UNIVERSIDADE da MADEIRA

Faculdade de Ciências Exatas e da Engenharia

Licenciatura em Engenharia Informática

Ano letivo: 2023/2024

**Projeto/Estágio**

2º Semestre

Regente e Orientador: Filipe Magno de Gouveia Quintal

# Deteção de sinais de trânsito em estradas da Região Autónoma da Madeira

João Vítor Afonso de Castro

**julho 2024**

# Índice

## Índice de figuras

# I. Introdução

O mundo está a ser cada vez mais virtualizado em várias áreas tais como educação, saúde e bem-estar, entretenimento, entre outras esferas da vida humana. A capacidade da realidade mista unir o mundo real com o virtual está a desencadear uma transformação na maneira como vivemos e nos relacionamos com a tecnologia.

No segundo semestre de 2023, a Meta lançou o aguardado Meta Quest Pro, um headset de realidade mista inovador que faz uma fusão entre o ambiente do mundo real com o ambiente virtual. Notável pelo seu design mais fino em comparação com os seus antecessores, o Meta Quest Pro apresenta lentes e câmeras de alta resolução. Os controladores do Meta Quest Pro são equipados com rastreamento de movimento. Além disso, o headset inclui recursos avançados como rastreamento facial e ocular. (*Meta Quest Pro*, n.d.)

O estudo neste trabalho foi possível graças à contribuição do docente e orientador Filipe Quintal, que forneceu o material e equipamento necessários para sua realização. O Meta Quest Pro foi o principal objeto deste estudo, visando resolver problemas e introduzir novas soluções e experiências numa atividade do quotidiano de grande parte da população mundial: a condução.

# II. Descrição do problema

O problema abordado neste estudo concentra-se na segurança e eficiência da condução. Embora esta seja uma atividade bastante praticada, ainda existem desafios significativos associados à mesma, tais como acidentes de trânsito, distrações ao volante, entre outros. Estes desafios podem resultar em problemas como danos consideráveis e outras adversidades.

A necessidade de abordar esses problemas de forma eficaz e inovadora levou à investigação do Meta Quest Pro, para melhorar a experiência de condução. A realidade mista proporcionada por este equipamento oferece a oportunidade de fundir elementos do mundo real com o virtual, permitindo a identificação de sinais e o fornecimento de avisos pertinentes acerca dos mesmos ao condutor.

Portanto, o objetivo da utilização do Meta Quest Pro é melhorar a eficiência, mas principalmente a segurança da condução, mitigando desafios e perigos associados a esta atividade diária.

# III. Solução

O headset oferece uma funcionalidade chamada Passthrough que permite ao utilizador, através dos sensores do equipamento, alternar da visão virtual para uma visão em tempo real do ambiente real ao seu redor. O Meta Quest Pro apresenta o

full-color Passthrough, uma melhoria significativa em relação a um dos seus antecessores, o Meta Quest 2, que apenas possui um Passthrough a preto e branco. Esta tecnologia aprimorada proporciona uma experiência mais confortável e realista, permitindo que o utilizador interaja diretamente com o mundo físico enquanto está imerso na visão de realidade virtual. (*Full-Color Passthrough on Meta Quest*, 2024; *Use Passthrough on Meta Quest*, 2024)

Ao integrar estas capacidades avançadas do Meta Quest Pro no contexto da condução, o headset pode melhorar a identificação de sinais de trânsito e fornecer avisos relevantes ao condutor, aumentando a segurança na estrada.

A solução inicial proposta envolvia o desenvolvimento de um software específico para o Meta Quest Pro, que utilizaria a funcionalidade Passthrough para identificar e destacar os sinais de trânsito na visão do utilizador. No entanto, devido à dificuldade e à falta de todos os equipamentos necessários – incluindo cabos adequados e um computador Windows capaz de rodar o software Oculus da Meta de forma fluida –, essa abordagem teve de ser adaptada.

A solução atual e final envolve o reconhecimento dos sinais de trânsito num vídeo gravado durante a utilização do Meta Quest Pro, através da funcionalidade Record Video. Durante uma viagem de carro, a utilização do headset é feita pelo passageiro do banco da frente, uma vez que é perigoso, proibido e desaconselhado o uso do Meta Quest Pro pelo condutor. O produto final consiste no vídeo capturado pelo headset, no qual os sinais de trânsito são destacados com figuras geométricas, proporcionando uma visualização clara e informativa dos mesmos. (*Record Video with Meta Quest*, 2024)

## IV. Tecnologias utilizadas

Para a execução deste projeto, foram utilizadas diversas tecnologias, incluindo o Sistema Operativo Windows, o headset Meta Quest Pro, a biblioteca OpenCV 4.9.0 e o IDE IntelliJ IDEA com suporte para Java 17 e Maven. Numa fase intermédia foi utilizado um iPhone SE (2022) para gravação de vídeos.

## V. Utilização das tecnologias

O projeto foi desenvolvido num computador com sistema operativo Windows, uma vez que a configuração de OpenCV no IntelliJ IDEA em macOS mostrou-se mais complicada em relação à mesma no Windows.

O Meta Quest Pro foi utilizado como base do projeto para a gravação dos vídeos, oferecendo uma visão mais realista do ponto de vista de um condutor, permitindo assim avaliar a precisão do headset e da sua funcionalidade Passthrough.

Como mencionado, o OpenCV foi a principal ferramenta para a deteção de sinais de trânsito. Esta biblioteca é otimizada com o foco para aplicações em tempo real e contém mais de 2500 algoritmos. Para além disso oferece suporte extensivo para C++, Python, Java, Windows e macOS. (*OpenCV*, n.d.)

Por fim, o IntelliJ IDEA foi escolhido como o IDE para o desenvolvimento do projeto devido à sua ampla adoção na criação de software em Java e outras linguagens baseadas em JVM (Java Virtual Machine), além da sua excelente compatibilidade com o OpenCV no ambiente Windows. (*IntelliJ IDEA*, n.d.)

## VI. Requisitos

Os principais requisitos de baixo nível do projeto são os seguintes:

1. O sistema deverá ser capaz de identificar diversos tipos de sinais de trânsito em vídeos gravados pelo Meta Quest Pro.
2. O sistema deverá incluir um mecanismo para destacar de alguma forma os sinais de trânsito num vídeo transcrito através do vídeo gravado pelo Meta Quest Pro.
3. O sistema deverá atingir uma taxa mínima de precisão no reconhecimento dos sinais de trânsito, garantindo uma identificação correta e confiável.
4. O sistema deverá operar de forma estável e confiável, minimizando falhas e garantindo uma experiência de utilizador consistente durante a utilização do headset.
5. O sistema deverá ser desenvolvido de forma compatível com atualizações de software do Meta Quest Pro.

## VII. Configuração do sistema

O projeto e todos os ficheiros utilizados foram ao longo do desenvolvimento colocados num repositório GitHub: https://github.com/jovitor03/TrafficSignDetector. Para a inicialização do projeto foi necessária configurar o OpenCV no IntelliJ IDEA. Para isso foi necessário seguir os seguintes passos:

1. Baixar o IntelliJ IDEA:

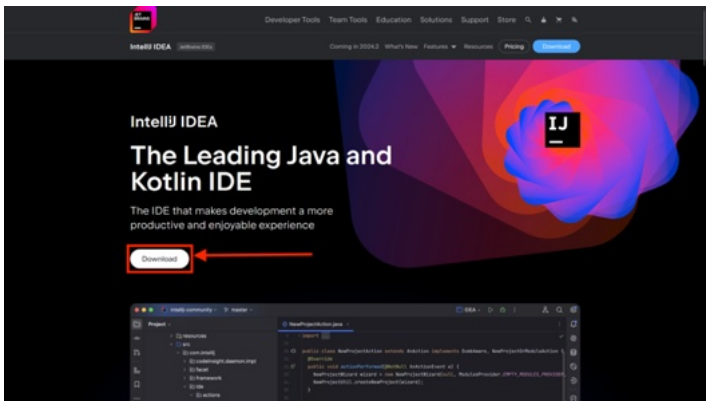


Figura 1 – Website para download do IntelliJ IDEA

2. Clonar o repositório – https://github.com/jovitor03/TrafficSignDetector.git:

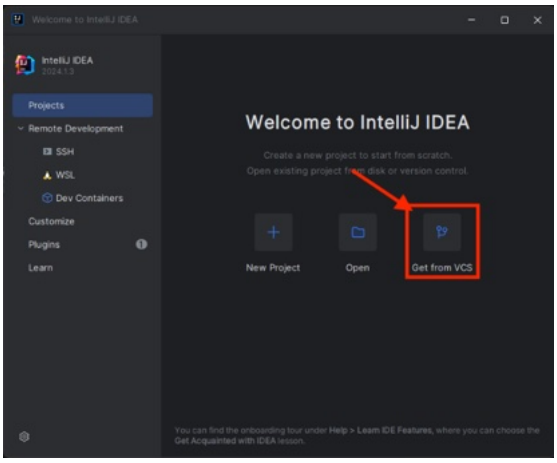


Figura 2 – Primeiro passo para clonar o repositório no IDE

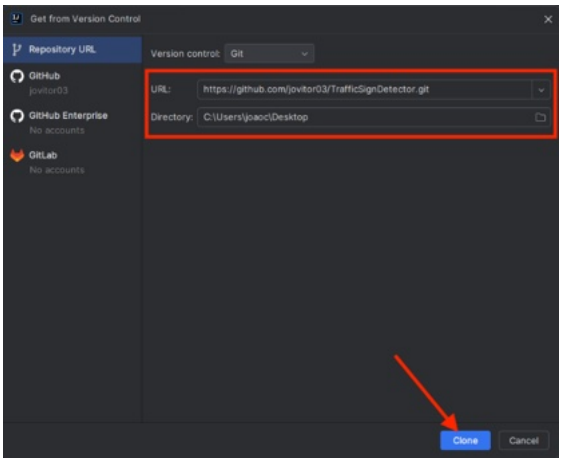


Figura 3 – Segundo passo para clonar o repositório no IDE

3. Instalar OpenCV – 4.9.0 em https://opencv.org/releases/:

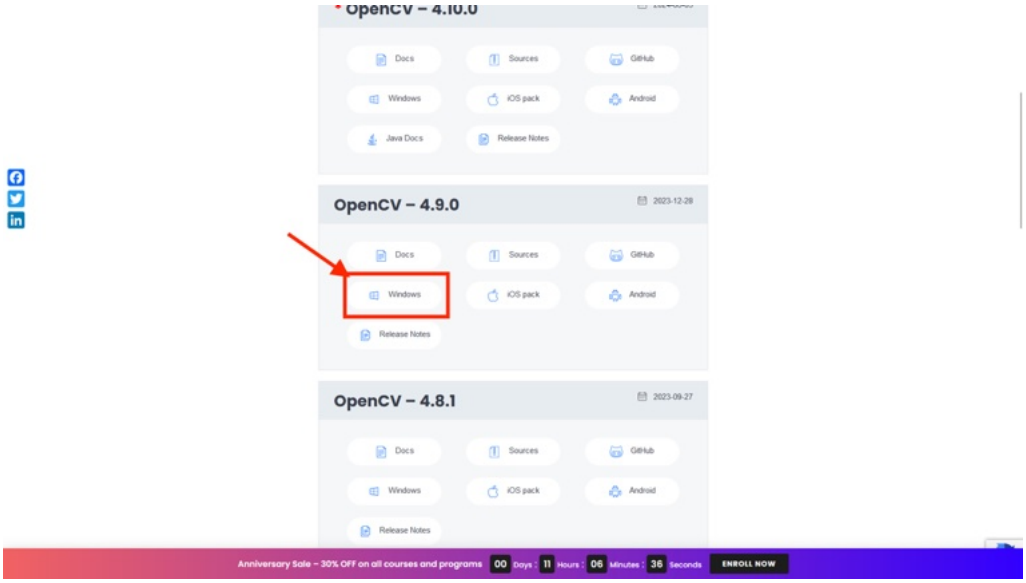


Figura 4 – Website para download do OpenCV - 4.9.0

4. Ir a File >> Project Structure...:

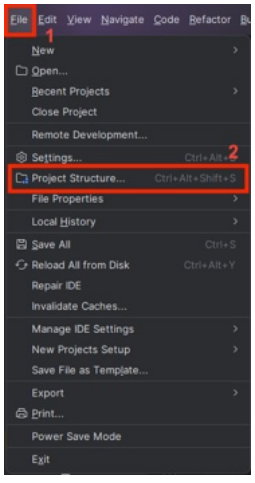


Figura 5 – Instruções para editar a estrutura do projeto

5. Em Project Settings, clicar em Modules e depois em Dependencies. Clicar posteriormente no + e em JARs or Directories...:

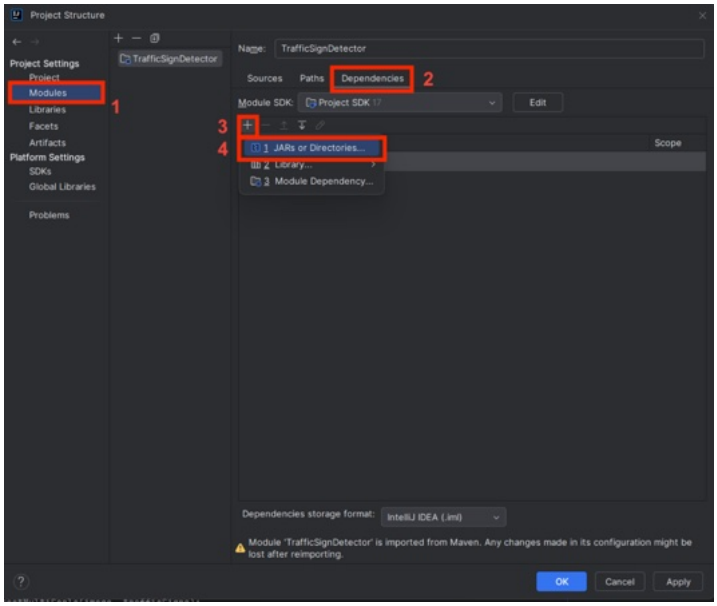


Figura 6 – Instruções para inserir e editar módulos e dependências

6. Procurar o ficheiro opencv-490.jar em opencv/build/java e adicionar em OK.

7. Clicar duas vezes no opencv-490.jar e depois no sinal de +:

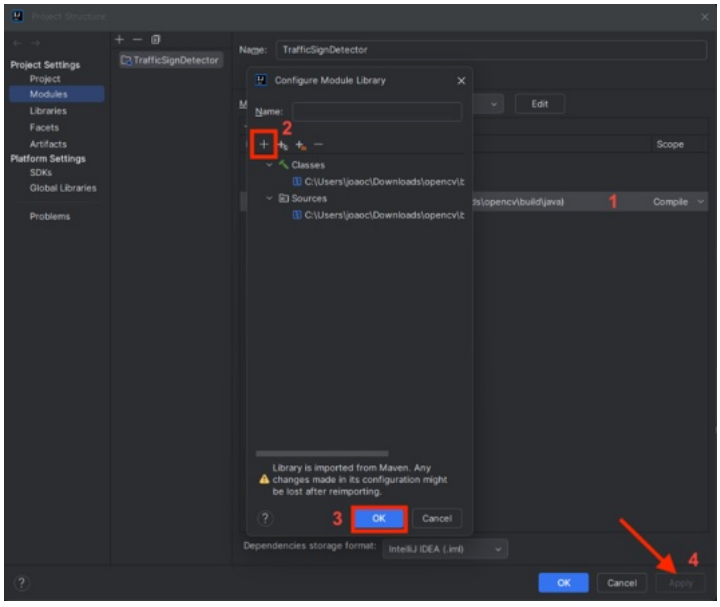


Figura 7 – Instruções para configurar módulos

8. Procurar o ficheiro opencv_java490.dll em opencv/build/java/x64 e adicionar em OK. Clicar novamente em OK. Por fim, clicar em Apply e OK.

Desta forma o projeto está pronto a ser utilizado.

# VIII. Implementação do sistema

## a. 1ª Fase

Numa primeira fase da implementação deste projeto, foram utilizados *cascade classifiers* (classificadores em cascata) para fazer a deteção dos sinais de trânsito.

A deteção de objetos utilizando classificadores em cascata baseados em características Haar é um método baseado em aprendizagem de máquina, onde uma função em cascata é treinada com muitas imagens positivas (contendo o objeto para deteção de interesse) e imagens negativas (sem o objeto de interesse). Após o treinamento, o classificador pode ser usado para detetar objetos em novas imagens.

O carregamento do classificador é feito através da classe CascadeClassifier que é utilizada para carregar um ficheiro .xml do classificador pré-treinado, que pode ser um classificador Haar ou LBP, através da função load. A deteção de objetos numa imagem estática ou num vídeo é realizada utilizando a função detectMultiScale, que retorna retângulos delimitadores para os objetos detetados.

O algoritmo baseia-se na extração de características Haar das imagens, calculadas utilizando retângulos pretos e brancos como filtros, onde cada característica é um valor obtido pela diferença entre a soma dos pixels dos retângulos branco e preto. Para otimizar os cálculos, utiliza-se a imagem integral, que permite calcular a soma dos pixels de qualquer retângulo na imagem original de forma eficiente. (*Cascade Classifier*, n.d.; Mittal, 2020)

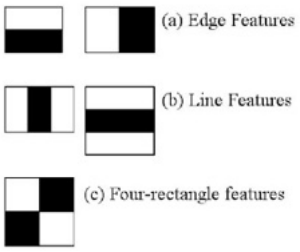


Figura 8 – Tipos de recursos Haar

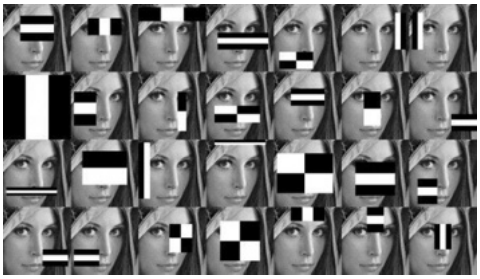

Figura 9 – Representação geral do treinamento de um classificador Haar

A seleção das características mais relevantes é feita com o algoritmo Adaboost, que escolhe as características com menor taxa de erro na classificação de imagens, combinando vários “classificadores fracos” em um “classificador forte”.

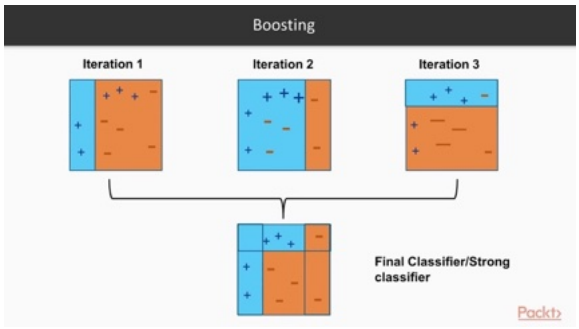


Figura 8 – Representação do algoritmo Adaboost

Para tornar o processo mais eficiente, os classificadores são organizados em uma cascata de estágios. Cada estágio consiste em um pequeno número de características. Se uma janela de imagem não passa por um estágio, ela é imediatamente descartada, evitando cálculos desnecessários. Apenas as janelas que passam por todos os estágios são consideradas como contendo o objeto de interesse.

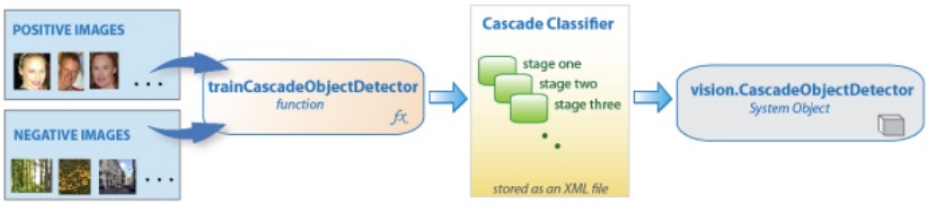


Figura 9 – Fluxograma da criação de classificadores em cascata

Na fase inicial do projeto, diversos classificadores em cascata foram aplicados na deteção de sinais de trânsito apenas em imagens. Tratam-se de imagens genéricas retiradas da internet. A imagem resultante é guardada no diretório /images com o nome <nome-antigo>_result, mantendo o formato original da imagem.

Os resultados obtidos foram satisfatórios, evidenciando uma boa performance dos classificadores em diversas condições e em diversos tipos de sinais de trânsito.


Figura 10 – Sinais de trânsito bem detetados em imagens com classificadores em cascata

No entanto, foi observado que algumas imagens apresentavam problemas específicos que impediam a identificação de alguns sinais ou qualquer sinal de trânsito.


Figura 11 – Sinais de trânsito mal detetados em imagens com classificadores em cascata

### b. 2ª Fase

Aproveitando a performance satisfatória dos classificadores em cascata nas imagens, foram realizados testes para avaliar a sua aplicação em vídeos, também retirados da internet. No primeiro teste de deteção de sinais de trânsito, o foco foi exclusivamente na deteção do sinal de STOP utilizando classificadores em cascata. Da mesma forma, o vídeo resultante é guardado no diretório /videos com o nome <nome-antigo>_result, mantendo o formato original do vídeo.

Inicialmente, a deteção do sinal de STOP foi bem sucedida. No entanto, observou-se que o classificador utilizado estava identificando outros objetos além do sinal de STOP pretendido, indicando assim a necessidade de refinamento da precisão na deteção de sinais.


Figura 12 – Tentativa de deteção do sinal de STOP num primeiro vídeo com um classificador em cascata


Figura 13 – Tentativa de deteção do sinal de STOP num segundo vídeo com um classificador em cascata

Para resolver o problema de deteção incorreta de objetos, foi implementada a combinação e utilização de diferentes classificadores em cascata para a deteção do sinal de STOP. Esta abordagem solucionou o problema inicial, eliminando a deteção de objetos que não eram sinais de STOP.


Figura 14 – Tentativa de deteção do sinal de STOP num primeiro vídeo com a combinação de vários classificadores em cascata


Figura 15 – Tentativa de deteção do sinal de STOP num segundo vídeo com a combinação de vários classificadores em cascata

No vídeo resultante com o sinal de STOP destacado, é possível observar que os erros de deteção de detetar objetos inexistentes ou diferentes do sinal de STOP deixaram de existir. No entanto, um pequeno problema ainda persiste: em alguns frames do vídeo, a combinação de diferentes classificadores em cascata não consegue detetar o sinal de trânsito. Apesar disso, para o observador do vídeo, o movimento dos frames torna evidente que o sinal de STOP está a ser corretamente identificado e destacado.

### c. 3ª fase

Nesta terceira fase, o objetivo inicial era utilizar os vídeos gravados pelo Meta Quest Pro durante viagens de carro para identificar sinais de trânsito, a fim de avaliar a qualidade de gravação e a qualidade dos vídeos capturados pelo headset. Contudo, após uma tentativa sem sucesso devido à baixa qualidade das câmeras e da funcionalidade Passthrough, o foco inicial do projeto foi alterado. A nova meta passou a ser a simples deteção de sinais de trânsito num percurso rodoviário na ilha da Madeira, utilizando vídeos gravados com um telemóvel. Inicialmente foi testado a deteção do sinal de STOP num vídeo gravado pelo telemóvel.


Figura 16 - Primeira tentativa de deteção do sinal de STOP num vídeo gravado através do telemóvel

Posteriormente, foi necessário a tentativa de deteção de outros sinais de trânsito. A eficiência dos classificadores em cascata disponíveis na web para fazer a deteção de sinais de trânsito, exceto o sinal de STOP, mostrou-se bastante reduzida e limitada. Para superar essa limitação, a solução encontrada seria fazer o treinamento de um classificador em cascata próprio. Ao invés da utilização de algoritmos e classificadores existentes, o treinamento personalizado de um classificador em cascata visa resolver o problema da precisão e aumentar a mesma na identificação de diversos sinais de trânsito.

Para o treinamento de classificadores em cascata, foi utilizado o software Cascade Trainer GUI (Version 3.3.1). Inicialmente foi tentado o treinamento de um

classificador em cascata que pudesse detetar qualquer tipo de sinais através de um dataset de mais de 12 mil imagens positivas e um grupo de 3 mil imagens negativas. Porém, de início o treinamento de um classificador em cascata para este enorme dataset, deu um erro inconcreto. (*Cascade Trainer GUI*, n.d.; Mykola, 2018)

```
OpenCV Error: Bad argument (Can not get new positive sample. The most possible reason is insufficient count of samples in given vec-file.
) in CvCascadeImageReader::PosReader::get, file D:\cv\opencv_3.2.0\sources_withTextModule\apps\traincascade\imagestorage.cpp, line 158
```

Figura 17 - Erro inconcreto do software para o dataset de 12 mil imagens positivas

A segunda tentativa de treinamento de um classificador em cascata, utilizando outro dataset, foi apenas a um tipo de sinal de trânsito. Neste caso ao sinal de trânsito de limite de velocidade de 60 km/h, tendo este sido escolhido aleatoriamente para testagem. Este não deu o erro que deu anteriormente por ter sido um número de imagens mais reduzido, porém o classificador em cascata testado no projeto, mostrou pior eficiência em comparação aos classificadores já existentes na web. (V N M Hemateja, 2021)


Figura 18 - Tentativa de deteção de um sinal de limite de velocidade (60 km/h) com classificador em cascata treinado no software

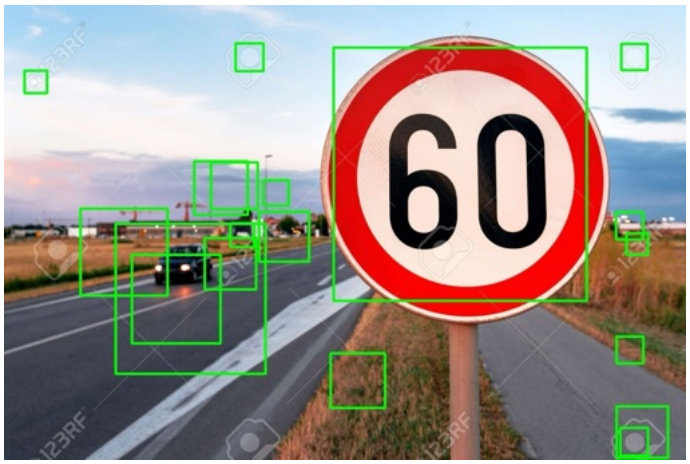


Figura 19 - Tentativa de deteção de um sinal de limite de velocidade (60 km/h) com classificador em cascata selecionado da web

Visto não ter encontrado algum estudo que tenha alguma vez determinado qual o número ideal de imagens positivas e imagens negativas, seria praticamente fazer tiro

ao alvo até chegar ao classificador em cascata ideal para cada sinal de trânsito ou até mesmo um classificador que pudesse detetar todos os sinais de trânsito. Para além disso, a qualidade das imagens positivas e negativas teriam influência no classificador em cascata.

### d. 4ª fase

Na quarta fase deste projeto, após ter sido estudado a eficiência de vários classificadores em cascata, foi escolhido um único classificador que apresentou melhores resultados em imagens e vídeos genéricos retirados da internet, já previstos neste estudo. Este classificador foi, não só escolhido por ter maior eficiência na deteção de sinais de trânsito, mas também por identificar menos objetos de forma errada em comparação a outros classificadores.


Figura 20 - Deteção de sinais de trânsito em imagens e vídeos genéricos utilizando o classificador em cascata escolhido pela sua melhor eficiência

Visto que cada classificador deteta o mesmo sinal de diversas formas diferentes e deteta objetos errados, a combinação do mesmo com outros classificadores foi impossível dado que a mesma não fazia a deteção de qualquer sinal de trânsito.

Para a realização dos testes com os vídeos gravados, foi necessário a divisão do vídeo completo em segmentos menores de 45 segundos, a fim de reduzir o tempo de processamento. Além disso, houve a necessidade de reduzir a qualidade do vídeo de 4k para 480p. Ambas as operações foram realizadas utilizando o software ffmpeg via consola de comandos. (*FFmpeg*, n.d.)

Passando então à testagem, do classificador em cascata escolhido, numa viagem de carro gravada através do telemóvel foi possível obter os seguintes resultados:

Figura 21 - Frames de uma boa deteção de sinais de trânsito num vídeo gravado pelo telemóvel durante a manhã

Figura 22 - Frames de uma má deteção de sinais de trânsito num vídeo gravado pelo telemóvel durante a manhã

Figura 23 - Frames de uma boa deteção de sinais de trânsito num vídeo gravado pelo telemóvel durante a tarde

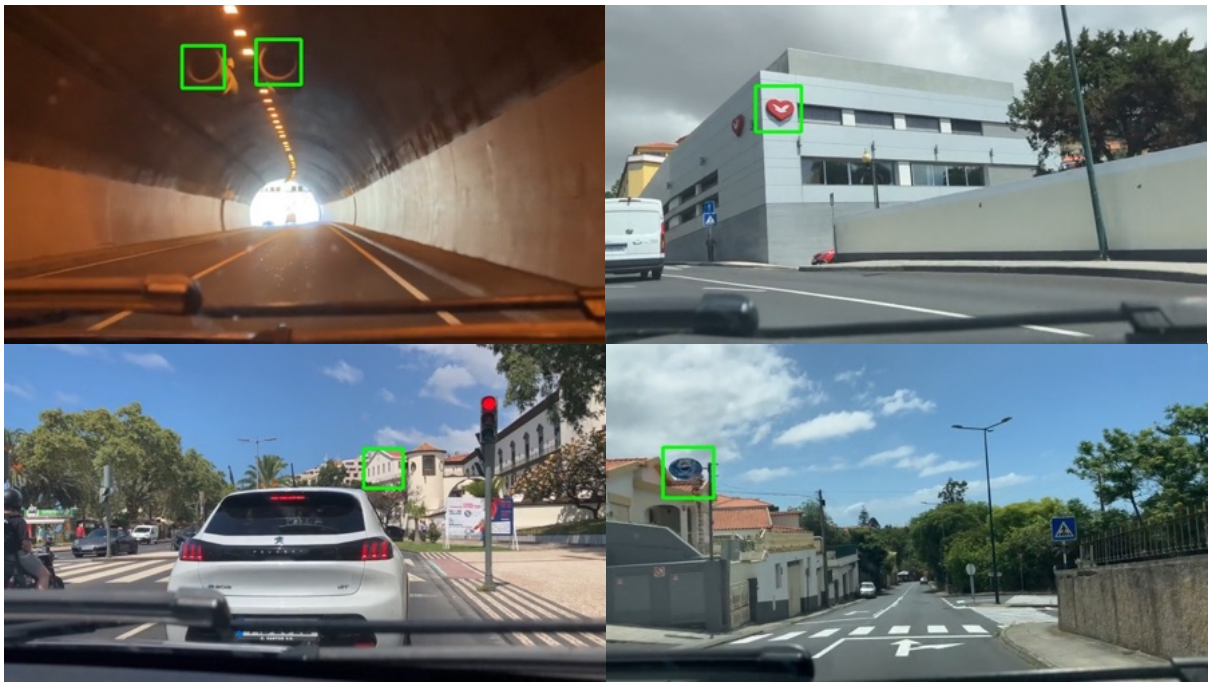

Figura 24 - Frames de uma má deteção de sinais de trânsito num vídeo gravado pelo telemóvel durante a tarde

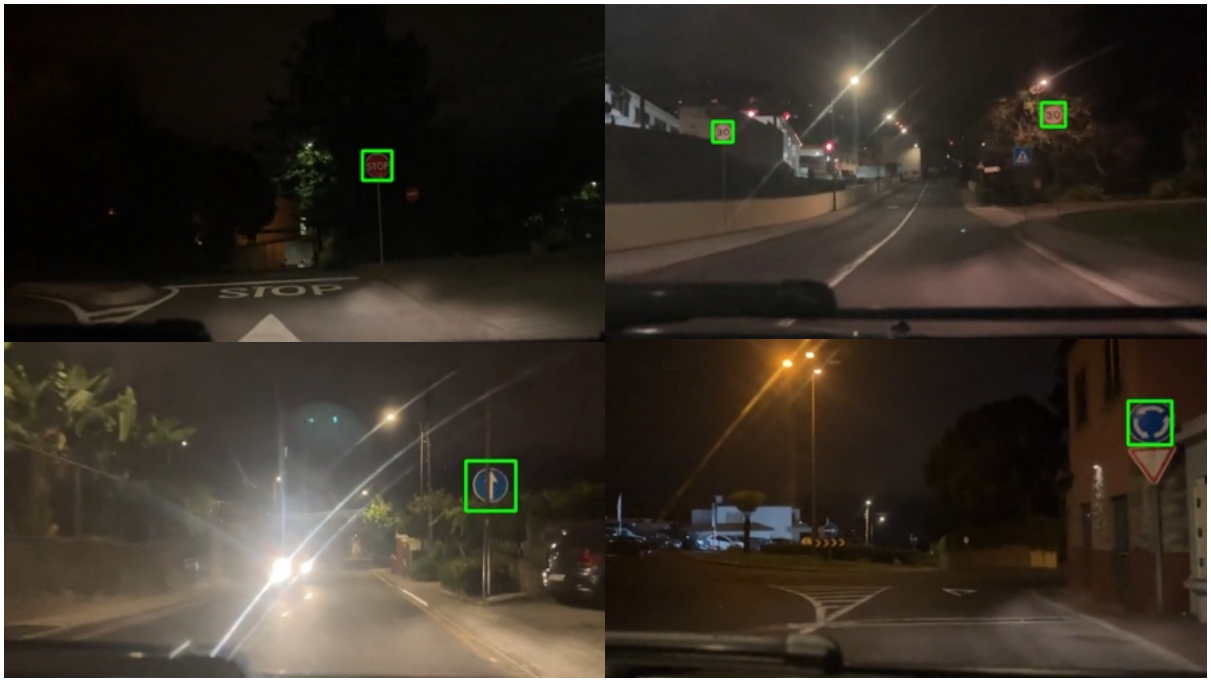


Figura 25 - Frames de uma boa deteção de sinais de trânsito num vídeo gravado pelo telemóvel durante a noite

Figura 26 - Frames de uma má deteção de sinais de trânsito num vídeo gravado pelo telemóvel durante a noite

Durante a análise dos resultados obtidos, observou-se que o classificador em cascata demonstra certa eficácia na deteção de sinais de trânsito. No entanto, foram identificados diversos sinais que não foram detetados pelo classificador, tais como sinais de perigo, de cedência de passagem e a maioria dos sinais de informação. Por outro lado, sinais como STOP, limites de velocidade e obrigações de rotunda foram consistentemente identificados ao longo dos vários vídeos testados.

É importante destacar que a eficácia da deteção de sinais se mostrou consistente nos vídeos gravados durante a noite em relação aos gravados durante o dia. Mesmo na ausência de luz natural, a deteção foi satisfatória quando os faróis do veículo e a iluminação dos postes de luz proporcionavam uma boa visibilidade.

Para complementar esta fase de avaliação, foi desenvolvido um código que direciona todos os frames nos quais foram detetados sinais de trânsito ou outros objetos relevantes para um diretório específico denominado "frames". Esse procedimento permitiu uma compreensão mais clara e concisa das deteções realizadas ao longo do processamento de cada vídeo.

### e. 5ª fase

Nesta última fase, foi criada uma interface que, para além da gravação de ficheiros em diretórios já definidos anteriormente, permite ao utilizador escolher de forma intuitiva a imagem ou vídeo que pretende testar a deteção de sinais de trânsito.

A interface de deteção de sinais de trânsito é projetada para facilitar a visualização e análise de imagens e vídeos previamente processados. Ela é composta por três botões principais que direcionam para diretórios específicos contendo uma variedade de imagens e vídeos relacionados.

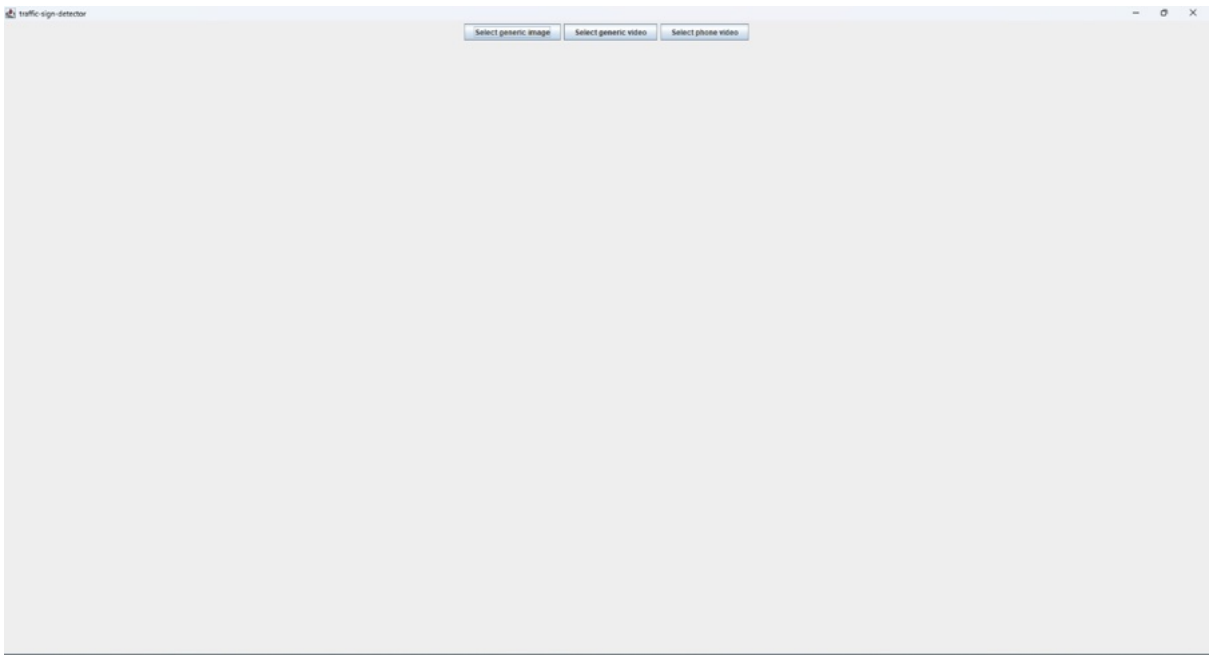


Figura 27 - Interface inicial quando o programa é corrido

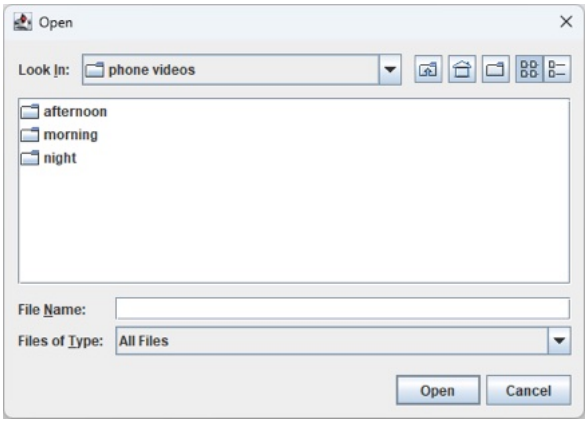


Figura 28 - Selecionador de ficheiros


Figura 29 - Visualização de uma imagem com o sinal de trânsito detetado com o caminho do ficheiro indicado

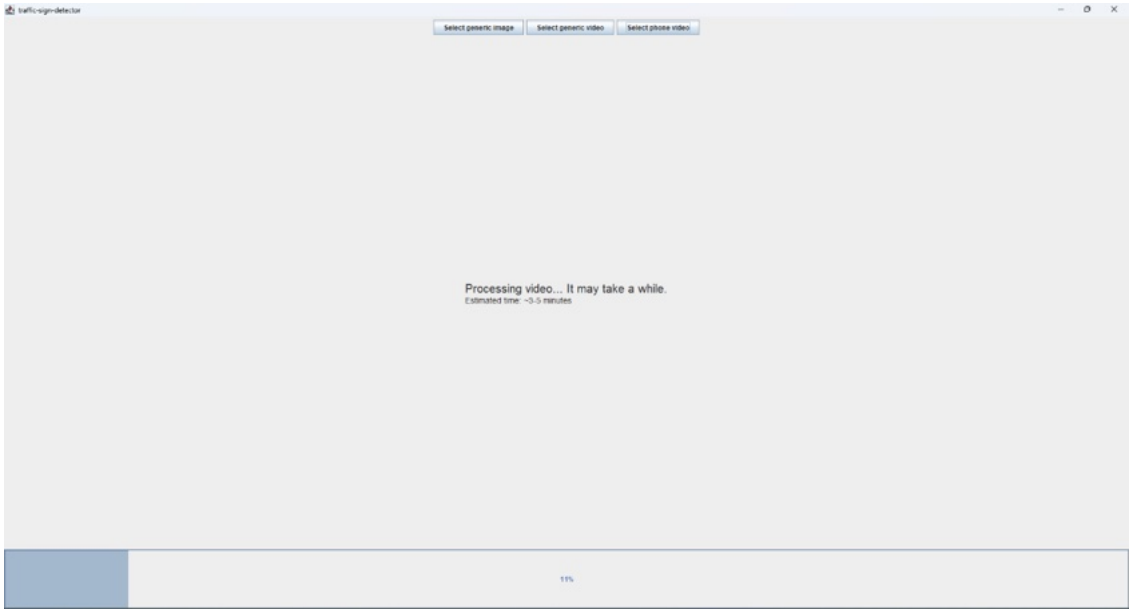


Figura 30 - Tela de processamento de vídeo com barra de progresso e tempo estimado


Figura 31 - Visualização de um vídeo com o sinal de trânsito detetado com o caminho do ficheiro indicado

## IX. Discussão do problema abordado

A realidade aumentada tem um potencial significativo para auxiliar os condutores no dia a dia, melhorando a segurança e a eficiência na condução. No entanto, dispositivos como o Meta Quest Pro ainda estão longe de serem viáveis para essa finalidade devido a limitações técnicas e questões de usabilidade.

O Meta Quest Pro, em particular, apresenta uma qualidade insuficiente na funcionalidade de Passthrough, essencial para a integração de realidade aumentada na condução. Além disso, o uso prolongado do headset pode causar desconforto, calor e até dores de cabeça, comprometendo ainda mais a segurança do condutor e aumentando os riscos na estrada.

Figura 32 - Exemplo de um frame de um vídeo gravado pelo Meta Quest Pro utilizando a funcionalidade Passthrough

Devido às limitações significativas dos headsets, o estudo avançou para a utilização de gravações via smartphone e classificadores em cascata. No entanto, os classificadores utilizados mostraram-se ineficientes, resultando em baixa precisão e potencialmente agravando ainda mais a insegurança na condução.

Para resolver a baixa eficiência dos classificadores em cascata, a utilização de métodos alternativos de deteção de objetos com OpenCV poderia ser considerada. O método com mais potencial é o YOLO (You Only Look Once), amplamente reconhecido por sua alta eficiência. Contudo, o YOLO possui grande compatibilidade com Python, e a sua implementação em Java, a linguagem escolhida para este estudo, apresentou desafios consideráveis, como a falta de recursos para configurar YOLO em Java ou em outras linguagens de programação. (Singh, 2021; *YOLO: Real-Time Object Detection*, n.d.)

No entanto, a solução apresentada e conseguida não deve de todo ser descartada, pois, se um classificador em cascata for bem treinado, pode ser utilizado de forma eficiente num projeto desenvolvido em Java ou até mesmo em outra linguagem de programação.

Este projeto destaca a necessidade de desenvolver soluções mais intuitivas e compatíveis com diversas linguagens de programação para permitir uma integração eficaz da realidade aumentada na condução diária.

Em suma, é possível concluir que os classificadores em cascata desenvolvidos em Java são em certa parte eficientes, porém ainda têm muito a evoluir para alcançar melhores resultados no futuro e poderem auxiliar e aprimorar a condução diária.

## X. Conclusão

Para concluir, a utilização do Meta Quest Pro e classificadores em cascata destaca tanto os avanços alcançados quanto as limitações encontradas ao longo do estudo. Inicialmente foi explorado o potencial do headset Meta Quest Pro para melhorar a segurança na condução através da realidade aumentada. No entanto, devido às limitações técnicas do dispositivo, especialmente relacionadas à funcionalidade Passthrough, foi necessário adaptar a abordagem para a utilização de vídeos gravados por um telemóvel.

A implementação inicial de classificadores em cascata mostrou-se promissora na deteção de sinais de STOP e limites de velocidade, proporcionando resultados satisfatórios durante a noite. Contudo, a precisão variou significativamente entre diferentes tipos de sinais de trânsito, com desafios na deteção de sinais de perigo, de cedência de passagem e de informação.

Para superar as limitações dos classificadores em cascata disponíveis na web, foram realizadas tentativas de treinamento personalizado. Apesar das dificuldades iniciais, o estudo demonstrou que, com o treinamento adequado e um dataset

apropriado, é possível melhorar significativamente a precisão de sinais de trânsito. No entanto, se for estudado qual a melhor forma de treinar um classificador em cascata, desde o número de imagens positivas e negativas até o número de etapas que o treinamento deve ter, a utilização de classificadores pode ser mais bem conseguida e mais eficiente.

Em resumo, embora os classificadores em cascata trabalhados em Java tenham mostrado eficácia em detetar alguns tipos de sinais de trânsito, há espaço para melhorias significativas, especialmente com a adoção de métodos mais avançados de deteção de objetos.

Concluindo, o estudo ressalta a importância de desenvolver soluções mais robustas e compatíveis com diversas plataformas para promover a utilização segura de tecnologias de realidade aumentada na condução diária, visando sempre aprimorar a segurança e a experiência dos condutores na estrada.

## XI. Referências


*Cascade Classifier*. (n.d.). https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

*Cascade Trainer GUI*. (n.d.). https://amin-ahmadi.com/cascade-trainer-gui/

*FFmpeg*. (n.d.). https://ffmpeg.org/

*Full-color Passthrough on Meta Quest*. (2024). https://www.meta.com/help/quest/articles/getting-started/getting-started-with-quest-pro/full-color-passthrough/

*IntelliJ IDEA*. (n.d.). https://www.jetbrains.com/idea/

*Meta Quest Pro*. (n.d.). https://www.meta.com/quest/quest-pro/

Mittal, A. (2020). *Haar Cascades, Explained*. Analytics Vidhya. https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d

Mykola. (2018). *GTSRB - German Traffic Sign Recognition Benchmark* [dataset]. https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

*OpenCV*. (n.d.). https://opencv.org/

*Record video with Meta Quest*. (2024). https://www.meta.com/help/quest/articles/in-vr-experiences/social-features-and-sharing/record-video-oculus/

Singh, A. (2021). *Top 6 Object Detection Algorithms*. Augmented Startups. https://medium.com/augmented-startups/top-6-object-detection-algorithms-b8e5c41b952f

*Use Passthrough on Meta Quest*. (2024). https://www.meta.com/help/quest/articles/in-vr-experiences/oculus-features/passthrough/

V N M Hemateja, A. (2021). *Traffic Sign Dataset—Classification* [dataset]. https://www.kaggle.com/datasets/ahemateja19bec1025/traffic-sign-dataset-classification

*YOLO: Real-Time Object Detection*. (n.d.). https://pjreddie.com/darknet/yolo/