DM

Interações Gestuais para Plataformas de Streaming de Vídeo

DISSERTAÇÃO DE MESTRADO

Francisco João Mendonça Teixeira

MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

maio | 2021

Interações Gestuais para Plataformas de Streaming de Vídeo

DISSERTAÇÃO DE MESTRADO

Francisco João Mendonça Teixeira
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Pedro Filipe Pereira Campos

CO-ORIENTAÇÃO
Diogo Nuno Crespo Ribeiro Cabral

Resumo

A popularidade do rato de computador como modo preferencial de interação com aplicações de computador é inigualável. Mesmo no tempo dos *smartphones* e *tablets*, que trouxeram ao público em geral inovações como o modo de controlo através do toque, não existe uma grande variedade nos modos de interação com os diversos dispositivos que usamos no quotidiano.

Este trabalho propõe uma nova abordagem para controlo, à distância, de aplicações de *streaming* de vídeo. A aplicação desenvolvida nesta dissertação procura um meio mais natural de interação com este tipo de plataformas usando interações gestuais, captadas pela câmara de profundidade RealSense D435. Câmaras de profundidade, ao contrário de câmaras de vídeo comuns, utilizam um fluxo de luz infravermelha para determinar a distância dos objetos relativamente à câmara. A câmara utilizada neste projeto possui capacidades de captura tanto neste modo, como no modo comum RGB.

Foi criada de raiz uma interface gráfica inspirada pela plataforma Netflix, para permitir o controlo dos seus botões e os vídeos reproduzidos, programaticamente. Foram desenvolvidos dois protótipos com tecnologias distintas para interagir com a interface. O primeiro, utiliza métodos de tratamento de imagem para retirar a informação pretendida, e um modelo pré-treinado de reconhecimento de gestos para identificar o gesto realizado. O segundo, integra um modelo de deteção e classificação de objetos otimizado para deteção e rastreamento de mãos entre *frames*. Foram feitas análises experimentais entre ambos e selecionado o segundo protótipo para prosseguir a testes com utilizadores.

A aplicação foi testada por 11 participantes, num estudo que comparou os seus tempos de utilização da aplicação por meio de interações gestuais com os tempos de utilização com o rato de computador. O sistema mereceu, por parte dos participantes, uma pontuação SUS “aceitável”.

Conclui-se que o sistema alcançou as expectativas iniciais, ainda que possua limitações aparentes que poderão ser mitigadas em iterações futuras.

Palavras-chave:

Interação gestual, Deteção de objetos, Subtração do plano de fundo, Segmentação do primeiro plano, Rastreamento gestual, MediaPipe

Abstract

The popularity of the computer mouse as the preferred interaction method with computer applications is unrivalled. Even in the age of smartphones and tablets, which have introduced touch screens to the masses, there seems to be a lack of variety regarding interaction methods with devices that we use every day.

This work proposes a new approach to interacting with video streaming applications from afar. The program developed for this dissertation looks for a more natural way of controlling these services by using gestural interactions, which are captured by the depth camera RealSense D435. Depth cameras, unlike regular video cameras, use an infrared feed to determine the distance of objects in frame, relative to the camera. The camera used in this project offers both infrared and RGB capturing modes.

A graphical interface, inspired by streaming platform Netflix, was created from scratch to allow control over its buttons and the video displayed programmatically. Two prototypes to interact with the GUI were developed using distinct technologies. The first one only uses image treatment methods to get the desired data and a pre-trained model for gesture recognitions to identify the gesture performed. The second one integrates an object detection and classification model optimized for hand detection and tracking between frames. Experimental analyses were run to compare both prototypes, resulting on the second being selected to proceed to the user tests stage.

The application was tested by 11 participants, in a study that tracked their times using the application via gestural interactions as well as with a computer mouse. The system merited an “acceptable” usability score by the participant.

We can conclude the system met the initial expectations despite its apparent limitations, which can be mitigated in future iterations.

Keywords:

Gestural interactions, Object detection, Background subtraction, Foreground segmentation, Hand tracking, MediaPipe

Agradecimentos

Esta dissertação conclui uma longa jornada de aprendizagem que não teria sido possível sem o apoio da minha família e amigos mais próximos.

Quero deixar um especial agradecimento à minha família, aos meus pais e irmão. Aos meus pais, devo eterna gratidão por todo o apoio incondicional e confiança depositada em mim. Ao meu irmão, que sempre esteve do meu lado, um sincero e profundo obrigado.

Um obrigado também a todos os meus colegas de curso com quem me cruzei durante este percurso académico. Um especial obrigado aos meus companheiros Diogo Cruz, Diogo Nóbrega e Marco Lima, por todos os momentos vividos e partilhados durante esta fase da minha vida.

Quero agradecer também aos meus orientadores, Dr. Diogo Cabral e Dr. Pedro Campos, por toda a orientação e apoio dado durante o desenvolvimento desta dissertação.

Um muito obrigado a todos os amigos que conheci até hoje, por me ajudarem a tornar na pessoa que sou.

Índice

Índice de figuras

Índice de tabelas

Lista de abreviaturas

2D	duas dimensões
3D	três dimensões
cm	centímetros
API	*Application Programming Interface*
BGR	*Blue, Green, Red*
CPU	*Central Processing Unit*
CTA	*Concurrent Think Aloud*
CUDA	*Compute Unified Device Architecture*
CuDNN	*CUDA Deep Neural Network*
ENet	*Efficient Neural Network*
fps	*frames per second*
GPU	*Graphics Processing Unit*
IoU	*Intersection over Union*
JSON	*JavaScript Object Notation*
NMS	*Non-maximum supression*
OpenCV	*Open Source Computer Vision Library*
PDA	*Personal Digital Assitant*
R-CNN	*Region-based Convolutional Neural Networks*
RGB	*Red, Green, Blue*
SD	*Standard Deviation*
SDK	*Software Development Kit*
SSD	*Single Shot MultiBox Detector*
SUS	*System Usability Scale*
USB	*Universal Serial Bus*
YOLO	*You Only Look Once*

1. Introdução

Nos últimos anos, o rato de computador continua a ser o modo preferencial de interação com os sistemas multimédia e eletrónicos em computadores. Mantendo-se relativamente inalterado desde os anos 80, a sua popularidade como acessório para computadores mantém-se inigualável, mesmo na era dos *smartphones* e *tablets* (Pogue, 2013; Shiels, 2008). Para além dos meios tradicionais e ferramentas de interação por toque, a maioria dos dispositivos que usamos no nosso dia-a-dia funcionam via feixes de luz infravermelha, como, por exemplo os comandos de controlo remoto. Estes aparelhos, sob a forma de comandos de televisão, portões, entre outros, permitem ao utilizador controlar sistemas à distância. Contudo, o bloqueio deste feixe de luz por parte de objetos é bastante comum. Por outro lado, estes controladores requerem habitualmente o uso de baterias, e possuem, por vezes, botões em demasia que criam ambiguidade e confusão ao utilizador.

No entanto, apesar do rato se manter como o dispositivo de mapeamento convencional mais preciso e exato, existem novas tecnologias que poderão, ainda que não substituir totalmente, levar à conceção de novos dispositivos focados em particulares áreas de utilização. Por exemplo, a caneta digital foi pensada com uma visão específica – caligrafia digital. Originalmente desenhada para a criação de notas em documentos, evoluiu até ao ponto de se tornar o ponto focal de controlo de certos sistemas (e.g. *PDAs*), bem como permitir a construção de retratos digitais de forma mais natural e fluída (Dormehl, 2021).
Este projeto pretende explorar as hipóteses de manipulação de informação num sistema de *streaming* de vídeo com recurso a interações gestuais entre o utilizador e a aplicação. Pretendemos, analogamente aos casos anteriores, encontrar um método que ofereça maior naturalidade no controlo dos dados apresentados no ecrã do computador.

Para encontrar um método de mais fácil assimilação para o utilizador, é preciso ter em conta os hábitos mais predominantes empregues na comunicação interpessoal e nos sistemas mais presentes no mercado atual. Além da fala, os movimentos gestuais são de extrema importância neste processo. Estes gestos ajudam-nos a transmitir uma mensagem

e a captar a atenção do público alvo (Goldin-Meadow and Alibali, 2013). A ideia base por detrás da implementação de técnicas de captação usando uma câmara de vídeo, apoia-se no vasto número de combinações possíveis entre a silhueta da mão e a sua posição no espaço. Pretende-se construir um protótipo que tentará associar os gestos que usamos no dia-a-dia com ações implementadas para controlo de uma aplicação de *streaming* de vídeo

As câmaras digitais padrão produzem uma imagem como uma matriz de duas dimensões de pixéis. Cada pixel tem valores RGB associados. Cada valor é um número natural entre 0 e 255. Um pixel com valor 0 nos três canais, (0, 0, 0), é considerado preto absoluto. Por conseguinte, (255, 0, 0) representa a cor vermelho, (255, 255, 255) o branco, etc. A combinação de todos estes pixéis com as suas variadas cores e tons formam as imagens que apreciamos no final. Nas câmaras de profundidade, os valores RGB são substituídos por um único valor de distância do objeto relativamente à câmara. Neste projeto serão analisados ambos modos de captura com a câmara RealSense D435. Explorar a utilização destes sistemas como meio de interação humano-computador pode provar-se viável, especialmente ao considerar a facilidade de montagem do sistema.

1.1. Objetivo

Atendendo ao contexto deste trabalho, propõe-se o desenvolvimento de uma aplicação, para computador, que explore diversos métodos de interação não tradicionais para controlo de um sistema de visualização de *streaming* de vídeo. Para este efeito, a aplicação desenvolvida terá, imprescindivelmente, que suportar o processamento de vídeo em tempo real. A capacidade de analisar o *feed* da câmara sem causar atrasos percetíveis é de extrema importância para o bom funcionamento do sistema, de forma a permitir uma operação natural, contínua e fluída com a interface.

O estudo das capacidades e limitações do sistema é a principal prioridade nesta dissertação, dado que o seu objetivo é entender a potencialidade de integração com plataformas usadas no dia-a-dia.

A utilização da aplicação deverá ser a mais simples possível e evitar uma curva de aprendizagem acentuada. Utilizadores comuns (i.e., utilizadores sem conhecimentos prévios do sistema ou as suas tecnologias), deverão ser capazes de interagir com a aplicação sem observação e intervenção por parte de terceiros.

Serão estudadas, brevemente, diversas abordagens para identificação e classificação de interações gestuais, para alcançar o objetivo proposto. Procura-se uma solução que não comprometa a fiabilidade dos resultados obtidos em troca de tempos de processamento mais reduzido.

De reforçar que esta dissertação, com o trabalho efetuado, não procura substituir o rato de computador comum, mas apenas estudar e explorar alternativas num contexto específico, com o intento de contribuir para a evolução e inovação neste espaço.

1.2. Contribuições

No âmbito desta dissertação, foi desenvolvido um programa para computador com uma interface gráfica que permita estudar a interação de utilizadores com um sistema não convencional de *streaming* de vídeo. Foram combinados vários temas abordados na revisão de literatura e procurado implementar os mais prometedores. Foram desenvolvidos dois protótipos com tecnologias distintas:

- Protótipo OpenCV – trabalha sobre a imagem com processos de tratamento da imagem e um modelo de classificação de gestos.
- Protótipo MediaPipe – utiliza um modelo de classificação de objetos para deteção da mão na imagem e retorna um objeto esqueleto com 21 pontos de referência, o qual analisamos para interpretar o gesto realizado.

Após o desenvolvimento de ambos os protótipos estar concluído, procedeu-se à comparação entre os mesmos.

- Foram feitas análises experimentais e comparações entre as duas abordagens. Destas análises resultou a seleção de um único protótipo para prosseguir com os testes com utilizadores.
- Foi elaborado um estudo com 11 participantes, no qual foi avaliada a usabilidade do sistema. Neste estudo, os utilizadores realizaram um conjunto de tarefas comuns à utilização do tipo de aplicação desenvolvida. As métricas registadas usando as técnicas implementadas foram comparadas com a utilização do sistema com o rato de computador.
- Por último, são discutidos os problemas encontrados, as limitações associadas à natureza do sistema, e sugeridas algumas considerações referentes a iterações futuras.

1.3. Estrutura do documento

Este documento está divido em cinco capítulos, que expõem um processo de pesquisa e subsequente prototipagem de duas abordagens distintas para encarar o problema apresentado.

No primeiro capítulo é feita uma breve introdução ao tema e à necessidade de inovação no espaço. De seguida, são expostos os objetivos do trabalho, bem como as suas contribuições.

No segundo capítulo é feita uma revisão de trabalhos relacionados e aplicações comerciais relevantes que façam uso das metodologias abordadas. São explicadas algumas abordagens e processos de análise de imagens, os problemas associados a cada uma, e justificada a sua utilidade para o objetivo proposto.

O terceiro capítulo encontra-se dividido em três subcapítulos. No primeiro ponto, é exposto o processo de criação do primeiro protótipo, que explora as técnicas de subtração do plano de fundo e segmentação do primeiro plano. É abordado o seu desenvolvimento e arquitetura, concluindo com os seus pontos fortes e fracos. No segundo ponto, é exposto o desenvolvimento do segundo protótipo, que implementa um modelo próprio de classificação de objetos, otimizado para deteção de mãos. É, novamente, abordado o processo de desenvolvimento e analisadas as limitações deste novo protótipo. Por fim, no terceiro subcapítulo, é examinada a interface desenvolvida, comum a ambos os protótipos, os seus elementos de navegação, e pistas visuais para ajudar na confirmação de ações bem-sucedidas por parte do utilizador.

No quarto capítulo são relatadas análises experimentais aos dois protótipos criados, de forma a selecionar um único protótipo para testagem com utilizadores reais. Seguidamente são feitos testes para averiguar a competência do sistema do ponto de vista do utilizador comum, e apreciados os resultados obtidos.

No quinto e último capítulo, são retiradas conclusões alusivas ao funcionamento e desempenho do sistema montado, bem como a viabilidade da sua integração em plataformas usadas no quotidiano. Por fim, são mencionados os principais obstáculos encontrados e sugeridas possíveis soluções para iterações futuras.

2. Trabalhos relacionados e contexto

Para o início do desenvolvimento deste trabalho, foi feita uma pesquisa de material relacionado com o tema de manipulação de informação, em ambientes que façam uso de tecnologias de *computer vision*. Por extensão, foram cobertos e explicados alguns métodos de deteção de objetos.

A câmara RealSense D435 ("Depth Camera D435," 2021) utilizada neste projeto permite-nos explorar abordagens particulares ao recorrer das suas lentes de luz infravermelha. Ao usar este modo de captura de imagens, conseguimos receber informação sobre a distância da câmara aos objetos no seu enquadramento, e criar mapas de profundidade para visualização dos resultados obtidos. A câmara RealSense permite-nos averiguar tanto esta abordagem, como as técnicas de processamento mais comuns de imagens RGB.

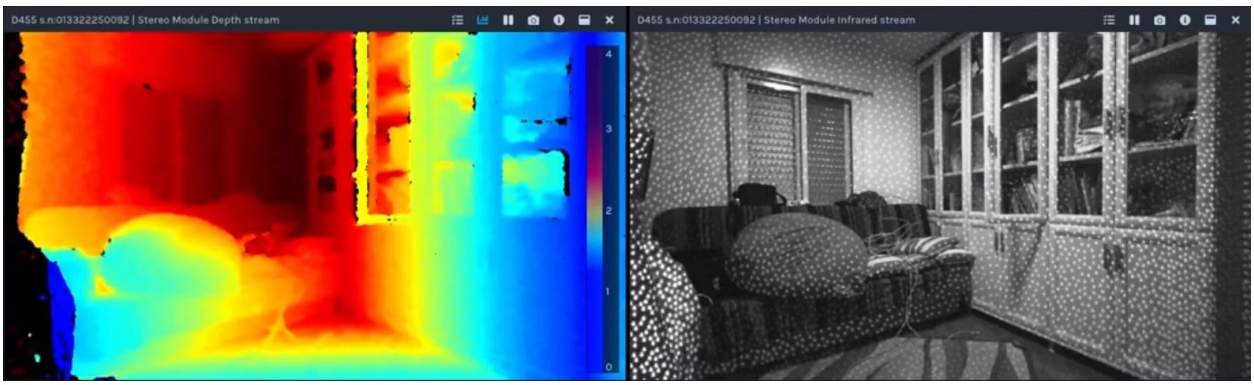


Figura 1 – Mapa de profundidade obtido com a câmara RealSense D455 (Intel RealSense, 2021)

Foram identificados dois modos gerais de proceder à primeira fase de deteção de mãos em imagens: Em primeiro lugar, é possível identificar um plano de fundo (*background*) na imagem, e tratar novos elementos sobrepostos como mãos. Outra alternativa é a possibilidade de utilizar *frameworks* específicas à deteção e classificação de objetos para reconhecer mãos e dedos em imagens.

Esta revisão de artigos e outros conteúdos multimédia provou-se vantajosa pois permitiu identificar certos problemas a evitar na construção do protótipo, para além de propor possíveis futuras utilizações do sistema.

2.1. Aplicações relacionadas

O primeiro material de pesquisa referenciado é uma dissertação de mestrado remontante ao ano 2018. Aguiar (2018) teve como intuito a manipulação de imagens recorrendo a uma *frame* háptica para exploração das capacidades de deteção de profundidade com a câmara RealSense D300.

O autor refere como objetivos a criação de um cenário de utilização da câmara mencionada para substituição dos tradicionais dispositivos de *input* (rato e teclado), juntamente com a incorporação de uma superfície tátil para fornecer *feedback* háptico ao utilizador. Este trabalho distingue-se dos demais neste subcapítulo devido à sua origem académica, e não comercial, como os restantes. No entanto, do mesmo resulta um protótipo físico e distinto de outros exemplos expostos, pelo que é analisado separadamente.

Em (Aguiar, 2018), o protótipo criado coloca uma folha de acetato entre dois quadros de madeira. O acetato, por ser um material transparente e semi-flexível, providencia ao utilizador um certo feedback quanto ao nível de profundidade imprimido nas suas ações. O autor refere ainda que o acetato reflete parte da luz infravermelha que a RealSense D300 usa para captação de informação. No entanto, os resultados foram considerados toleráveis para prosseguir com o projeto.

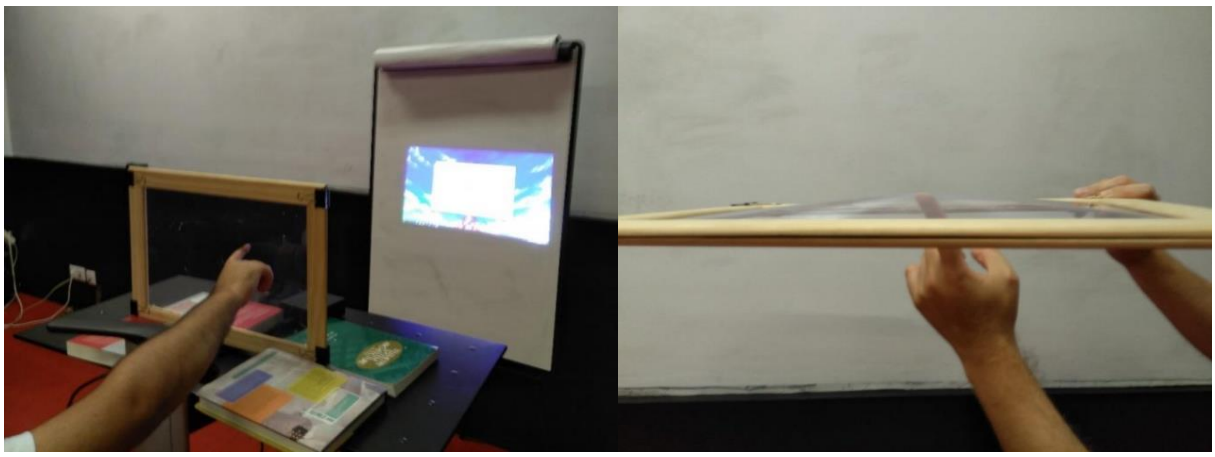


Figura 2 – Frame háptica desenvolvida pelo autor (Aguiar, 2018)

O projeto utiliza uma versão anterior da câmara utilizada nesta dissertação. Porém, a Intel descontinuou a primeira versão do seu *SDK* (versão usada no projeto referenciado) e removeu quaisquer ferramentas de rastreamento de faces ou mãos na segunda versão agora disponível, pelo que não é possível recriar a solução analisada.

Certas aplicações para *smartphones* e *tablets* tentam também fazer uso de tecnologias da área de *computer vision*; é o caso da nova funcionalidade na *app* "Djay Pro AI" para iPad. Uma atualização da *app* introduziu uma opção de controlo sem toque (*touchless*), recorrendo à *Vision – framework* de *computer vision* da Apple ("Gesture Control | Algoriddim," 2021; Turi, 2020). A nova *framework* da Apple permite a deteção de faces e marcos de referência, bem como classificação de objetos a partir de modelos *machine learning* customizados ("Vision | Apple Developer Documentation," n.d.).

Figura 3 – Esquema de exemplo de utilização do modo sem toque ("Gesture Control | Algoriddim," 2021)

O novo modo de controlo usa a câmara do iPad Pro para detetar as duas mãos do utilizador que deverão estar sobre a câmara. Atualmente, apenas são possíveis aplicar três efeitos com esta abordagem sem toque. O utilizador usa uma das mãos para "agarrar" o efeito pretendido e movimenta a mão livre para o aplicar.

Figura 4 – Exemplo de utilização e aproximação da interface da aplicação (DJ Ravine, 2020)

As principais limitações deste sistema, para além da falta de reconhecimento de gestos complexos, é a necessidade de boa iluminação no espaço envolvente e um campo de visão desimpedido entre a câmara e as mãos. Por outro lado, não requer a adição de nenhum componente adicional, apenas o dispositivo em si.

Um conceito algo diferente dos expostos ao longo desta dissertação é a integração destas ideias num ambiente de realidade virtual. Os óculos de realidade virtual Oculus

Quest 2 têm um modo experimental de rastreamento de mãos disponível para usar com a interface do Oculus e certas aplicações compatíveis (“Hand Tracking | Oculus,” 2021).

Os óculos fazem a deteção das mãos, e do espaço em redor, com as quatro câmaras inseridas na parte frontal do aparelho.

Figura 5 – Óculos de realidade virtual Oculus Quest 2 e as suas quatro câmaras

As câmaras presentes, com o seu modo infravermelho, conseguem captar os objetos em redor e a sua distância relativa aos óculos, para uma representação precisa em ambiente 3D. Isto estende-se às mãos do utilizador. O Quest 2 reconhece e deteta as mãos do utilizador e implementa algoritmos que permitem para segui-las e as suas rotações no espaço. Estão previstos dois gestos para controlar as interfaces compatíveis no sistema: o *pinch* (juntar o polegar e o dedo indicador) e o *grab* (gesto de fechar a mão). Tivemos a oportunidade de testar extensivamente este sistema. Para selecionar um elemento na interface, apontamos a mão até o laser virtual estar na posição desejada. Depois, geralmente, o gesto *pinch* realiza a ação principal. Para fazer *scroll*, vertical ou horizontal, basta fazer *pinch* e mover a mão no sentido desejado.

Figura 6 – Aspeto do sistema visto pelo utilizador (Virtual Reality Oasis, 2019)

O sistema oferece excelente resultados de elevada confiança, estando o espaço bem iluminado. A sua única limitação percetível resume-se a manter as mãos dentro do campo de visão das câmaras. Por vezes, por ser uma implementação bastante natural e bem conseguida, temos tendência a olhar para um lado e tentar controlar um objeto para lá da nossa visão periférica.

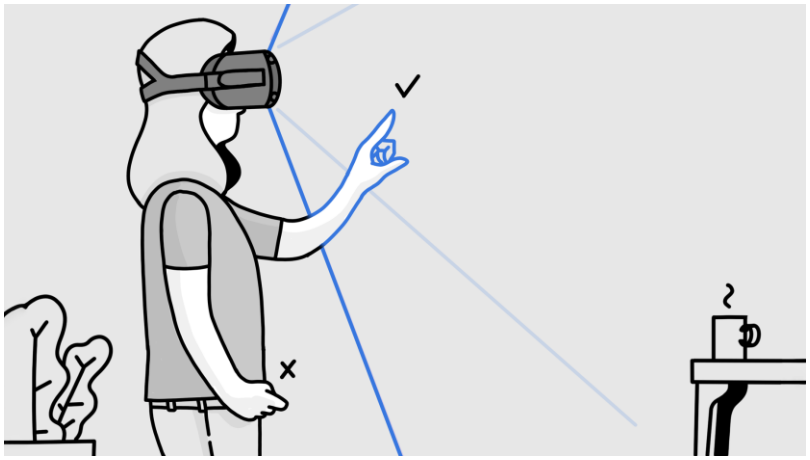


Figura 7 – Representação do campo de visão do Oculus Quest 2 (Lang, 2020)

As implementações sobreditas influenciaram o projeto desenvolvido nesta dissertação, pois ofereceram dados importantes acerca das limitações esperadas na montagem deste tipo de sistemas e a sua usabilidade moderada.

2.2. Subtração do plano de fundo

A abordagem explorada nesta secção implementa técnicas de subtração do plano de fundo (*background subtraction*) para determinar a posição e aspeto de objetos. O conceito de subtração do plano de fundo passa por estudar uma imagem estática, e subtrair os novos elementos inseridos na *frame*. Este método não classifica os novos elementos encontrados; apenas sabe que existem novos objetos na imagem que não se encontravam presentes anteriormente. De notar que todos os métodos explorados neste capítulo estão incluídos de raiz na biblioteca OpenCV, introduzida no ponto 3.1.1.2.

A subtração do plano de fundo – por vezes chamada segmentação do primeiro plano (*foreground segmentation*) – é normalmente aplicada em contextos de vídeo em tempo real. Resumidamente, numa primeira instância, é retirada uma *frame* (ou uma média de n *frames*) do *feed* da câmara e denominada de plano de fundo. Nas seguintes *frames*, procede-se ao processamento da diferença entre a *frame* atual e o plano de fundo. Segundo este método, os componentes estáticos serão si próprios na nova *frame*, enquanto os novos elementos em questão serão objetos ou pessoas em movimento. Geralmente, é feito um *thresholding* na máscara resultante da subtração efetuada, transformando-a num objeto completamente branco e com a silhueta da máscara original.

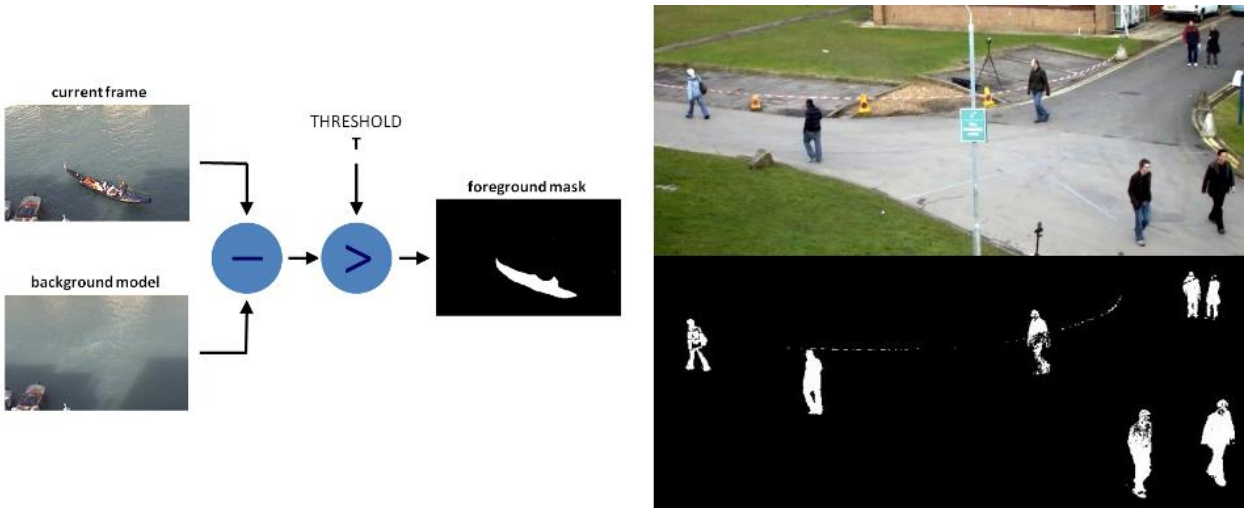


Figura 8 – Exemplos de background subtraction ("Background Subtraction - OpenCV-Python Tutorials," 2021; "OpenCV: How to Use Background Subtraction Methods," n.d.)

Na prática, são efetuadas várias operações para separar a máscara do primeiro plano do plano de fundo e reduzir o ruído existente.

O primeiro passo é identificar o plano de fundo – por vezes referido como modelo de fundo (*background model*). Isto pode ser conseguido utilizando apenas uma única *frame*, ou a média de um número arbitrário de *frames*. Este segundo método denomina-se filtro de média temporal (*temporal median filter*). Liu et al. (2016b) e Babaee et al. (2019) apresentam soluções mais evoluídas para estimar, na íntegra, o plano de fundo em vídeos em que o mesmo se encontra relativamente ocluído. Nesta revisão usaremos uma aplicação do filtro de média temporal mais simples e usualmente implementada.

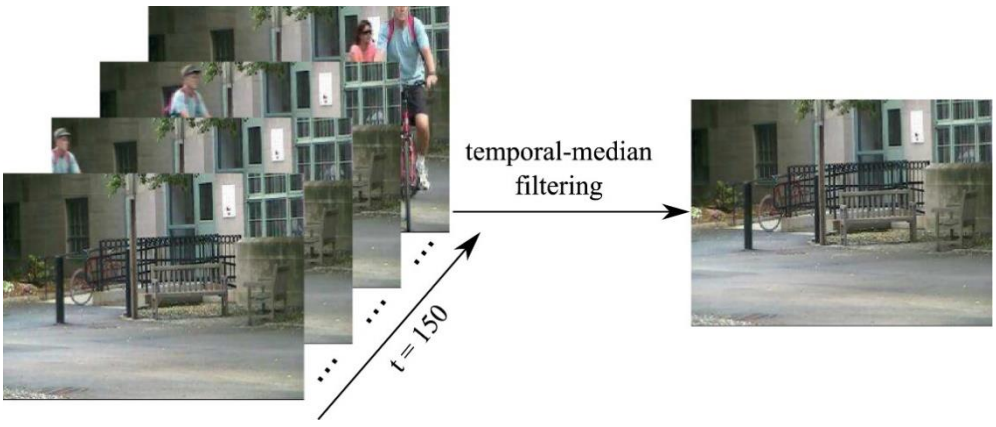


Figura 9 – Estimação do plano de fundo ao longo de 150 frames (Babaee et al., 2019)

Agora que se tem informação sobre a parte estática da cena, pode-se simplesmente subtrair à *frame* atual o plano de fundo (*background*). O resultado desta segmentação é uma máscara composta pelas diferenças entre as duas imagens (em que, idealmente, para a nossa aplicação, se encontrará apenas a mão). Esta máscara chama-se máscara *foreground* (de primeiro plano).

Figura 10 – Exemplo de máscara foreground (Li, 2021)

De seguida introduzimos filtros, como *Gaussian blur* (Misra and Wu, 2020; "Wikipedia - Gaussian blur," 2021), que desfoca a imagem de forma a suavizar a transição entre cores e os contornos da máscara, reduzindo o ruído existente.

Figura 11 – Máscara foreground após aplicado o filtro "Gaussian blur" (Li, 2021)

Seguidamente, usamos *thresholding* binário para converter o primeiro plano branco e o plano de fundo preto (Padilha, 2013). Esta técnica traz duas vantagens: primeiramente,

cria contornos nítidos e mais bem definidos; em segundo lugar, é um modo de generalizar os resultados obtidos, de forma que um modelo de reconhecimento gestual incorporado posteriormente possa acomodar utilizadores com diferentes tons de pele. Ficamos, desta maneira, com uma máscara que se assemelha à silhueta da mão.

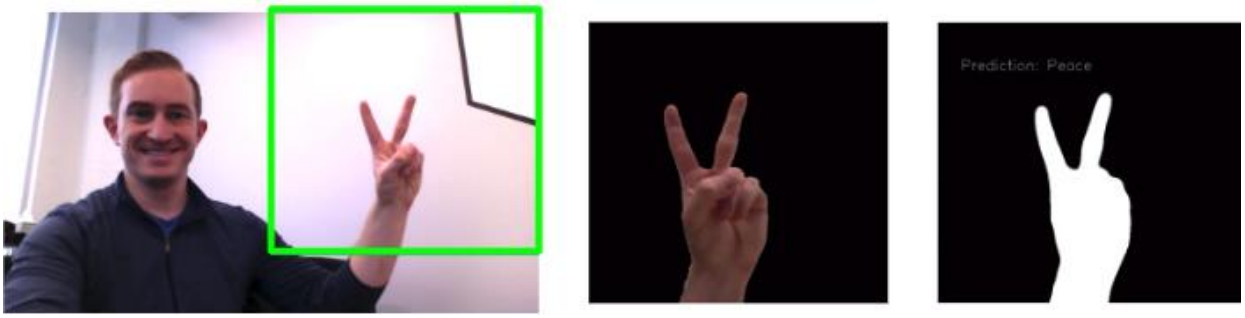


Figura 12 – Da esquerda para a direita: imagem original, máscara foreground, máscara com threshold binário (Heintz, 2018)

Aqui, podemos escolher implementar modelos treinados com redes neuronais para estimar poses e gestos efetuados com a mão, ou usar técnicas de análise da máscara para, por exemplo, saber quantos dedos se encontram erguidos.

2.2.1. Modelo de reconhecimento gestual

Analisemos a primeira opção. Para tal, é corrido um modelo de reconhecimento gestual pré-treinado, ao qual passamos a máscara *threshold*. O modelo estará treinado para reconhecer gestos específicos que sejam semelhantes às imagens de referência utilizadas para o treino da rede neuronal.

Geralmente, os modelos criados destinados à interpretação destas máscaras *threshold* são treinados com fotografias a preto e branco e com um aspeto fixo. Esta abordagem permite generalizar o processo de reconhecimento para qualquer utilizador, não sendo influenciado, por exemplo, pelos diferentes tons de pele possíveis.

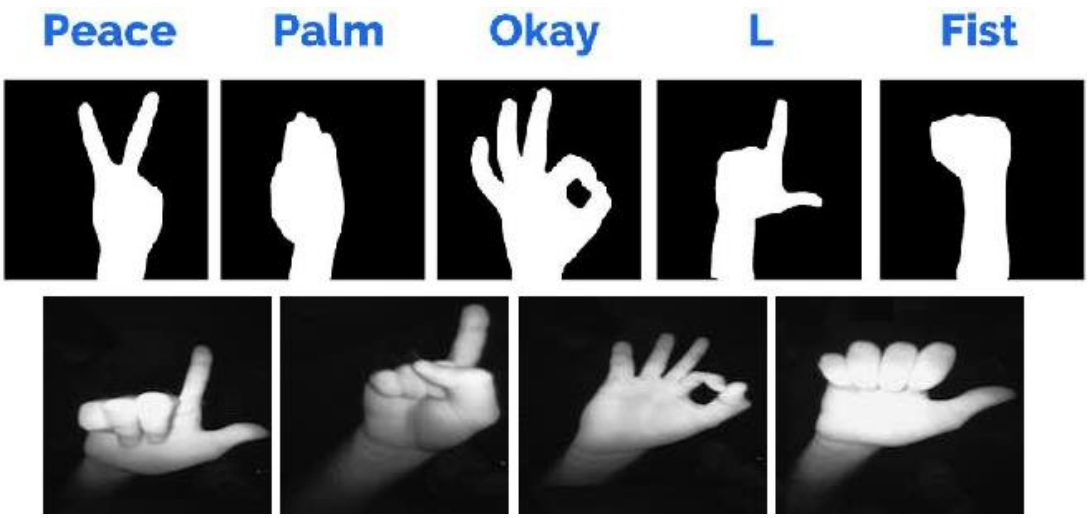


Figura 13 – Exemplos de gestos identificados pelo modelo em (Heintz, 2018): em cima, poses associadas a cada gesto; em baixo, algumas das imagens de referência usadas para treinar o modelo

Após as comparações, o detetor retorna a probabilidade associada a cada um dos gestos. O gesto que apresente uma probabilidade maior que um certo valor estabelecido, corresponderá a uma certeza de estarmos perante uma suposição correta. No código, cada ação está relacionada com uma ação a realizar. Na figura seguinte, Heintz (2018) cria um simples programa, que integra com a sua *smart home*, realizando ações como "diminuir volume" e "acender luzes, música ligada".

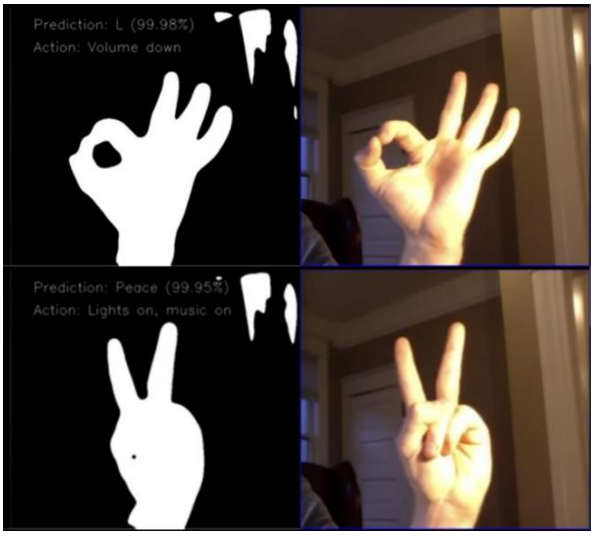


Figura 14 – Exemplo das predições dos gestos "L" e "Peace", respetivamente (scgrinchy, 2018)

2.2.2. Análises das máscaras de primeiro plano

Por outro lado, podem ser levadas a cabo análises da máscara *threshold* sem que seja necessário recorrer a modelos e redes neuronais. Ilango (2017) apresenta um modo distinto para contar o número de dedos erguidos a partir da máscara obtida. Com a máscara de primeiro plano, ao examinar os seus contornos com a biblioteca OpenCV, conseguimos isolar o segmento de maior área numa região da imagem, e assim encontrar a mão na *frame*.

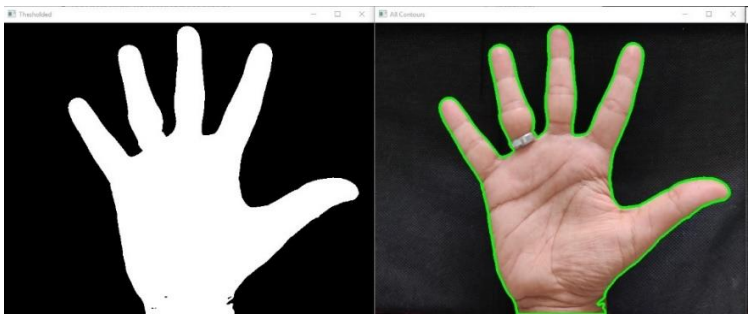


Figura 15 – À esquerda, a máscara threshold; à direita, os contornos da área a verdes, desenhados sobre a imagem original (Ilango, 2017)

Novamente, recorrendo a funções nativas do OpenCV, procedemos ao encontro dos extremos (topo, base, esquerda e direita) dos contornos. Ao conhecer as coordenadas destes quatro pontos, uma média para descobrir o ponto central. Este ponto não alinha perfeitamente com o centro da palma da mão, e tende a inclinar-se mais para o lado do polegar. No caso da figura seguinte, a média para a coordenada central no eixo horizontal é feita com as extremidades dos dedos polegar e mínimo, resultando num ponto médio mais à direita que o real.

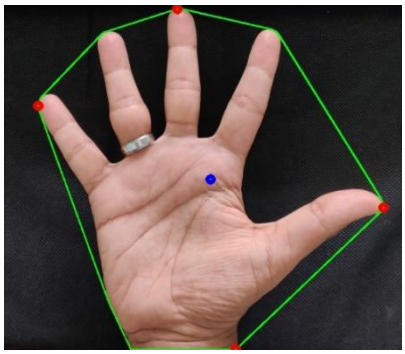


Figura 16 – Exemplo de mão com os contornos (a verde), pontos extremos (a vermelho) e ponto central (a azul) (Ilango, 2017)

De centro no ponto azul, é criada uma circunferência, de raio ligeiramente menor que a distância aos extremos. Ao fazer uma operação *AND* a nível binário entre a máscara *threshold* e a circunferência, resulta uma circunferência cortada, onde restam apenas as zonas em que as extremidades (dedos e pulso) ultrapassam o perímetro do círculo.

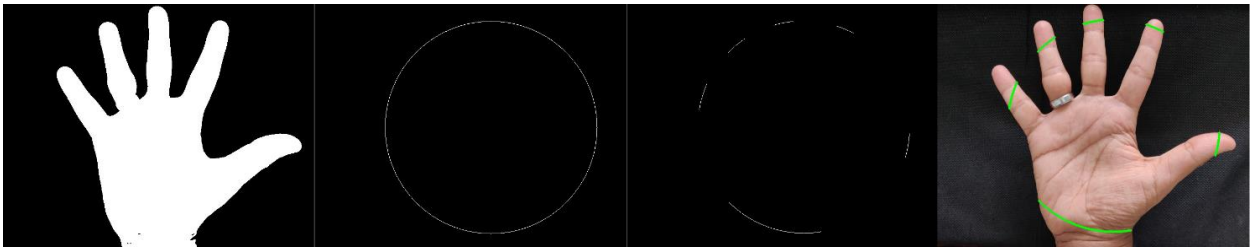


Figura 17 – Interceção da máscara threshold e a circunferência criada (Ilango, 2017)

Após eliminar a zona do pulso – por eliminação da maior interceção – resta um número de interceções equivalente ao número de dedos erguidos. Este método de calcular o número gesticulado com os dedos à câmara, sem a utilização de modelos e detetores, é preferível ao anterior, porquanto se reduz a memória alocada necessária ao projeto.

Em (Li, 2021), consta-se uma abordagem diferente ao mesmo problema. A ideia executada pelo autor consiste em, primeiro, detetar os contornos da mão (como na Figura 15) e, em seguida, calcular quantas deformidades de ângulo superior a 90 graus existem. A proposta baseia-se no facto das regiões entre dedos serem bastante distintas na anatomia da mão, identificadas por um ângulo agudo acentuado. Se forem encontrados quatro pontos com esta descrição, compreende-se estarem cinco dedos erguidos.

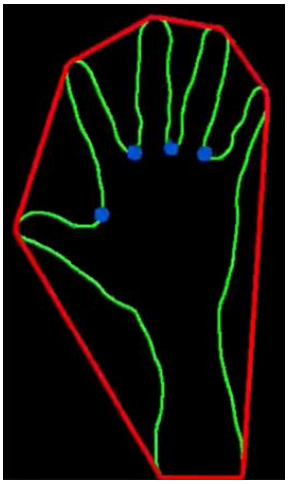


Figura 18 – Pontos azuis demarcam os ângulos agudos entre dedos (Li, 2021)

Em suma, existem várias técnicas e abordagens de analisar imagens e retirar informação relevante recorrendo unicamente a métodos de manipulação de imagens, sem ser necessário equipamento específico ou dispendioso. Não obstante, todas apresentam as suas limitações. O primeiro método explorado neste capítulo faz uso de um modelo pré-treinado que requer tempo e recursos únicos. A segunda abordagem apenas reconhece um dedo como extenso se ultrapassar a circunferência definida. Por fim, o último método, ao obter resultados mais consistentes que o anterior, não permite diferenciar entre estar apenas um ou nenhum dedo erguido.

Ademais, todas estas técnicas partilham limitações em comum. Por se tratarem de análises a *frames* RGB obtidas sem informação extra sobre os dados na imagem, súbitas mudanças na iluminação do ambiente são o suficiente para tornar o plano de fundo (*background*) estimado inutilizável. Outra restrição ao operar, concretamente, o sistema via interação gestual à distância, é a necessidade de regiões estritamente definidas na imagem, onde deverá encontrar-se apenas a mão a controlar o sistema. Isto resulta da impossibilidade de diferenciar sobre a natureza do objeto em primeiro plano. Uma limitação final é a obrigação de um plano de fundo estático. Um *background* em movimento não permitirá, com as técnicas analisadas, criar uma imagem uniforme sobre a informação que deverá ser ignorada. Assim, os sistemas montados para captação destas imagens têm, também eles mesmos, que permanecer em repouso total.

Os exemplos analisados neste subcapítulo utilizam todos eles câmaras comuns com imagens do tipo RGB. Poderá ser vantajoso estudar a adaptação destas técnicas a imagens de profundidade ao longo deste projeto, sendo a RealSense D435 capaz de capturar vídeo com ambas configurações.

2.3. NVIDIA CuDNN

Foi explorada a biblioteca NVIDIA CuDNN (*CUDA Deep Neural Network*) ("NVIDIA cuDNN," 2021), durante a fase de pesquisa para esta dissertação, por ser uma biblioteca disponibilizada ao público que permite utilizar o poder de processamento de placas gráficas NVIDIA para diminuir consideravelmente o tempo de processamento necessário em modelos de classificação de objetos ("NVIDIA CUDA Zone," 2021). Foi instalado o *toolkit* CUDA localmente, segundo Rosebrock (2020), e compilado o OpenCV, com as respetivas *flags*, processo descrito em ("OpenCV: Installation in Windows," 2021), de modo a tirar proveito da sua pronta integração com a CuDNN.

Ferramentas de deteção e classificação de objetos são a fundação sobre a qual grandes tecnologias, como a condução autónoma, assentam. Foram feitos vários avanços recentemente, numa tentativa de ir ao encontro da exigência e procura destes novos sistemas.

A NVIDIA lançou a *API* CUDA em 2007. Com a sua disponibilização ao público, surgiu a possibilidade de adaptar, livremente, placas gráficas NVIDIA para tratamento de elevadas cargas de trabalho nos algoritmos de deteção de objetos. A contínua evolução na indústria das *GPUs* é especialmente benéfica para a área de *computer vision*. Placas gráficas, face a uma arquitetura de computação paralela, juntamente com o seu elevado número de *cores*, distinguem-se na rápida execução dos processos tópicos. Ademais, *GPUs* recentes gozam de grandes quantidades de memória e largura de banda (*bandwidth*). Tais características reduzem o custo de transferência de dados entre *CPU* e *GPU*, prevenindo possíveis estrangulamentos no sistema, indicando a viabilidade da futura utilização de placas gráficas neste campo (Coates et al., 2009; Kashinkunti and Chabert, 2019).

Ao instalar o *toolkit* CUDA e compilar o OpenCV com os módulos necessários ao CuDNN, tem-se as ferramentas necessárias para proceder à sua testagem, discutida posteriormente neste capítulo. Optou-se, desde início, pela integração das *frameworks* CUDA e CuDNN, comprovada a sua superioridade sobre o processamento unicamente em *CPU*.

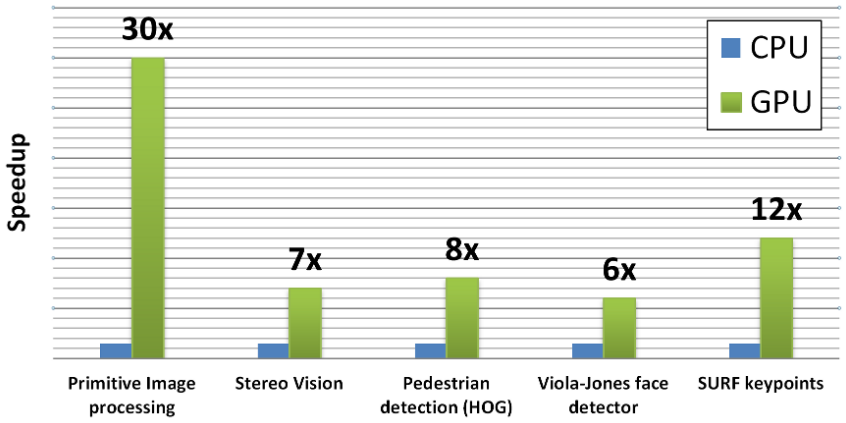


Figura 19 – Comparação de velocidade de processamento entre CPU e GPU, com o OpenCV, em diversos métodos ("OpenCV: CUDA," 2021))

Hangün and Eyecioğlu (2017) fazem uma análise detalhada da diferença de velocidade de processamento entre *CPU* e *GPU*. Os autores reportam as configurações do computador usado:

- *CPU*: Intel Core i7-6700 @3.40GHz ("Intel® Core™ i7-6700 Product Specifications," 2021)
- *GPU*: NVIDIA GeForce GTX 970 ("NVIDIA GeForce GTX 970 Specs," 2021)

Foi selecionado como principal critério de avaliação a relação entre tempo de processamento e resolução de imagem. As imagens de teste foram processadas com a sua resolução original, bem como redimensionadas para resoluções reduzidas e resoluções aumentadas. As figuras seguintes evidenciam a desigualdade dos tempos de tratamento das imagens entre processador e placa gráfica.

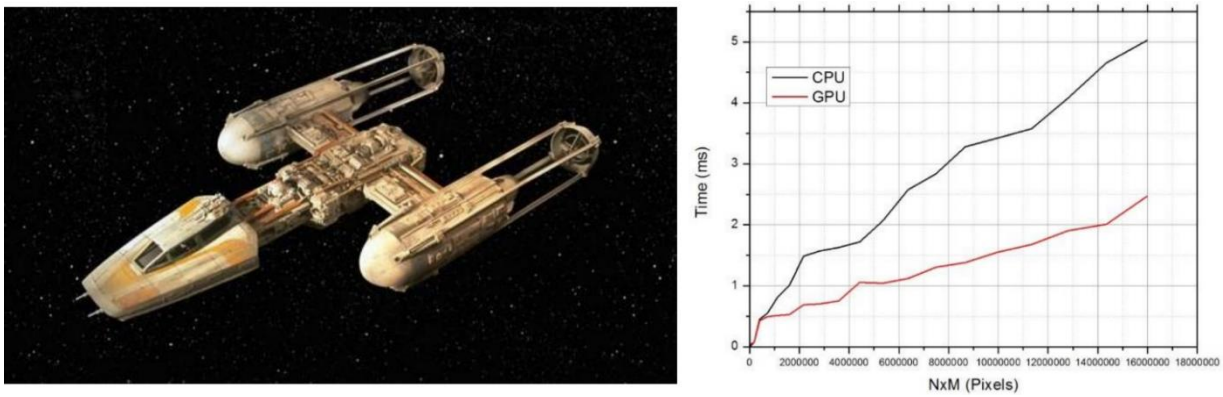


Figura 20 – Comparação de tempos no redimensionamento da imagem à esquerda (Hangün and Eyecioğlu, 2017)

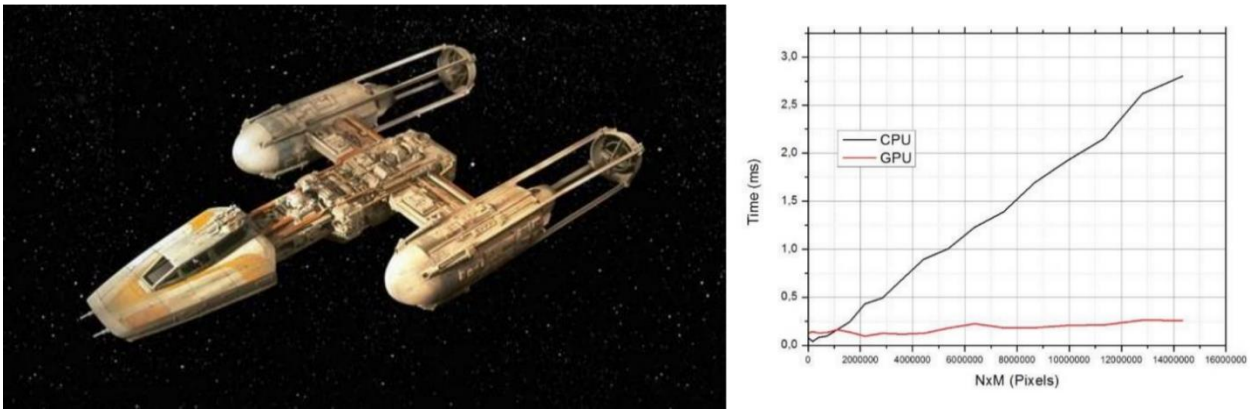


Figura 21 – Comparação de tempos no thresholding da imagem à esquerda (Hangün and Eyecioğlu, 2017)

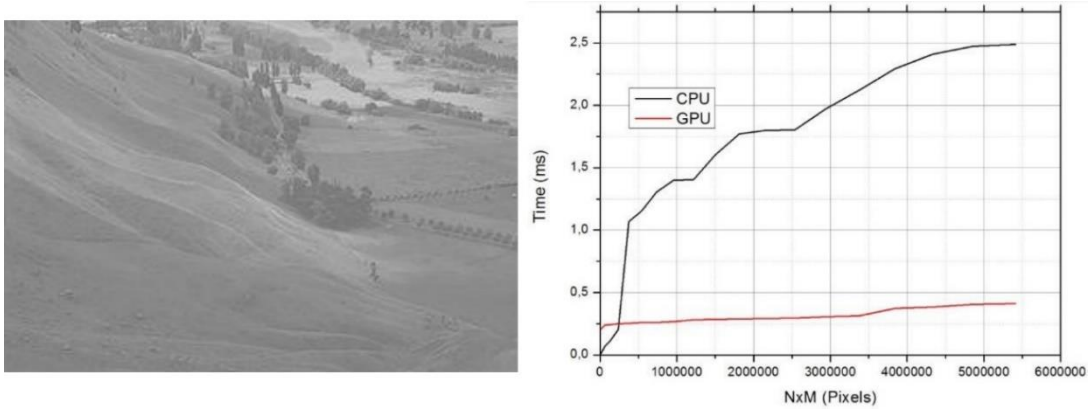


Figura 22 – Comparação de tempos na equalização do histograma da imagem à esquerda (Hangün and Eyecioğlu, 2017)

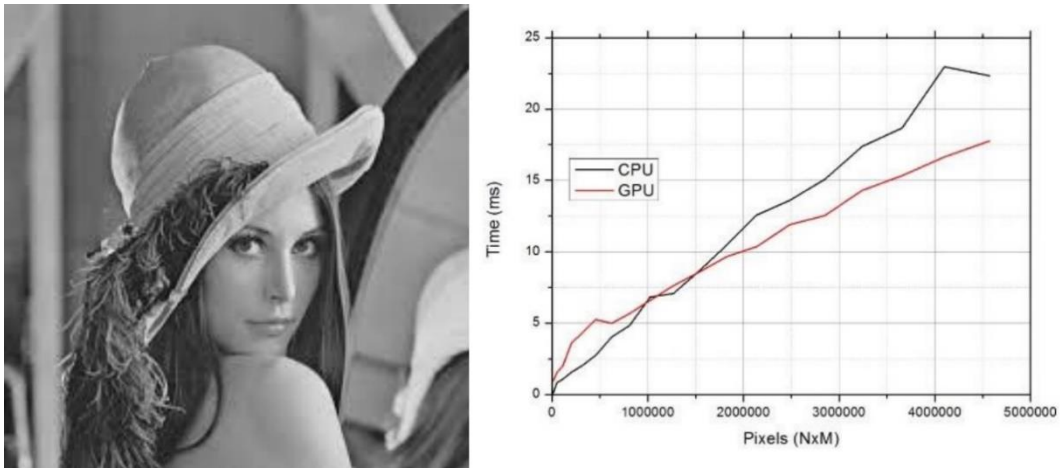


Figura 23 – Comparação de tempos na deteção de arestas da imagem à esquerda (Hangün and Eyecioğlu, 2017)

Para correr um módulo de deteção e classificação de objetos é necessário acesso a um modelo pré-treinado. Para tal, foi usado o modelo e código disponíveis em (Rosebrock, 2020b).

As configurações do computador em que foi testado o modelo, e em que será desenvolvido o projeto desta dissertação, são os seguintes:

- *CPU*: Intel Core i7-4770 @3.40GHz ("Intel® Core™ i7-4770 Product Specifications," 2021)
- *GPU*: NVIDIA GTX 1060 6GB ("NVIDIA GeForce GTX 1060," 2021)

Apesar dos testes terem sidos realizados no computador pessoal descrito, os seus resultados foram publicados precedentemente em (Cruz, 2020).

Em primeiro lugar, foram efetuados testes a um trecho, aproximadamente 30 segundos, de um vídeo de monitorização de trânsito disponível em (Panasonicsecurity, 2015). Foi escolhido o vídeo mencionado por ser de elevada qualidade e com grande número de deteções e classificações.

Formato:	Matrovska (AVC)
Bit rate:	17.0 Mbps
Largura:	3840 pixels
Altura:	2160 pixels
Frame rate:	29.970 FPS
Bit depth:	8 bits
Duração:	30.063s

Tabela 1 – Especificações do vídeo de teste

Com as especificações supramencionadas, tenciona-se testar a biblioteca frente a um exemplo computacionalmente exigente.

O vídeo foi analisado recorrendo a cinco modelos distintos de classificação de objetos, todos eles fazendo uso da tecnologia *CUDA*. Com esta, podemos tirar proveito de melhoramento na velocidade de processamento ao utilizar a GPU em vez do CPU.

	SSD	YOLOv3	Enet	Mask R-CNN	Enet + Mask R-CNN
Média de frames/s	8.052	**8.393**	2.895	1.332	0.626
Tempo médio de deteção (s)	0.093	**0.040**	0.058	0.656	0.71
Tempo médio de processamento e desenho (s)	**0.001**	0.076	0.255	0.061	0.856
Tempo total (s)	**111.778**	149.742	310.866	675.638	1438.795

Tabela 2 – Resultados de testes a cinco modelos de classificação de objetos com tecnologia CUDA

Os melhores resultados obtidos em cada categoria estão realçados a negrito. É percetível que o modelo mais rápido é o SSD. O seu funcionamento é exposto em detalhe no próximo capítulo.

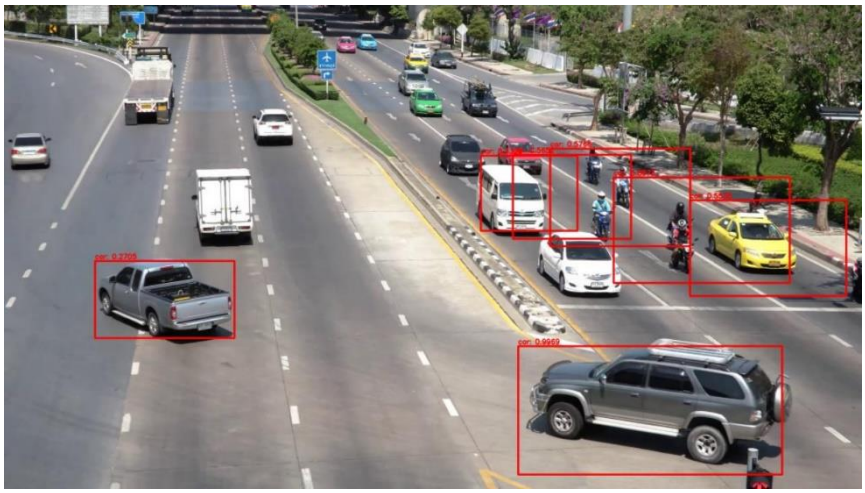


Figura 24 – Exemplo de deteção usando o modelo SSD (Cruz, 2020)

Por ser o melhor, o SSD foi selecionado para proceder a experiências incorporando a câmara RealSense D435. Foram feitos testes informais, de modo a perceber a utilidade da biblioteca CuDNN integrada no OpenCV, em Python, para o âmbito deste projeto. Com uma mão presente no enquadramento da câmara, o SSD apresentou uma *frame rate* média de 6 *fps*. Em primeira análise, este pode não parecer um valor demasiadamente baixo; porém, torna visível e comum o fenómeno desfoque de movimento (*motion blur*).

Figura 25 – Efeito de desfoque de movimento (Freewell Gear, 2019)

Este efeito impossibilita a implementação desta *framework* em protótipos desta dissertação, devido ao facto deste desfoque interromper o processo de deteção das mãos. A contínua inferência do esqueleto da mão fica, assim, comprometida, não sendo possível garantir uma boa utilização do sistema a desenvolver. Ocasionalmente, a área desfocada é inserida na imagem processada pelo modelo, resultando numa imagem que o modelo não está treinado para reconhecer. Outra eventualidade é a área desfocada ser pequena o suficiente para o modelo encontrar, com êxito, um resultado para a *frame* processada. Contudo, devido ao ruído presente na imagem, o resultado e esqueleto obtidos não serão, geralmente, satisfatórios.

2.4. MediaPipe

A *framework* MediaPipe destaca-se das restantes por funcionar excecionalmente bem em tempo real em simples dispositivos como telemóveis ou navegadores de internet. A MediaPipe inclui numerosas soluções que permitem a deteção e rastreamento de mãos, íris de olhos, faces e objetos com demarcação no espaço. Estes últimos podem ser detetados em 2D ou 3D (“MediaPipe: Object Detection,” 2021; “MediaPipe: Objectron (3D Object Detection),” 2021). Ao providenciarem estas bibliotecas livremente, os autores esperam

despertar a criatividade da comunidade para novas aplicações, tais como o reconhecimento de linguagem gestual (Bazarevsky and Zhang, 2019). Um exemplo algo inesperado desta inventividade é o produto da Alfred Camera: uma aplicação, para os sistemas operativos *Android* e *iOS*, que torna telemóveis comuns em câmaras de segurança. A Alfred Camera utiliza a MediaPipe para classificação de objetos em movimento. Após a montagem e configuração do sistema, o utilizador é notificado quando a aplicação identifica um objeto em movimento ("Alfred Camera," 2020).

A única solução de relevância para este projeto é a *Hands*, responsável por reconhecer mãos em imagens e, para cada uma, inferir 21 pontos de referência com coordenadas 2.5D (coordenadas *x*, *y* e *profundidade relativa* ao pulso) (Zhang et al., 2020), possibilitando a visualização dos seus respetivos esqueletos. Assim, serão apenas analisados em detalhe os mecanismos intrínsecos a esta.

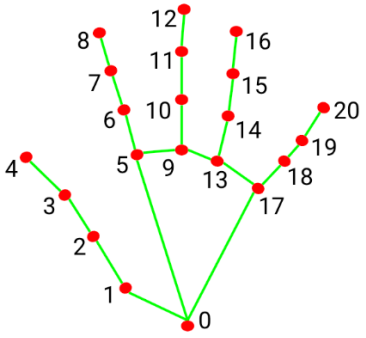


Figura 26 - Os 21 pontos de referência da mão ("MediaPipe Hands," 2021)

Zhang et al. (2020) referem fazer uso do modelo de deteção de objetos *Single Shot MultiBox Detector* (SSD) idêntico ao utilizado na *BlazeFace*, *framework* inserida noutras soluções apresentadas pela MediaPipe. Bazarevsky et al. (2019) reportam o desenvolvimento de um modelo mais leve adaptado do SSD, capaz de correr a uma velocidade entre 200 e 1000 *fps* nos dispositivos móveis mais relevantes (*flagship devices*). Uma abordagem tradicional do SSD usa uma rede de duas partes. A rede base é, habitualmente, uma rede pré-treinada de classificação de imagens usada apenas como extratora de características, sendo-lhe retirada a sua camada final de classificação de objetos. São criados mapas de características com variadas resoluções, dependendo da rede base escolhida, a partir dos resultados previamente obtidos. Posteriormente, o

modelo é complementado adicionando uma estrutura auxiliar de extração de características, que itera sobre a imagem transmutada para resoluções cada vez menores, nos mapas produzidos anteriormente. São aplicadas *bounding boxes* (áreas de deteção retangulares delimitativas) de variados tamanhos e proporções em cada resolução, permitindo a deteção tanto de objetos de grande escala (imagem com resolução reduzida) como objetos diminutos (imagem com maior resolução).

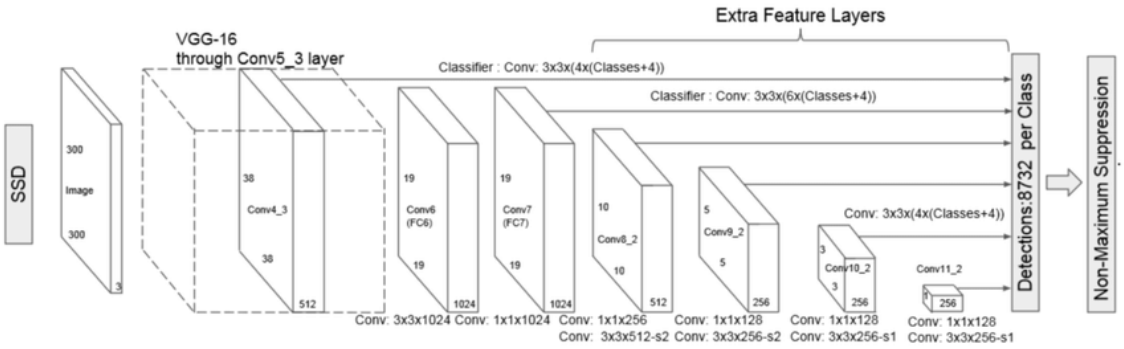


Figura 27 - Arquitetura do modelo SSD (Liu et al., 2016a)

Por fim, é aplicado o algoritmo *non-maximum supression* (NMS) às *bounding boxes* resultantes da combinação de ambas as partes do SSD. Para explicar como funciona o NMS, temos primeiro que perceber a métrica IoU (*Intersection over Union*).

Um objeto numa imagem terá uma *bounding box* verdadeira (*ground-truth*) correspondente. Um modelo de deteção de objetos (i.e. SSD) produz *bounding boxes* estimadas do objeto que deteta, cada uma com uma pontuação associada; pontuações mais altas incorrem na maior certeza do resultado ser correto. Entende-se por IoU o quociente entre a área de interseção da *bounding box* verdadeira com a estimada, e a área de união da *bounding box* verdadeira com a estimada.

Figura 28 - Visualização da métrica IoU (Rosebrock, 2016)

O NMS funciona escolhendo a estimativa com melhor pontuação e descartando as *bounding boxes* de estimativas piores do mesmo objeto. Ao encontrar a *bounding box* com pontuação mais alta, esta é selecionada como a correta e será a apresentada pelo algoritmo no fim do processo. De seguida, são processadas as *bounding boxes* do mesmo objeto, analisadas as suas IoU, comparando com a *bounding box* já selecionada como correta, e descartadas as que apresentam um valor elevado. Compreende-se que áreas com IoU muito altas pertencem à mesma instância de um objeto na imagem.

Figura 29 - Visualização do NMS em funcionamento. Estimativas mais claras apresentam maior pontuação e são selecionadas, estimativas mais escuras apresentam alta IoU e são descartadas (Ng, n.d.)

A implementação proposta em (Bazarevsky et al., 2019), adaptada do SSD, passa por substituir a utilização de 2 *anchor boxes* (*bounding boxes* de diversos tamanhos e proporções pré-definidas) por *pixel* nas resoluções 8x8, 4x4 e 2x2, pela utilização de 6 *anchor boxes* por *pixel* na resolução 8x8.

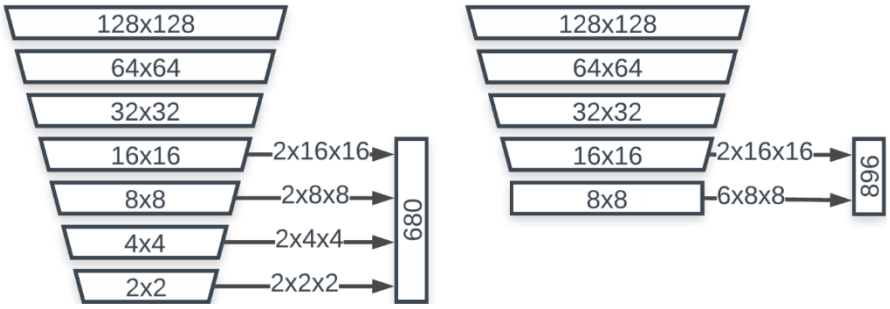


Figura 30 - Comparação de "anchor boxes" entre o SSD (esquerda) e o BlazeFace (direita) (Bazarevsky et al., 2019)

Adicionalmente, quanto ao NMS, o método de seleção é modificado para retornar uma *bounding box* gerada a partir da média ponderada das *bounding boxes* de um objeto,

ao invés de optar simplesmente pela que apresente maior pontuação. Esta alteração deve-se ao facto do NMS original, quando aplicado a vídeos, tender a flutuar entre diferentes *bounding boxes*, causando instabilidade entre *frames* (denominada *jitter*), e provocando fenómenos de ruído visual. Os autores realçam o facto desta alteração não pressupor um aumento no custo de computação do algoritmo.

A solução *Hands*, presente na *framework* da MediaPipe, utiliza um modelo baseado no *BlazeFace*, designado *BlazePalm*. Detetar mãos é, no entanto, um problema mais complexo, no âmbito de *computer vision*, que detetar faces. Mãos carecem de padrões de alto contraste (e.g. áreas dos olhos e boca), podendo ainda ocluírem-se a si próprias (e.g. punho cerrado) ou outras (e.g. apertos de mão). Assim, foi treinado um detetor, que processa a imagem na sua totalidade, destinado a encontrar, não mãos na sua integralidade, mas apenas as suas palmas. Os autores reforçam esta decisão ao salientar que palmas podem ser modeladas usando apenas *bounding boxes* quadrangulares, ignorando outras proporções, o que reduz o número de *anchor boxes* necessárias por um fator de 3 a 5 vezes.

De acordo com (Bazarevsky et al., 2019), inferimos que o *BlazePalm* retorna uma área expandida ao plano em redor da palma, que incluirá os dedos articulados. Após a localização da palma na imagem, é executado um modelo para deteção ulterior dos 21 pontos de referência supramencionados. O principal benefício desta abordagem é a possibilidade de reutilizar esta área nas *frames* subsequentes para rastreamento da mão, sem a necessidade de executar incessantemente o modelo sobre a imagem original. Apenas aquando da falha de deteção da mão nesta área, é corrido outra vez o *BlazePalm* que, novamente, localizará a palma da mão na *frame*.

Estes modelos são treinados com técnicas de *machine learning* e milhares de imagens. São usados *datasets* que incluem imagens com deteções e resultados verdadeiros, e imagens que ajudam o modelo a melhor lidar com eventuais falsos positivos.

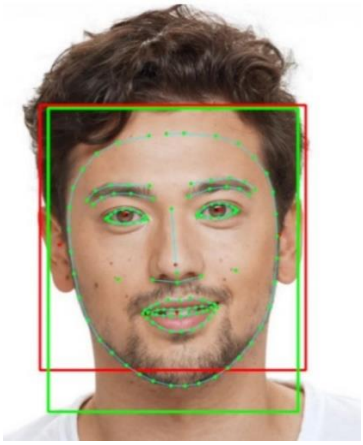


Figura 31 - No BlazeFace, é detetada a "bounding box" da face (a vermelho), e retornada uma área maior (a verde) (Bazarevsky et al., 2019)

Por fim, os autores enunciam os *datasets* usados no treino de cada modelo (Zhang et al., 2020). O modelo destinado à inferência dos pontos de referência da mão foi treinado com recurso a três datasets:

- *Dataset in-the-wild* composto por 6.000 imagens de grande diversidade nas condições de iluminação e aparência das mãos, contudo com reduzida complexidade de poses. *Dataset* demarcado manualmente com 21 pontos de referência.
- *Dataset in-the-house* composto por 10.000 imagens de "todos os gestos fisicamente possíveis" (Zhang et al., 2020) apesar de apresentar pouca variedade de planos de fundo. *Dataset* demarcado manualmente com 21 pontos de referência.
- *Dataset* sintético para o qual foi criado um modelo de alta-fidelidade de uma mão a três dimensões. Foram depois criados vídeos da mão com sequências de transição entre poses e retiradas 100.000 *frames* para formar o *dataset*. Cada pose é capturada com iluminação realista e de três ângulos diferentes. *Dataset* demarcado com 21 pontos de referência projetados pelo programa.

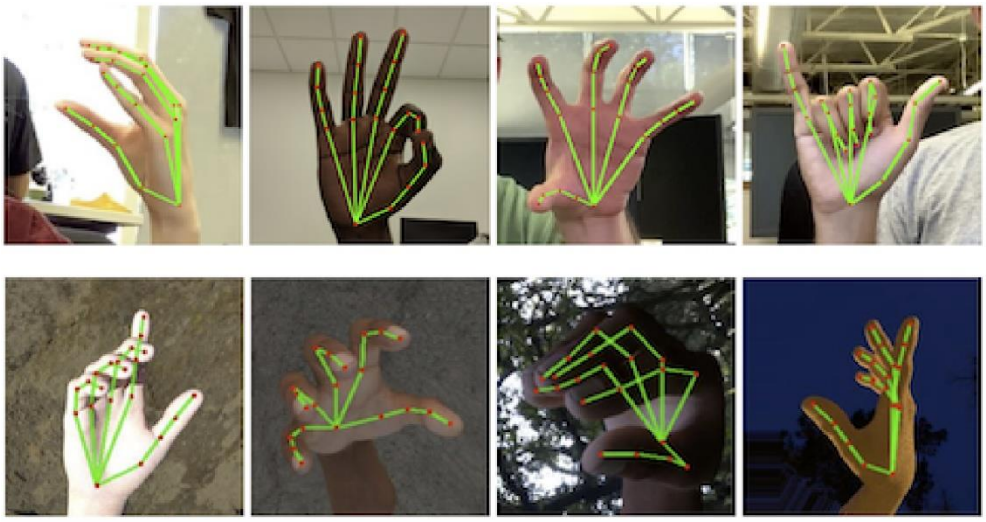


Figura 32 - Exemplos dos datasets utilizados: em cima, imagens reais demarcadas manualmente; em baixo, imagens sintéticas demarcadas por projeções (Zhang et al., 2020)

Os autores realizaram testes com diversas combinações destes *datasets*. Concluiu-se que a junção dos *datasets* sintéticos com os de fotografias reais proporciona os melhores resultados. Além do mais, ao utilizar um *dataset* sintético tão extenso, foi possível registar uma diminuição do *jitter* observado entre *frames*. A tabela seguinte mostra a variação do Erro Quadrático Médio (EQM) conforme os diferentes *datasets* utilizados para treino.

Dataset	**EQM normalizado pelo tamanho da palma da mão**
Apenas fotografias reais	16.1%
Apenas sintético	25.7%
Ambos	13.4%

Tabela 1- Comparação de treino do modelo com os diferentes datasets (Zhang et al., 2020)

Por sua vez, o *BlazePalm* (o modelo encarregue de detetar as palmas das mãos na figura), foi treinado apenas com o *dataset in-the-wild*. Os autores referem ser suficiente usar apenas este *dataset*, visto oferecer uma grande variedade de proporções e aspetos.

Estas inovações provaram ser consideráveis, pois permitem a implementação, neste projeto, de um modelo de deteção mais robusto e completo do que o estudado no ponto 2.2, que, no entanto, evita as principais limitações do modelo no ponto 2.3.

A *framework* desenvolvida pela MediaPipe está disponível publicamente com módulos prontos a usar. O seu funcionamento é extremamente simples e serve como excelente fundação para suporte de desenvolvimento de projetos como o proposto nesta dissertação. No entanto, o modelo *BlazePalm* foi desenvolvido para iterar sobre imagens RGB, pelo que não será possível explorar as funcionalidades de profundidade oferecidas pela RealSense D435. Contudo, os resultados obtidos aparentam ter um elevado grau de fiabilidade com um tempo de processamento desejado para tratamento de vídeo em tempo real. Não sendo necessário a informação de profundidade das *frames* obtidas, pode-se escolher implementar um outro modo de receber as *frames* RGB da câmara, substituindo o SDK da Intel por um outro modo de captura que permita a conexão a qualquer câmara de vídeo comum. Esta abordagem cumpre os requisitos iniciais impostos para integração com o projeto a desenvolver, pelo que será incorporada e estudada em maior detalhe no próximo capítulo.

3. Protótipos

Para alcançar os objetivos mencionados na secção 1.1, foi desenvolvido um programa que permite ao utilizador controlar a interface gráfica de uma aplicação de *streaming* de vídeo. De forma a testar a utilidade desta abordagem num contexto prático, foi criada uma aplicação de raiz, ao invés de tentar integrar com uma existente, para permitir o controlo dos botões na interface através do *input* da câmara. A interface assemelha-se à plataforma *Netflix* para permitir usar várias secções e janelas com modos de interação diferentes. Com esta, podemos fazer *scroll* na vertical para ver mais categorias, *scroll* na horizontal para ver mais filmes pertencentes a uma categoria, selecionar um filme para visualizar, e implementar todos os controlos básicos associados à visualização de vídeos (pausar, resumir, parar, aumentar e diminuir volume).

De notar que, ainda que tenham sido desenvolvidos dois protótipos distintos, a interface criada é comum a ambos.

Foram construídos dois sistemas distintos, com base nos trabalhos analisados, para processamento do *feed* recebido da câmara e controlo do cursor no sistema. O primeiro, apoia-se apenas no uso de métodos de processamento de imagem, com posterior implementação de um modelo de classificação de gestos. Neste protótipo inicial, foi feito uso do modo de profundidade da câmara RealSense utilizada. O segundo protótipo desenvolvido apenas utiliza o fluxo RGB da RealSense, podendo então ser reproduzido com qualquer câmara de vídeo comum. O procedimento adotado neste sistema faz uso de tecnologias de classificação de objetos e rastreamento do esqueleto da mão entre *frames*.

Neste capítulo são descritos, de forma mais elaborada, os métodos empregados na implementação das diferentes abordagens.

3.1. Protótipo OpenCV

Primeiramente, serão abordadas as estratégias postas em prática no protótipo desenvolvido apenas com recurso a métodos de tratamento de imagem, bem como as implementações comuns a ambos os protótipos. Este protótipo faz uso de alguns métodos expostos na secção 2.2.

3.1.1. Bibliotecas externas utilizadas

Nesta subsecção serão expostas as bibliotecas e *frameworks* externas utilizadas neste primeiro protótipo.

3.1.1.1. Front-end

Para gerar a interface foi selecionada a biblioteca *PyQt5* (versão 5.15.1), que é um conjunto de *bindings* da *framework Qt*, originalmente desenvolvida em C++, adaptada para Python. Foi escolhida esta biblioteca pela facilidade de criação de interfaces devido aos números módulos prontos a usar que oferece. O *PyQt5* possui ainda módulos e funcionalidades para reprodução de ficheiros de vídeo, o que permitiu criar uma *GUI* consistente sem ser necessário integrar bibliotecas externas complementares.

Adicionalmente, fez-se uso do *OpenCV* para mostrar as janelas com as diferentes máscaras aplicadas às *frames* da câmara. Todavia, esta biblioteca foi utilizada quase exclusivamente na parte de processamento de imagens no *back-end* e é abordada na secção seguinte.

3.1.1.2. Back-end

Foi utilizada a biblioteca pyrealsense2 (versão 2.36.0.2038) para aceder ao *feed* de infravermelhos da câmara Intel RealSense D435 e captação das imagens em tempo real. O *pyrealsense2* é o *wrapper* oficial, com as *bindings* para Python necessárias, que permite aceder à Intel RealSense SDK 2.0, desenvolvida em C++. A vantagem de conseguir aceder às funcionalidades disponibilizadas pela *SDK* são descritas mais à frente.

A biblioteca usada mais extensamente na parte de processamento das imagens captadas foi o OpenCV (*Open Source Computer Vision Library*). Esta é uma biblioteca *open source* focada no desenvolvimento de aplicações na área de *computer vision* e *machine learning*. O OpenCV conta com mais de 2500 algoritmos otimizados que permitem, entre outros, identificar e reconhecer rostos, objetos e gestos/ações em vídeos ou imagens. Está disponível para C++, Java, Python e MATLAB ("OpenCV: About," 2021).

De forma a proceder ao processamento de imagens com o OpenCV é necessário usar ainda a biblioteca *numpy*. Os dados de uma imagem, no computador, são guardados em matrizes de dados ilegíveis para o ser humano. O *numpy* é a biblioteca universal usada para operações com este tipo de dados, visto proporcionar um grande aumento em eficiência e escalabilidade quando comparada com as listas padrão da linguagem Python. O *numpy* oferece ainda imensas operações prontas a usar sobre vetores e matrizes, facilitando a manipulação dos dados da imagem a processar ("What is NumPy?," 2021).

Ao longo da evolução deste protótipo, foram testados vários métodos para a captação do *feed* da câmara RealSense D435. O método que se provou mais eficaz para esta aplicação foi utilizar o modo de infravermelhos da câmara para recolha de *frames*, ao contrário do modo comum RGB. Para isto, é necessário recorrer à *pyrealsense2* de forma a poder receber as imagens de profundidade. Ao captar imagens em tempo real com o fluxo infravermelho, conseguimos uma menor sensibilidade a alterações na iluminação, algo que afeta, consideravelmente, os resultados obtidos quando trabalhamos com o modo RGB. Por exemplo, utilizando este último fluxo, a sombra de uma mão projetada numa parede no plano de fundo pode ser, erradamente, reconhecida pelo algoritmo, por existir um

contraste entre a cor dos pixéis da parede e os da sombra. Em contrapartida, o fluxo de infravermelhos facilmente ignora estas pequenas discrepâncias.

Para além desta vantagem, ao utilizar a lente de infravermelhos temos acesso a diversas funcionalidades proveitosas, tal como a opção de apenas aceitar a informação que se encontre entre certas distâncias. Assim, ao ter um ambiente de trabalho fixo, podemos predefinir um limite até o qual queremos captar, ignorando os restantes pixéis que provêm de superfícies para lá dessa distância limite.

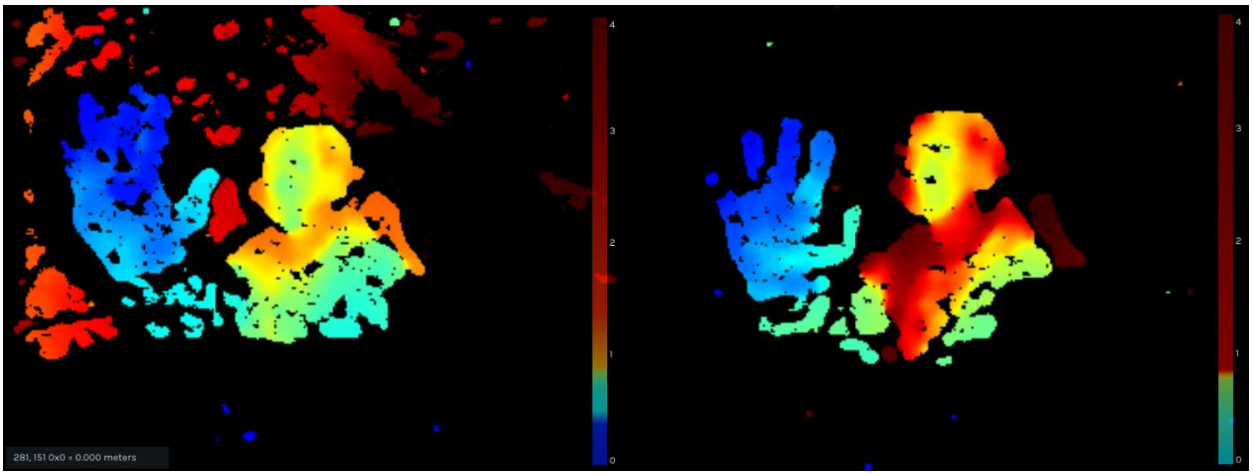


Figura 33 – Imagens de profundidade: à esquerda, sem limite de distância; à direita, com um limite máximo de 90cm

Para serem reconhecidos os gestos a capturar, é utilizado um modelo pré-treinado que reconhece certas poses com a mão, disponível em (Chin, 2021). Este modelo faz uso das imagens criadas pelas máscaras *threshold* obtidas. As poses reconhecidas pelo modelo identificado são: *five* (mão aberta), *swing* (gesto a imitar um telefone), *fist* (punho cerrado) e *point* (gesto de apontar com o dedo indicador).

Figura 34 – Máscaras threshold com os gestos reconhecidos, pela ordem supramencionada

Este tipo de modelos é treinado com centenas, se não milhares, de fotografias de mãos em diversas poses, distâncias da câmara, localização na imagem, entre outras variáveis. Normalmente encontram-se a preto e branco para diminuir a variação que a cor da pele e iluminação podem introduzir. Infelizmente, não é possível encontrar muitos modelos disponíveis publicamente e prontos a integrar. O modelo pré-treinado usado na aplicação oferece boa fidelidade e exatidão nos resultados e torna possível a fácil identificação do gesto realizado. Para ter acesso aos dados do modelo e prever a resposta ao *input*, são necessárias as bibliotecas *Keras* (versão 2.1.3) e *Tensorflow* (versão 1.13.1). Pelo modelo usado datar de janeiro de 2018, foi preciso instalar uma versão mais antiga do *Tensorflow* que ainda suportasse certas funções que foram, desde então, descontinuadas.

Para fazer a ligação entre a aplicação e o cursor no ecrã usa-se a *PyAutoGUI* (versão 0.9.50). Esta biblioteca é um módulo multiplataforma para Python que permite, programaticamente, controlar o rato e teclado.

Por último, para efeitos de diversidade e autenticidade da aplicação, é reproduzido o *trailer* do filme da *thumbnail* clicada. Neste âmbito, foram usadas as bibliotecas *Pytube* (versão 10.8.2) e *python-youtube* (versão 0.8.0). A *python-youtube* é um *wrapper* para Python da *YouTube API DATA v3*. Esta biblioteca efetua uma pesquisa pelo trailer do filme selecionado e retorna o URL do resultado encontrado. Seguidamente, com conhecimento do URL pretendido, tratamos de transferir o vídeo para visualização, com recurso ao *Pytube*.

3.1.2. Desenvolvimento

O protótipo foi criado e evoluindo de forma modular, o que tornou simples todo o processo de adição de novas funcionalidades e métodos. O diagrama seguinte representa a arquitetura do sistema com os módulos principais da aplicação. São eles:

- **Módulo câmara –** controla e interpreta o *input* proveniente do *feed* da câmara quanto à sua posição no espaço. Demarca as áreas da *frame* correspondentes

a cada mão e passa como argumento, na chamada da função principal, ao módulo da mão correspondente.

- **Módulo *RightHand* –** extrai a mão direita da imagem original e controla o movimento do cursor, com uma velocidade proporcional ao número de dedos erguidos.
- **Módulo *LeftHand* –** extrai a mão esquerda da imagem original e delega a análise e comparação para o módulo de reconhecimento gestual. Após retorno de qual o gesto produzido, efetua a ação associada na interface em primeiro plano.
- **Módulo de identificação –** analisa a imagem recebida e compara com o *dataset* de referência para encontrar o melhor resultado, usando o modelo em (Chin, 2021). Retorna pontuações associadas a cada tipo de gesto ao módulo LeftHand.
- **Módulo de *download* –** Despoleta as ações de pesquisa do vídeo no YouTube (com a biblioteca *python-youtube*) e a sua descarga (com a biblioteca *Pytube*) para o diretório do projeto. Após o encerramento do leitor de vídeo, remove o ficheiro do sistema.
- **GUI –** desenha a interface e recebe *inputs* vindos dos módulos das mãos, convertendo-os em ações com os elementos da interface (e.g. botões).

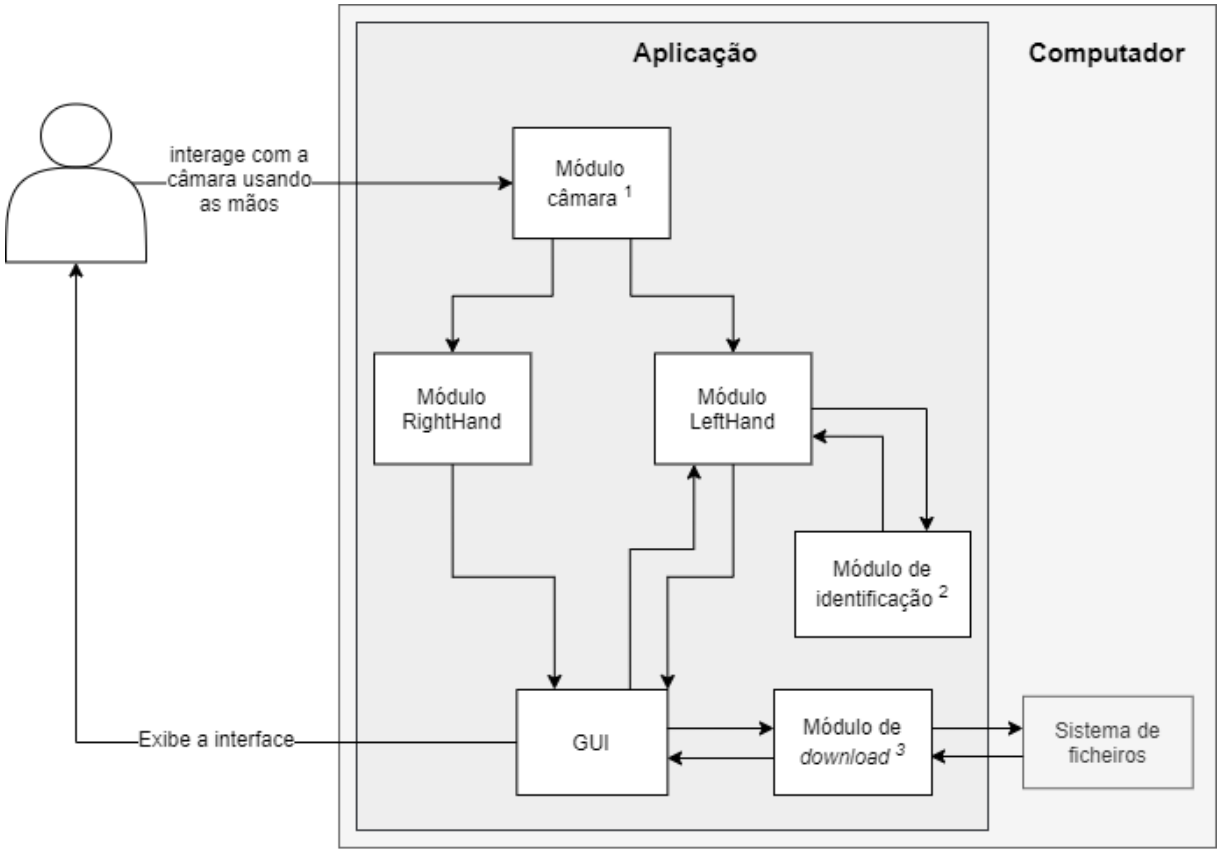


Figura 35 – Arquitetura da aplicação com o protótipo OpenCV

O protótipo final foi desenvolvido gradualmente, evoluindo a partir de módulos individuais, testando as diversas técnicas presentes na versão final, bem como outras ultimamente descartadas.

O módulo câmara foi, naturalmente, a origem do projeto. Neste, é inicializada a *pipeline* da câmara, com a configuração pré-definida pela *SDK* das câmaras RealSense indicada para o modo de profundidade. Foi escolhida uma combinação da resolução *640x480* com uma *framerate* de 30 *fps* para obter os melhores resultados possíveis. Uma resolução mais alta implica uma redução na *framerate*, o que verificou ser algo desfavorável devido à necessidade de rastreio da mão em tempo real. Inversamente, diminuir a resolução para além da estabelecida, mesmo que signifique um aumento de 30 para 60 *fps*,

torna a imagem insuficientemente nítida para uma segmentação satisfatória entre as máscaras da mão e plano de fundo.

Com a *SDK* da RealSense, são capturadas as imagens de profundidade que serão depois interpretadas e convertidas para imagens de cor. O esquema de cor escolhido foi o *White To Black*, em que os objetos mais próximos da câmara são representados a branco e os mais distantes a preto.

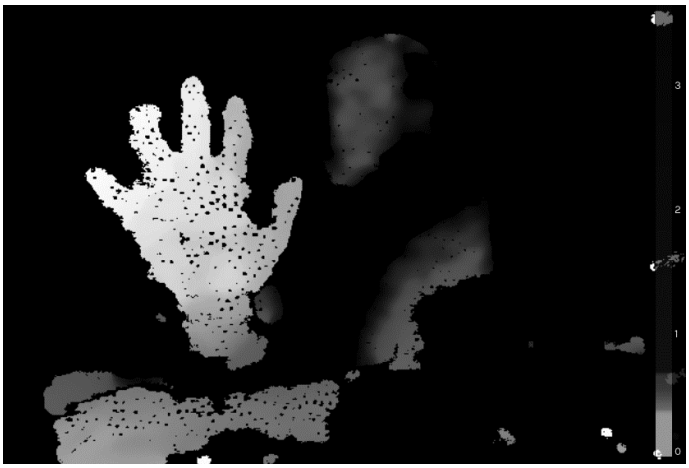


Figura 36 – Visualização da frame de profundidade com o esquema "White To Black"

Foi usado este esquema sobre os outros visto que o início do processo de tratamento de dados com o OpenCV envolve uma transformação de imagens RGB para preto e branco. Ao utilizar o esquema *White To Black* de raiz, reduzimos potenciais complicações na transformação da imagem pelo OpenCV este esquema de cor. Após convertida a *frame* de profundidade para uma imagem RGB, é passada a ambas as classes *RightHand* e *LeftHand*, para interpretação das poses ou movimentos da mão correspondente.

Ambos os módulos *RightHand* e *LeftHand* necessitam de recortar da *frame* a região em que se encontrará a mão. Será nestas áreas que é aplicada a técnica de subtração do plano de fundo e segmentação do primeiro plano. Desta abordagem resulta uma máscara contendo apenas a mão presente na região recortada. Esta delimitação de uma zona pré-definida para cada mão é essencial, pois o programa não consegue distinguir a informação pretendida (a mão) de informação extra a ignorar (por exemplo um braço ou ombro). Ao

delimitar uma região da imagem para cada mão consegue-se ter a certeza de que apenas extraímos a informação pretendida.

O módulo *RightHand* começa por transmudar a imagem para preto e branco e aplicar variados métodos (e.g. erosão, dilatação, *opening*, *closing*) pré-configurados com o propósito de diminuir o ruído presente.

Posteriormente são retirados os contornos da mão e contados os dedos erguidos para controlar a velocidade de movimento do cursor no ecrã. Num extremo, um dedo erguido desloca o cursor mais lentamente (mais exato), enquanto que cinco dedos erguidos deslocam o cursor mais rapidamente (menos exato).

O módulo *LeftHand* é o mais complexo do projeto, pois necessita dos mesmos tratamentos base do módulo supramencionado, bem como processos mais elaborados para a comparação com o modelo pré-treinado. Começamos por aplicar os mesmos procedimentos introdutórios do módulo *RightHand* até à extração da máscara da mão. A imagem a comparar é então redimensionada para corresponder às dimensões das fotografias de referência usadas para treinar originalmente o modelo.

De seguida, é prevista uma resposta fazendo uso dos métodos implementados, recorrendo a processos específicos a redes neuronais. É recebida uma lista com uma pontuação associada a cada uma das quatro possibilidades (*five*, *swing*, *fist*, *point*) e escolhida como resposta correta a que apresenta uma pontuação mais alta. Cada gesto despoleta uma ação específica consoante a interface com que o utilizador se encontra a interagir no momento.

A totalidade dos elementos visuais dispostos no ecrã são gerados no módulo interface. Esta classe é iniciada na abertura do programa e cria, dinamicamente, uma *GUI* similar à do serviço de *streaming* Netflix. Os elementos de organização estrutural e elementos interativos são guardados num dicionário – de chave "nome do elemento" e valor "referência do elemento". Esta foi a abordagem mais simplista encontrada para facilitar a criação da interface com um bloco de código em ciclo, reduzindo assim o código a manter e evitando discrepâncias entre padrões de nomenclatura no projeto.

A aplicação trata de adicionar automaticamente qualquer pasta presente no diretório do projeto como uma nova categoria de vídeos, para além de adicionar os ficheiros de imagem existentes nessa mesma pasta como filmes pertencentes a essa categoria. Ao clicar numa *thumbnail*, é aberto o *trailer* do filme selecionado.

Figura 37 – Pastas com as thumbnails e interface que as adiciona dinamicamente

Em termos técnicos, após o clique na *thumbnail* de um filme, é feita uma pesquisa na página YouTube com o nome do mesmo, seguido da palavra "*trailer*", e descarregado o primeiro resultado para reprodução. Uma nova janela destinada à visualização do vídeo é exibida. No encerramento do reprodutor, o ficheiro temporário descarregado é eliminado do disco.

3.2. Protótipo MediaPipe

Neste protótipo foi utilizada a biblioteca *MediaPipe* (versão 0.8.3.1). Esta *framework* é extremamente leve e rápida, e procede ao reconhecimento do esqueleto da mão na imagem, bem como o rastreamento da mesma nas *frames* seguintes.

3.2.1. Bibliotecas externas utilizadas

Esta subsecção tratará de expor as bibliotecas e *frameworks* externas utilizadas neste segundo protótipo.

3.2.1.1. Front-end

Foi utilizada mais uma vez a biblioteca *PyQt5* com o mesmo propósito de no protótipo supramencionado.

Foi também usado um método da biblioteca *MediaPipe* que permite o desenho das ligações entre os pontos chave identificados na(s) mão(s).

3.2.1.2. Back-end

Novamente, foi utilizada a biblioteca *pyrealsense2* para captação das imagens da câmara D435 em tempo real, com a distinção de ter sido usado o *feed* RGB no presente protótipo (em contraste com a utilização do *feed* de infravermelhos anteriormente).

A *framework* usada mais extensamente na parte de processamento das imagens captadas foi a *MediaPipe*. Esta é uma biblioteca *open source* focada no desenvolvimento de soluções de alta-fidelidade de deteção de objetos, com soluções disponíveis otimizados para deteção de corpos, mãos, olhos, faces, entre outros, fazendo usos de técnicas de *machine learning*. O ponto de destaque do produto disponibilizado pela *MediaPipe*, comparando com outros semelhantes, é o facto de ser concebido para atingir resultados em tempo real em telemóveis, não dependendo de *hardware* presente apenas em computadores potentes e dispendiosos. A solução empregada neste projeto é somente a *Hands*, que é escalável a várias mãos.

A imagem de cor, produzida pelo *feed* da câmara com recurso à *pyrealsense2,* é do formato BGR (*Blue, Green Red*). Pela *MediaPipe* usar o formato RGB (*Red, Green, Blue*), as bibliotecas *numpy* e *OpenCV* são necessárias para transformar os dados que compõem a *frame* num *array* passível de conversão do padrão BGR para RGB.

Ao contrário do que acontece com o protótipo desenvolvido inicialmente, a *mediapipe* trabalha com imagens de cor e não com imagens a preto e branco, o que força a utilização do *feed* RGB nesta porção do projeto. Esta opção não seria ideal numa abordagem tradicional, porém, o modelo desenvolvido pela equipa da MediaPipe está treinado para reconhecimento de mãos com diferentes tons de pele e diferenciação dos planos em redor. A *framework* apresenta resultados de grande fidelidade com desempenho em tempo real, sem causar atrasos percetíveis na captura de imagens. É inferido um esqueleto satisfatório mesmo em condições de auto-oclusão de porções significativas da mão.

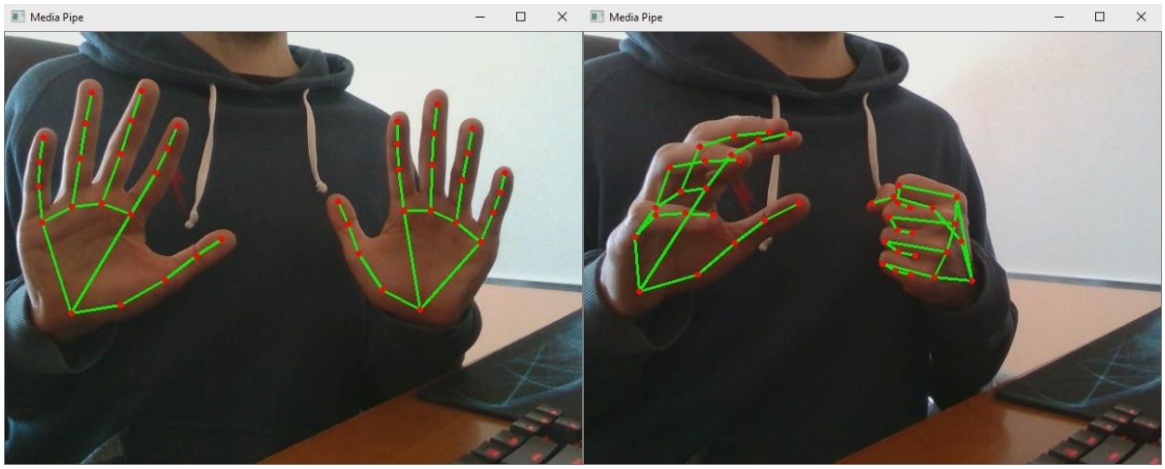


Figura 38 – Estimativas corretas dos esqueletos das mãos

Contudo, as suas limitações são evidentes principalmente quando uma mão oclui a outra, embora os autores classifiquem a solução como adequada para estas situações. Nestes casos, os esqueletos gerados revelam-se, por vezes, inutilizáveis.

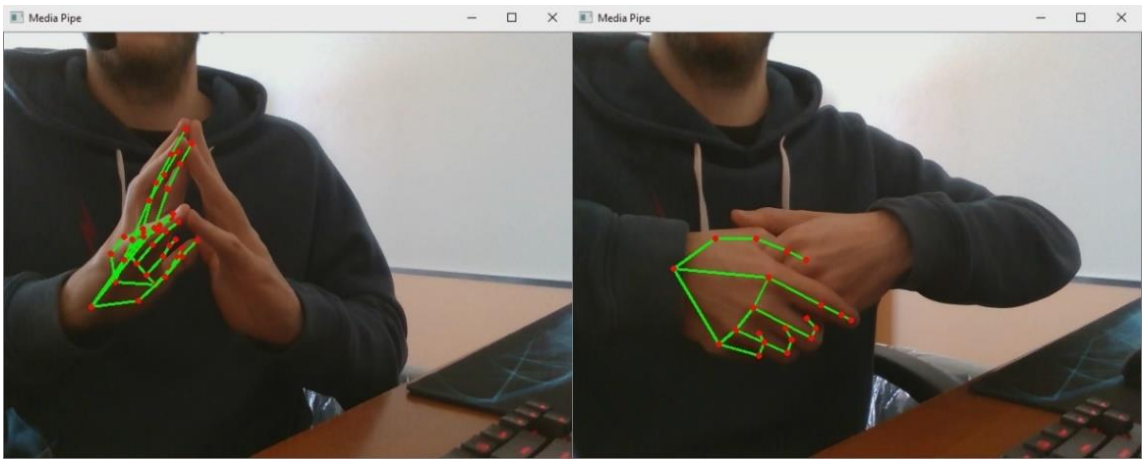


Figura 39 – Estimativas erradas dos esqueletos das mãos

O *BlazePalm* aparenta definir a etiqueta *handedness* (mão esquerda ou direita) pela localização do polegar em relação à palma/costas da mão. Ocasionalmente, a *MediaPipe*, dependendo da posição da mão quanto à câmara, define a etiqueta com a *handedness* errada. A figura seguinte mostra, nas duas primeiras capturas de ecrã, uma leitura correta da mão filmada, e, na última, uma leitura errónea.

Figura 40 – Capturas da mesma mão, com três poses diferentes

Nas instâncias anteriores, os resultados tendem a ser consistentes; isto é, mostrar à câmara a mão como nas duas primeiras imagens retorna sempre a etiqueta "*Right*", tal como a terceira retorna sempre "*Left*". No entanto, com a mão apontada para o solo, o modelo regista, por vezes, a *handedness* oposta.

Figura 41 – A mesma pose, com diferenças desprezáveis, apresenta resultados diferentes

Apesar dos vários momentos de respostas erráticas, o sistema montado apenas prevê uma utilização com as mãos voltadas para a câmara e na direção do monitor (Figura 38 e primeira captura na Figura 40). Com a instalação descrita os resultados são, geralmente, fiáveis e livres de problemas, não comprometendo a eficácia do *BlazePalm* no protótipo criado.

É também importante referir que existiram ocasiões em que o vestuário influenciou o desempenho da *MediaPipe* e foram identificadas mãos erradas, o que por vezes despoletou ações na interface.

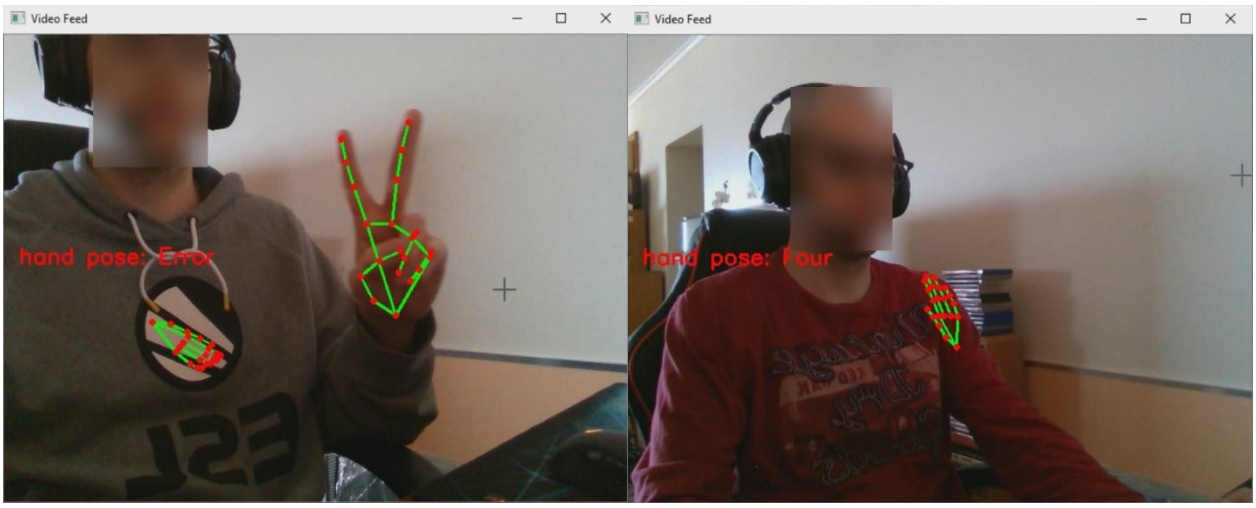
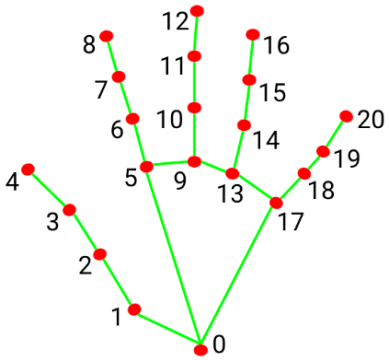


Figura 42 – Deteções erradas com diferentes vestuários

Uma vantagem notável na implementação da biblioteca *MediaPipe* está no reconhecimento de gestos efetuados. Por ter acesso a 21 pontos-chave que compõem a estrutura da mão, é possível criar combinações de dedos para representar ações, excluindo do projeto o uso de modelos pré-treinados. Por exemplo, ao saber que os dedos indicador e médio estão estendidos, e os restantes fechados, percebe-se que o utilizador está a gesticular o número dois. Do mesmo modo, se todos os dedos se encontrarem fechados, trata-se do gesto "punho cerrado". Um dedo considera-se fechado quando o ponto de referência da polpa (pontos acabando em -*TIP*) se encontra abaixo, nas coordenadas *y*, da primeira articulação desse mesmo dedo (pontos 2, 6, 10, 14 e 18). No dedo indicador, isto significa que o mesmo se encontra fechado caso o ponto 8 esteja, nas coordenadas *y*, abaixo do ponto 5.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figura 43 – Os 21 pontos-chave da mão com os seus nomes em código ("MediaPipe Hands," 2021)

A *PyAutoGUI* continua a ser necessária para realizar as realizar as mesmas operações, descritas na secção equivalente no protótipo anterior. Do mesmo modo, recorreu-se também às bibliotecas *Pytube* e *python-youtube*.

3.2.2. Desenvolvimento

O protótipo atual assenta na base do discutido anteriormente neste capítulo. Todavia, foram feitas alterações no código que alteraram ligeiramente as responsabilidades de cada módulo, bem como as relações entre si. Desta maneira, os módulos passam a agir da seguinte forma:

- **Módulo de câmara e deteção –** controla e interpreta o *input* proveniente do *feed* da câmara. Corre a solução *Hands* na *frame* e classifica as mãos encontradas quanto à sua *handedness*. Por último, delega a referência do objeto encontrado para o módulo da mão correspondente.
- **Módulo de interpretação –** responsável pelas funções que tratam de transformar os dados dos 21 pontos de referência da mão em informações que permitam saber: se certo dedo se encontra estendido ou dobrado, a coordenada central da mão, se dois dedos se estão a tocar, o gesto realizado pela mão, etc.
- **Módulo *RightHand* –** processa o objeto da mão direita e controla o movimento do cursor com uma velocidade proporcional ao número de dedos erguidos.
- **Módulo *LeftHand* –** processa o objeto da mão esquerda e chama o módulo de interpretação. Após retorno de qual o gesto produzido, efetua a ação associada na interface em primeiro plano.
- **Módulo de *download* –** Despoleta as ações de pesquisa do vídeo no YouTube (com a biblioteca *python-youtube*) e a sua descarga (com a biblioteca *Pytube*)

para o diretório do projeto. Após o encerramento do leitor de vídeo, remove o ficheiro do sistema.

- **GUI –** desenha a interface e recebe *inputs* vindos dos módulos das mãos, convertendo-os em ações com os elementos da interface (e.g. botões).

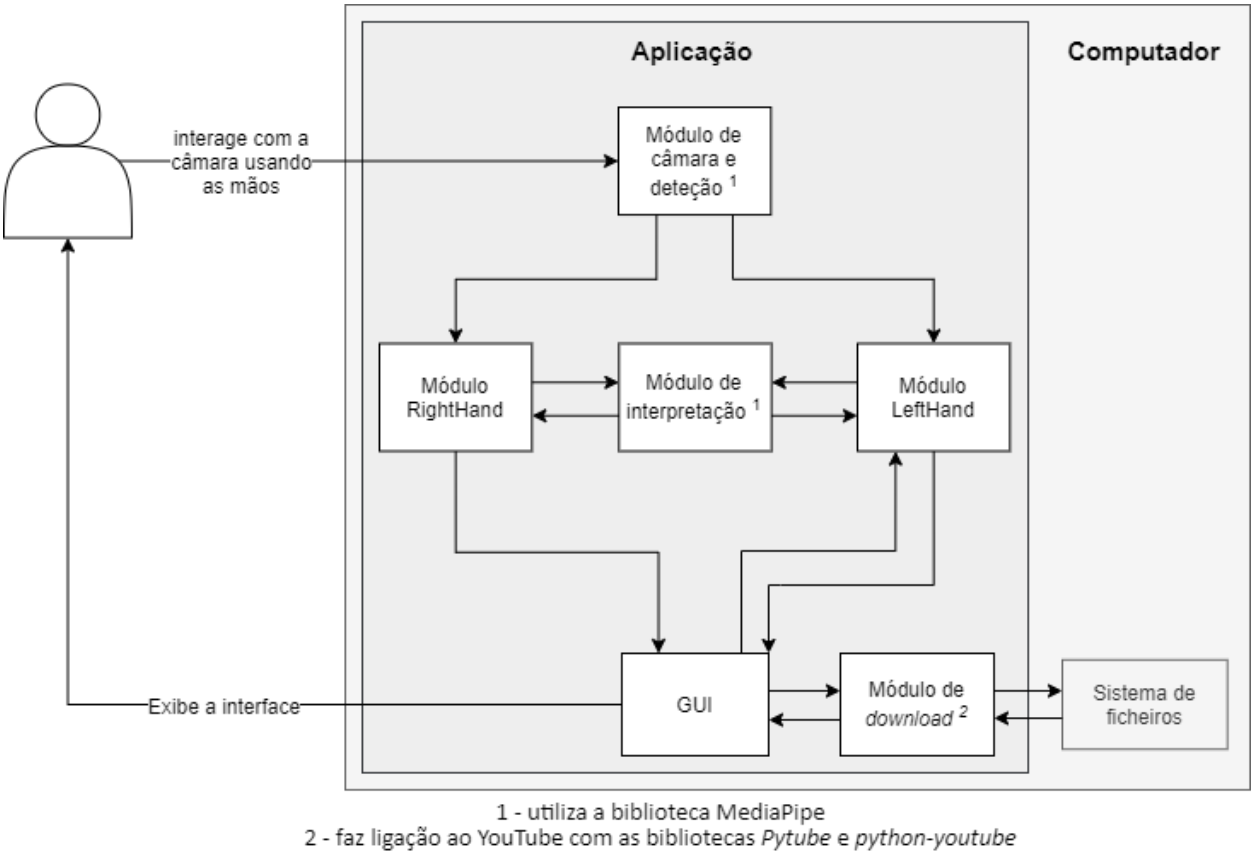


Figura 44 – Arquitetura da aplicação com o protótipo MediaPipe

Tal como o protótipo OpenCV, o novo protótipo foi desenvolvido e testado modularmente. Procedeu-se à adaptação, por partes, do código escrito anteriormente para OpenCV para código suportado pela MediaPipe. Foram ainda refatoradas funções comuns a ambos os módulos das mãos para um único ficheiro, reduzindo o código a manter.

Analogamente ao protótipo anterior, a aplicação origina no módulo câmara. É, desta vez, inicializada a *pipeline* da câmara com a configuração pré-definida pela *SDK* das câmaras ReaslSense indicada para o modo a cor. A escolha da resolução *640x480* a 30 *fps* continua a ser a mais adequada para obter os melhores resultados possíveis. Por ter que se analisar

palmas e dedos quanto à sua cor e aspetos, com contornos e limites bem definidos, torna-se imperativo reduzir ao máximo fenómenos como o desfoque de movimento (*motion blur*).

O *feed* da câmara é capturado com a *pyrealsense2*, em modo a cor, que será depois convertido do formato BGR8 para RGB8 com recurso ao *OpenCV* e *numpy*. Com a *frame* no formato correto, segue-se o seu processamento recorrendo à solução *Hands* disponibilizada. São analisadas as mãos encontradas na imagem e, conforme a sua etiqueta *Handedness*, é chamada a classe correspondente com o objeto detetado.

A mudança para a *framework* MediaPipe vem simplificar o código em várias áreas. Foi pensado um modo de controlo do cursor que permite tanto movimentos rápidos como exatos. São contados o número de dedos erguidos, recorrendo ao módulo de interpretação, e aplicado um multiplicador à velocidade de movimento do cursor dependente do número gesticulado pelo utilizador. Um dedo erguido permite deslocar o cursor mais lentamente (controlo mais exato), enquanto que cinco dedos erguidos permitem percorrer uma maior distância no ecrã com movimentos mais pequenos (controlo menos exato).

No protótipo OpenCV, o módulo *LeftHand* era o que apresentava maior complexidade devido à integração de modelos criados por terceiros e os seus processos envolventes para reconhecimento dos gestos efetuados pelo utilizador. Contudo, nesta abordagem, o módulo resume-se a receber o objeto com os pontos de referência da mão, perguntar ao módulo de interpretação qual o gesto realizado (calculado a partir de combinações de dedos estendidos e dobrados), e despoletar a ação na interface associada à resposta obtida.

O módulo de interpretação é composto por funções que recebem o objeto com os pontos-chave da mão, analisam as suas coordenadas, e traduzem os resultados para um sistema legível por humanos. Tais funções incluem: saber se dois dedos se estão a tocar, o número de dedos estendidos, o ponto central da mão e o gesto sendo realizado. A definição de cada pose gestual está relacionada com quais dedos estão abertos e quais estão fechados. Por exemplo, o gesto *three* implica que os dedos indicador, médio e anelar se encontrem abertos, e os dedos mínimo e polegar fechados. Outro exemplo é o *rock*, composto pelos dedos indicador e mínimo abertos e os restantes fechados. A lista de gestos possíveis a detetar inclui: *one*, *two*, *three*, *four*, *five*, *fist*, *okay*, *spiderman*, *rock* e *telephone*.

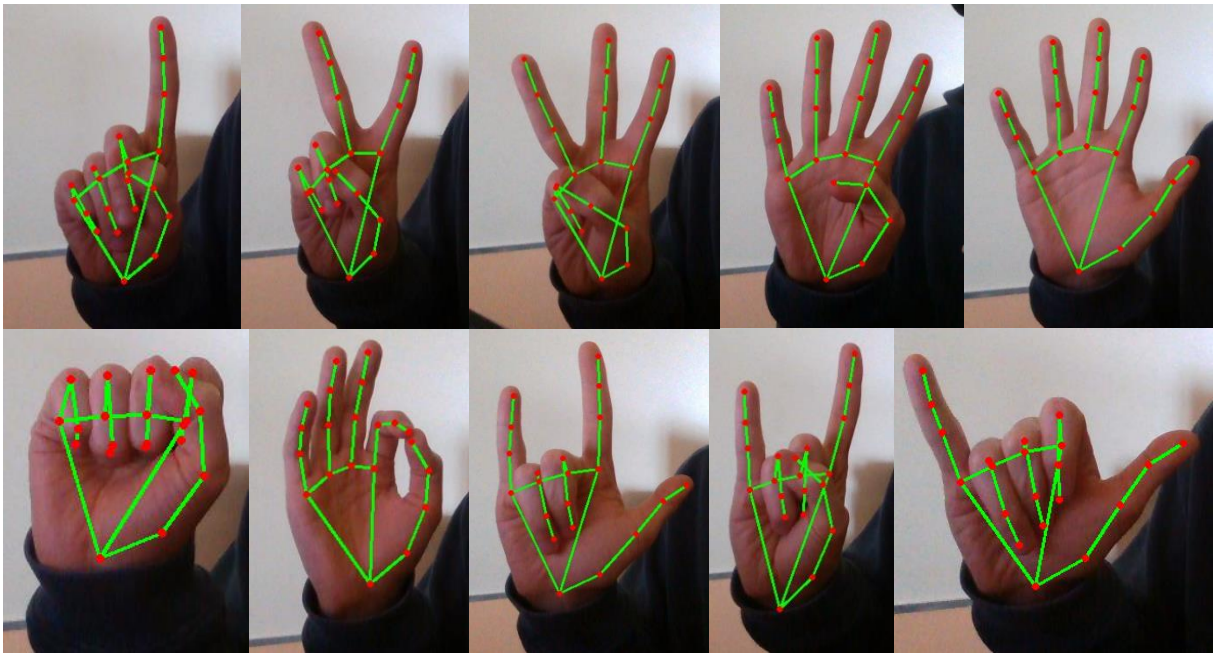


Figura 45 – Gestos reconhecidos pelo programa, pela ordem supramencionada

A execução da interface e os seus elementos mantém-se inalterada, gerada a partir do módulo interface. O módulo é inicializado na abertura do programa e criado dinamicamente. Toda a abordagem inerente a esta classe é igual à descrita no capítulo precedente.

O processo responsável pela construção das categorias de vídeos através da adição de pastas no diretório do projeto é também ele idêntico, bem como o procedimento para descarregar e reproduzir vídeos da página YouTube.

3.3. Interface

O protótipo final da aplicação desenvolvida assenta sobre duas janelas: a primeira sendo a janela inicial com os diferentes filmes das diferentes categorias; a segunda sendo o leitor de vídeo criado. Cada uma das interfaces determina as ações realizadas pelo módulo *LeftHand*, por existirem gestos comuns com ações diferentes entre as mesmas. Ao

desenvolver de raiz a interface gráfica desta aplicação, conseguimos prontamente identificar os elementos (i.e., botões, *thumbnails*, secções, janelas) por debaixo do cursor e despoletar a ação desejada. Serão usados os nomes dos gestos presentes no protótipo MediaPipe para os exemplos dos pares gesto-ação referidos neste subcapítulo.

Na interface principal, as diversas categorias são empilhadas verticalmente, com os seus respetivos filmes dispostos em fila horizontal com possibilidade de *scroll* nesta orientação.

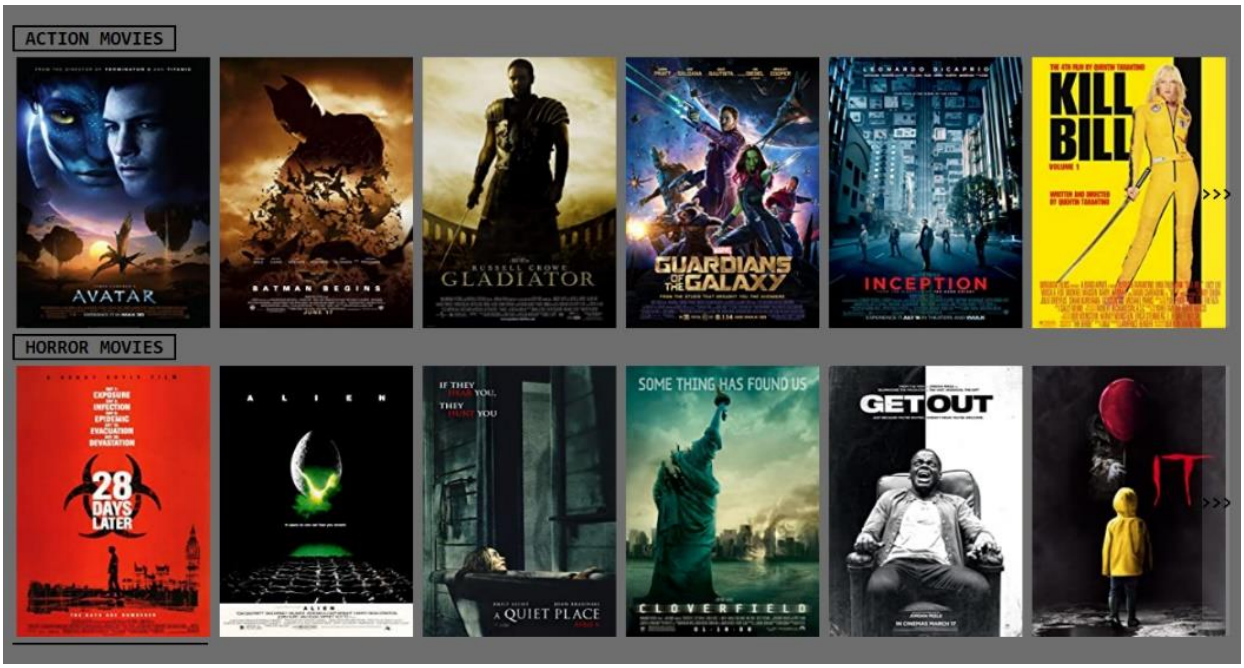


Figura 46 - Interface principal

O nome da categoria é criado de acordo com o nome da pasta presente no diretório raiz. A secção diretamente abaixo é povoada pelas imagens presentes na pasta da categoria.

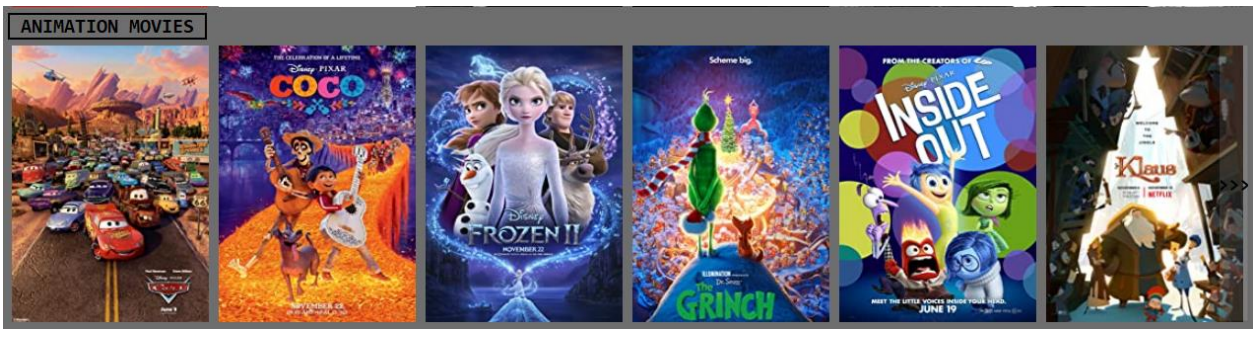


Figura 47 - Elementos da interface da categoria "*Animation Movies*"

Ao ter controlo sobre os elementos de uma categoria, facilmente consegue-se associar a vários elementos uma ação em comum. Na Figura 47, por exemplo, com o gesto *five*, realizado com o cursor sobre qualquer um dos filmes, somos capazes de descobrir a referência do botão de *scroll* horizontal desta secção e simular um clique, sem necessidade de colocar o cursor explicitamente por cima deste. Analogamente, com o *telephone*, podemos fazer *scroll* para a esquerda.

Figura 48 – "*Animation Movies*" com botão de *scroll* para a esquerda

Do mesmo modo, o gesto *rock* permite fazer *scroll* para baixo para ver outras categorias, enquanto o *three* produz a ação oposta. A ação de abrir um filme é feita ao manter o cursor (mão direita) sobre o filme pretendido e realizar a pose *fist* (mão esquerda).

Ao selecionar um filme, é aberta uma nova janela sobreposta à interface principal. O espaço central, adequado para uma resolução 16:9, está reservado para a reprodução do ficheiro de vídeo, com os botões de controlo multimédia dispostos por baixo.

Figura 49 - Reprodutor de vídeo

São apresentados ao utilizador os botões para reproduzir, pausar e parar o vídeo. Cada um destes apresenta um *feedback* visual, explicado mais à frente, quando clicado. À sua frente, são visíveis contadores com o tempo decorrido e a duração total do vídeo, bem como uma barra de progresso controlável entre estes. Por fim, é também dado ao utilizador controlo sobre o volume da aplicação através de um controlo deslizável.

Figura 50 - Ampliação dos botões de controlo multimédia

Atendendo ao facto que esta será uma interface manipulada através do rastreamento gestual no espaço – e não dependente de uma interação direta com os botões introduzidos – foi pensado e implementado um sistema de *feedback* visual que confirma uma ação realizada com êxito. Com um gesto processado bem-sucedidamente pelo módulo *LeftHand*, o programa simula o pressionar do seu botão associado na interface. Surge no ecrã, momentaneamente, sobre o vídeo reproduzido, o ícone da ação executada. Este sistema permite ao utilizador saber, inequivocamente, qual a ação levada a cabo, assim como quando foi efetuada.

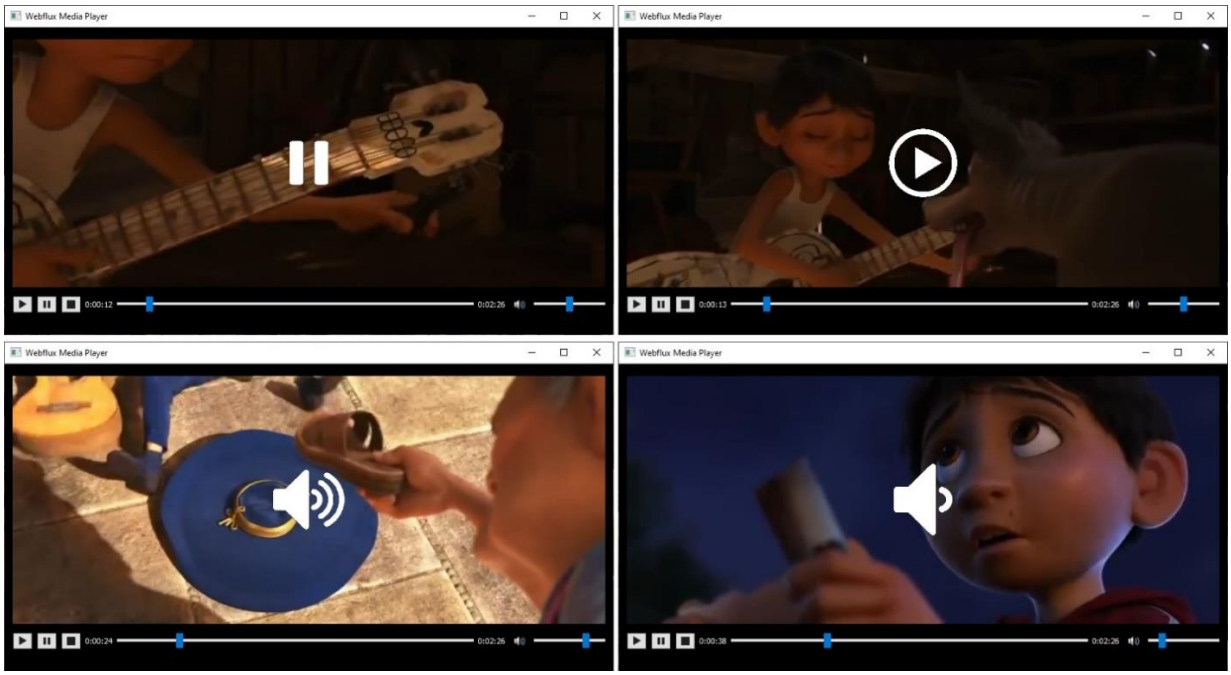


Figura 51 - Ícones de ações bem-sucedidas sobrepostos ao vídeo

Quando o utilizador se encontra nesta janela, as interações gestuais utilizadas na interface principal despoletam ações diferentes. Seguindo a ordem da figura anterior, o gesto *three* é responsável por pausar e retomar a reprodução do vídeo; a pose *rock* está associada ao aumento do volume; e o gesto *fist* despoleta a diminuição do volume. Para fechar esta janela e regressar à interface principal, utiliza-se o *telephone*.

4. Avaliação

Neste capítulo começamos por comparar ambos os protótipos desenvolvidos, prosseguindo com o protótipo superior para testes-piloto e testes com utilizadores. Ao longo destes últimos, focamo-nos em problemas comuns aos vários participantes, tentando chegar a uma solução futura funcional.

4.1. Análise experimental

Após a elaboração de ambos os protótipos, realizaram-se análises experimentais que permitiram comparar diretamente os desempenhos de cada protótipo. Tentou-se simular um uso habitual da aplicação seguindo uma lista ordenada de tarefas a realizar. Registou-se o tempo gasto, ações bem-sucedidas e malsucedidas, tais como quaisquer notas relevantes. As tarefas a efetuar, por ordem, foram:

a) Na secção "***Animation Movies***", abrir o filme ***Shrek***
b) Ao fim de 15 segundos, pausar a reprodução
c) Fechar a janela de visualização
d) Ir até à secção "***Romance Movies***" e abrir o filme ***La La Land***
e) Diminuir o volume para 0%
f) Pausar a reprodução
g) Retomar a reprodução
h) Aumentar o volume para 50%
i) Fechar a janela de visualização

Os resultados expostos abaixo foram obtidos com um plano de fundo relativamente desimpedido e num espaço bem iluminado por luz natural.

O primeiro protótipo testado foi o OpenCV. Os seus resultados são como seguem:

Iteração nº	*Tempo (min:s)*	*Ações erradas*
1	2:46	0
2	2:26	0
3	2:29	0
4	2:57	2
5	2:45	1
média	2:41 (SD = 0:12)	–

Tabela 3 – Testes informais com o protótipo OpenCV

A principal desvantagem observada durante a testagem deste protótipo foi a necessidade de reajustar constantemente a mão direita de volta à área predefinida, para continuar o movimento do cursor até à posição desejada. Além disto, a máscara da mão, ao sair da área no fim de cada tentativa de movimento do cursor, provocava um pequeno desvio na posição final do cursor. Em alguns casos, isto levou a que tivesse de ser feito um movimento adicional para colocar o cursor na posição desejada.

De um modo geral, não ocorreram erros por parte do modelo no reconhecimento que afetassem o bom funcionamento da aplicação. Contudo, nos dois últimos testes realizados, o programa, em três ocasiões, reconheceu gestos diferentes dos apresentados à câmara, que despoletaram ações não pretendidas na interface. Assim, tiveram que ser realizadas as ações opostas para correção destas gafes, pelo que estes dois testes se prolongaram em relação aos anteriores.

O protótipo falhou a proporcionar uma utilização simples, principalmente pelo tempo necessário para mover o cursor a distância pretendida e a necessidade de repetir os mesmos gestos várias vezes. No entanto, não produziu um grande número de ações erradas quando considerado o tempo total de uso com a interface.

De seguida, foram feitos os mesmos testes com o protótipo MediaPipe.

Iteração nº	*Tempo (min:s)*	*Ações erradas*
1	2:31	0 *
2	2:03	0 *
3	2:38	2 *
4	2:08	0 *
5	1:58	0
média	2:28 (SD = 0:20)	–

Tabela 4 – Testes informais com o protótipo MediaPipe

Os testes realizados relatam tempos menores comparativamente ao protótipo anterior. De referir que na vasta maioria das iterações (assinaladas com um asterisco), a *MediaPipe* identificou mãos na imagem que não existiam, normalmente apenas durante algumas *frames*. Apenas na terceira iteração estas "mãos-fantasma" despoletaram ações erradas no programa: a primeira detetou duas vezes o gesto de abrir o filme escolhido, o que levou a ter que fechar uma destas duas janelas; a segunda ocorrência foi a identificação de uma mão direita durante alguns *frames*, o que foi suficiente para mover o cursor para fora da interface e ter que o reajustar para efetuar a tarefa pretendida. Nas restantes iterações assinaladas, as duplas deteções existentes não impediram o funcionamento tolerável da aplicação.

A principal limitação apontada a este sistema, para além das duplas deteções ocasionais, é identificar quais as mãos na imagem que controlam a aplicação. Os testes efetuados apenas tiveram um utilizador em frente à câmara. No entanto, se estiverem duas pessoas presentes na *frame*, torna-se um desafio decidir qual o utilizador com o controlo sobre a aplicação. Por outro lado, revelou-se uma utilização mais natural da interface, sobretudo por não ter áreas estritamente definidas para cada mão, podendo movê-las livremente sem receio de não serem detetadas.

Nenhum dos dois protótipos teve um desempenho aceitável em condições não ideais. Ao testar ambos os modos com fraca iluminação interior, as *frames* captadas pela câmara, tanto em modo RGB como infravermelho, exibiam demasiados artefactos. Estes, em ambos os protótipos, criavam elementos aleatórios na *frame* que eram lidos e interpretados pelo

programa, ao invés de serem ignorados. A existência destes artefactos que interferiam com o desempenho dos modelos de deteção, impossibilitaram a realização das análises experimentais sob estas condições de luminosidade até ao fim.

Figura 52 – Alguns artefactos (a verde) presentes na frame captada pela câmara

Por fim, foi feita uma última medição, desta vez usando um rato de computador. Estes valores reportados são considerados "valores de controlo", visto representarem o uso habitual de uma aplicação no computador.

Iteração nº	***Tempo (min:s)***
1	1:11
2	1:15
3	1:08
4	1:14
5	1:09
média	1:11 (SD = 0:03)

Tabela 5 – Testes informais usando o rato de computador

Com estes valores, podemos confirmar que qualquer um dos protótipos criados teve um desempenho pior que a utilização com o rato. Isto pode ser devido aos problemas discutidos anteriormente, ao tempo necessário para executar uma ação com um gesto (definido no código como 2 segundos), habituação a trabalhar com computadores com o auxílio do rato, ou uma combinação destes e outros fatores. Espera-se que as diferenças sejam ainda mais acentuadas na testagem com utilizadores que se deparam com a aplicação pela primeira vez.

Com as avaliações efetuadas ao projeto desenvolvido, e por ter superado todas as métricas do protótipo OpenCV, foi selecionado o protótipo MediaPipe para prosseguir com os testes com utilizadores. A liberdade de movimentos deste último é uma grande melhoria na usabilidade da aplicação, não comprometendo a qualidade dos resultados obtidos.

4.2. Testes-piloto

Após as análises experimentais iniciais, foram feitos testes-piloto com 2 utilizadores de 23 anos de idade, 1 homem e 1 mulher. Estes testes-piloto foram testes de usabilidade *CTA* (*Concurrent Think Aloud*) (Lewis, 1982), em que os participantes interagem com o sistema ao mesmo tempo que verbalizam os seus pensamentos e ações.

Estes estudos não foram cronometrados e apenas se procurou detetar os principais problemas da interface e interações gestuais, observando os participantes e anotando os pontos mais relevantes.

Com estes, foi possível apurar quais as áreas no uso da aplicação que necessitavam de uma explicação mais aprofundada para melhor preparar os utilizadores à sua utilização. Destes testes não resultaram alterações à interface ou às poses designadas para despoletar ações na interface.

4.3. Testes com utilizadores

Foram realizados testes com utilizadores usando o protótipo MediaPipe, pelas métricas registadas no primeiro subcapítulo desta secção. Este estudo contou com a participação de 11 utilizadores, 8 homens e 3 mulheres, de idades entre os 21 e 61 (média = 39.27; SD = 16.32) anos.

Para avaliar o funcionamento da aplicação desenvolvida, foi pedido aos utilizadores que realizassem as tarefas descritas anteriormente, controlando o programa, à vez, com o rato e com interações gestuais. Os testes foram feitos individualmente e com uma explicação prévia da interface e gestos incorporados, bem como um período de experimentação com o reconhecimento gestual por parte da câmara. De notar que nos participantes de idade mais avançada esta explicação prolongou-se mais. Por fim, foi pedido a cada utilizador que respondesse ao questionário padrão *SUS* (Brooke, 1995) e, facultativamente, adicionasse algum comentário ou sugestão à sua avaliação.

De forma a evitar ao máximo *biases* nas comparações entre os tempos dos diferentes modos de interação, 6 dos 11 participantes controlaram primeiro a interface via interação gestual, com os restantes 5 a começarem pelo rato. A tabela seguinte mostra, para cada participante, o tempo gasto em cada teste e a pontuação das suas respostas ao questionário *SUS*. A interação inicial de cada utilizador tem o seu tempo destacado a negrito.

Participante	***Idade***	***Rato (min:s)***	***Mãos (min:s)***	***Pontuação SUS***
1	60	1:30	**3:41**	90
2	39	**2:45**	4:46	65
3	24	1:01	**3:03**	72.5
4	58	**4:30**	5:59	45
5	21	1:11	**3:18**	77.5
6	24	**1:08**	1:56	77.5
7	60	2:23	**6:18**	52.5
8	61	**2:21**	3:58	67.5
9	36	1:13	**2:32**	72.5
10	24	**1:15**	3:18	77.5
11	25	1:02	**2:32**	92.5
*média (*sem outlier)*	39 (SD = 16)	1:35 (SD = 0:37)*	3:46 (SD = 1:20)	71.8 (SD = 13.53)

Tabela 6 – Resultados dos testes com utilizadores

Em geral, o estudo revelou uma pontuação média de usabilidade (*SUS*) de 71.8 (SD = 13.53). Bangor et al. (2008) apresentam uma escala na qual uma pontuação abaixo de 50 é classificada "inaceitável", e um valor acima de 70 como "aceitável", grupo em que se insere esta aplicação. No total, os participantes levaram em média 3 minutos e 46 segundos (SD =

1 minuto e 20 segundos) a concluir as tarefas via interação gestual. Para análise dos resultados de tempo de execução com recurso ao rato, foram excluídos os *outliers* (participante 4 na Tabela 6). O tempo médio de 1 minuto e 35 segundos (SD = 37 segundos) com o rato é consideravelmente mais baixo que os tempos reportados no controlo da aplicação com movimentos gestuais. Estes resultados sugerem maior facilidade em identificar as ações necessárias na interface com a utilização do rato.

De notar que é possível agregar os participantes em 2 grupos de faixas etárias distintas que apresentam valores semelhantes nas métricas registadas para uma análise mais compreensiva dos valores obtidos. A tabela seguinte reorganiza as informações já apresentadas.

Participante	***Idade***	***Rato (min:s)***	***Mãos (min:s)***	***Pontuação SUS***
8	61	**2:21**	3:58	67.5
1	60	1:30	**3:41**	90
7	60	2:23	**6:18**	52.5
4	58	**4:30**	5:59	45
2	39	**2:45**	4:46	65
média	56 (SD = 8)	2:42 (SD = 0:59)	4:56 (SD = 1:03)	64 (SD = 12.38)
9	36	1:13	**2:32**	72.5
11	25	1:02	**2:32**	92.5
3	24	1:01	**3:03**	72.5
6	24	**1:08**	1:56	77.5
10	24	**1:15**	3:18	77.5
5	21	1:11	**3:18**	77.5
média	26 (SD = 5)	1:08 (SD = 0:05)	2:57 (SD = 0:19)	78.3 (SD = 6.72)

Tabela 7 – Resultados dos testes com utilizadores separados em dois grupos

A tabela, agora ordenada decrescentemente por idade, permite observar dois universos com discrepâncias acentuadas. Os utilizadores de idades entre 39 e 61 (média = 55.6; SD = 8.36) anos gastaram, em média, 2 minutos e 42 segundos (SD = 59 segundos) a realizar as tarefas com o rato. Os participantes de idades entre os 21 e 36 (média = 26.67; SD = 4.78) anos gastaram apenas, em média, 1 minuto e 8 segundos (SD = 5 segundos) com os mesmos objetivos.

A análise ao método de interação gestual revela resultados análogos. Utilizadores do grupo etário mais jovem levaram, em média, 2 minutos e 57 segundos (SD = 19 segundos), em contraste com os 4 minutos e 56 segundos (SD = 1 minuto e 3 segundos) da sua contraparte.

A avaliação média de 78.3 (SD = 6.72) dos questionários *SUS* pelo segundo conjunto significa que a aplicação mantém a classificação "aceitável" segundo (Bangor et al., 2008). Para o primeiro grupo, a aplicação teve uma pontuação de usabilidade média de 64 (SD = 12.38), classificando-se como "marginalmente aceitável". Considera-se que a aplicação desenvolvida, numa análise geral, não criou grandes obstáculos aos participantes deste estudo.

Apesar das suas pontuações finais não aparentarem grandes divergências, questões individuais oscilam razoavelmente nas respostas obtidas. A Figura 53 expõe as respostas às questões do questionário padrão *SUS* sob forma de diagrama de caixa. No diagrama seguinte, conferimos os limites inferior e superior, o primeiro (azul-claro) e terceiro (azul-escuro) quartil, bem como a mediana dos valores representados (linha entre o primeiro e terceiro quartil).

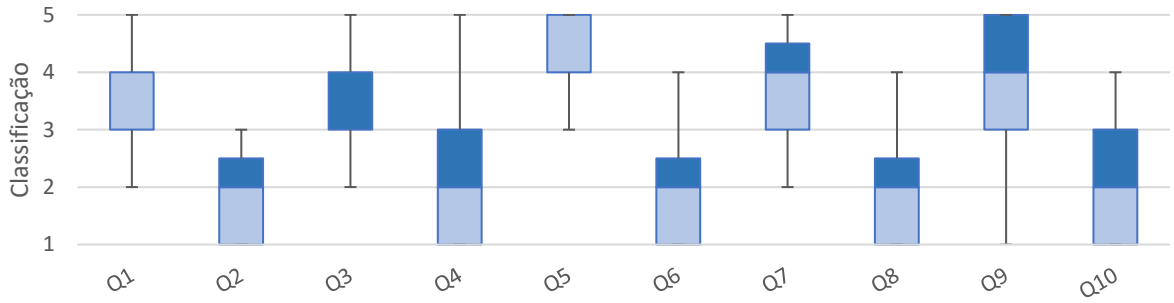


Figura 53 – Diagrama de caixa das questões nos questionários SUS

Observamos que 80% das questões obtiveram respostas em que, numa escala de apenas 5 valores, existe uma distância de pelo menos 4 valores entre a classificação mais baixa e a classificação mais alta da mesma questão. Em 20% das perguntas totais, todos os valores foram encontrados entre as respostas da população analisada. Estas oscilações mostram uma necessidade futura de aperfeiçoar o sistema e a sua interface, na procura de

melhorar a experiência do utilizador e alcançar um maior consenso sobre a aplicação desenvolvida.

Durante os testes realizados, foi possível notar que 6 dos 11 (55%) participantes tendiam a descansar a mão esquerda (responsável pelos gestos que despoletam ações na aplicação) em pose de punho, enquanto reliam a tarefa seguinte ou se concentravam na ação a efetuar. Isto levou a que, por vezes, o sistema reconhecesse a pose efetuada acidentalmente e, inconvenientemente, abrisse o filme por debaixo do cursor.

No total, 3 utilizadores (27%) mostraram dificuldade a realizar e manter os gestos *rock* e *telefone*. Por outro lado, 2 participantes (18%) exibiram sinais de confusão quando deparados com cenários *multitasking* e pelos mesmos gestos despoletarem ações diferentes na interface principal e durante a visualização do filme.

Por fim, 3 utilizadores (27%) sugeriram alterar o gesto de pausar e retomar a reprodução do vídeo (gesto *three*), embora tenham os 3 proposto gestos diferentes – *five*, *fist* e *two*. Por estas sugestões terem surgido inicialmente com os 2 primeiros utilizadores, os restantes 9 participantes foram questionados diretamente sobre a alteração de qualquer gesto e ação, ao qual apenas 1 respondeu positivamente.

4.4. Discussão de resultados

Os testes com utilizadores permitiram identificar uma nova limitação do sistema, equivalente a uma já identificada durante a fase de desenvolvimento. Na Figura 42 é verificada a ocorrência de deteções erradas, dependendo do vestuário utilizado. Aquando da realização dos testes com o participante 7, foi necessário realocar para um local de teste distinto, visto que o plano de fundo (parede interior de tijolo) causava complicações à MediaPipe no reconhecimento das mãos na imagem. O sistema funcionou satisfatoriamente nos restantes ambientes.

No mesmo intuito, foi necessária atenção extra à montagem do sistema em ambiente de teste. O computador portátil em que decorreram os testes, bem como a câmara RealSense posicionada junto ao ecrã, tiveram que estar a uma distância maior que a habitual entre o monitor de computador e o utilizador, para permitir um plano amplo o suficiente para os movimentos gestuais efetuados.

Em sumário, foi demonstrado que é possível controlar uma aplicação de *streaming* de vídeo através somente de interações gestuais, com uma usabilidade aceitável. Não obstante esta possibilidade, o rato continuou a ser o modo preferencial de interação por parte dos participantes deste estudo inicial. Verificou-se também alguma variação entre as curvas de aprendizagem de diferentes faixas etárias, o que convida a exploração de novas formas de introdução da população de idade mais avançada a técnicas de interação por *computer vision*.

5. Conclusões

Esta dissertação implicou o contacto com várias tecnologias do meio *computer vision,* com o intuito de desenvolver uma aplicação de *streaming* de vídeo para computador, controlável através apenas por interações gestuais. Para tal, foi imprescindível encontrar uma solução viável e capaz de produzir resultados em tempo real.

Foi feita uma revisão de literatura extensiva que permitiu identificar diferentes abordagens a implementar no projeto. Processos como subtração do plano de fundo e deteção e classificação de objetos foram explorados, este último abordando dois métodos distintos.

Ao longo do desenvolvimento deste projeto, foram criados dois protótipos que implementam dois funcionamentos adversos. O protótipo OpenCV faz uso das capacidades de leitura de profundidade da câmara RealSense D435 utilizada, captando imagens recorrendo ao seu *feed* de luz infravermelha. Para este protótipo, são estritamente definidas áreas reservadas para localização de cada mão na *frame*. O programa trabalha sem contexto sobre o objeto que se encontra dentro desta região, e assume que se trata efetivamente de uma mão. A silhueta da mão é interpretada recorrendo a um modelo pré-treinado que a compara com as suas imagens de referência, e retorna, para cada pose possível, uma pontuação.

O protótipo MediaPipe, por sua vez, utiliza o *feed* RGB da mesma câmara, podendo ser adaptado a qualquer câmara de vídeo comum. Este protótipo incorpora um modelo de deteção e classificação de objetos otimizado para a deteção de mãos em imagens (*BlazePalm*). Quando detetada uma mão na *frame*, retorna um esqueleto composto por 21 pontos de referência, com coordenadas *x* e *y*, que permitem facilmente interpretar quais dedos se encontram abertos ou fechados ou. O modelo atribui ainda uma etiqueta mão "esquerda" ou "direita" ao esqueleto encontrado, o que elimina a necessidade de regiões predefinidas para cada mão do protótipo anterior. Os gestos realizados pelo utilizador são calculados no sistema pelas coordenadas dos pontos de referência na imagem.

Após a fase de desenvolvimento, análises experimentais e comparações entre os dois protótipos permitiram identificar a vasta superioridade do protótipo MediaPipe. Apesar das suas limitações, esta abordagem oferece uma grande liberdade de movimentos e maior precisão no movimento do cursor no ecrã. Por ter superado o protótipo OpenCV, tanto em métricas quantitativas como qualitativas, optou-se por prosseguir exclusivamente com o protótipo MediaPipe para testes com utilizadores.

Procedeu-se à avaliação da aplicação num estudo com o qual contaram 11 participantes. Em testes individuais, foram registados os tempos de execução de uma lista ordenada de instruções, tanto com o sistema completo de rastreio gestual, como com a utilização do rato de computador. Estes últimos valores são considerados "valores de controlo", que correspondem às interações habituais entre utilizadores e aplicações de computador. Seguidamente, os participantes preencheram o questionário de usabilidade *SUS* para apreciação do sistema. Verificou-se uma utilização mais demorada com interações gestuais em todos os utilizadores, mantendo-se o rato o método de controlo mais rápido e exato. Contudo, o sistema mereceu uma avaliação de usabilidade "aceitável" por parte dos participantes, que reagiram positivamente às inovações introduzidas.

Considera-se que os objetivos definidos inicialmente foram cumpridos com sucesso, proporcionando uma alternativa ao rato de computador no contexto proposto. Contudo, este projeto no seu estado atual requer iterações futuras para suplantar as limitações encontradas.

5.1. Limitações

As principais limitações encontradas durante o desenvolvimento da aplicação provêm dos limites inerentes a cada método de processamento e deteção de objetos. Em especial, a dificuldade em obter resultados de alta confiança com um tempo de processamento adequado ao uso de aplicações em tempo real.

Um dos objetivos deste trabalho era estudar e comparar os vários métodos de interação gestual com computador existentes atualmente. Constatou-se que a maioria dos métodos disponíveis ao público pecam, geralmente, ou pelo tempo de processamento, ou pela confiança e fiabilidade dos seus resultados. No entanto, é possível encontrar um reduzido número de modelos satisfatórios que ambicionam solucionar este dilema, tal é o caso da *framework* MediaPipe.

A qualidade dos resultados obtidos por esta *framework* são, por norma, excelentes. Contudo, as limitações descobertas em casos de oclusão de duas mãos, bem como quanto ao plano de fundo e vestuário usados, podem tornar os resultados inutilizáveis.

Por último, a identificação do utilizador que controla a aplicação, num cenário de vários utilizadores no enquadramento da câmara, não foi abordada neste trabalho.

5.2. Trabalho futuro

Esta dissertação teve um foco no desenvolvimento de um sistema que permitisse ao utilizador controlar uma aplicação de *streaming* de vídeo, através de interações gestuais. Como tal, seria uma mais-valia para este trabalho a adição de outras abordagens de reconhecimento e rastreio gestual ao capítulo revisão de literatura, com o objetivo de encontrar os resultados mais fiáveis possíveis com o tempo de processamento exigido por uma aplicação desta natureza.

O trabalho desenvolvido retornou resultados interessantes no estudo realizado. Seria útil prosseguir com os testes realizados e obter uma maior amostra de utilizadores, de preferência continuando a análise dos resultados observados entre participantes de diferentes faixas etárias.

Outra sugestão para uma iteração futura do sistema é o desenvolvimento de um novo conjunto de gestos que melhor representem as ações despoletadas na interface. Os gestos *rock* e *three*, por exemplo, despoletam na interface *scroll* a nível vertical, ações

normalmente não associadas com estas poses gestuais. O conceito de gestos semelhantes às suas ações no ecrã deverá ajudar a diminuir a curva de aprendizagem existente.

Como consideração final, sendo este um programa de visualização e manipulação de conteúdos multimédia, seria interessante explorar a viabilidade da sua expansão a outros dispositivos, nomeadamente *smart TVs*. Especula-se que, com investigação do tópico e contínuo desenvolvimento deste projeto, esta seria uma evolução natural do trabalho apresentado.

6. Bibliografia

Aguiar, D.G. de, 2018. Interações Gestuais e Faciais Detetadas com Câmara de Profundidade.

Alfred Camera: Smart camera features using MediaPipe, 2020. . Google Dev. Blog. URL https://developers.googleblog.com/2020/03/alfred-camera-smart-camera-features-using-mediapipe.html (accessed 4.30.21).

Babaee, M., Dinh, D.T., Rigoll, G., 2019. A deep convolutional neural network for video sequence background subtraction. Pattern Recognit. 76, 635–649. https://doi.org/10.1016/j.patcog.2017.09.040

Background Subtraction - OpenCV-Python Tutorials [WWW Document], 2021. URL https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html (accessed 4.23.21).

Bangor, A., Kortum, P.T., Miller, J.T., 2008. An Empirical Evaluation of the System Usability Scale. Int. J. Human–Computer Interact. 24, 574–594. https://doi.org/10.1080/10447310802205776

Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., Grundmann, M., 2019. BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. ArXiv190705047 Cs.

Bazarevsky, V., Zhang, F., 2019. Google AI Blog: On-Device, Real-Time Hand Tracking with MediaPipe [WWW Document]. URL https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html (accessed 4.20.21).

Brooke, J., 1995. SUS: A quick and dirty usability scale. Usability Eval Ind 189.

Chin, J., 2021. jrobchin/Computer-Vision-Basics-with-Python-Keras-and-OpenCV.

Coates, A., Baumstarck, P., Le, Q., Ng, A.Y., 2009. Scalable Learning for Object Detection with GPU Hardware 7.

Cruz, D.H. da S., 2020. Interfaces e métodos de pesquisa visual em imagens.

Depth Camera D435 [WWW Document], 2021. . Intel® RealSense™ Depth Track. Cameras. URL https://www.intelrealsense.com/depth-camera-d435/ (accessed 8.27.21).

DJ Ravine, 2020. DJ Ravine's NO HANDS DJ Mix with Gesture Control & Machine Learning (djay pro AI).

Dormehl, luke, 2021. The History and Evolution of the Computer Stylus [WWW Document]. Digit. Trends. URL https://www.digitaltrends.com/features/evolution-history-of-the-stylus/ (accessed 5.5.21).

Freewell Gear, 2019. MOTION BLUR EXPLAINED IN 2 MINUTES (shutter speed and frame rate 180 degree rule).

Gesture Control | Algoriddim [WWW Document], 2021. URL https://www.algoriddim.com/gesturecontrol (accessed 5.7.21).

Goldin-Meadow, S., Alibali, M.W., 2013. Gesture's Role in Speaking, Learning, and Creating Language. Annu. Rev. Psychol. 64, 257–283. https://doi.org/10.1146/annurev-psych-113011-143802

Hand Tracking | Oculus [WWW Document], 2021. URL https://support.oculus.com/2720524538265875/ (accessed 5.7.21).

Hangün, B., Eyecioğlu, Ö., 2017. Performance Comparison Between OpenCV Built in CPU and GPU Functions on Image Processing Operations 8.

Heintz, B., 2018. Training a Neural Network to Detect Gestures with OpenCV in Python [WWW Document]. Data Sci. URL https://towardsdatascience.com/training-a-neural-network-to-detect-gestures-with-opencv-in-python-e09b0a12bdf1 (accessed 4.23.21).

Ilango, G., 2017. Hand Gesture Recognition using Python and OpenCV - Part 2 [WWW Document]. Gogul Ilango. URL https://gogul.dev/software/hand-gesture-recognition-p2 (accessed 4.25.21).

Intel® Core™ i7-4770 Product Specifications [WWW Document], 2021. URL https://ark.intel.com/content/www/us/en/ark/products/75122/intel-core-i7-4770-processor-8m-cache-up-to-3-90-ghz.html (accessed 4.20.21).

Intel® Core™ i7-6700 Product Specifications [WWW Document], 2021. URL https://ark.intel.com/content/www/us/en/ark/products/88196/intel-core-i7-6700-processor-8m-cache-up-to-4-00-ghz.html (accessed 4.20.21).

Intel RealSense, 2021. Intel® RealSense™ Depth Camera D455 Quick Start optimization.

Kashinkunti, P., Chabert, F., 2019. Creating an Object Detection Pipeline for GPUs [WWW Document]. NVIDIA Dev. Blog. URL https://developer.nvidia.com/blog/object-detection-pipeline-gpus/ (accessed 4.20.21).

Lang, B., 2020. Oculus "Designing for Hands" Introduces Best Practices for Quest Hand-tracking. Road VR. URL https://www.roadtovr.com/oculus-designing-for-hands-quest-hand-tracking-best-practices/ (accessed 5.7.21).

Lewis, C., 1982. Using the "Thinking-aloud" Method in Cognitive Interface Design. RC9265 6.

Li, Z., 2021. lzane/Fingers-Detection-using-OpenCV-and-Python.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016a. SSD: Single Shot MultiBox Detector. ArXiv151202325 Cs. https://doi.org/10.1007/978-3-319-46448-0_2

Liu, W., Cai, Y., Zhang, M., Li, H., Gu, H., 2016b. Scene background estimation based on temporal median filter with Gaussian filtering, in: 2016 23rd International Conference on Pattern Recognition (ICPR). Presented at the 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 132–136. https://doi.org/10.1109/ICPR.2016.7899621

MediaPipe Hands [WWW Document], 2021. . mediapipe. URL https://google.github.io/mediapipe/solutions/hands.html (accessed 4.20.21).

MediaPipe: Object Detection [WWW Document], 2021. . mediapipe. URL https://google.github.io/mediapipe/solutions/object_detection.html (accessed 4.30.21).

MediaPipe: Objectron (3D Object Detection) [WWW Document], 2021. . mediapipe. URL https://google.github.io/mediapipe/solutions/objectron.html (accessed 4.30.21).

Misra, S., Wu, Y., 2020. Gaussian Blur - an overview | ScienceDirect Topics [WWW Document]. URL https://www.sciencedirect.com/topics/engineering/gaussian-blur (accessed 4.23.21).

Ng, A., n.d. Non-max Suppression - Object Detection [WWW Document]. Coursera. URL https://www.coursera.org/lecture/convolutional-neural-networks/non-max-suppression-dvrjH (accessed 4.20.21).

NVIDIA CUDA Zone [WWW Document], 2021. . NVIDIA Dev. URL https://developer.nvidia.com/cuda-zone (accessed 4.20.21).

NVIDIA cuDNN [WWW Document], 2021. . NVIDIA Dev. URL https://developer.nvidia.com/cudnn (accessed 4.20.21).

NVIDIA GeForce GTX 970 Specs [WWW Document], 2021. . TechPowerUp. URL https://www.techpowerup.com/gpu-specs/geforce-gtx-970.c2620 (accessed 4.20.21).

NVIDIA GeForce GTX 1060 [WWW Document], 2021. . NVIDIA. URL https://www.nvidia.com/pt-br/geforce/products/10series/geforce-gtx-1060/ (accessed 4.20.21).

OpenCV: About, 2021. . OpenCV. URL https://opencv.org/about/ (accessed 4.20.21).

OpenCV: CUDA, 2021. . OpenCV. URL https://opencv.org/platforms/cuda/ (accessed 4.20.21).

OpenCV: How to Use Background Subtraction Methods [WWW Document], n.d. URL https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html (accessed 4.22.21).

OpenCV: Installation in Windows [WWW Document], 2021. URL https://docs.opencv.org/master/d3/d52/tutorial_windows_install.html (accessed 4.20.21).

Padilha, A.J., 2013. Processamento e Análise de Imagem [WWW Document]. URL https://web.fe.up.pt/~padilha/PAI/ficheiros/Cap4-ac.pdf (accessed 4.23.21).

Panasonicsecurity, 2015. 4K camera example for Traffic Monitoring (Road).

Pogue, D., 2013. The Trouble with Touch Screens. Sci. Am. 308, 25–25.

Rosebrock, A., 2020a. How to use OpenCV's "dnn" module with NVIDIA GPUs, CUDA, and cuDNN. PyImageSearch. URL https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/ (accessed 4.20.21).

Rosebrock, A., 2020b. OpenCV "dnn" with NVIDIA GPUs: 1549% faster YOLO, SSD, and Mask R-CNN. PyImageSearch. URL https://www.pyimagesearch.com/2020/02/10/opencv-dnn-with-nvidia-gpus-1549-faster-yolo-ssd-and-mask-r-cnn/ (accessed 4.20.21).

Rosebrock, A., 2016. Intersection over Union (IoU) for object detection. PyImageSearch. URL https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (accessed 4.20.21).

scgrinchy, 2018. Gesture Recognition Demo with Python & OpenCV.

Shiels, M., 2008. Firm makes one billionth mouse.

Turi, J., 2020. Djay Pro AI for iPad now has touchless gesture controls [WWW Document]. Engadget. URL https://www.engadget.com/djay-pro-ai-ipad-touchless-gesture-controls-150015498.html (accessed 5.7.21).

Virtual Reality Oasis, 2019. Oculus Quest Hand Tracking Is HERE.

Vision | Apple Developer Documentation [WWW Document], n.d. URL https://developer.apple.com/documentation/vision (accessed 5.7.21).

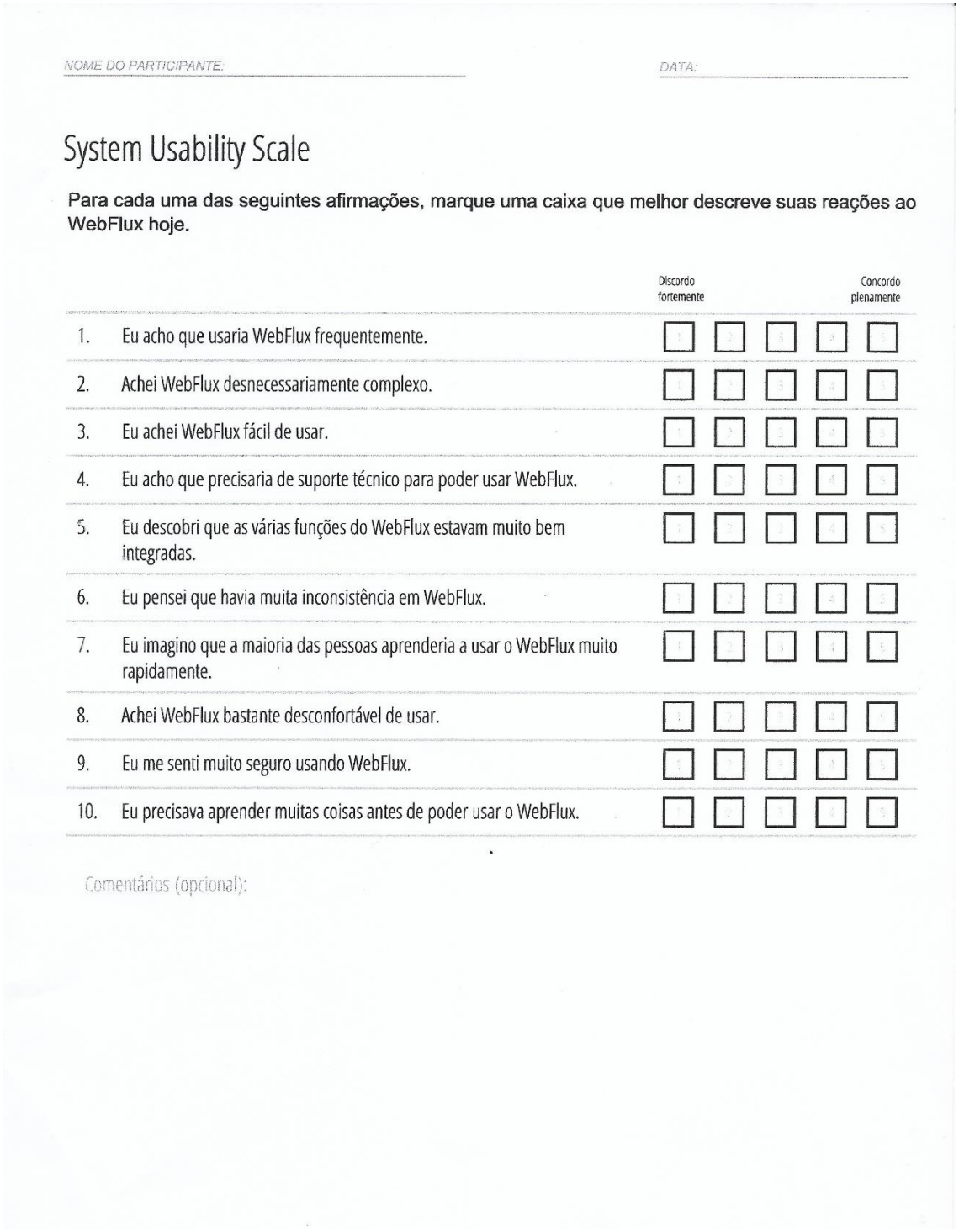
What is NumPy? [WWW Document], 2021. URL https://numpy.org/devdocs/user/whatisnumpy.html (accessed 5.7.21).

Wikipedia - Gaussian blur, 2021. . Wikipedia.

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., Grundmann, M., 2020. MediaPipe Hands: On-device Real-time Hand Tracking. ArXiv200610214 Cs.

Anexos

I. Questionário SUS utilizado nos testes com utilizadores



NOME DO PARTICIPANTE: _____ DATA: _____

System Usability Scale

Para cada uma das seguintes afirmações, marque uma caixa que melhor descreve suas reações ao WebFlux hoje.

		Discordo fortemente				Concordo plenamente
1.	Eu acho que usaria WebFlux frequentemente.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
2.	Achei WebFlux desnecessariamente complexo.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
3.	Eu achei WebFlux fácil de usar.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
4.	Eu acho que precisaria de suporte técnico para poder usar WebFlux.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
5.	Eu descobri que as várias funções do WebFlux estavam muito bem integradas.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
6.	Eu pensei que havia muita inconsistência em WebFlux.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
7.	Eu imagino que a maioria das pessoas aprenderia a usar o WebFlux muito rapidamente.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
8.	Achei WebFlux bastante desconfortável de usar.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
9.	Eu me senti muito seguro usando WebFlux.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
10.	Eu precisava aprender muitas coisas antes de poder usar o WebFlux.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5

Comentários (opcional):