

**Deteção, reconhecimento e alerta de objetos em vídeo**

Licenciatura em Engenharia Informática

Marco António da Silva Ferreira

Válter Santos Pedrosa

Leiria, junho de 2024



**Deteção, reconhecimento e alerta de objetos em vídeo**

Licenciatura em Engenharia Informática

Marco António da Silva Ferreira

Válter Santos Pedrosa

Trabalho de Projeto da unidade curricular de Projeto Informático realizado sob a orientação do Professor Doutor Paulo Costa e do Professor Doutor Filipe Neves

Leiria, junho de 2024

# Agradecimentos

Queremos agradecer aos professores orientadores do Instituto Politécnico de Leiria Professor Doutor Paulo Costa e Professor Doutor Filipe Neves pela disponibilidade de reunir todas as semanas de modo a certificar que o trabalho estava a ser desenvolvido dentro dos parâmetros pretendidos e por se mostrarem sempre disponíveis para resolver qualquer problema e esclarecer qualquer dúvida sobre a realização deste projeto.

De mencionar também o aluno do 2º ano de Comunicação e Media Emanuel José Brás Nunes da Escola Superior de Educação e Ciências Sociais do Instituto Politécnico de Leiria por ajudar a construir os elementos gráficos presentes na aplicação que se refletem numa experiência melhorada para o utilizador final.

# Resumo

*Este relatório apresenta o desenvolvimento de uma aplicação de vigilância com capacidade de associação de câmaras de vídeo onde são detetados objetos em tempo real e emitidos alertas quando detetados objetos imóveis relevantes. Este software recorre ao modelo de deteção YOLOV8 para alcançar uma deteção precisa e eficiente em vídeo.*

*O projeto foi motivado pela crescente relevância da Inteligência Artificial e as suas aplicações práticas em diversas áreas, como segurança, automação industrial, medicina e veículos autónomos. O objetivo principal foi criar uma ferramenta robusta e confiável para segurança, capaz de monitorizar áreas e notificar a presença de objetos suspeitos imóveis em tempo real.*

*Adotando a metodologia ágil Scrumban, desenvolvemos uma aplicação capaz de associar múltiplos dispositivos para deteção de objetos e geração de alertas. Os requisitos funcionais foram cuidadosamente definidos e atendidos com sucesso.*

*Os testes demonstraram que a aplicação opera de forma eficaz em diferentes cenários de vigilância, proporcionando uma resposta rápida e precisa. Durante o desenvolvimento, adquirimos conhecimentos valiosos sobre modelos de deteção de objetos e consolidamos a capacidade de desenvolver sistemas, o que resultou numa aplicação de desktop.*

*Para futuras melhorias, sugere-se a implementação da funcionalidade de adicionar novos objetos para deteção, além dos pré-treinados pelo modelo, permitindo uma maior personalização aos utilizadores. Esta evolução potencializaria mais ainda a utilidade e aplicabilidade da aplicação desenvolvida.*

**Palavras-chave:** deteção, objetos, yolo, reconhecimento, IA.

# Abstract

This report presents the development of a surveillance application with the capability to associate video cameras, where objects are detected in real-time and alerts are issued when relevant stationary objects are detected. This software uses the YOLOV8 detection model to achieve precise and efficient detection in video.

*The project was motivated by the growing relevance of Artificial Intelligence and its practical applications in various fields, such as security, industrial automation, medicine, and autonomous vehicles. The primary objective was to create a robust and reliable tool for security, capable of monitoring areas and notifying the presence of suspicious stationary objects in real-time.*

*By adopting the agile methodology Scrumban, we developed an application capable of associating multiple devices for object detection and alert generation. The functional requirements were carefully defined and successfully met.*

*The tests demonstrated that the application operates effectively in different surveillance scenarios, providing a quick and accurate response. During the development, we acquired valuable knowledge about object detection models and consolidated our ability to develop systems, resulting in a desktop application.*

*For future improvements, it is suggested to implement the functionality to add new objects for detection, beyond those pre-trained by the model, allowing greater customization for users. This evolution would further enhance the utility and applicability of the developed application.*

**Keywords:** detection, objects, yolo, recognition, AI

Índice

Trata-se de um elemento **obrigatório**. Nota: **o índice nunca figura do índice.**

[Agradecimentos iii](#_Toc171468291)

[Resumo iv](#_Toc171468292)

[Abstract v](#_Toc171468293)

[Lista de Figuras x](#_Toc171468294)

[Lista de tabelas xiii](#_Toc171468295)

[Lista de siglas e acrónimos xiv](#_Toc171468296)

[1. Introdução 1](#_Toc171468297)

[1.1. Enquadramento 1](#_Toc171468298)

[1.2. Motivação 1](#_Toc171468299)

[1.3. Objetivos 2](#_Toc171468300)

[1.4. Estrutura 2](#_Toc171468301)

[2. Análise teórica 4](#_Toc171468302)

[2.1. Representação de uma imagem digital 4](#_Toc171468303)

[2.2. Processamento de imagens 4](#_Toc171468304)

[2.2.1. Visualização 4](#_Toc171468305)

[2.2.2. Reconhecimento 5](#_Toc171468306)

[2.2.3. “Sharpening” e restauração 5](#_Toc171468307)

[2.2.4. Reconhecimento de padrões 5](#_Toc171468308)

[2.2.5. Recuperação 5](#_Toc171468309)

[2.3. Métodos de deteção de objetos 6](#_Toc171468310)

[2.3.1. Métodos tradicionais 6](#_Toc171468311)

[2.3.1.1. Viola-Jones Detector 6](#_Toc171468312)

[2.3.1.2. Histogram of Oriented Gradients Detector 7](#_Toc171468313)

[2.3.1.3. Deformable Part-based Model 7](#_Toc171468314)

[2.3.2. Métodos Deep Learning 7](#_Toc171468315)

[2.3.2.1. Redes Neuronais Convolucionais 8](#_Toc171468316)

[2.3.2.1.1. Camada convolucional 8](#_Toc171468317)

[2.3.2.1.2. Camada de pooling 10](#_Toc171468318)

[2.3.2.1.3. Camada fully-connected 11](#_Toc171468319)

[2.4. Classificadores de imagens 11](#_Toc171468320)

[2.4.1. VGG16 11](#_Toc171468321)

[2.4.2. ResNet 12](#_Toc171468322)

[2.5. Detetores de objetos 14](#_Toc171468323)

[2.5.1. One-stage detectors 14](#_Toc171468324)

[2.5.1.1. SSD 14](#_Toc171468325)

[2.5.1.2. YOLO 15](#_Toc171468326)

[2.5.1.3. RetinaNet 16](#_Toc171468327)

[2.5.1.4. DETR 17](#_Toc171468328)

[2.5.2. Two-stage detectors 18](#_Toc171468329)

[*2.5.2.1.* *R-CNN* 18](#_Toc171468330)

[2.5.2.2. Faster *R-CNN* 19](#_Toc171468331)

[3. Metodologias e tecnologias 20](#_Toc171468332)

[3.1. Metodologia de trabalho 20](#_Toc171468333)

[3.1.1. Método tradicional 20](#_Toc171468334)

[3.1.2. Metodologias ágeis 21](#_Toc171468335)

[3.1.2.1. Scrum 22](#_Toc171468336)

[3.1.2.2. XP 23](#_Toc171468337)

[3.1.2.3. Kanban 25](#_Toc171468338)

[3.1.2.4. Scrumban 25](#_Toc171468339)

[3.1.3. Metodologia de trabalho escolhida 26](#_Toc171468340)

[3.2. Funcionalidades da aplicação 27](#_Toc171468341)

[3.2.1. Análise de requisitos 27](#_Toc171468342)

[3.2.2. Mockups 28](#_Toc171468343)

[4. Implementação 33](#_Toc171468344)

[4.1. Escolha do modelo de deteção 33](#_Toc171468345)

[35](#_Toc171468346)

[4.2. Ambiente de desenvolvimento 35](#_Toc171468347)

[4.3. Arquitetura da aplicação 36](#_Toc171468348)

[4.4. Implementação dos requisitos funcionais 37](#_Toc171468349)

[4.4.1. RF1. Adicionar dispositivo 37](#_Toc171468350)

[4.4.2. RF2. Editar dispositivo 37](#_Toc171468351)

[4.4.3. RF3. Pausar captura 38](#_Toc171468352)

[4.4.4. RF4. Retomar captura 40](#_Toc171468353)

[4.4.5. RF5. Capturar live 40](#_Toc171468354)

[4.4.6. RF6. Expandir dispositivo 40](#_Toc171468355)

[4.4.7. RF7. Remover dispositivo 40](#_Toc171468356)

[4.4.8. RF8. Alterar layout 41](#_Toc171468357)

[4.4.9. RF9. Alterar delay 41](#_Toc171468358)

[4.4.10. RF10. Adicionar objetos a detetar 42](#_Toc171468359)

[4.4.11. RF11. Remover objetos a detetar 42](#_Toc171468360)

[4.4.12. RF12. Definir tempo para alerta de objeto 42](#_Toc171468361)

[4.4.13. RF13. Filtrar alertas 43](#_Toc171468362)

[4.4.14. RF14. Eliminar alertas 43](#_Toc171468363)

[4.4.15. RF15. Exibir detalhes de alerta 44](#_Toc171468364)

[4.4.16. RF16. Gravar configuração 44](#_Toc171468365)

[4.4.17. RF17. Importar configuração 45](#_Toc171468366)

[4.4.18. RF18. Configurar email para receber alerta 45](#_Toc171468367)

[4.4.19. RF19. Configurar telemóvel para receber alerta 46](#_Toc171468368)

[4.4.20. RF20. Nova Configuração 46](#_Toc171468369)

[5. Testes 47](#_Toc171468370)

[5.1. Teste de margem ideal para definir posição do objeto 47](#_Toc171468371)

[5.1.1. Condições do teste 47](#_Toc171468372)

[5.1.2. Resultados obtidos: 48](#_Toc171468373)

[5.1.2.1. Class “backpack” 48](#_Toc171468374)

[5.1.2.2. Class “sports ball” 50](#_Toc171468375)

[5.1.2.3. Class “Person” 51](#_Toc171468376)

[5.1.3. Análise dos resultados 52](#_Toc171468377)

[5.2. Teste da arquitetura para recolher frames de dispositivos 53](#_Toc171468378)

[5.2.1. Grab & Retrieve 53](#_Toc171468379)

[5.2.2. Parallel 53](#_Toc171468380)

[5.2.3. Open & Close 53](#_Toc171468381)

[5.2.4. Open & Close threading 53](#_Toc171468382)

[5.2.5. Condições de teste 53](#_Toc171468383)

[5.2.6. Resultados obtidos 54](#_Toc171468384)

[5.2.6.1. Grab & Retrieve 54](#_Toc171468385)

[5.2.6.2. Parallel 54](#_Toc171468386)

[5.2.6.3. Parallel 5 sec 55](#_Toc171468387)

[5.2.6.4. Parallel 10 sec 55](#_Toc171468388)

[5.2.6.5. Open & Close 56](#_Toc171468389)

[5.2.6.6. Open & Close threading 56](#_Toc171468390)

[5.2.7. Análise dos Resultados 57](#_Toc171468391)

[5.2.8. Conclusão 58](#_Toc171468392)

[5.3. Teste de limite de dispositivos 58](#_Toc171468393)

[5.3.1. Condições do teste 58](#_Toc171468394)

[5.3.2. Resultados obtidos 59](#_Toc171468395)

[5.3.3. Análise dos resultados 62](#_Toc171468396)

[5.4. Teste de geração de alerta 62](#_Toc171468397)

[5.4.1. Condições do teste 62](#_Toc171468398)

[5.4.2. Resultados obtidos 63](#_Toc171468399)

[5.4.3. Análise dos resultados 66](#_Toc171468400)

[6. Conclusão 67](#_Toc171468401)

[Bibliografia 68](#_Toc171468402)

# Lista de Figuras

Elemento a figurar, **quando aplicável**.

[Figura 2.1 - Aplicação filtro sobre imagem. Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 9](#_Toc171468404)

[Figura 2.2 - Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 9](#_Toc171468405)

[Figura 2.3 - Movimento do Kernel Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 10](#_Toc171468406)

[Figura 2.4 - Estrutura do VGG16, fonte: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918 12](#_Toc171468407)

[Figura 2.5 - Comparação das taxas de erro de teste e treino em CNN's com número de camadas diferente, fonte: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/ 13](#_Toc171468408)

[Figura 2.6 - Skip Connections, fonte: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/ 13](#_Toc171468409)

[Figura 2.7 – Algoritmo *SSD*, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 15](#_Toc171468410)

[Figura 2.8 – Algoritmo *YOLO*, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 16](#_Toc171468411)

[Figura 2.9 - Algoritmo RetinaNet, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 16](#_Toc171468412)

[Figura 2.10 - Arquitetura *DETR*, fonte: https://arxiv.org/pdf/2005.12872 17](#_Toc171468413)

[Figura 2.11 - Algoritmo de deteção *R-CNN*, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 18](#_Toc171468414)

[Figura 2.12 - Algoritmo de deteção Faster *R-CNN*, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries 19](#_Toc171468415)

[Figura 3.1 - Quadro kanban, fonte: https://www.kanbanchi.com/wp-content/uploads/2013/05/kanban-board-700x380.png 25](#_Toc171468416)

[Figura 3.2 - Mockup início 29](#_Toc171468417)

[Figura 3.3 - Mockup layout horizontal 29](#_Toc171468418)

[Figura 3.4 - Mockup adicionar dispositivo 30](#_Toc171468419)

[Figura 3.5 - Mockup adicionar dispositivo IP 30](#_Toc171468420)

[Figura 3.6 - Mockup definições dispositivo 31](#_Toc171468421)

[Figura 3.7 - Mockup emitir alerta para um dispositivo 31](#_Toc171468422)

[Figura 3.8 - Mockup dispositivo expandido 32](#_Toc171468423)

[Figura 3.9 - Mockup página alertas 32](#_Toc171468424)

[Figura 4.1 - Gráfico de comparação *YOLO*v7 vs. *YOLO*v5 vs. *YOLO*R and Vit Transformers, fonte: https://viso.ai/deep-learning/object-detection/ 34](#_Toc171468425)

[Figura 4.2 – Gráfico de comparação *YOlO*v7 e *YOLO*v8, fonte: https://docs.ultralytics.com/models/yolov8/ 35](#_Toc171468426)

[Figura 4.3 – Código editar dispositivo 38](#_Toc171468427)

[Figura 4.4 Código atualizar lista de objetos a detetar 38](#_Toc171468428)

[Figura 4.5 - Código stop\_lock 1 39](#_Toc171468429)

[Figura 4.6 - Código stop\_lock 2 39](#_Toc171468430)

[Figura 4.7 – Limpeza dos dados do dispositivo removido 41](#_Toc171468431)

[Figura 4.8 - Stacked Layout 41](#_Toc171468432)

[Figura 4.9 - Filtragem dos alertas 43](#_Toc171468433)

[Figura 4.10 - Código remover alerta 44](#_Toc171468434)

[Figura 4.11 - Guardar configurações 45](#_Toc171468435)

[Figura 4.12 – Importar configuração 45](#_Toc171468436)

[Figura 5.1 - Gráfico cantos backpack a 5M 48](#_Toc171468437)

[Figura 5.2 - Gráfico cantos backpack a 10M 49](#_Toc171468438)

[Figura 5.3 - Gráfico cantos backpack a 15M 49](#_Toc171468439)

[Figura 5.4 - Gráfico cantos bola a 5M 50](#_Toc171468440)

[Figura 5.5 - Gráfico cantos bola a 10M 50](#_Toc171468441)

[Figura 5.6 - Gráfico cantos bola a 12M 51](#_Toc171468442)

[Figura 5.7 - Gráfico cantos person a 5M 51](#_Toc171468443)

[Figura 5.8 - Gráfico cantos person a 35M 52](#_Toc171468444)

[Figura 5.9 - Gráfico cantos person a 70M 52](#_Toc171468445)

[Figura 5.10 - Gráfico performance parallel 54](#_Toc171468446)

[Figura 5.11 - Gráfico performance parallel 5 segundos 55](#_Toc171468447)

[Figura 5.12 - Gráfico performance parallel 10 segundos 55](#_Toc171468448)

[Figura 5.13 - Gráfico performance open & close 56](#_Toc171468449)

[Figura 5.14 - Gráfico performance open & close threading 56](#_Toc171468450)

[Figura 5.15 – Gráfico stress LIVE 59](#_Toc171468451)

[Figura 5.16 - Gráfico stress 1 segundo 60](#_Toc171468452)

[Figura 5.17 - Gráfico stress 5 segundos 61](#_Toc171468453)

[Figura 5.18 - Gráfico stress 10 segundos 61](#_Toc171468454)

[Figura 5.19 - Lista alertas do objeto "chair" a 5 metros 63](#_Toc171468455)

[Figura 5.20 - Lista alertas do objeto "chair" a 10 metros 64](#_Toc171468456)

[Figura 5.21 - Lista alertas do objeto "chair" a 15 metros 64](#_Toc171468457)

[Figura 5.22 - Lista alertas do objeto "car" a 5 metros 65](#_Toc171468458)

[Figura 5.23 - Lista alertas do objeto "car" a 10 metros 65](#_Toc171468459)

[Figura 5.24 - Lista alertas do objeto "car" a 15 metros 66](#_Toc171468460)

# Lista de tabelas

Elemento a figurar, **quando aplicável**.

[Tabela 1- Tabela comparativa entre metodologias tradicionais e ágeis 22](#_Toc171468461)

[Tabela 2 - Tabela comparativa entre Scrum e *XP*, fonte: https://www.geeksforgeeks.org/difference-between-scrum-and-xp/ 25](#_Toc171468462)

[Tabela 4 – Tabela comparativa entre modelos de deteção 33](#_Toc171468463)

[Tabela 5 – Tabela comparativa dos resultados obtidos do teste das várias arquiteturas 57](#_Toc171468464)

[Tabela 6 – Tabela representativa dos limites dos parâmetros 57](#_Toc171468465)

# Lista de siglas e acrónimos

Elemento a figurar, **quando aplicável**.

|  |  |
| --- | --- |
| CNN  CV  DETR  DL  DPM  FPN  FPS  HOG  IA  IoU | Convolutional Neural Network  Computer Vision  Detection Transformer  Deep Learning  Deformable Part-based Model  Feature Pyramid Network  Frames Por Segundo  Histogram of Oriented Gradients  Inteligência Artificial  Intersection of Union |
| ML | Machine Learning |
| MRI  NLP  PO  R-CNN  RoI  RPN  SM | Magnetic Resonance Imaging  Neural Language Processing  Produt Owner  Region-Based Convolutional Neural Networks  Region of Interest  Region Proposal Network  Scrum Master |
| SSD  SVM  XP  YOLO | Single Shot Detector  Support Vector Machine  Extreme Programming  You Only Look Once |
|  |  |
|  |  |
|  |  |
|  |  |

# Introdução

Paragrafo de intorucção do capitulo

## Enquadramento

Nos últimos anos, tornou-se evidente o crescimento exponencial e sem precedentes da Inteligência Artificial (IA), especialmente desde o fim do ano de 2022 com o aparecimento do Chat GPT. Uma das áreas que mais beneficiou o aparecimento de abordagens baseadas em *IA* foi a área de Computer Vision (CV) e deteção de objetos uma vez que estas adquiriram maior relevância com os modelos baseados em Deep Learning (DL) que é uma das muitas aplicações da *IA*.

A deteção de objetos é uma das principais técnicas da área de Computer Vision e permite a identificação e localização de objetos específicos em imagens ou vídeos. Esta capacidade de deteção tem uma variedade de aplicações, tais como: segurança e vigilância, automação industrial, veículos autónomos, aplicações médicas, entre outras. A capacidade de deteção de objetos permitiu avanços enormes nas áreas referidas, nomeadamente na evolução da condução de veículos autónomos (sem intervenção humana), no diagnóstico de doenças através da identificação de padrões em exames médicos difíceis de interpretar por humanos que consequentemente permitem atingir um diagnóstico prematuro e garantir uma maior probabilidade de tratamento com sucesso. Esta capacidade também se reflete em avanços significativos na área da segurança, uma vez que viabiliza sistemas de vigilância modernos que, através da deteção de objetos, monitorizam áreas em tempo real, identificam atividades suspeitas e proporcionam a emissão de alertas quando é detetada alguma anomalia.

É neste contexto de vigilância que se enquadra este projeto que tem como objetivo identificar objetos específicos que se encontrem imóveis tempo suficiente para serem considerados suspeitos. Na secção 1.3 exploramos os objetivos com maior detalhe.

## Motivação

Este projeto enquadra-se na UC de Projeto Informático onde tínhamos diversas possibilidades de propostas de projeto para escolher. A nossa primeira escolha foi precisamente este tema com o nome “Deteção, reconhecimento e alerta de objetos em vídeo”. A Inteligência Artificial, por ser uma área muito recente, tem um vasto campo ainda para ser explorado e por nos considerarmos curiosos e atuais queremos sempre acompanhar as novas tecnologias e metodologias, e a *IA* é uma delas. A área de foco deste projeto é a Computer Vision que é um dos muitos ramos da *IA* e este contacto com tecnologia permitiu-nos compreender como funcionam os métodos de deteção de objetos, o que esperar destes métodos e como trabalhar com os mesmos. MUDAR LINGUAGEM PARA 3 PESSOA

## Objetivos

Este projeto insere-se na categoria de deteção de objetos, tendo como objetivo principal o desenvolvimento de uma aplicação de vigilância capaz de associar múltiplos dispositivos a detetar objetos de forma automática e precisa enquanto emite alertas à entidade responsável quando necessário. Existe uma infinidade de aplicações para um software deste género, especialmente na segurança tanto de espaços públicos como privados.

Pretende-se com este projeto permitir que o utilizador monitorize áreas de interesse, quer seja para uso pessoal a vigiar espaços privados ou para o ramo empresarial onde se vigiem espaços comuns onde é exigida uma segurança reforçada. Esta aplicação surge como uma excelente ferramenta para auxílio na tarefa de vigilância uma vez que permite ao utilizador receber alertas importantes com base nas configurações definidas mesmo que este não se encontre no computador no momento de alerta ou esteja focado noutras câmaras e perca ações suspeitas.

Para o desenvolvimento da aplicação foi adotada uma metodologia ágil bastante conhecida e utilizada atualmente denominada Scrumban. As metodologias ágeis têm um grande foco na área de desenvolvimento de software e caracterizam-se pela flexibilidade, trabalho em equipa e colaboração com o cliente. O planeamento adaptativo é a principal característica desta abordagem.

## Estrutura

Este relatório conta com um capítulo de análise teórica onde são abordados os temas subjacentes aos componentes criados com maior profundidade para uma melhor compreensão do projeto desde o conceito mais básico como a maneira que um computador interpreta uma imagem até os temas mais complexos nomeadamente os vários métodos que existem para a deteção de objetos.

De seguida são apresentadas as metodologias de trabalho mais utilizadas atualmente, identificam-se as duas principais abordagens, metodologias tradicionais e ágeis. Após análise destas metodologias apresenta-se o método de trabalho escolhido e devida justificado.

O tópico seguinte reflete os requisitos funcionais que foram idealizados para a aplicação. Estes requisitos foram pensados de maneira a atender às necessidades mais críticas do utilizador. Para complementar os requisitos escritos, apresentamos mockups que são representações visuais e detalhadas que refletem o funcionamento desejado para a aplicação, o que permite uma melhor compreensão dos requisitos.

O capítulo que se segue temo o propósito de explicar a implementação dos requisitos formulados no tópico anterior. Estes requisitos são analisados um a um e numa abordagem mais técnica do que funcional. Além dos requisitos, é explicado ainda o ambiente de desenvolvimento que foi escolhido e a sua justificação. Por último, é exposta a arquitetura da aplicação, os elementos que a compõem e como os diversos componentes comunicam.

De seguida, encontra-se um capítulo de testes que permite retirar algumas conclusões sobre os cenários explorados. Existem testes preliminares, que serviram como base para escolher determinados parâmetros do desenvolvimento da aplicação e testes de sistema realizados à aplicação no fim desta ser desenvolvida para testar o seu comportamento perante os cenários escolhidos.

Para finalizar o relatório encontra-se o capítulo de conclusões onde se verifica o sucesso dos objetivos estipulados, com o desenvolvimento de uma aplicação que mostra alta precisão nos testes realizados, que opera em tempo real e responde de forma eficaz aos diversos cenários de vigilância a que foi proposta. Ao longo do projeto, foi possível adquirir conhecimentos significativos sobre os modelos de deteção de objetos e consolidar a capacidade de desenvolver sistemas desktop que atendam às necessidades dos utilizadores. Apresentam-se ainda sugestões para possíveis futuras melhorias que não foram atendidas no tempo proposto para o desenvolvimento do projeto.

# Análise teórica

Neste capítulo vamos explorar os conceitos teóricos necessários para compreender por completo a teoria em que se baseia esta aplicação, como funciona e perceber as bases que justificam as escolhas tomadas durante o desenvolvimento da aplicação.

## Representação de uma imagem digital

Uma imagem é representada pelas suas dimensões (altura e largura) consoante o número de pixéis. Um pixel é um ponto na imagem que tem uma tonalidade, opacidade ou cor específica. Normalmente é representado de uma das seguintes maneiras:

* Grayscale – Um pixel é um inteiro compreendido entre 0 e 255 em que 0 é completamente preto e 255 completamente branco.
* RGB – Um pixel é composto por 3 inteiros compreendidos entre 0 e 255 em que cada um representa a intensidade de vermelho (R), verde (G) e azul (B).
* RGBA – É uma extensão de RGB que acrescenta um campo alfa, que reflete a opacidade do pixel.

## Processamento de imagens

O processamento de imagens consiste no processo de transformar uma imagem numa forma digital e executar determinadas operações através de algoritmos e modelos matemáticos para extrair informação útil sobre a mesma. Vamos explorar os 5 tipos principais de processamento de imagens.

### Visualização

Este primeiro tipo de processamento envolve a conversão de dados na imagem para um formato visual que possa ser facilmente interpretado por humanos. Existem várias técnicas possíveis a ser utilizadas, nomeadamente melhoramento de contraste, segmentação ou criação de imagens tridimensionais a partir de dados de imagens bidimensionais. Um bom exemplo de aplicação desta técnica é na medicina, onde são aplicadas técnicas de visualização em Magnetic Resonance Imaging (MRI) para obter um diagnóstico ou explorar conectividade no cérebro de um indivíduo.

### Reconhecimento

No contexto de processamento de imagens, o reconhecimento refere-se à identificação e categorização de objetos ou padrões específicos dentro de uma imagem. As aplicações desta categoria podem ser a deteção de rostos humanos, identificação de caracteres de texto ou reconhecimento de objetos em cenas complexas. Para melhorar a precisão e a eficácia dos sistemas de reconhecimento de imagens são utilizadas frequentemente técnicas avançadas de Machine Learning (ML) e Redes Neurais Convolucionais (CNNs do inglês Convolutional Neural Networks). É neste tipo de processamento de imagem que este projeto se insere.

### “Sharpening” e restauração

“Sharpening” e restauração de imagens são processos usados para melhorar a qualidade de imagens que tenham sido distorcidas, quer seja por ruído, desfoque ou compressão. “Sharpening”, por um lado, foca em realçar detalhes e bordas na imagem, tornando os elementos mais nítidos e claros. A restauração, por outro, tenta recuperar a imagem original corrigindo distorções e removendo ruídos. Estes processos utilizam, geralmente, técnicas matemáticas e algoritmos de filtragem.

### Reconhecimento de padrões

O reconhecimento de padrões envolve a análise de imagens para identificar e categorizar padrões ou características específicas que podem ser repetidas ou significativas. Identificação de texturas, formas, ou regularidades em conjuntos de dados de imagens são alguns exemplos das capacidades desta categoria. Esta área é fundamental em aplicações como análise de imagens médicas, inspeção de qualidade de produtos em fábricas e análise de solo em imagens de satélite. Para treinar sistemas que sejam capazes de reconhecer e interpretar padrões com alta precisão são frequentemente utilizados algoritmos de *ML*.

### Recuperação

A recuperação é uma técnica de *CV* que envolve encontrar imagens semelhantes a uma original numa grande base de dados. A recuperação de imagens é crucial em áreas como base de dados multimédia, sistemas de vigilância e arquivos digitais, onde é essencial o acesso rápido e preciso a imagens relevantes.

Existem duas técnicas para realizar a recuperação de imagens, CBIR e TBIR. CBIR significa Content-Based Image Retrieval e retrata que o modelo compara a imagem fornecida com as da base de dados através do conteúdo visual e as suas características intrínsecas, ou seja, analisa a cor, textura, forma entre outros atributos. TBIR significa Text-Based Image Retrieval e esta abordagem baseia-se em descrições textuais e metadados associados às imagens para realizar a procura. As imagens são indexadas com base em palavras-chave, legendas, anotações ou outras informações textuais fornecidas.

## Métodos de deteção de objetos

Como expresso anteriormente, este projeto enquadra-se na categoria de reconhecimento dentro dos vários tipos de processamento de imagens. Com este projeto pretendemos identificar objetos selecionados em frames extraídos de câmaras conectadas à nossa aplicação. Existem dois tipos principais de processamento de imagens em Computer Vision: classificação de imagens e deteção de objetos. Os algoritmos de classificação apenas identificam a melhor classe que caracteriza a imagem designando uma legenda à imagem inteira baseado no seu conteúdo predominante, enquanto os algoritmos de deteção identificam quantos objetos de uma determinada categoria estão na imagem e onde se encontra cada objeto. Existem vários métodos relacionados à deteção de objetos, sendo que estes podem pertencer a abordagens tradicionais ou Deep Learning.

### Métodos tradicionais

Os métodos tradicionais na deteção de objetos envolvem geralmente extração de características e algoritmos de *ML* como Viola-Jones Detector, Histogram of Oriented Gradients (HOG) Detector e Deformable Part-based Model (DPM). Estas abordagens tradicionais contribuíram significativamente para o desenvolvimento de métodos de deteção de objetos, oferecendo “insights” na consideração de diferentes características e desafios na deteção de objetos.

### Viola-Jones Detector

Este algoritmo utiliza "Haar-like features" que são características formadas por elementos retangulares que capturam diferenças na intensidade dos pixéis. Estas características são combinadas em secções em cascata, onde cada estágio descarta rapidamente as áreas da imagem que não contêm elementos de interesse. As localizações que passam por todos os estágios são consideradas como contendo um objeto. A eficiência deste método reside na capacidade de filtrar rapidamente a maioria das áreas negativas (sem objeto), analisando detalhadamente apenas as áreas promissoras, o que o torna adequado para aplicações em tempo real.

### Histogram of Oriented Gradients Detector

Este método funciona calculando gradientes de intensidade em células locais, que capturam orientações de bordas e contornos na imagem. Para cada célula são gerados histogramas de gradientes, que são normalizados em blocos sobrepostos para garantir robustez contra variações de brilho e contraste. O vetor de atributos resultante representa a distribuição das arestas em toda a imagem e é utilizado como entrada para um classificador, como o Support Vector Machine (SVM), para detetar objetos específicos. O *SVM* é um tipo de algoritmo de *ML* supervisionado usado para classificação e regressão. Neste contexto, o *SVM* é utilizado para identificar as classes de interesse, distinguindo entre a presença ou ausência de um objeto específico com base nos vetores de características gerados pelo *HOG*.

### Deformable Part-based Model

*DPM* é um método avançado de deteção de objetos que modela um objeto como uma coleção de partes interconectadas, onde cada parte pode mover-se ligeiramente em relação às outras. Este método utiliza um modelo hierárquico em que a aparência de cada parte e as relações espaciais entre as partes são capturadas por descritores de características, como o *HOG*. O *DPM* envolve um processo de aprendizagem que determina a melhor configuração das partes e as suas deformações permitidas, criando um modelo flexível que pode lidar com variações de pose, escala e forma do objeto. Esta técnica é particularmente eficaz para a deteção de objetos em cenários complexos, onde a rigidez dos modelos tradicionais anteriores pode falhar.

### Métodos Deep Learning

*ML* é um ramo de Inteligência Artificial e assenta na premissa que a máquina adquire dados e aprende com eles. Em vez de escrever código específico num programa com instruções específicas para completar uma tarefa, *ML* permite que o sistema aprenda e reconheça padrões por si próprio e faça previsões.

Deep Learning é um campo especializado de *ML* assente em *CNN*s que envolve aprendizagem em diferentes estágios. Este método é caracterizado pela sua estrutura em camadas, cada camada adicionando conhecimento à anterior. Deste modo, o Deep Learning treina a máquina para tentar espelhar o funcionamento natural do cérebro humano.

Os métodos de *DL* têm ganho maior atenção e sucesso nos últimos anos. As Redes Neuronais Convolucionais são uma tecnologia chave no *DL* para deteção de objetos. Estas conseguem aprender a extrair automaticamente características relevantes de imagens e aprender padrões complexos que são representativos de diferentes categorias de objetos. Existem dois tipos de arquiteturas de deteção de objetos baseadas em *CNN*s, one-stage detectors e two-stage detectors.

### Redes Neuronais Convolucionais

Rede Neuronal Convolucional é um algoritmo de Deep Learning. O pré-processamento exigido numa *CNN* é muito menor quando comparado a outros algoritmos de classificação. Enquanto em métodos primitivos os filtros são desenhados manualmente (hand-engineered), as *CNN*s, com treino suficiente, têm a habilidade de aprender estes filtros/características. A arquitetura de uma *CNN* é análoga à do padrão de conectividade entre os neurónios no cérebro Humano e foi inspirada pela organização do córtex visual.

As *CNN*s diferem das outras redes neuronais pela sua performance superior com entradas de vídeo, áudio ou fala. Estas redes dispõe de três camadas principais: camada convolucional, camada de pooling e camada fully-connected.

### Camada convolucional

Esta é a primeira camada de uma rede convolucional, é onde ocorre a maior parte do processamento e tem como objetivo reduzir as imagens numa forma mais leve computacionalmente de modo a ser mais fácil de processar sem perder características críticas que permitam manter uma boa precisão.

Na Figura 2.1 podemos verificar este processo de convolução sobre uma imagem com apenas um canal. Está representado, a verde, uma imagem de entrada de dimensões 5x5x1 e a amarelo o filtro ou kernel (k) aplicado à imagem representado pela seguinte matriz ([1,0,1]; [0,1,0]; [1,0,1]). À direita podemos ver o output a vermelho que é a imagem após o processo de convolução.

Uma imagem com texto, captura de ecrã, quadrado

Descrição gerada automaticamente

Figura 2.1 - Aplicação filtro sobre imagem. Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

No caso de imagens com vários canais, como é o caso do RGB que possui três canais, a convolução da imagem é efetuada como mostra a Figura 2.2.

Uma imagem com texto, diagrama, captura de ecrã, Esquema

Descrição gerada automaticamente

Figura 2.2 - Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

A Figura 2.3 representa o movimento em que o filtro opera sobre uma imagem. O filtro move-se da esquerda para a direita até atingir toda a largura passando para baixo para a esquerda e repete o processo até toda a imagem ser percorrida.

Uma imagem com esboço, design, cubo

Descrição gerada automaticamente

Figura 2.3 - Movimento do Kernel Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

As *CNN*’s não necessitam de estar limitadas a uma única camada convolucional. Por norma, a primeira camada convolucional é responsável por capturar características de baixo nível como arestas ou cor. Com camadas adicionais, a arquitetura adapta-se para reconhecer características mais detalhadas possibilitando uma rede com uma compreensão completa das imagens.

### Camada de pooling

Na camada de pooling ocorre a redução da dimensionalidade, reduzindo o número de parâmetros no input. Tal como a camada convolucional, esta camada varre um filtro sobre toda a matriz de entrada com a diferença que o filtro não contém qualquer peso. Ao invés, o filtro aplica uma função de agregação aos valores dentro do campo recetivo, preenchendo o vetor de saída. Existem dois tipos de pooling principais:

* **Max pooling**: Nesta abordagem o filtro escolhe o pixel com maior valor e envia-o para o vetor de saída.
* **Average pooling**: Neste método, o filtro calcula a média dos valores dos pixéis no campo recetivo e envia o valor para o vetor de saída

Embora se perca muita informação nesta camada, acaba por ser vantajoso para a *CNN* uma vez que reduz a complexidade, melhorando a eficiência e limita o risco de overfitting. Entende-se por overfitting o fenómeno que pode ocorrer em *CNN*s quando o modelo se ajusta muito bem ao conjunto de treino e perde a capacidade de generalizar para novos exemplos que não constem no dataset de treino. Por exemplo, o modelo reconhece imagens específicas no conjunto de treino em vez de padrões gerais que lhe permitam detetar novas imagens.

### Camada fully-connected

Os valores dos pixéis das imagens de entrada não estão diretamente relacionados à camada de saída em camadas parcialmente conectadas. Por outro lado, nesta camada, como o nome sugere, as camadas estão totalmente conectadas (fully-connected) pelo que cada nó na camada de saída está diretamente conectado a um nó da camada anterior.

Esta camada executa a tarefa de classificação com base nas características extraídas através das camadas anteriores e os seus diferentes filtros. Enquanto as camadas convolucionais e de pooling tendem a usar funções ReLu, esta última normalmente recorre a uma função de ativação softmax para classificar apropriadamente os inputs originando uma probabilidade entre 0 e 1.

## Classificadores de imagens

As duas arquiteturas de *CNN* que serão agora apresentadas inserem-se na categoria de classificadores e não de deteção de objetos. No entanto é importante abordá-las uma vez que são utilizadas frequentemente em métodos de deteção que vamos abordar mais a frente, principalmente na etapa de extração de mapas de características de uma imagem.

### VGG16

VGG16 é uma arquitetura de uma *CNN* desenvolvida pelo Visual Geometry Group e é considerada um dos melhores métodos de *CV*. A estrutura desta rede conta com 13 camadas convolucioanais, divididas em blocos onde cada bloco é seguido por uma camada de pooling do tipo Max Polling, contando com 5 destes elementos. Para concluir, a arquitetura conta com 3 camadas fully connected como se pode ver na Figura 2.4.

Uma imagem com texto, Tipo de letra, file, captura de ecrã

Descrição gerada automaticamente

Figura 2.4 - Estrutura do VGG16, fonte: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

O 16 no nome refere-se às 16 camadas, 13 convolucionais mais 3 fully connected, com peso (que possibilitam aprendizagem) das 21 que constituem o modelo.

As camadas de convolução aplicam filtros de 3x3 pixéis com stride de 1 e sempre o mesmo padding e maxpool layer de 2x2 com stride de 2. Estas configurações de filtros pequenos nas camadas de convolução permitem manter a resolução das características enquanto o pooling ajuda a reduzir o custo computacional, reduzindo para metade a resolução das características.

As seguintes (e últimas) camadas são as camadas fully connected. As primeiras duas integram 4096 canais cada e a terceira, por executar uma classificação “1000-way ILSVRC” contém 1000 canais, um para cada classe. A última camada é a camada de softmax que classifica as imagens nas diferentes classes.

### ResNet

Após a primeira versão de arquiteturas baseadas em *CNN*s (AlexNet), de um modo geral, as arquiteturas seguintes consistiam sempre em camadas adicionais para reduzir a taxa de erro. Com o crescimento do número de camadas, verificou-se que a taxa de erro de treino e testes também aumenta como mostra a Figura 2.5. Percebeu-se que o problema estava relacionado com o gradiente tornar-se 0 ou demasiado grande, a este fenómeno chama-se “Vanishing gradient”.

Uma imagem com texto, file, Tipo de letra, diagrama

Descrição gerada automaticamente

Figura 2.5 - Comparação das taxas de erro de teste e treino em CNN's com número de camadas diferente, fonte: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

A Residual Network ou ResNet, propõe um conceito novo chamado Blocos Residuais. Este método propõe utilizar uma técnica (Skip Connections) que permite saltar camadas intermediárias conectando ativações entre camadas, formando Blocos Residuais como mostra a Figura 2.6. A arquitetura desta rede consiste num empilhamento destes Blocos Residuais.

Uma imagem com texto, diagrama, Tipo de letra, file

Descrição gerada automaticamente

Figura 2.6 - Skip Connections, fonte: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

A abordagem por trás desta rede é que, em vez das camadas aprenderem o mapeamento subjacente, permitimos que a rede se ajuste ao mapeamento residual. Por isso, em vez de aprender uma função de mapeamento H(x), os blocos residuais aprendem a função residual F(x), onde H(x) = F(x) + x, o que facilita o treino de redes muito profundas. A vantagem de adicionar este tipo de Skipping Connection é que se alguma camada prejudicar o desempenho da arquitetura, é ignorada pela regularização, o que possibilita o treino de uma rede neuronal profunda sem ocorrer o fenómeno Vanishing Gradient.

## Detetores de objetos

### One-stage detectors

Os One-stage detectores são desenhados para prever coordenadas da bounding box (caixa que delimita um objeto detetado) e probabilidades de classe para vários objetos numa única passagem pela rede. Estes modelos são conhecidos pela sua simplicidade e eficiência, uma vez que eliminam a necessidade de uma etapa separada para elaboração de propostas. O objetivo destes detectors é examinar amostras densas de possíveis localizações de objetos e prever a presença de um objeto e a sua bounding box associada numa única fase (single-shot). Por realizarem a deteção numa passagem única, estes algoritmos, são mais rápidos na deteção, o que os torna ideais para aplicações com deteção em tempo real. Esta abordagem sacrifica ligeiramente a precisão dos modelos para se focar na sua rapidez, e tendem a ter dificuldade a detetar objetos pequenos e tratar instâncias de objetos com variações de tamanho significativas. Os one-stage detectors mais populares são You Only Look Once (YOLO), Single-Shot Detector (SSD), RetinaNet e CenterNet.

### SSD

*SSD*, Single-Shot Detector, é um dos modelos mais rápidos para previsões multi-box de deteção de objetos para aplicações de real-time. Este modelo atinge uma taxa de frames por segundo (FPS) quase 5x maior que o Region-Based Convolutional Neural Networks (*R-CNN*). A sua arquitetura pode ser dividida em três componentes principais. A primeira etapa é a extração de características onde os principais mapas de características são selecionados recorrendo a uma *CNN* pré-treinada como VGG16 ou ResNet. A seguir este modelo conta com várias camadas convolucionais, cada uma responsável por detetar objetos de diferentes escalas. As camadas mais profundas são responsáveis pela deteção de objetos maiores, enquanto as camadas mais superficiais detetam objetos menores. Por último encontramos camadas de supressão não máxima para reduzir a taxa de erro causada por bounding-boxes repetidas uma vez que cada ponto em cada camada convolucional pode ter várias anchor boxes associadas, toda esta arquitetura pode ser vista na Figura 2.7. Entende-se por Anchor boxes as formas e tamanhos predefinidos pelo algoritmo usadas para prever as bouding boxes.



Figura 2.7 – Algoritmo *SSD*, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

### YOLO

O *YOLO* (You Only Look Once) é um dos métodos mais populares para deteção de objetos da atualidade. Existem várias versões deste modelo, à data deste artigo há registo até a versão 9. Este modelo utiliza uma das melhores arquiteturas de redes neuronais que lhe conferem alta precisão e velocidade de processamento, características que justificam a sua popularidade.

O *YOLO* trata a deteção de objetos como um problema de regressão única, direto da imagem para as bounding boxes e probabilidades da classe. Ao contrário dos métodos que usam regiões propostas como os métodos da família *R-CNN*, este método divide a imagem numa grade e faz previsões para cada célula da grade.

A arquitetura do *YOLO* conta com três etapas principais para atingir o seu objetivo como mostra a Figura 2.8. Na primeira destas fases, o algoritmo define blocos residuais, isto é, divide a imagem numa grade de SxS células onde cada célula atua como um ponto central para previsão. Na segunda técnica, cada ponto central para uma previsão particular é considerado para a criação das bouding boxes. Embora as tarefas de classificação funcionem bem para cada grade, a segregação das bounding boxes para cada previsão feita é mais complexa. Na terceira e última técnica, é implementada a métrica de Intersection of Union (IoU) para calcular a melhor bounding box para cada objeto.



Figura 2.8 – Algoritmo *YOLO*, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

### RetinaNet

Introduzido em 2017, o RetinaNet tornou-se um dos melhores modelos com single shot detection que ultrapassava outros métodos populares. Quando foi lançado, as suas capacidades excediam os modelos *SSD* e *YOLO*v2 uma vez que mantinha a velocidade destes modelos e ainda competia com os da família *R-CNN* em termos de precisão. Este modelo foi desenvolvido para resolver um problema comum dos modelos de deteção de objetos, que é a deteção de objetos pequenos.

A arquitetura do modelo RetinaNet, representada na Figura 2.9, é construída com base em três padrões, nomeadamente o modelo ResNet, feature pyramid network (FPN) e perda focal. O ResNet é uma arquitetura de *CNN* desenhada para superar o problema de desaparecimento de gradientes durante o treino de redes muito profundas. A *FPN* é a principal inovação do RetinaNet e permite ao modelo detetar objetos em várias escalas uma vez que ajuda a combinar os recursos semanticamente ricos de imagens de resolução mais baixa com os recursos semanticamente fracos das imagens de resolução mais alta. Por fim, neste modelo a perda de entropia cruzada nos modelos anteriores é substituída pela perda focal que permite ao modelo concentrar-se nas regiões que são mais difíceis de classificar durante o treino, o que pode melhorar o desempenho em classes minoritárias ou em objetos pequenos.

Figura 2.9 - Algoritmo RetinaNet, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

### DETR

DEtection TRansformer (DETR) é um algoritmo de deteção de objetos relativamente recente, introduzido em 2020 por investigadores do Facebook AI Research (FAIR). Este modelo aproveita a arquitetura Transformer, originalmente construída para tarefas de processamento de linguagem natural (NLP), para abordar o problema de deteção de objetos de maneira única e eficaz.

O Transformer é uma arquitetura de *CNN* conhecida pela sua capacidade de “self-attention” que permite capturar relações e dependências complexas entre elementos de uma sequência ou conjunto de dados. Esta capacidade de self-attention permite que o *DETR* entenda contextos locais e globais, melhorando as suas capacidades de deteção de objetos.

Como podemos ver na Figura 2.10, este modelo conta com um backbone que consiste numa *CNN* tradicional (ex:ResNet) para extrair um mapa de características de uma imagem. Esse mapa de características é processado pelo encoder que tem a capacidade de aprender e representar relações entre partes distantes de uma imagem. De seguida o decoder recebe uma sequência de embeddings de consulta (query embeddings) em que cada consulta tenta prever a presença e a localização de um objeto na imagem. As consultas são refinadas através de camadas de atenção cruzada e feed-forward, aprendendo a focar em diferentes partes da imagem. Por fim, existe uma camada de previsão que identifica a classe dos objetos e desenha bounding boxes caso eles existam.

Uma imagem com texto, captura de ecrã, diagrama

Descrição gerada automaticamente

Figura 2.10 - Arquitetura *DETR*, fonte: https://arxiv.org/pdf/2005.12872

### Two-stage detectors

Os two-stage detectors, ao contrário do one-stage, seguem um processo de duas etapas. Na primeira etapa, estes detetores geram um conjunto de propostas de regiões candidatas a localizações com objetos. Estas propostas são depois refinadas e classificadas na segunda etapa. Two-stage detectors tendem a ter uma precisão maior, mas a ser mais lentos quando comparados com os one-stage detectors, devido à etapa adicional de elaboração de propostas. Os two-stage detectors mais populares são *R-CNN*, SPPNet, Fast *R-CNN*, Faster *R-CNN*, *FPN* e S2ANet.

### *R-CNN*

O método *R-CNN* (Region-Based Convolutional Neural Networks) tenta extrair as características mais importantes da imagem, como mostra a Figura 2.11, permitindo focar apenas nas zonas selecionadas. Este processo de seleção permite gerar sub-segmentações (normalmente perto de 2000) e selecionar entradas candidatas para o algoritmo de deteção. Uma vez que o algoritmo de seleção termina com sucesso, cada proposta de região é redimensionada para um tamanho fixo e passa por uma *CNN* pré-treinada para extrair características. Por fim, as características extraídas são classificadas sobre presença de objetos e identificação da classe presente. Para obter o melhor resultado, a previsão é feita por um modelo de classificação enquanto um modelo de regressão é usado para classificar as bouding boxes para as regiões propostas.



Figura 2.11 - Algoritmo de deteção *R-CNN*, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

### Faster *R-CNN*

Embora o modelo anteriormente descrito (*R-CNN*) ser capaz de atingir resultados desejáveis, há alguns problemas associados, principalmente na velocidade do modelo. Foi então introduzido o modelo Fast *R-CNN* para tentar combater estes problemas associados à velocidade do modelo. Neste método toda a imagem é passada para a *CNN* pré-treinada em vez de sub-segmentações para ober um mapa de características e, posteriormente passa por uma camada adicional chamada Region-of-Interest (RoI) que recebe duas entradas do modelo pré-treinado e aplica o algoritmo Selective Search para devolver uma camada totalmente conectada com um output.

O modelo Faster *R-CNN* é uma das melhores versões da família *R-CNN* que melhora consideravelmente a velocidade do modelo quando comparada com os seus antecessores. Enquanto o *R-CNN* e o Fast *R-CNN* utilizam um algoritmo de Selective Search para calcular as propostas de região, o Faster *R-CNN* substitui este algoritmo com uma rede superior de propostas de região como mostra a Figura 2.12. Esta rede de propostas de região (RPN, Region Proposal Network) calcula imagens de uma ampla variedade e diferentes escalas para produzir resultados eficazes e permite reduzir a margem de tempo computacional em 10ms por imagem. A *RPN* consiste numa camada convolucional que obtém os mapas de características essenciais de cada pixel e para cada mapa tem várias anchor boxes (anchor boxes são caixas predefinidas usadas como referência para calcular as bounding boxes) com várias escalas, tamanhos e proporções diferentes.



Figura 2.12 - Algoritmo de deteção Faster *R-CNN*, fonte: https://neptune.ai/blog/object-detection-algorithms-and-libraries

# Metodologias e tecnologias

Nesta secção pretendemos dar a conhecer algumas metodologias de trabalho tanto de uma abordagem tradicional como Agile. Após discutir as várias metodologias vamos escolher uma e justificar a escolha.

Ainda neste capítulo apresentamos os requisitos funcionais que o utilizador deve esperar da nossa aplicação bem como mockups (protótipos) que desenhamos antes de começar o desenvolvimento da mesma com o objetivo de explorar a melhor maneira de estruturar a aplicação. A interface da aplicação foi projetada com foco na simplicidade para o utilizador de forma a garantir uma experiência agradável.

## Metodologia de trabalho

Quando se fala em desenvolvimento de software, existem duas maneiras de abordar a sua gestão e planeamento de modo a tornar a gestão de projeto mais eficiente e eficaz. Por um lado, temos as metodologias tradicionais, como o modelo cascata, que apostam numa abordagem sequencial e planeamento detalhado desde o início do projeto. Enquanto, em contraste, as metodologias ágeis dão prioridade à flexibilidade, colaboração e adaptação contínua ao longo do projeto.

### Método tradicional

A abordagem tradicional de gestão de projetos, também conhecida como método cascata, segue um processo linear e sequencial onde só se passa à fase seguinte quando a anterior é completada. É uma abordagem estruturada e orientada a planos com foco no planeamento completo e documentação antecipada. Todos os projetos, independentemente da sua natureza, seguem o mesmo ciclo de vida que incluem as fases de viabilidade, planeamento, design, construção, teste, produção e suporte. O projeto é planeado todo no início e não há qualquer mudança de requisitos, uma vez que nesta abordagem assume-se que os requisitos são fixos e o tempo e custo variáveis. Como tal, não é adequado a projetos de grande escala em que os requisitos para todo o projeto não se conseguem definir numa fase inicial.

### Metodologias ágeis

Ao contrário do método anterior, o foco das metodologias ágeis é o trabalho em equipa, colaboração com o cliente e flexibilidade. É uma abordagem iterativa que se concentra mais na incorporação do cliente e lançamentos contínuos em cada iteração do desenvolvimento do projeto. O conceito chave desta abordagem é que aprofunda a evolução das mudanças e esforço colaborativo para obter resultados em vez de um processo predefinido. O planeamento adaptativo é a principal característica das metodologias ágeis que convence muitos gestores de projeto.

Na Tabela 1 podemos ver a comparação de alguns aspetos comuns a ambas as abordagens.

|  |  |  |
| --- | --- | --- |
| Parâmetros | Metodologias Ágeis | Gestão de projeto tradicional |
| Flexibilidade e adaptabilidade | Ênfase na flexibilidade e adaptabilidade | Ênfase no planeamento e previsibilidade |
| Requisitos do projeto | Prioriza a satisfação do cliente e software funcional | Prioriza o seguimento de um plano rigoroso e atender aos requisitos do projeto |
| Comunicação | Encoraja comunicação e colaboração cara-a-cara | Confia na documentação escrita. Comunicação formal |
| Fases de projeto e planificação | Sprints curtos para planeamento e entrega. Permite alterações e ajustes ao longo do projeto | Fases de projeto mais longas para planeamento e entrega. Tem um plano rígido que e difícil alterar |
| Funcionalidade de equipa | Equipas multifuncionais | Equipas com funções divididas |
| Organização | Equipas auto-organizadas | Equipas organizadas hierarquicamente |
| Estrutura organizacional | Iterativa | Linear |
| Escala de projeto | Grande escala | Escala pequena e média |
| Modelo de desenvolvimento | Modelo de entrega evolutivo | Modelo de ciclo de vida |
| Requisitos do utilizador | Inputs interativos | Claramente definidos antes de implementação |
| Envolvimento do cliente | Alto | Baixo |

Tabela 1- Tabela comparativa entre metodologias tradicionais e ágeis

Fonte: <https://www.knowledgehut.com/blog/agile/agile-project-management-vs-traditional-project-management> (com ajustes)

Optamos por seguir uma metodologia ágil por não saber de antemão os requisitos que vai ter a nossa aplicação, uma vez que esta abordagem permite-nos desenvolver a aplicação de maneira mais flexível e colaborativa. Vamos então analisar as principais metodologias ágeis e a forma como abordam a gestão dos projetos.

### Scrum

O Scrum é a framework mais utilizada de todas as metodologias ágeis e caracteriza-se por etapas ou ciclos de desenvolvimento, conhecidos como sprints e pela maximização do tempo de desenvolvimento de um produto de sofware em direção ao Product Goal.

Uma equipa de Scrum é composta por 3 elementos com papeis bem definidos, o Product Owner (PO), o Scrum Master (SM) e a equipa de desenvolvimento. O *PO* é o elemento que liga o cliente à equipa de desenvolvimento, assumindo a função garantir que a visão do cliente é entendida pela equipa de desenvolvimento e definindo os critérios de sucesso. É este que define o que deve ser feito, e em que ordem deve ser feito, atribuindo valor aos requisitos. O *SM* é referido como o facilitador do projeto uma vez que este certifica que as boas práticas da metodologia Scrum são seguidas, verifica relações saudáveis entre o *PO* e os programadores e elimina quaisquer obstáculos que impeçam uma colaboração produtiva. Por fim, a equipa de desenvolvimento, como o nome indica, é o conjunto de membros que trabalha em conjunto para criar, testar e lançar versões incrementais do produto final.

Além da equipa Scrum, esta metodologia conta com os seguintes eventos Scrum: Daily Scrum, Sprint, Sprint planning, Sprint review e Sprint retrospective.

O Daily Scrum consiste numa curta reunião diária que ocorre, geralmente, todos os dias no mesmo sítio à mesma hora. Nesta reunião a equipa avalia o trabalho realizado no dia anterior e planeia o que será feito nas próximas 24 horas.

A Sprint representa o período em que o trabalho deve estar concluído, não maior que um mês. Uma nova sprint começa quando a anterior acaba.

A Sprint Planning meeting marca o início de uma nova sprint. Nesta reunião, toda a equipa Scrum participa e define objetivos, sendo que no final pelo menos um incremento de software utilizável deve ser produzido.

No fim de cada sprint, acontece a Sprint Review em que a equipa Scrum mostra o trabalho realizado ao *PO* e stakeholders.

Por fim, a Sprint Retrospective, dá lugar à equipa para discutir o que correu bem e o que pode ser melhorado em relação à última sprint com o objetivo de atingir uma melhoria contínua.

No Scrum existem ainda 2 backlogs. O Product backlog que contém a lista das tarefas para sprints futuras e s Sprint Backlog que inclui as tarefas a realizar na sprint atual.

### XP

Extreme Programming (XP) é outra metodologia ágil bastante presente no mundo dos projetos de desenvolvimento de software. Tem como objetivo criar sistemas de alta qualidade, assente nos princípios de proximidade com o cliente, testes constantes e ciclos de desenvolvimento curtos. Os 5 componentes básicos do *XP* são: comunicação, simplicidade, feedback, respeito e coragem. Esta abordagem aposta na velocidade e simplicidade com ciclos curtos de desenvolvimento e menos documentação.

No *XP* o cliente deve participar ativamente do processo de desenvolvimento. Tudo precisa da comunicação com o cliente. Este deve receber o melhor resultado possível a cada sprint, ver o progresso no sistema e ser informado de mudanças de planos para que possa dar a sua opinião.

Outra prática fundamental no *XP* é o pair programming. O pair programming consiste numa programação a pares num único computador em que um tem o papel de escrever o código enquanto o outro está sempre ao lado a rever. Desta maneira, erros e detalhes que possam passar despercebidos a um programador, o outro consegue cobri-los diminuído a probabilidade de falhas.

Para perceber mais facilmente o que difere o *XP* do Scrum podemos consultar a Tabela 2:

|  |  |
| --- | --- |
| Scrum | Extreme Programming |
| Trabalho em iterações (sprints) podem ir até 1 mês | Trabalho em iterações de apenas 1-2 semanas |
| Modelos Scrum não permitem mudanças no seu cronograma ou diretrizes. | Permite mudanças nos cronogramas definidos |
| Ênfase na auto-organização | Ênfase em fortes práticas de engenharia |
| Equipa determina a sequência na qual o produto é desenvolvido | Equipa segue uma ordem de prioridade pré-determinada |
| A framework Scrum não está completamente descrita. Para adotá-la é necessário completá-la com métodos de trabalho, como *XP* ou Kanban | Pode ser aplicado diretamente a uma equipa. O *XP* é também conhecido pelas suas características “Ready-to-apply” |
| Não enfatiza as práticas de engenharia de software que os programadores devem usar | Enfatiza as técnicas de programação que os programadores devem seguir para garantir um melhor resultado |
| Requer que os programadores estejam conscientes da adoção de métodos de engenharia para garantir um melhor progresso ou qualidade | É muito rígido na adoção de métodos de engenharia, como pair programming, design simples, reestruturação, para garantir um melhor progresso ou qualidade. |
| O Product Owner define a prioridade das tarefas de acordo com os requisitos | O cliente define a prioridade das tarefas e analisa os lançamentos. |
| Existe flexibilidade para a equipa ajustar a prioridade das tarefas se necessário | A equipa não pode alterar a prioridade dos requisitos definidos pelo cliente |
| Valores: - Abertura  - Foco  - Compromisso | Valores: - Comunicação  - Simplicidade  - Feedback |
| Menor envolvimento do cliente no projeto | Maior envolvimento do cliente no projeto |

Tabela 2 - Tabela comparativa entre Scrum e *XP*, fonte: https://www.geeksforgeeks.org/difference-between-scrum-and-xp/

### Kanban

A metodologia kanban visa uma melhoria contínua, flexibilidade na gestão de tarefas, e fluxo de trabalho melhorado. Por ser uma abordagem ilustrativa, o progresso do projeto como um todo é facilmente compreendido em poucos segundos.

O Kanban, ao contrário do scrum, não possui um prazo definido para quando as tarefas devem ser concluídas. Em vez disso, esta metodologia é gerida pela prioridade dos requisitos num quadro Kanban, como se pode ver na Figura 3.1. Este quadro é dividido em três estados para uma tarefa: pendente (to do), em desenvolvimento (doing) e concluída (done). As tarefas são representadas por cartões visuais que transitam entre os estados até serem finalizadas. Assim, a equipa consegue ter uma noção visual do desenvolvimento do projeto.



Figura 3.1 - Quadro kanban, fonte: https://www.kanbanchi.com/wp-content/uploads/2013/05/kanban-board-700x380.png

### Scrumban

O Scrumban, como o nome sugere, é uma metodologia ágil que combina elementos do Scrum e Kanban. Esta abordagem envolve aplicar os princípios de visualização do fluxo de trabalho e processos flexíveis do Kanban a uma equipa da Framework Scrum. Este princípio, no entanto, remove alguns dos aspetos mais rígidos do Scrum e deixa as equipas adotar uma abordagem personalizada para o desenvolvimento do projeto.

O Scrunban adota um quadro similar ao quadro Kanban na medida em que é a ferramenta principal de fluxo de trabalho com a diferença que permite adicionar as colunas necessárias ao quadro para marcar a fase de progresso para cada requisito.

O Scrum, define limites de tempo e tarefas por cada sprint, o contrário acontece no Kanban que se foca no fluxo de trabalho contínuo. No Scrumban é estabelecido um limite para a quantidade de trabalho que a equipa pode realizar num momento e este limite é o total de cartões no quadro.

Enquanto no Scrum cada tarefa é atribuída a um membro específico da equipa, no Scrumban o foco é estabelecer a ordem de prioridade das tarefas no quadro e cabe à equipa distribuir as tarefas a realizar pelos diversos membros.

Embora esta abordagem não tenha todas as reuniões típicas da estrutura Scrum, no Scrumban pode haver reuniões curtas durante as sprints para a equipa discutir planos e desafios encontrados.

### Metodologia de trabalho escolhida

Das metodologias ágeis apresentadas, excluímos já de partida o *XP* por não termos o cliente sempre presente no desenvolvimento da aplicação nem condições para aplicar o pair programming que esta abordagem requere. Das restantes, Scrum, Kanban e Scrumban decidimos escolher a última por oferecer o melhor de cada uma das metodologias em que esta se baseia enquanto descarta os aspetos menos favoráveis das mesmas. Deste modo conseguimos aproveitar não só a flexibilidade que o Kanban nos permite, mas também os princípios de planeamento Scrum permitindo-nos adaptar nosso processo de desenvolvimento de acordo com as necessidades.

Para aplicar a metodologia ágil Scrumban, realizamos uma reunião semanal com o Product Owner e Scrum Master, interpretados pelos professores orientadores. Nesta reunião discutimos o trabalho desenvolvido na última sprint que acaba nessa mesma reunião e determinamos o que deve ser feito para a próxima. Utilizamos a framework Jira Software da Atlassian que nos permite criar Issues que representam as tarefas/requisitos que a nossa aplicação tem ou vai ter. Aqui também conseguimos criar o quadro kanban que nos proporciona uma noção visual do desenvolvimento do trabalho e onde podemos colocar os Issues previamente criados. Conseguimos ainda criar e terminar as várias sprints ao longo do processo de desenvolvimento.

## Funcionalidades da aplicação

### Análise de requisitos

RF1. Adicionar dispositivo: Enquanto utilizador quero adicionar novos dispositivos à página de dispositivos para começar a captura e deteção nesses dispositivos.

RF2. Editar dispositivo: Enquanto utilizador quero editar dispositivos presentes na página de dispositivos para alterar as configurações dos mesmos.

RF3. Pausar captura: Enquanto utilizador quero parar a captura de frames de um dispositivo para parar de atualizar os frames desse dispositivo.

RF4. Retomar captura: Enquanto utilizador quero retomar a captura de frames de um dispositivo pausado para voltar a capturar frames desse dispositivo.

RF5. Capturar live: Enquanto utilizador quero capturar frames sem delay (ver em live) para ver em direto o estado de um dispositivo.

RF6. Expandir dispositivo: Enquanto utilizador quero expandir a janela de captura de um dispositivo para ver com maior detalhe os frames capturados do mesmo.

RF7. Remover dispositivo: Enquanto utilizador quero remover dispositivos da janela de dispositivos para remover dispositivos que já não tenha interesse.

RF8. Alterar layout: Enquanto utilizador quero alterar o layout da disposição dos dispositivos para escolher qual o layout que quero que os dispositivos estejam dispostos.

RF9. Alterar delay: Enquanto utilizador quero alterar a frequência com que um dispositivo captura e processa um frame para escolher o tempo que achar adequado para cada dispositivo.

RF10. Adicionar objetos a detetar: Enquanto utilizador quero adicionar objetos que quero detetar num dispositivo para escolher os objetos que considero relevantes para deteção no dispositivo.

RF11. Remover objetos a detetar: Enquanto utilizador quero remover objetos que já não quero detetar num dispositivo para retirar os objetos que já não tenha interesse a detetar num dispositivo.

RF12. Definir tempo para alerta de objeto: Enquanto utilizador quero definir o tempo máximo de um objeto parado antes de emitir um alerta para ser notificado quando um objeto estiver parado num dispositivo há mais tempo que o definido.

RF13. Filtrar alertas: Enquanto utilizador quero filtrar os alertas emitidos para visualizar apenas os alertas que se enquadram nos parâmetros que procuro.

RF14. Eliminar alertas: Enquanto utilizador quero eliminar os alertas emitidos para eliminar os alertas que considero irrelevantes ou que já foram analisados e tratados.

RF15. Exibir detalhes de alerta: Enquanto utilizador quero visualizar os detalhes de um alerta para analisar mais informações associadas ao alerta emitido.

RF16. Gravar configuração: Enquanto utilizador quero guardar os dispositivos e as suas configurações para quando sair da aplicação poder reabrir no estado em que estava.

RF17. Importar configuração: Enquanto utilizador quero importar uma configuração que tenha guardado previamente para poder importar os dispositivos e as suas configurações.

RF18. Configurar email para receber alerta: Enquanto utilizador quero configurar um ou mais emails onde quero receber os alerta emitidos pela aplicação.

RF19. Configurar telemóvel para receber alerta: Enquanto utilizador quero configurar um ou mais telemóveis onde quero receber os alerta emitidos pela aplicação.

### Mockups

Neste capítulo apresentamos os mockups desenhados para aplicar os requisitos definidos na secção anterior. A aplicação foi pensada contendo duas páginas principais, a de dispositivos onde o utilizador pode configurar tudo o que é relacionado aos dispositivos e a de alertas onde o utilizador pode configurar os parâmetros relacionados com os alertas.

Uma imagem com texto, multimédia, captura de ecrã, software

Descrição gerada automaticamente

Figura 3.2 - Mockup início

Uma imagem com texto, captura de ecrã, multimédia, Software de multimédia

Descrição gerada automaticamente

Figura 3.3 - Mockup layout horizontal

Uma imagem com texto, software, Software de multimédia, multimédia

Descrição gerada automaticamente

Figura 3.4 - Mockup adicionar dispositivo



Figura 3.5 - Mockup adicionar dispositivo IP



Figura 3.6 - Mockup definições dispositivo



Figura 3.7 - Mockup emitir alerta para um dispositivo



Figura 3.8 - Mockup dispositivo expandido



Figura 3.9 - Mockup página alertas

# Implementação

Este capítulo serve para justificar as escolhas tomadas a respeito da implementação das funcionalidades descritas no capítulo anterior.

## Escolha do modelo de deteção

Sendo esta aplicação uma aplicação de deteção de objetos, o primeiro passo é escolher o modelo a utilizar para a deteção dentro dos muitos existentes nos dias de hoje e apresentados no capítulo 2.3. Para isso vamos analisar resultados de alguns estudos comparativos de modelos de deteção de objetos.

A Tabela 4 foi retirada de uma pesquisa (Sahil e Zaidi, 2022) e permite comparar a eficácia de vários modelos atuais destacando a precisão através dos parâmetros AP[0,5:0,95] e AP0.5 e a velocidade através do parâmetro *FPS*. AP significa Average Precision e quando seguida de 0.5 representa a precisão média de todas as classes quando preveem bounding boxes com *IoU* > 0.5 sendo uma métrica menos rigorosa, enquanto quando seguida de [0,5:0,95] avalia a média da precisão de múltiplos valores de *IoU* (de 0.5 a 0.95), fornecendo uma visão mais completa e rigorosa do desempenho do modelo. *FPS* representa o número de frames processados por segundo e avalia a velocidade do modelo. As linhas a cinzento identificam modelos com *FPS* > 30.

Uma imagem com texto, captura de ecrã, menu, número

Descrição gerada automaticamente

Tabela 4 – Tabela comparativa entre modelos de deteção

Uma vez que um dos requisitos para a nossa aplicação é função de captura em real-time, focamo-nos apenas nas linhas a cinzento uma vez que apenas estas são indicadas para real-time atingindo pelo menos 30 *FPS*. Dentro destas, a que apresenta maior precisão tanto AP[0,5:0,95] como AP0.5 é o modelo *YOLO*-v4 que permite capturar 31 *FPS* e mantém um AP0.5 de 64,90% e AP[0,5:0,95] de 43%. O modelo EfficientDet-D2 apresenta também AP[0,5:0,95] de 43% conseguindo 41.7 *FPS*.

Num estudo mais recente (Boesch, 2024) está presente o seguinte gráfico, representado pela Figura 4.1.

Figura 4.1 - Gráfico de comparação *YOLO*v7 vs. *YOLO*v5 vs. *YOLO*R and Vit Transformers, fonte: https://viso.ai/deep-learning/object-detection/



Neste gráfico no eixo do X verifica-se o tempo de inferência que quanto menor, melhor e no eixo do Y a precisão média. Aqui verifica-se que a versão mais recente do *YOLO* apresentada (*YOLO*v7) é claramente favorita atingindo 56,8% de precisão, a maior de todos os modelos apresentados e é a que apresenta menor tempo de inferência, apenas igualando outras versões do *YOLO* (*YOLO*v5 e *YOLO*R) que atingem uma precisão menor.

Esta segunda comparação mostra que o modelo antes equiparado ao *YOLO*, o EfficientDet-D2, perde significativamente quando comparado a versões mais recentes do *YOLO*.

O *YOLO*v8 foi lançado em janeiro de 2023 e mostra melhorias relativamente ao seu antecessor (*YOLO*v7) tanto na precisão como na velocidade como se pode observar na Figura 4.2.

## YOLO algorithm comparison YOLOv7 vs YOLOv8

Figura 4.2 – Gráfico de comparação *YOlO*v7 e *YOLO*v8, fonte: https://docs.ultralytics.com/models/yolov8/

Uma vez que o modelo *YOLO*v7 mostrou-se o favorito perante as necessidades da nossa aplicação e o modelo *YOLO*v8 é mais preciso e mais rápido que o anterior, vamos implementar o modelo *YOLO*v8 como o nosso modelo de deteção de objetos integrado na aplicação.

## Ambiente de desenvolvimento

Para integrar o modelo de deteção de objetos *YOLO*v8 na nossa aplicação seguimos a documentação oficial da ultralytics. Esta aplica o *YOLO* em ambiente python pelo que nós também utilizamos esta linguagem para desenvolver os nossos scripts relacionados com o *YOLO* no ficheiro yoloScripts.py.

Para integrar estes scripts com uma interface gráfica para que o utilizador possa interagir com o modelo de deteção de objetos, recorremos ao PyQT5. Esta é a Framework para GUI´s em python mais popular por vários motivos, entre eles:

* Suporta vários sistemas operativos, entre eles Android, IOS, Linux, macOS e Windows;
* Tem uma grande biblioteca de widgets;
* Recorre à framework Qt, uma biblioteca C++ amplamente usada para criar GUIs ricas e robustas;
* Permite a criação de aplicações GUI profissionais e completas;
* Pode ser facilmente integrada com outras bibliotecas Python, como NumPy, Pandas e bibliotecas de machine learning, tornando-se ideal para a criação de aplicações científicas e de análise de dados;
* Suporta programação orientada a objetos;
* Oferece boa performance para a maioria das aplicações GUI e pode ser otimizada conforme necessário para atender a requisitos específicos de desempenho.

## Arquitetura da aplicação

Este projeto pode ser dividido em duas grandes componentes, o ficheiro “yoloScript.py” que contém a chamada à função do *YOLO* e toda a lógica de deteção de objetos e compreende também os métodos relacionados com a emissão de alertas. A outra componente com o nome de “App.py“ abrange a lógica da aplicação que interage com o utilizador onde recorremos ao PyQT5.

Para a chamada à função de previsão do *YOLO*, enquadrada na funcionalidade deteção de objetos, escolhemos uma abordagem das quatro inicialmente consideradas: Grab & Retrieve, Parallel, Open & Close e Open & Close threading. Na secção de teste 5.2 explicamos em detalhe cada um destes métodos de captura e o cenário de teste efetuado que nos permitiu escolher a melhor alternativa atendendo aos nossos requisitos. A arquitetura escolhida foi Parallel que para cada dispositivo de captura é criada uma thread que lê um frame, descodifica-o e envia para a função do *YOLO* para que possa detetar objetos na imagem. No mesmo teste verificamos ainda que o ideal é que esta chamada à função *YOLO* ocorra com intervalo mínimo de 10 segundos.

Estas duas componentes comunicam através de um elemento queue presente na biblioteca multiprocessing. Este elemento permite manter um canal aberto entre estes dois ficheiros que funciona como uma pilha manipulando itens da maneira First In First Out. A queue é criada no ficheiro App.py, enviada como parâmetro para o ficheiro yoloScript e é partilhada pelos dois permitindo a comunicação.

## Implementação dos requisitos funcionais

Vamos explicar de um modo mais técnico como foram implementados os requisitos funcionais descritos na secção 3.2.

### RF1. Adicionar dispositivo

A função adicionar dispositivo, acedida pelo botão na página de dispositivos (DispositivoWindow), cria um widget (DispositivoWidget) com os parâmetros definidos no painel de configuração, adiciona-o à lista de widgets global que guarda todos os DispositivosWidgets e ao dicionário de dispositivos da janela DispositivoWindow. De seguida cria uma thread que acede ao método predict do ficheiro yoloScript que começa a capturar frames do dispositivo e processa-os com o *YOLO* para detetar objetos adicionados à lista de objetos a detetar na página de configuração. Por fim, caso este seja o primeiro dispositivo a ser adicionado, este método cria uma thread que é responsável pela leitura dos itens da queue que provém da deteção do *YOLO*.

### RF2. Editar dispositivo

A função editar dispositivo, acedida através do botão de settings presente no DispositivoWidget, ao clicar no botão o sistema abre uma janela de configuração com os campos preenchidos com os parâmetros do dispositivo. A classe do DispositivoWidget depois fica à espera de que a janela ConfigurarDispositivo envie um sinal de que o utilizador terminou as alterações e altera os seus atributos para os recentes editados como se pode ver na Figura 4.3.

 Finalmente chama-se o método “update\_obj\_to\_find” do ficheiro yoloScript para que a lista de objetos a detetar e o tempo para emitir alertas sejam atualizados também nos métodos deste ficheiro como mostra a Figura 4.4.

Figura 4.3 – Código editar dispositivo



Figura 4.4 Código atualizar lista de objetos a detetar

### RF3. Pausar captura

Outro botão presente no DispositivoWidget é o botão de pausa. Este está associado ao método “stop\_button\_clicked” que começa por chamar o método “change\_stop” do ficheiro yoloScript que altera o valor do dicionário stop\_dict na posição do dispositivo a ser pausado para true utilizando para o efeito um lock. A função que está continuamente a capturar frames para deteção (função predict), na próxima iteração, entra nos blocos de código contidos na condição “with stop\_lock: ” e interrompe a captura de frames até que o stop\_dict para este dispositivo volte a ter o valor false. É utilizado um “Lock” (stop\_lock) para garantir que a operação seja thread-safe e não exista concorrência ao aceder e modificar os valores do dicionário stop\_dict como se pode ver na Figura 4.5 e Figura 4.6.



Figura 4.5 - Código stop\_lock 1



Figura 4.6 - Código stop\_lock 2

Regressando ao ficheiro com o código da aplicação, após notificarmos a função de captura para parar de capturar, alteramos a imagem do DispositivoWidget para a última imagem capturada com um icon de pausa sobreposto.

### RF4. Retomar captura

Do mesmo modo que temos uma função para avisar a função predict que pare a captura, a mesma função avisa a última a retomar a captura. Para isto a basta chamar a função change\_stop na mesma, mas passando o valor false em vez de true. Com isto, o dicionário stop\_dict na posição do dispositivo readquire o valor false, deixando de entrar nos blocos de código dentro da condição “with stop\_lock:” retomando à normal funcionamento da função.

O próximo passo é ir buscar o valor do delay na comboBox, porque este pode ter sido alterado, e chamar a função change\_delay do ficheiro yoloScript que, paralelamente ao funcionamento do stop\_dict e stop\_lock, utiliza o delay\_dict e delay\_lock para alterar a variável local delay na função predict acedendo ao valor presente no dicionário delay\_dict na posição do dipositivo.

### RF5. Capturar live

Quando o botão com o ícone “live“ é pressionado, é chamada a função change\_stop do ficheiro yoloScript com o parâmetro stop=false para que, caso o estado anterior tenha sido pausa, a função predict voltar a capturar frames. É também chamada a função change\_delay com o valor de delay 0 para que a função predict do yoloScript.py funcione sem qualquer time.sleep, o que espelha uma funcionalidade de captura live, em real-time.

### RF6. Expandir dispositivo

Este botão quando pressionado, cria uma janela com a imagem presente no widget e redimensiona-a para 800px de largura mantendo a proporção original da imagem.

### RF7. Remover dispositivo

É possível remover um dispositivo da lista de dipositivos na janela principal. Para isso, criamos a função “remove\_button\_clicked” que chama a função “remove\_device” do ficheiro yoloScript. Esta, por sua vez, adiciona a lista delete\_devices o dispositivo a adicionar, e, na próxima iteração do predict, toda a lógica de capturar e processar frames é saltada e termina o ciclo com a liberação da câmara e limpeza dos dados associados ao dispositivo como mostra a Figura 4.7, dentro da função predict existem vários pontos onde é lida o valor da variável delete\_devices para verificar se o dispositivo está na lista isto para melhorar a experiencia do utilizador para que quando este clique no botão de eliminar o dispositivo o dispositivo seja eliminado o mais rápido possível.



Figura 4.7 – Limpeza dos dados do dispositivo removido

### RF8. Alterar layout

Para permitir alternar entre layouts, o layout de dispositivos é do tipo QStackedLayout como mostra a Figura 4.8. Este tipo de layout permite associar outros layouts e alternar entre os mesmos.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamenteExiste um botão para cada layout que ao ser pressionado, altera para o layout escolhido, alterando o índice do stacked\_layout e remove os widgets associados ao layout anterior e adiciona os ao escolhido.

Figura 4.8 - Stacked Layout

### RF9. Alterar delay

Na página inicial de dispositivos, o utilizador pode alterar a frequência que os frames são atualizados. Para isso, basta alterar o valor da comboBox de delay no widget do dispositivo. A comboBox tem um listener para quando o valor é alterado ativando a função “change\_delay” que vai buscar o novo valor da comboBox e envia como parâmetro para a função change\_delay do ficheiro yoloScript.py. Esta função altera a variável local delay na função predict acedendo ao valor presente no dicionário delay\_dict na posição do dipositivo.

### RF10. Adicionar objetos a detetar

Na página de configuração do dispositivo (que pode ser acedida ao adicionar novo dispositivo ou ao editar um existente) o utilizador pode adicionar novos objetos a detetar no dipositivo. Nesta página estão listadas as classes pré-treinadas do modelo numa QListWidget ordenadas alfabeticamente e permitem seleção múltipla. Por baixo da lista está um botão com a label “Add (+)” que vai buscar os itens selecionados na QListWidget, guarda-os numa lista, adiciona-os à QListWidget de objetos selecionados e remove-os da lista dos objetos disponíveis. Quando o utilizador terminar a configuração, o sistema envia um sinal à instância da janela DispositivosWindow em que um dos parâmetros é a lista dos objetos selecionados para deteção. Caso o dispositivo já exista, o sistema altera os parâmetros editados no widget do dispositivo e chama a função “update\_obj\_to\_find” do ficheiro yoloScript que altera o valor no dicionário global “obj\_find\_dict” na posição do dispositivo para a nova lista de objetos a detetar. Caso o dispositivo seja novo, o procedimento está descrito no RF1, localizado na subsecção 4.4.1.

### RF11. Remover objetos a detetar

Na página de configuração do dispositivo (que pode ser acedida ao adicionar novo dispositivo ou ao editar um existente) o utilizador pode remover objetos a detetar no dipositivo previamente adicionados. Nesta página estão listadas as classes selecionadas para deteção numa QListWidget ordenadas alfabeticamente e permitem seleção múltipla. Por baixo da lista está um botão com a label “Remove (-)” que vai buscar os itens selecionados na QListWidget de objetos selecionados e remove-os da lista dos objetos selecionados e adiciona-os à lista de objetos disponíveis. De seguida eliminamos o item do dicionário “tempo\_alertas” que é o dicionário que associa os objetos a detetar com o tempo definido para emitir alerta e eliminamos o widget na QListWidget de definir tempo de alertas para os dispositivos.

### RF12. Definir tempo para alerta de objeto

Na página de configuração do dispositivo, podemos navegar para a página à direita que contém as configurações de alertas. Esta página tem um QListWidget com os dispositivos selecionados para deteção e duas dropdowns associadas a cada dispositivo para definir o tempo para emitir um alerta. A primeira dropdown guarda o valor e a segunda a ordem de tempo (segundos, minutos ou horas). Quando um objeto é adicionado, o tempo de alerta vem a 0, o que significa que não tem um tempo definido, logo não está a emitir alerta. Após a configuração, quando se pressiona o botão “Done”, é enviado o dicionário que relaciona os objetos ao tempo para emitir alerta para o ficheiro yoloScript.py. Caso o dispositivo seja novo, é enviado como parâmetro para o criar, caso seja uma alteração de um dispositivo existente, é alterada a lista através da função “update\_obj\_to\_find” que recebe a lista de objetos e o dicionário dos alertas.

### RF13. Filtrar alertas

Na página onde são exibidos os alertas emitidos, existe um menu de filtragem no topo. Quando o utilizador pressiona o botão “Apply filters” é chamada a função “filter\_alertas”. Esta começa por eliminar os todos widgets com os alertas exibidos. De seguida extrai os parâmetros escolhidos nos filtros e por fim filtra os alertas pelos parâmetros escolhidos e chama a função “mostrar\_alertas” que trata de os listar. Esta última função recebe os alertas a filtrar e cria os widgets para que possam ser exibidos, como se pode ver na Figura 4.9.



Figura 4.9 - Filtragem dos alertas

### RF14. Eliminar alertas

Os widgets com a informação dos alertas têm um botão que permite removê-los da lista. O método associado a este botão, começa por criar uma lista “alertas” vazia e tentar ler os alertas do ficheiro “alertas.bin” através do “pickle.load(f)” e extrai-los para a lista “alertas” e o loop termina quando é encontrado o EOF indicando o fim do ficheiro. A seguir a lista de alertas é filtrada removendo o alerta onde foi pressionado o botão remover. Recorrendo ao “pickle.dump” os alertas filtrados são reescritos no ficheiro alertas.bin. O acesso a ficheiros está contido num try, catch para lidar com qualquer erro que possa acontecer. Por fim, o widget do alerta é removido da página de alertas, como se pode ver na Figura 4.10.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura 4.10 - Código remover alerta

### RF15. Exibir detalhes de alerta

Ao carregar no alerta, é exibida uma janela com os detalhes do alerta. Para isto, apenas é criada uma janela que recebe os detalhes do alerta como atributos e mostra-os, sendo estes o nome do dispositivo em que o alerta foi originado, o tempo que o objeto esteve parado, o objeto e a data e horas do alerta.

### RF16. Gravar configuração

A aplicação conta com uma barra de menu com a opção File. Dentro das ações desta opção encontram-se o “Save” e “Save as”, ao clicar no “Save as” é aberto o explorador de ficheiros de forma a ser possível guardar um ficheiro com as configurações atuais. A ação “Save” guarda no mesmo ficheiro caso este já tenha sido aberto ou então caso não tenha sido aberto nenhum ficheiro de configuração acontece o mesmo processo que “Save as”, como se pode ver na Figura 4.11.

Uma imagem com texto, software, Software de multimédia, captura de ecrã

Descrição gerada automaticamente

Figura 4.11 - Guardar configurações

### RF17. Importar configuração

No menu da aplicação na opção “File” existe a ação “Open” ao ser clicado primeiro apaga todos os DispositivosWidgets que já estão colocados e de seguida abre o explorador de ficheiros para que se escolha de que ficheiro queremos importas as configurações. O ficheiro esperado é do tipo .json onde são guardados todos os elementos necessários para criar todos os DispositivosWidget que foram guardados anteriormente, como se pode ver na Figura 4.12.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 4.12 – Importar configuração

### RF18. Configurar email para receber alerta

No menu da aplicação na opção “Preferences” existe a opção “Emails to send alert” onde o utilizador pode adicionar e remover emails onde pretende ser notificado dos alertas. Para tal, é aberta uma janela chamada "EmailDialog" que contém uma lista com os endereços de email para os quais os alertas serão enviados. Existe também um campo para escrever um email, para adicionar este email à lista, é necessário clicar no botão “Adicionar Email”, para remover emails da lista é necessário selecionar o email a remover e clicar no botão “Remover Email”. Ao fechar a janela, a função emails\_to\_send\_alert do script yoloScript.py é chamada, atualizando a lista de emails para onde os alertas serão enviados.

### RF19. Configurar telemóvel para receber alerta

No menu da aplicação na opção “Preferences” existe a opção “Phone Numbers to send alert” onde o utilizador pode adicionar e remover números de telemóvel onde se pretende receber os alertas. Para tal, é aberta uma janela chamada “PhoneDialog” que contém a lista com os números de telemóvel onde pretende receber os alertas. Para adicionar um número de telemóvel à lista, é necessário selecionar o código numérico do país na QComboBox e, em seguida, escrever o número em si no campo à direita da QComboBox. Tal como no “EmailDialog” existem botões para adicionar o número à lista “Adicionar Número de Telefone” e o que remove os números da lista “Remover Número de Telefone”. Ao fechar a janela, primeiro o espaço entre o código do país e o número é removido e, de seguida é chamada a função “phone\_numbers\_to\_send\_alert” do yoloScript.py, atualizando a lista de números de telemóvel para onde os alertas são enviados.

### RF20. Nova Configuração

No menu da aplicação, na opção "File", existe a opção "New". Ao ser clicado, verifica se existem dispositivos ativos e, caso estes não estejam guardados numa configuração, é criada uma QMessageBox a perguntar se o utilizador quer guardar a configuração. Caso ele selecione a opção "Sim", a aplicação seguirá para o RF16. Se a resposta for "Não" ou se a configuração já tiver sido guardada, a aplicação remove todos os dispositivos e apresenta um ambiente limpo.

# Testes

No capítulo de testes vamos apresentar testes conduzidos por nós que no decorrer do desenvolvimento do projeto achamos necessários.

## Teste de margem ideal para definir posição do objeto

O *YOLO*, após detetar um objeto, desenha uma bounding box que delimita o objeto detetado. Mesmo que o input não se altere, no caso deste projeto em que o input são camaras de vídeo, significa objeto na mesma posição, esta box pode variar ligeiramente devido a vários fatores. Estes fatores compreendem: variabilidade de rede, anchor boxes ou Non-Maximum Supression (NMS). Variabilidade de rede porque, a rede *YOLO*, como qualquer rede neuronal de deep lerning, tem variabilidade inerente nas suas previsões, devido à variação de valores inicias de peso, processos de otimização e descendente estocástico de gradiente (Stochastic Gradient Descent - SGD). Se as anchor boxes não corresponderem exatamente à forma e tamanho dos objetos na imagem, pode também levar a variações nas bounding boxes. Por fim, NMS é uma técnica de pós-processamento usada pelo *YOLO* para filtrar as bounding boxes previstas. Neste processo é escolhida a box com maior grau de confiança e suprime-se as boxes sobrepostas com menor grau de confiança. O limite, isto é, o valor de confiança mínimo necessário para que uma bounding box seja considerada válida, usado nesta técnica pode afetar que bounding boxes são selecionadas, resulta em variações nas previsões finais.

Posto isto, submetemo-nos à criação de um cenário de testes para definir qual a margem ideal a definir para compensar a variação da posição da bounding box em objetos parados. Deste modo garantimos uma deteção de movimento mais rigorosa, e consequentemente alertas de segurança mais precisos e confiáveis.

### Condições do teste

Para avaliar qual a margem que confere a melhor exatidão dos resultados, foram simulados 17 cenários de testes variando os objetos a detetar e a distância a que se encontram do dispositivo de gravação. A posição da camara manteve-se a mesma durante todo o procedimento bem como o cenário onde se encontraram os objetos a detetar. Este cenário consistia num dia de sol, céu limpo com presença de sombra onde os objetos foram detetados.

Os cenários de teste foram os seguintes:

* Mochila (backpack) a 5,10 e 15 metros
* Bola (sports ball) a 5,10 e 12 metros
* Pessoa (person) a 5,10,15,20,25,30,35,40,50,60 e 70 metros

### Resultados obtidos:

Os gráficos seguintes representam, à esquerda, a distância entre os cantos inferior esquerdo (canto 1) do frame anterior e o atual e a distância do canto superior direito (canto 2) entre o frame anterior e atual à direita.

### Class “backpack”

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.1 - Gráfico cantos backpack a 5M

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.2 - Gráfico cantos backpack a 10M



Figura 5.3 - Gráfico cantos backpack a 15M

### Class “sports ball”

Uma imagem com diagrama, file, texto, Gráfico

Descrição gerada automaticamente

Figura 5.4 - Gráfico cantos bola a 5M

Uma imagem com texto, file, Gráfico, diagrama

Descrição gerada automaticamente

Figura 5.5 - Gráfico cantos bola a 10M

O próximo teste foi feito a 12 metros porque a 15 a bola já não era detetada.

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.6 - Gráfico cantos bola a 12M

### Class “Person”

Uma imagem com file, texto, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.7 - Gráfico cantos person a 5M

Uma imagem com file, diagrama, texto, Gráfico

Descrição gerada automaticamente

Figura 5.8 - Gráfico cantos person a 35M

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.9 - Gráfico cantos person a 70M

### Análise dos resultados

Após teste dos vários cenários, seguidos da análise e discussão dos resultados obtidos, consideramos 4 px como margem ideal a aplicar para desprezar as possíveis variações das bounding boxes geradas pelo modelo de deteção usado em objetos parados. Isto deve-se ao facto dos maiores desvios obtidos serem na ordem dos 3px, sendo o máximo deles 3.15 no caso de teste backpack a 5 metros. Sendo 4 o próximo inteiro, 4px cobre todos os desvios que obtivemos na fase de testes e permite-nos desprezar as possíveis variações da bounding box num objeto parado.

## Teste da arquitetura para recolher frames de dispositivos

Serve o presente teste para testar qual a arquitetura que melhor se adequa à nossa aplicação e aos seus requisitos. Para isso temos 4 modelos para teste sendo eles os seguintes.

### Grab & Retrieve

Neste modelo, após inicializarmos as câmaras com o videCapture de cada dispositivo, criamos uma thread de captura por dispositivo. Esta thread apenas agarra (grab) continuamente frames do dispositivo para posterior descodificação e análise do frame. Na thread principal, de X em X segundos (definido pelo utilizador) chamamos a função retrieveFrames que percorre o vetor das câmaras previamente inicializadas e recolhe (retrieve) o último frame agarrado (grabbed) pela thread de captura do dispositivo associado descodificando-o. De seguida enviamos cada frame para a função *YOLO* para o processar.

### Parallel

Nesta arquitetura, para cada dispositivo de captura selecionado, é criada uma thread que lê, descodifica e envia o frame ao *YOLO* para detetar objetos na imagem.

### Open & Close

Neste padrão (ausente de threads) cada dispositivo de captura é inicializado, captura o frame, processa-o para deteção através do *YOLO* e fecha o dispositivo, seguindo para o próximo continuamente, N vezes, sendo N o número de dispositivos selecionado.

### Open & Close threading

Neste modelo, inicializa-se os dispositivos e cria-se uma thread por dispositivo para ler um frame. Após todas as threads terem lido um frame, cria-se uma thread por frame para o processar no *YOLO*. Finalmente, depois das threads terminarem o processamento, fecham-se os dispositivos inicializados no início.

### Condições de teste

Para comparar os vários modelos de implementação, utilizamos a biblioteca “psutil” do python que nos permite avaliar a % de CPU usage bem como a % de RAM usage durante todo o processo. Para avaliar os cenários, utilizamos as mesmas 5 câmaras, das quais 2 locais e 3 por IP e definimos 40 iterações para avaliar o tempo de execução.

### Resultados obtidos

### Grab & Retrieve

Neste padrão não foi possível concluir os testes porque ao usar câmaras IP em vez de locais no videoCapture, o retrieve não consegue descodificar (retrieve) o frame agarrado (grabbed) anteriormente, pelo que excluímos das opções.

### Parallel

Uma imagem com texto, file, diagrama, captura de ecrã

Descrição gerada automaticamente

Figura 5.10 - Gráfico performance parallel

Ao analisar a percentagem elevada de CPU usage nesta arquitetura, acrescentamos dois novos casos de teste para tentar diminuir a % de CPU usage sacrificando tempo de execução. Para isso, adicionamos um time.sleep de 5 e 10 segundos.

### Parallel 5 sec

Uma imagem com texto, file, captura de ecrã, diagrama

Descrição gerada automaticamente

Figura 5.11 - Gráfico performance parallel 5 segundos

### Parallel 10 sec

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.12 - Gráfico performance parallel 10 segundos

### Uma imagem com texto, file, Gráfico, diagrama Descrição gerada automaticamenteOpen & Close

Figura 5.13 - Gráfico performance open & close

### Open & Close threading

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.14 - Gráfico performance open & close threading

### Análise dos Resultados

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | Tempo execução | Máx – Init CPU % | Máx – Init RAM % | Mean – Init CPU % | Mean – Init RAM % |
| Parallel | 1m 44s | 100 | 10,80 | 85,88 | 10,03 |
| Parallel 5 sec | 5m 24s | 100 | 8,50 | 34,55 | 7,67 |
| Parallel 10 sec | 7m 39s | 35,20 | 8,30 | 20,46 | 4,11 |
| Open & Close | 17m 58s | 28,30 | 19,50 | 9,86 | 13,53 |
| Open & Close threading | 10m 25s | 32,40 | 13 | 18,14 | 8,77 |

Tabela 5 – Tabela comparativa dos resultados obtidos do teste das várias arquiteturas

Através dos dados retirados podemos criar a seguinte tabela:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Menor tempo execução | Maior tempo  Execução | Menor CPU usage | Maior CPU usage | Menor RAM usage | Maior RAM usage |
| Parallel | Open & Close | Open & Close | Parallel | Parallel 10 sec | Open & Close threading |

Tabela 6 – Tabela representativa dos limites dos parâmetros

Neste projeto, priorizamos o menor custo computacional em relação ao tempo de execução. Dito esto, excluímos o modelo Parallel pois utiliza o esforço máximo do CPU e o Open & Close threading por representar o maior esforço sobre a RAM.

Entre o Parallel 5 sec e o Parallel 10 sec, o último representa o melhor comparativamente porque demora 1,42 vezes mais para concluir o teste mas utiliza menos 1,69 vezes o CPU e 1,87 vezes menos a RAM.

O Open & Close é o que apresenta menor custo para o processador (9,86%) mas demora demasiado tempo em relação ao Parallel 10 sec que utiliza 20,46% do processador, o que representa 2,08 vezes mais que o anterior, mas acaba o teste 2,35 vezes mais cedo, o que representa um rácio maior de tempo / CPU usage, pelo que também exclui o Open & Close em comparação ao Parallel 10 sec.

### Conclusão

Após análise e discussão dos resultados obtidos, escolhemos a arquitetura Parallel 10 sec como a mais adequada para a nossa aplicação por ser o mais equilibrado quando aplicamos os nossos critérios.

## Teste de limite de dispositivos

Este teste, diferente dos anteriores, é um teste de sistema na medida em que testamos o comportamento da aplicação como o conjunto de todas as funcionalidades interligadas e não a componentes isolados como os testes anteriores. Neste teste vamos adicionar dispositivos à aplicação, interagir com eles e testar vários cenários para perceber qual o limite da mesma.

O objetivo deste este teste é avaliar o comportamento da aplicação, bem como a taxa de esforço para o processador e de memoria RAM quando a aplicação é levada ao seu limite. É fundamental testar as capacidades máximas da aplicação para documentar o máximo de dispositivos recomendado para o utilizador final perceber as capacidades da aplicação e quando deve esperar que a mesma possa a falhar.

### Condições do teste

Para avaliar o limite da aplicação vamos analisar o comportamento da mesma em 4 cenários diferentes. O primeiro cenário é adicionar câmaras e capturar frames em real-time (funcionalidade Live). O segundo é adicionar câmaras e capturar frames com um delay de 1 segundo para todas as câmaras. O terceiro é com delay de 5 segundos e no quarto o delay é de 10 segundos. Vamos utilizar estes tempos para delay por serem os 4 mais baixos (e consequentemente computacionalmente mais exigentes) que existem disponíveis na aplicação.

O teste foi conduzido num computador com as seguintes especificações. Processador AMD RYZEN 7 4800H 2.9GHz 8 núcleos 16 threads, 16gb de RAM DDR4 e sistema operativo Windows 10.

### Resultados obtidos

Os gráficos que vamos apresentar têm todos a mesma estrutura. Do lado esquerdo podemos observar a % de CPU à medida que capturamos frames e do lado direito a % de RAM usage. Os picos inferiores nos gráficos relativos ao processador representam os momentos em que são adicionados novos dispositivos.

O primeiro cenário é testar a capacidade máxima com câmaras a capturar em real-time o que originou o seguinte gráfico.

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 5.15 – Gráfico stress LIVE

O gráfico começa a ser desenhado no momento que o primeiro frame é capturado, logo o frame 1 representa o primeiro dispositivo adicionado que atinge os 50% de CPU usage. O segundo dispositivo é adicionado por volta do frame 20 que dispara para os 100% de CPU. Existem ainda 2 dispositivos adicionados, um por volta do frame 60 e outro depois do 200, mas em termos de CPU não acrescenta informação adicional uma vez que já atingiu os 100% logo no 2 dispositivo. A RAM parece não ser muito afetada negativamente, tendo uma amplitude de apenas 9,30%.

Além dos dados extraídos do gráfico, verificamos que a partir da 3 câmara Live adicionada inclusive, o delay apresentado começa a apresentar-se significativo e a partir da 5 a aplicação começa a não responder devido à utilização excessiva de recursos do sistema.

No segundo cenário adicionamos câmaras com um delay de 1 segundo para a captura de novos frames em cada dispositivo o que deu origem ao seguinte gráfico.

Uma imagem com diagrama, texto, file, Gráfico

Descrição gerada automaticamente

Figura 5.16 - Gráfico stress 1 segundo

Neste gráfico os dispositivos adicionados foram nos frames 1, 17 e 65. Podemos observar que atingimos os 100% de CPU usage a partir da terceira câmara.

No terceiro cenário adicionamos dispositivos que capturam frames a cada 5 segundos. Obtivemos o seguinte gráfico.

Uma imagem com file, diagrama, Gráfico, texto

Descrição gerada automaticamente

Figura 5.17 - Gráfico stress 5 segundos

Neste cenário verificamos que a partir do sexto dispositivo adicionado, o processador tem alguns picos para os 100% e a partir do oitavo dispositivo, o gráfico verifica-se a maior parte do tempo localizado nos 100% de CPU usage. Também podemos aferir que a partir do décimo primeiro dispositivo adicionado o delay das imagens apresentadas na aplicação após processamento é notável.

O quarto e último cenário de teste foi feito com dispositivos a capturar frames a cada 10 segundos e originou o seguinte gráfico.

Uma imagem com file, Gráfico, diagrama, texto

Descrição gerada automaticamente

Figura 5.18 - Gráfico stress 10 segundos

Neste cenário verificou-se que a partir de dez câmaras adicionadas, o processador atinge os 100% de CPU usage periodicamente e a partir da décima sexta câmara o gráfico entra em regime permanente atingindo os 100%.

### Análise dos resultados

Após teste dos cenários mais críticos e pesados para o software, podemos prever as seguintes recomendações.

Para dispositivos de captura em live recomenda-se o uso de no máximo 2 dispositivos para manter os frames atualizados em real-time. A partir de 5 dispositivos podemos esperar um mau funcionamento da aplicação.

Para dispositivos de captura com delay de 1 segundo recomenda-se o uso de no máximo 3 dispositivos de modo a manter o processador fora dos 100% de usage.

Para dispositivos de captura com delay de 5 segundos recomenda-se o uso de no máximo 8 dispositivos de modo a manter o processador fora dos 100% de usage e no máximo 11 se queremos garantir que os frames atualizam sem delays significativos.

Para dispositivos de captura com delay de 10 segundos recomenda-se o uso de no máximo 16 dispositivos de modo a manter o processador fora dos 100% de usage.

## Teste de geração de alerta

É importante perceber se a nossa aplicação final perdeu qualidade na tarefa de detetar objetos em relação aos testes conduzidos ao modelo de deteção antes de estar integrado na aplicação. Este teste permite-nos perceber se os alertas estão a ser devidamente emitidos nas condições pretendidas e sem erros. Para a realização do teste vamos utilizar 2 objetos de diferentes dimensões a 3 distâncias distintas e ver se são emitidos alertas quando pretendido. O objetivo é deixar um objeto parado, cronometrar o tempo passado e ver quando (e se) o alerta é emitido em 6 cenários resultantes dos 2 objetos escolhidos a 3 distâncias.

### Condições do teste

Os objetos utilizados para o teste são uma cadeira de dimensões 45x70x57 cm e um carro de 434x173x140 cm. Vamos analisar a capacidade da aplicação emitir alertas a 5, 10 e 15 metros. Nas configurações do dispositivo vamos definir o tempo para emitir alerta 20 segundos parado em todos os cenários de teste e a captura e processamento de frames com 1 segundo de intervalo.

O dispositivo de captura utilizado é a câmara traseira de um iPhone 14. Para condições atmosféricas encontra-se um dia nublado com alguma incidência solar. O cenário para deteção dos objetos é o fim de uma rua ampla às 14:42 GMT.

### Resultados obtidos

Para análise dos resultados vamos verificar o momento em que foi emitido o alerta na lista de alertas. Podemos ver isto na coluna do meio que começa com “Hora: “ seguido do timestamp do alerta. De acordo com as configurações descritas anteriormente, e se tudo estiver a funcionar como espectado, é esperado que a cada 20 segundos seja emitido um alerta.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamenteO primeiro cenário é uma cadeira a 5 metros da câmara, representado pela Figura 5.19.

Figura 5.19 - Lista alertas do objeto "chair" a 5 metros

O segundo cenário é a mesma cadeira a 10 metros, representado pela Figura 5.20.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura 5.20 - Lista alertas do objeto "chair" a 10 metros

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamenteO terceiro cenário é o último da cadeira e esta encontra-se a 15 metros, representado pela Figura 5.21.

Figura 5.21 - Lista alertas do objeto "chair" a 15 metros

Passando para o objeto seguinte, um carro. O quarto cenário é um carro a 5 metros, representado pela Figura 5.22.

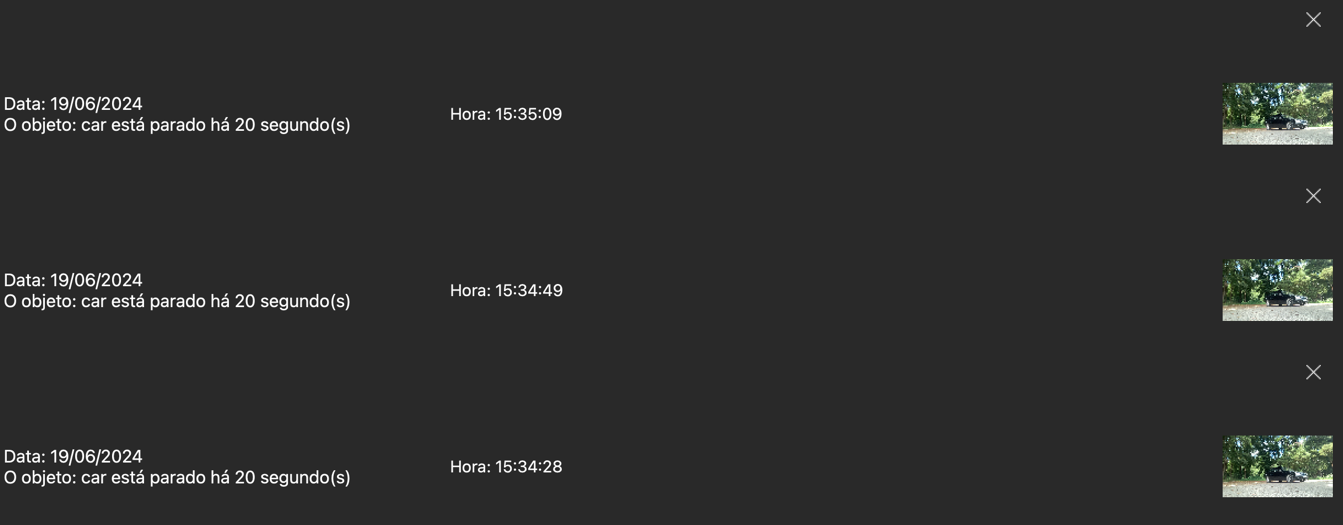


Figura 5.22 - Lista alertas do objeto "car" a 5 metros

O quinto cenário é o carro a 10 metros, representado pela Figura 5.23.

Figura 5.23 - Lista alertas do objeto "car" a 10 metros

Uma imagem com captura de ecrã, texto

Descrição gerada automaticamente

O sexto e último cenário é o carro a 15 metros, representado pela Figura 5.24.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura 5.24 - Lista alertas do objeto "car" a 15 metros

### Análise dos resultados

Todos os resultados mostram alertas emitidos a 20 segundos, como esperado, com 1 segundo de tolerância. Este segundo pode ser desprezado porque os frames só estão a ser atualizados a cada 1 segundo, e o alerta pode ser emitido com esse segundo de atraso caso o tempo para emitir o alerta tenha coincidido no intervalo em que o frame não é atualizado. Nestes casos o alerta é emitido na próxima passagem pela função que processa a imagem.

O único alerta que não vai de acordo com o esperado é no segundo cenário onde se monitoriza a cadeira a 10 metros, mas deve-se ao facto da câmara ter-se movido com o vento às 15:08:30 o que interrompeu a contagem de frames em que o objeto estava parado e explica o intervalo sem alerta entre os alertas às 15:08:24 e 15:08:51. Podemos com certeza afirmar isto uma vez que durante o teste estávamos a monitorizar o output do objeto estar parado ou em movimento.

Podemos concluir que a aplicação mantém a capacidade de detetar objetos parados com precisão e emite os alertas quando espectado.

# Conclusão

Neste relatório, apresentamos o desenvolvimento de uma aplicação de vigilância para a deteção de objetos e emissão de alertas. O projeto teve como objetivo principal a criação de uma ferramenta eficaz para a segurança, capaz de identificar objetos imóveis e considerados suspeitos. Com o auxílio de Computer Vision, nomeadamente o modelo de deteção *Yolo*V8 conseguimos desenvolver uma aplicação precisa como foi verificado nos testes à mesma.

Os objetivos estipulados foram atingidos, desenvolvemos com sucesso uma aplicação que consegue identificar objetos e alertar quando estes estão parados com alta precisão. Todos os requisitos idealizados foram implementados e o sistema comporta-se como esperado. Os testes realizados mostraram que a aplicação é capaz de operar em tempo real e responder de forma eficaz a diferentes cenários de vigilância.

No decorrer do projeto adquirimos conhecimentos valiosos sobre modelos de deteção de objetos, como estes operam e quais as diferenças entre os vários que podemos recorrer. Consolidamos a nossa capacidade de desenvolver um sistema na forma de aplicação desktop capaz de responder às necessidades do utilizador.

Numa futura melhoria desta aplicação sugerimos adicionar a funcionalidade de adicionar objetos novos para deteção além dos pré-treinados pelo modelo. Com esta melhoria, permitimos ao utilizador treinar qualquer objeto do seu interesse e passar a detetá-lo nos seus dispositivos o que tornaria a aplicação ainda mais interessante.

# Bibliografia

Altamira team. (2019, abril 9). **Scrum vs. other methodologies**. Retirado de <https://www.altamira.ai/blog/scrum-vs-other-methodologies/>

Anwar A. (2019, junho 7). **Difference between AlexNet, VGGNet, ResNet, and Inception**. Retirado de <https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96>

ArcGIS Developers. (s.d.). **How SSD works**. Retirado de <https://developers.arcgis.com/python/guide/how-ssd-works/>

BasicAI Marketing Team. (s.d.). **Leading object detection algorithms in 2023: A comprehensive overview**. Retirado de <https://www.basic.ai/blog-post/leading-object-detection-algorithms-in-2023%3A-a-comprehensive-overview>

Bhalerao C. (2023, novembro 2). **A Deep Dive Into** **Non-maximum suppression (NMS)**. Retirado de <https://builtin.com/machine-learning/non-maximum-suppression>

Bharath K. (2024, abril 15). **Object detection algorithms and libraries**. Retirado de <https://neptune.ai/blog/object-detection-algorithms-and-libraries>

Boesch G. (2023, novembro 1). **What´s the difference between machine learning and deep learning?.** Retirado de <https://viso.ai/deep-learning/deep-learning-vs-machine-learning/>

Boesch G. (2024, Janeiro 4). **Object detection in 2024: The Definitive Guide**. Retirado de <https://viso.ai/deep-learning/object-detection/>

Fixe Consulting. (2017, abri 15). **Agile x Scrum x Lean: O que cada um é**. Retirado de <https://fixeconsulting.com/2017/04/agile-x-scrum-x-lean-o-que-cada-um-e/>

GeeksforGeeks. (2024, junho 19). **Difference between Scrum and XP**. Retirado de <https://www.geeksforgeeks.org/difference-between-scrum-and-xp/>

Goetz C. (2023, maio 7). **Agile vs. traditional: Which method is right for you?**. Retirado de <https://www.linkedin.com/pulse/agile-vs-traditional-which-method-right-you-carlos-goetz/>

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. (2015, dezembro 10). **Deep Residual Learning for Image Recognition**. Retirado de <https://arxiv.org/pdf/1512.03385v1>

Keita Z. (2022, setembro). **YOLO object detection explained**. Retirado de <https://www.datacamp.com/blog/yolo-object-detection-explained>

Kissflow. (2023, novembro 16). **Kanban methodology: The Simplest Agile Framework**. Retirado de <https://kissflow.com/project/agile/kanban-methodology/>

Kumar S. (2024, maio 29). **What is Extreme programming (XP)**. Retirado de <https://www.geeksforgeeks.org/software-engineering-extreme-programming-xp/>

Laoyan S. (2024, fevereiro 2). **What is** **Agile methodology?**. Retirado de <https://asana.com/pt/resources/agile-methodology>

LAPix. (s.d.). **Deep Learning: Deteção de objetos em imagens**. Retirado de <https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deteccao-de-objetos-em-imagens/>

Linhares G. (2022, maio 22). **GUIs com Python e PyQt5: Introdução**. Retirado de <https://www.linkedin.com/pulse/guis-com-python-e-pyqt5-introdu%C3%A7%C3%A3o-guilherme-trevisan-linhares/?originalSubdomain=pt>

Lucidchart. (s.d.). **Agile vs. Waterfall vs. Kanban vs. Scrum**. Retirado de <https://www.lucidchart.com/blog/agile-vs-waterfall-vs-kanban-vs-scrum>

Matviiok T. (2020, dezembro 22). **How to choose your optimal development methodology**. Retirado de <https://keenethics.com/blog/how-to-choose-your-optimal-development-methodology>

Mostafid T. (2023, dezembro 11). **Overview of VGG16, Xception, MobileNet, and ResNet50 neural networks**. Retirado de <https://medium.com/@t.mostafid/overview-of-vgg16-xception-mobilenet-and-resnet50-neural-networks-c678e0c0ee85>

Potrimba P. (2023, setembro 25). **What is DETR?**. Retirado de <https://blog.roboflow.com/what-is-detr/>

Olorunshola1 O. et al. (2023, setembro 20). **Comparative Study of Some Deep Learning Object Detection Algorithms: R-CNN, FAST RCNN, FASTER R-CNN, SSD, and YOLO**. Retirado de <https://nilejeas.com/fulltext/262-1681947210.pdf?1717505346>

ProductPlan. (s.d.). **Scrumban**. Retirado de <https://www.productplan.com/glossary/scrumban/>

Raeburn A. (2024, fevereiro 13). **What is Extreme programming (XP)**. Retirado de <https://asana.com/pt/resources/extreme-programming-xp>

Redmon J. et al. (2015, junho 8). **You Only Look Once: Unified Real-Time Object Detection**. Retirado de <https://arxiv.org/abs/1506.02640>

Rohini G. (2021, setembro 22). **Everything you need to know about VGG16**. Retirado de <https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918>

Rustamy F. (2023, agosto 20). **Detection transformer (DETR) vs YOLO for object detection**. Retirado de <https://medium.com/@faheemrustamy/detection-transformer-detr-vs-yolo-for-object-detection-baeb3c50bc3>

Saha S. (2018, dezembro 15). **A comprehensive guide to convolutional neural networks—The ELI5 way**. Retirado de <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

Simplilearn. (2023, outubro 11). **Image processing**. Retirado de <https://www.simplilearn.com/image-processing-article>

Srinivasan A. (2019, setembro 7). **Stochastic gradient descent - clearly explained**. Retirado de <https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31>

Yadoshchuk V. (2024, abril 29). **Why use Python for AI and Machhine Learning**. Retirado de <https://waverleysoftware.com/blog/python-for-ai-and-ml/>

Zaidi S. et al. (2022, junho 30). **A survey of modern deep learning based object detection models**. Retirado de <https://www.sciencedirect.com/science/article/pii/S1051200422001312#se0590>