NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

JOÃO RAFAEL MARQUES DUARTE

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

DESENHO E DESENVOLVIMENTO DE SOLUÇÃO MÓVEL COM RECURSO A PROCESSAMENTO DE IMAGEM PARA OPERAÇÕES PORTUÁRIAS

MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Universidade NOVA de Lisboa
Novembro, 2021

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

DESENHO E DESENVOLVIMENTO DE SOLUÇÃO MÓVEL COM RECURSO A PROCESSAMENTO DE IMAGEM PARA OPERAÇÕES PORTUÁRIAS

JOÃO RAFAEL MARQUES DUARTE

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Orientador: André Dionisio Bettencourt da Silva Parreira Rocha
Professor Auxiliar Convidado, FCT-NOVA - Universidade Nova de Lisboa

Coorientador: Ricardo Alexandre Fernandes da Silva Peres
Professor Auxiliar Convidado, FCT-NOVA - Universidade Nova de Lisboa

Júri

Presidente: Name of the committee chairperson
Full Professor, FCT-NOVA

Arguentes: Name of a rapporteur
Associate Professor, Another University
Name of another rapporteur
Assistant Professor, Another University

MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Universidade NOVA de Lisboa
Novembro, 2021

Desenho e desenvolvimento de solução móvel com recurso a processamento de imagem para operações portuárias

Aos meus pais,

Agradecimentos

Todo o apoio recebido ao longo da realização deste projeto teve um impacto essencial no sucesso do mesmo.

Aos Orientadores deste projeto, André Rocha e Ricardo Peres, pela mestria na orientação e acompanhamento, não só durante a realização deste projeto, mas também ao longo do meu percurso académico em diversas UC's.

À Opertri , pela oportunidade, pelo disponibilização de recursos, e pelo acompanhamento ao longo do desenvolvimento do projeto.

À Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa, um enorme agradecimento por me ter acolhido há 5 anos e me ter proporcionado uma formação exemplar na vertente académica, pessoal e social.

Aos meus pais, os grandes impulsionadores do que sou hoje, fontes de sabedoria e críticas construtivas, sem as quais seria impossível o sucesso deste percurso.

Um agradecimento especial à Flávia não só pelo amor e carinho, mas também pela crítica, por todas as aventuras e vivências, por tudo.

Por fim, mas não menos importante, aos amigos, à família que ganhei, a todos os que fizeram, de uma forma ou outra, parte do Gang do VII. Aos que me acompanharam nos momentos de vitoria e de derrota, de alegria e de tristeza, com o mesmo amor que eu sempre de capa negra aos ombros.

A todos, um grande obrigado.

"Let him think that I am more man than I am and I will be so."
(Ernest Hemingway)

Resumo

Nas últimas décadas, existiu um crescimento exponencial da tecnologia, mais precisamente na indústria. Sendo que a logística é um dos pilares do funcionamento da indústria mundial, esta teve de evoluir, tecnologicamente, para satisfazer as necessidades industriais. Esta tem sido feita de forma heterogénea, o que leva a que os avanços tecnológicos cheguem a certas regiões mais rápido do que a outras.

Neste sentido,a Opertri, a empresa responsável pela gestão de entradas e saídas de contentores no porto da Horta, ilha do Faial, faz, ainda, a recolha dos identificadores dos contentores de carga de forma manual, em papel. Para simplificar o processo de recolha dos identificadores e torná-lo mais eficiente e avançado tecnologicamente, desenvolveu-se uma aplicação móvel que pode ser utilizada em qualquer dispositivo móvel com acesso a Internet, que através da câmara ou galeria faz a captação de imagens e as envia para um servidor , o qual contem um algoritmo que utilizando métodos de *Deep Learning* (DL) faz inicialmente a localização do ID e posteriormente o seu reconhecimento. É retornado para a aplicação o ID, onde este pode ser verificado manualmente pelo operário e seguidamente é feito o envio automático para o servidor da empresa.

Palavras-chave: Classificação, ISO:6346, Visão Computacional, *Deep Learning*, *OpenCv*, *React Native*.

Abstract

In recent decades, there has been an exponential growth in technology, more precisely in industry. As logistics is one of the pillars of the functioning of global industry, it had to evolve technologically to meet industrial needs. This evolution has been done in a heterogeneous manner, which means that technological advances reach certain regions faster than others.

In this sense,Opertri, the company responsible for managing the entry and exit of containers at the port of Horta, Faial Island, still collects the identifiers of the cargo containers manually on paper. To simplify the process of collecting identifiers and make it more efficient and technologically advanced,a mobile application was developed that can be used on any mobile device with Internet access, which through the camera or gallery captures images and sends them to a server, which contains an algorithm that, using DL methods,initially locating the ID and subsequently recognizing it. The ID is returned to the application, where it can be manually verified by the operator and then automatically sent to the company's server.

Keywords: Classification, ISO 6346, Computer Vision, *Deep Learning*, *OpenCv*, *React Native*.

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Siglas

ABS *Anti-lock Braking System*
ALPR *Automatic License Plate Recognition*
APP Aplicação Móvel

BIC *Bureau International des Containers*

CNN *Convolutional Neural Network*
CPS *Cyber-Physical System*

DL *Deep Learning*

FCN *Fully Convolutional Network*

GPS *Global Positioning System*
GPU Graphics Processing Unit

I4.0 Indústria 4.0
ID Identificador
IoT *Internet of Things*
IoU *Intersection over Union*
ISO *International Organization for Standardization*
IT *Information Technology*
ITS *Intelligent Transportation System*

ML *Machine Learning*

OCR *Optical Character Recognition*
OT *Operational Technology*

PIB Produto Interno Bruto

RFID *Radio-Frequency Identification*
RN *React Native*
RNN *Recurrent Neural Networks*
ROI *Region of Interest*

SO Sistemas Operativos

TMS *Transport Management System*

UI/UX *User Interface / User Experience*

WMS *Warehouse Management System*

YOLO *You Only Look Once*

1

Introdução

Neste capítulo é apresentada a motivação, onde far-se-á um enquadramento atual dos sistemas e processos logísticos. Será, também, decifrado qual o problema que esta dissertação pretende resolver e, ainda, referidos os principais objetivos desta dissertação. A última secção deste capítulo apresenta uma sucinta explicação da organização estrutural desta dissertação, isto é, a temática abordada em cada um dos capítulos.

1.1 Motivação

Com a evolução da tecnologia nos últimos séculos, com maior incidência no último e no presente, a indústria evoluiu de forma bastante acentuada comprovado com a existência de quatro revoluções industriais. Estas revoluções criaram necessidades nas economias que rodeiam a indústria, como o caso da logística e, mais precisamente dos processos logísticos, que tiveram de evoluir de forma a fornecer matérias-primas com a rapidez necessária e, também, de maneira a ser possível entregar o produto final ao cliente.

O setor da logística, um dos mais importantes da economia global, tem vindo a crescer na Europa, beneficiando da recuperação financeira da maioria dos países europeus. Em 2016 as receitas deste setor pesavam entre 6 e 10% do PIB em grande parte dos países europeus com é visível na Figura 1.1. Segundo estudos da *Transparency Market Research* (TMR) em 2016 o valor de mercado mundial deste setor seria de 8.1 milhões de milhões de dólares e era previsto um aumento para 15.5 milhões de milhões de dólares no ano de 2023 [1].

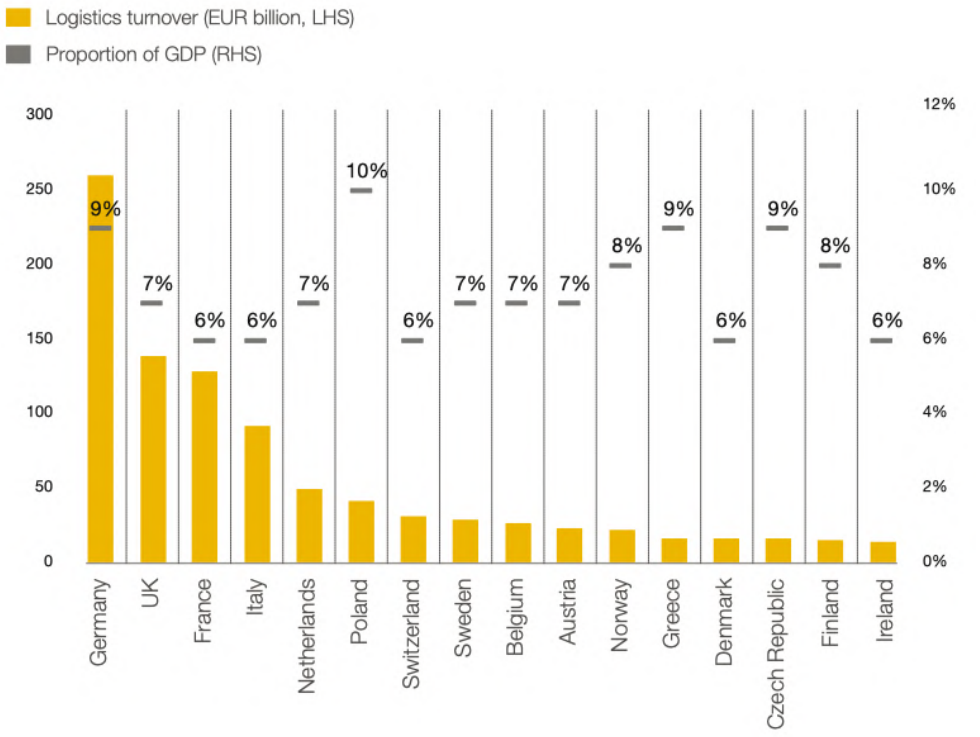


Figura 1.1: Faturação na área da Logística e proporção do PIB [2].

Os valores apresentados na figura anterior são demonstrativos da importância do setor na economia europeia , pelo que se deve proporcionar uma evolução continua do setor logístico, optimizando processos e tornando-os mais avançados tecnologicamente.

1.2 Problema e Objetivos

A evolução tecnológica chegou aos diferentes locais do mundo em épocas distintas, o que faz com que a tecnologia utilizada em vários pontos do planeta seja bastante diferente.

Um processo logístico bastante importante é o registo dos contentores de carga a serem carregados ou descarregados aquando da chegada de um navio de mercadorias a um porto. Existem portos bastante avançados tecnologicamente, onde o registo dos contentores é feito de forma automática no momento em que são colocados ou retirados do navio, como é o caso do porto de Antuérpia onde em 41 equipamentos de carga e descarga de navios é utilizado o sistema *BoxCatcher* para o reconhecimento do ID do contentor a ser movimentado[3]. Por outro lado, existem portos onde o processo ainda é feito de forma manual, ou seja, um operário do porto percorre os contentores, registando no papel os caracteres de cada um sendo que, posteriormente, são introduzidos, igualmente de forma manual, no sistema informático. Este é o caso da Opertri[4] situada no Porto da Horta, na ilha do Faial, do Arquipélago dos Açores.

Cada contentor, para ser considerado legal, deve conter em cinco, das suas seis faces,

um sistema identificador standardizado. Este sistema de identificação consiste num código alfanumérico com quatro letras seguidos de sete números. O sistema foi inventado na década de 70 pelo *Bureau International des Containers* (BIC) e introduzido em 1984 pela *International Organization for Standardization* (ISO). Este foi atualizado em 1995 e passou a denominar-se de *ISO 6346:1995*.

Segundo a ISO e o BIC em [5] e percetível no exemplo da Figura 1.2, o identificador consiste em:

- **Prefixo (BIC *code*)** – três letras maiúsculas que identificam o dono, ou principal operador do contentor, devidamente registado no BIC.
- **Categoria** – uma letra maiúscula que identifica a categoria do contentor. Neste sentido, existem três categorias diferentes:
  - **U** - Para todos os contentores de carga.
  - **J** - Para equipamentos que possam ser acoplados a um contentor, como por exemplo os contentores frigoríficos.
  - **Z** - Para reboques aos quais são acoplados um contentor.
- **Número de Série** – seis dígitos com um número de série que ainda não se encontre registado.
- **Dígito de Validação** – um dígito de validação que fornece um meio de validação das precisões de gravação e transmissão do código do proprietário e do número de série.

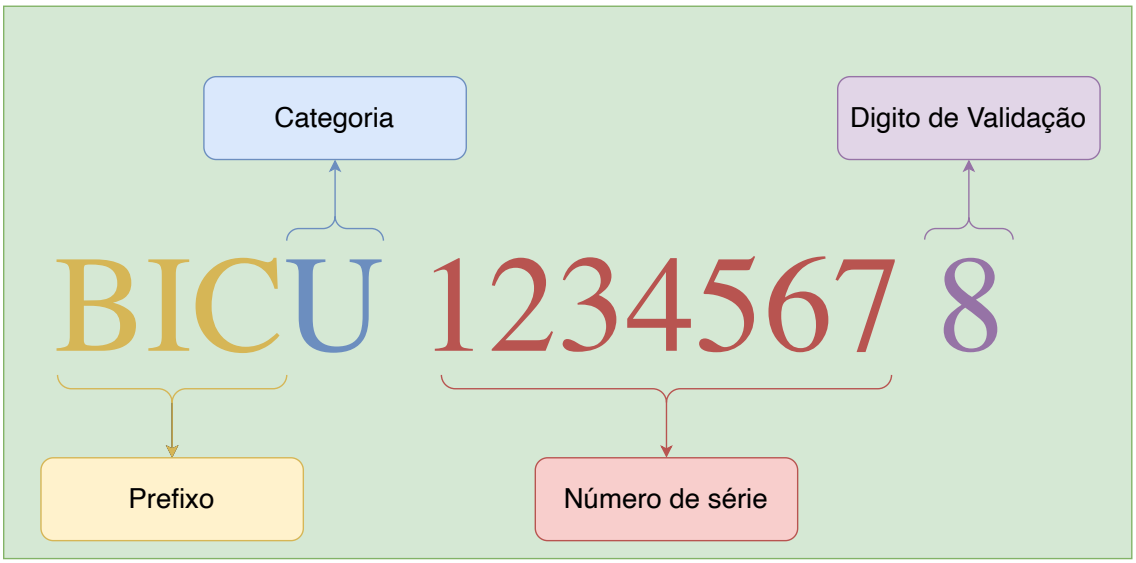


Figura 1.2: Exemplo de um Identificador.

Esta dissertação tem como objetivo desenhar e desenvolver um sistema que possa ser utilizado num dispositivo móvel e, que recorrendo a processamento de imagem, seja capaz de reconhecer o identificador específico de cada contentor e o envie para o sistema informático da empresa de operações portuárias do porto da Horta de modo a melhorar a eficiência desta mesma gestão, facilitando a tarefa do operário encarregue de realizar o registo dos contentores e, ainda, reduzindo a probabilidade de existir erro humano neste mesmo registo.

1.3 Estrutura do Documento

A presente dissertação seguirá a seguinte estrutura:

- **Estado de Arte** - Neste capítulo é feito uma contextualização geral da evolução da logística e da sua importância. Apresenta referências à temática dos Sistemas Ciber-Físicos e, por fim, é feita uma abordagem completa e atenta ao Reconhecimento Ótico de Caracteres onde, inicialmente, se faz uma abordagem histórica ao tópico, uma descrição do tipos e fases desta tecnologia seguidas de uma comparação entre as abordagens tradicionais e abordagens que usam *Machine Learning* (ML) e DL;
- **Arquitetura do sistema** - Apresentam-se as descrições iniciais dos requisitos funcionais e não funcionais da dissertação. Segue-se a exposição da arquitetura do sistema, que contem as arquiteturas tanto da aplicação como do *Back-end*.
- **Implementação** - Neste capítulo são apresentados os métodos e tecnologias utilizados em cada um dos componentes do sistema.
- **Testes e Validação** - Este capítulo apresenta uma avaliação do sistema desenvolvido partindo de testes qualitativos e quantitativos realizados ao sistema como um todo, e especialmente, a cada componente.
- **Conclusão** - Aqui apresentam-se as conclusões retiradas pela análise do sistema, já totalmente finalizado, e sugestões de melhoria.

2

Estado Da Arte

Neste Capitulo um apanhado do Estado da Arte de alguns tópicos como a Logística e os Sistemas Ciber-Físicos. É, ainda, feita uma abordagem completa e atenta ao Reconhecimento Ótico de Caracteres onde, inicialmente, se faz uma abordagem histórica ao tópico e uma descrição do tipos e fases desta tecnologia, seguidas de um comparativo entre as abordagens tradicionais e abordagens que usam Machine Learning (ML) e Deep Learning (DL)

2.1 Logística

A logística evolui de forma a acompanhar os avanços tecnológicos dos setores parceiros como a indústria e o comércio.

2.1.1 Evolução

- **Logística 1.0**

  A primeira alteração percetível no mundo da indústria foi a transferência de trabalho manual para a produção mecanizada e que, como foi tão grande, se considerou uma revolução. Não existe uma data especifica para a tal revolução, mas sabe-se que ocorreu na segunda metade do século XVII no Reino Unido e que se alastrou para a Europa Ocidental e América do Norte nas décadas seguintes. Em 1765 James Watt criou o primeiro motor a vapor eficiente estando esta criação na base da grande revolução. Em 1769 Joseph Cugnot criou aquele que ficou conhecido pelo primeiro automóvel "motorizado". Anos mais tarde estes motores foram adaptados a locomotivas e barcos mudando drasticamente o panorama dos sistemas de transporte criando, assim, os primeiros sistemas logísticos [6].

  A raiz da palavra "Logística" é proveniente do termo Francês "Logis" que significa acomodação de tropas. Desta forma, no início do século XIX, logística foi introduzida e definida pelos militares como o planeamento e movimento de tropas [7].

Segundo Yossi Sheffi & Peter Klaus em [8], a noção original e mais elementar da logística abrange um conjunto de 3 atividades operacionais (movimento, armazenamento e reorganização) que acrescentam valores aos bens. O valor acrescentado inerente a estas atividades baseia-se na otimização dos "3 P's":

- ***Place*** - Adiciona o valor de local aos produtos, movendo-os de locais com menos valor para locais de mais valor, segundo a ótica do cliente.
- ***Period & Pace*** - Adiciona valor temporal aos produtos, armazenando-os, e assim:
  - Mover artigos de períodos em que estão disponíveis (após fabrico ou colheita) para períodos em que exista procura por parte dos clientes.
  - Tornar todos os processos mais eficazes - desacoplar os processos ao longo da cadeia de valor para que, cada um, possa executar o seu ritmo de acordo com as possibilidades financeiras.
- ***Patern*** - Adiciona valor de encomenda aos produtos, organizando-os em quantidades e padrões desejados pelos clientes.

A evolução da Logística, ou do termo "Logística 1.0", começou nas aplicações militares existindo, posteriormente a sua aplicação em organizações empresariais, nos anos 50. Esta aplicação, focava-se apenas na otimização do transporte e da movimentação de mercadorias (distribuição física) dentro de uma organização, que foi potencializada pela maquinação das ferramentas de transporte. Desta forma, a Logística 1.0, satisfez as necessidades industriais sentidas pelos primeiros pelos clientes[9].

- **Logística 2.0**

  Nos anos 60 ocorreu um novo período de mudanças, mudanças estas, que foram consideradas como uma evolução e não como uma revolução do ponto de vista tecnológico, ao contrário do que havia acontecido anteriormente. Foram feitas descobertas de novos materiais como por exemplo o aço, o cobre ou o alumínio, bastante importantes no desenvolvimento de nova maquinaria. Complementando essa evolução, a indústria química teve uma expansão sem precedentes conciliando-se esta com a possibilidade do uso de energia elétrica e do petróleo, verificando-se, desta forma avanços significativos nos serores das comunicações e dos transportes[10].

  Posto isto, foi a globalização da indústria que tornou clara a necessidade de produção em massa. Com esta realidade de produção em massa foram feitos, também, bastantes avanços na área da logística, mais concretamente na forma como a carga era manuseada. Este manuseamento começou a ser feito de forma automática começando, desta forma, a aparecer os primeiros equipamentos de triagem automática

bem como os primeiros armazéns automáticos. À semelhança dos avanços de produção em massa, nos navios de contentores existiram, também, avanços no que toca à mecanização, sendo as zonas portuárias as mais afetadas[10].

No âmbito empresarial, as empresas começaram a negociar entre si, de modo a gerir e coordenar o fluxo de mercadoria, tanto dentro como fora das próprias organizações. Neste período surgiu o termo *Supply Chain Management* que foi definido pelo CSCMP[1] como a junção do planeamento e da gestão de todas as atividades envolvidas no fornecimento e aquisição de produtos, com a conversão desses mesmos produtos e ainda todas as atividades de gestão logística. Tudo isto, inclui ainda a coordenação e colaboração com parceiros sendo eles fornecedores, intermediários, prestadores de serviços a terceiros ou, ainda, clientes. *Supply Chain Management* integra, principalmente, a gestão de procura e oferta dentro e fora de cada empresa.

Para Yossi Sheffi & Peter Klaus em [8] *Supply Chain Management* é definida como logística, adicionando-se, desta forma, um 4º 'P':

- **Process Coordination ou partnerships management** - A gestão das cadeias de abastecimento envolvem a realização de parcerias (dentro da empresa ou entre empresas) e é necessário que se faça a gestão dessas parcerias ou processos.

Concluindo, a Logística 2.0 não se focava exclusivamente no fluxo físico da carga ou na sua automatização, mas sim em aumentar os níveis de otimização para obter melhorias no sistema, levando, isto, a um incremento nos níveis de interação entre empresas.

- **Logística 3.0**

  A terceira revolução industrial iniciou-se em 1964 com o aparecimento do primeiro robot e com a introdução, na indústria, das primeiras máquinas NC[2]. Com a introdução de computadores na indústria, a logística teve, também, uma evolução própria com o aparecimento de vários *Software* como o *Warehouse Management System* (WMS) e o *Transport Management System* (TMS), que surgiram com o objetivo de realizar uma gestão e controlo de todo o processo logístico de forma computacional.[10]

  Ao nível da *Supply Chain* o melhor fornecedor de um mercado global era contratado e pequenas relações eram efetuadas. Já na logística, mais precisamente dentro das empresas, fábricas ou armazéns, o fluxo de carga era feito em linhas automáticas, por empilhadores ou, ainda, por robots com as "rotas" previamente programadas. Em relação à frota fora das instalações, os veículos tinham rotas pré-planeadas, calendarizadas e otimizadas por *Software* e que tratavam do transporte de matérias primas e produtos acabados.

[1] *Council of Supply Chain Management Professionals*
[2] *Numerical Control* - Máquinas controladas de forma numérica.

De acordo com Yossi Sheffi & Peter Klaus em [8] se observássemos as empresas que pareciam ser as mais avançadas e boas praticantes de logística no final da década de 90 veríamos que, para além de *Placing* , *Pacing* e *Patterning* e da mera coordenação de *Partnerships* , essas empresas ainda "programavam" os seus processos e operações de modo a mobilizar fluxos de informação e produtos através dos seus sistemas de uma forma ágil e flexível. Dessa forma, os autores introduziram o 5º 'P':

- ***Pliant flow*** - Ver os negócios como redes de fluxos não relacionados que trazem valor aos seus clientes, e que, por isso se focam em estruturas de rede que permitem um fluxo desobstruído e rápido de bens, informação, dinheiro e ideias que facilmente se ajustam de acordo com as exigências do cliente.

2.1.2 Logística 4.0

A introdução de *Internet of Things* (IoT) e serviços para o ambiente de manufatura despoletou a 4ª revolução industrial[11].

Para Lasi *et al.*[12], esta nova mudança de paradigma na manufatura é resultante do uso de internet, que permite a comunicação entre outras máquinas e seres humanos em tempo real e, ainda, o uso do que é conhecido como *Smart Products and Smart Services* que disponibiliza o avanço da digitalização dentro das fábricas. As futuras *Smart Factorys* permitirão a conexão de todos os elementos envolvidos nos processos de manufatura e disponibilizarão a aplicação de conceitos como adaptabilidade, interconectividade, eficiência e ergonomia.

Segundo o WEF[3] em [13] a 4ª Revolução Industrial é caracterizada pela convergência de tecnologias inovadoras como a robótica avançada, a inteligência artificial, IoT e a realidade virtual e aumentada. Estas estão a transformar os processos de produção e os modelos de negócio ao longo de diferentes indústrias. Os líderes de negócios já não podem focar-se apenas no desenvolvimento e tendências do seu próprio setor, mas sim perceber potenciais transformações e perturbações em todo o leque de fornecedores, clientes e mercados adjacentes.

Esta evolução e o uso destas tecnologias inovadoras estão a ter um grande impacto nos sistemas de manufatura bem como nas cadeias de abastecimento. Todas as tecnologias acima referidas, combinadas, irão abrir um leque de novas oportunidades em várias dimensões do mercado, desde a indústria até ao próprio cliente. Segundo dados da associação alemã de logística BVL[4] apresentados no WEF em 2017, mostram que se prevê uma diminuição de 34,2% nos custos e um aumento de 33,6% nas receitas nos serviços logísticos.

Para Kesheng Wang em [14], a Logística 4.0 define-se como um termo que engloba

[3] *World Economic Forum*
[4] *Bundesvereinigung Logistik*

tecnologias e conceitos na cadeia de valor de uma empresa, sendo que esta cadeia se foca nos passos que criam valor, começando nos fornecedores e acabando nos consumidores, com o objetivo de maximizar o valor entregue. Para chegar a esse objetivo a cadeia avalia as necessidades dos consumidores e não apenas prevê a procura de mercado. O conceito é alcançado por meio do envolvimento dos consumidores no design do produto para transformar as necessidades em valor de negócio, consequentemente, dependente desse valor encontrado, uma nova seleção é feita por parte da empresa preenchendo as necessidades entregando ao cliente um valor maior. Podemos, portanto, também dizer que a cadeia de abastecimento é parte constituinte da cadeia de valor.

Dentro da temática da Indústria 4.0 (I4.0), a logística e cadeia de abastecimento podem ser definidas como um *Cyber-Physical System* (CPS) colaborativo, denominado *Smart Logistics* [15]. Este pode ser definido como um sistema logístico que pode aumentar a flexibilidade para as alterações de mercado e fazer com que a empresa fique mais perto das necessidades do consumidor final.

Isto poderá aumentar o nível do atendimento ao cliente, a otimização da produção e ainda diminuir os custos de armazenamento e produção. Seguindo esta ordem de ideias a Logística 4.0 deve depender e usar as seguintes aplicações tecnológicas:

- **Planeamento de Recursos**

  Os procedimentos de gestão do planeamento de recursos, de acordo com o paradigma da I4.0 e a implementação de [16], que potenciem a otimização dos recursos/processos, o tempo de alinhamento do mercado e o aumento do emprego dos ativos[17].

  O nível de sofisticação do ambiente onde será implementado aumentará substancialmente, através de não só IoT mas também no grau de especialização dos recursos humanos. Este grau de especialização mudará drasticamente no sentido em que haverá uma crescente necessidade de competências computacionais e analíticas

- **Sistemas de gestão de Armazenamento**

  Os armazéns sempre foram um centro vital no fluxo de mercadorias dentro de uma cadeia de abastecimento. No entanto, no atual clima económico, também precisam de servir de fonte de vantagem competitiva para os prestadores de logística[18].

  O paradigma da I4.0 introduzirá notáveis mudanças na forma como um armazém funciona atualmente. Em especial, a introdução de uma gestão inteligente e a implementação adequada dos WMS que transformará as atividades do armazém de acordo com os requisitos, futuros, da logística de acordo com o paradigma da I4.0[19].

  A integração necessária dos diferentes intervenientes e partes interessadas da cadeia de abastecimento garantirá uma coordenação e um alinhamento total entre todas as

fases da cadeia de valor. Assim, como exemplo, as transportadoras poderão comunicar a sua posição e prever o tempo de chegada ao armazém pelo sistema de gestão, que será capaz de selecionar e preparar a plataforma onde o transportador irá entregar a sua carga, otimizando, desta forma, a entrega. Simultaneamente, sensores (*Radio-Frequency Identification* (RFID) por exemplo) revelarão o que foi entregue e enviarão os dados de rastreio para toda a cadeia de fornecimento. O WMS atribuirá, automaticamente, espaço de armazenamento de acordo com as especificidades da entrega e solicitará o equipamento apropriado para mover as mercadorias para o local correto, de forma autónoma. Uma vez que as paletes sejam transferidas para o local atribuído, as etiquetas transmitirão sinais para o WMS para fornecer visibilidade, em tempo real, para os níveis de inventário, o que poderá prevenir situações de rutura de *stock*, bem como melhorar a capacidade de decisão da gestão para ajustes que possam ser necessários[15].

- **Sistemas de Gestão de Transporte**

  Um TMS permite interações entre um sistema de gestão de encomendas (SGE) e um centro de distribuição (CD) ou um armazém. À medida que a TMS cresceu, estes sistemas foram procurados para ajudar as empresas a controlar e gerir os custos de transporte, para lidar com comunicações electrónicas com clientes, parceiros comerciais e transportadora e, também, para que existisse uma integração com outras tecnologias da cadeia de abastecimento (como os Sistemas de Gestão de Armazéns).

  À medida que a amplitude de ofertas foi aumentando, a TMS foi-se tornando uma escolha popular para empresas de todos os tamanhos e de todas as indústrias.

  Com o uso de IoT, um sistema TMS é, certamente, um elemento essencial no conceito de Logística 4.0. A logística 4.0 utiliza dados em tempo real para obter mais eficiência e eficácia num processo logístico.

  Um sistema TMS é essencial para que uma empresa possa usar a tecnologia GPS (glossário) para localizar com precisão os seus próprios veículos, enquanto estes estiverem na estrada, monitorizar o movimento de carga, negociar com as transportadoras, consolidar envios, usar as funcionalidades avançadas da plataforma e interagir com um *Intelligent Transportation System* (ITS).

  As funcionalidades da TMS continuam a expandir-se de ano para ano o que, num futuro próximo, deverão levar mais empresas a adotar estes sistemas no esforço de melhorar a gestão global dos transportes e o atendimento ao cliente [15].

- **Sistemas Transporte Inteligente**

  O ITS é um campo novo que opera em diferentes áreas de sistemas de transporte, tais como a gestão de transportes, o controlo, as infraestruturas, as operações, as políticas e os métodos de controlo. A ITS adota novas tecnologias como *Hardware* de

computação, sistema de posicionamento, tecnologias de sensores, telecomunicações, processamento de dados, operação virtual e técnicas de planeamento. Uma vez que vivemos num mundo global, o sistema ITS torna-se uma parte vital dele. A ideia de integração de tecnologias virtuais é uma questão nova no domínio dos transportes e desempenha um papel essencial para ultrapassar as questões no mundo global.

Os ITS são importantes para aumentar a segurança e a fiabilidade, as velocidades de viagem, o fluxo de tráfego e contribuem para a redução de riscos e da taxa de acidentes, das emissões de carbono e da poluição atmosférica[20].

2.1.3 Breve Resumo

De acordo com a literatura revista e com a ligação histórica entre as revoluções industriais e a logística podemos definir logística 4.0 como uma direção tecnológica e estratégica que integra diferentes tipos de tecnologias inovadoras com o objetivo de aumentar a eficiência e eficácia da cadeia de abastecimento, mudando o seu foco para a cadeia de valor e, assim, maximizando o valor entregue ao consumidor final, aumentando, consequentemente, o nível de competitividade. Este será alcançado aumentando os níveis de transparência e descentralização entre as diferentes partes devido à digitalização.

2.2 Sistemas Ciber-Físicos

Primeiramente, há que salientar e ter em conta que o conceito de I4.0 é o progresso continuo das tecnologias da informação e da comunicação combinadas com um crescimento exponencial da capacidade de computação, transmissão e armazenamento que permite o surgimento de novos sistemas tecnológicos cada vez mais poderosos e interligados. Estes novos sistemas são chamados de Sistemas Ciber-Físicos, ou *Cyber-Physical System* (CPS), sendo que, para estes, não existe uma definição concisa e aceite.

2.2.1 Conceito e Componentes

CPS são sistemas inteligentes que incluem redes projetadas para haver interação de componentes físicos e computacionais. Estes sistemas, altamente interligados e integrados, fornecem novas funcionalidades de forma a melhorar a qualidade de vida e permitir avanços em aplicações críticas, tais como cuidados de saúde personalizados, resposta de emergência, gestão de tráfego, manufatura inteligente, defesa e segurança interna e, ainda, no fornecimento e uso de energia[21].

Os Sistemas Ciber-Físicos referem-se à integração da computação com processos físicos. Redes e computadores monitorizam e controlam, geralmente com ciclos de feedback onde os processos os cálculos e vice-versa[22].

Apesar das inúmeras definições que possam ser apresentadas, não há uma descrição nítida do que constitui um CPS, mas todas tem em comum uma característica principal

que pode ser vista como a ligação direta (combinação inteligente) dos mundos "reais" e "virtuais"[23]. Dito isso, o acoplamento de componentes de processamento de informação e objetos físicos não é algo novo sendo que já era usado na automação desde 1970. A inovação está na interconetividade entre objetos e processos através de redes de informação abertas e globais como, por exemplo, a internet.

O principal condutor tecnológico para o crescimento dos CPS tem origem em *Hardware* e *Software*. Por um lado, a infraestrutura tecnológica consiste em sistemas incorporados, sensores e atuadores de alta performance e, ainda, em interfaces de comunicação que fornecem capacidades de *Hardware* inevitáveis. Por outro, o uso da web empresarial, plataformas de integração e serviços basados em *cloud* abre a oportunidade de negócios totalmente novos[23].

Tendo em conta as duas definições acima descritas, é relevante destacar que não são apenas os sistemas incorporados que podem constituir os CPS, mas também coisas e termos mais abstratos, como processos operacionais e de gestão. Desta forma, os CPS podem, ainda, ser descritos como um conceito teórico bastante abstrato. Outra característica que pode ser utilizada para descrever um CPS, e simultaneamente, mostrar diversas dimensões desses sistemas, é o grau de descentralização da sua estrutura, ou seja, um CPS pode ser implementado apenas num *microship*, incluído vários sensores e um microprocessador para o processamento de informação, mas por outro lado pode ser implementado numa fábrica e todos os componentes dessa fábrica são também componentes do Sistema Ciber-Físico. Existe ainda a opção de um CPS de escala global onde, por exemplo, uma empresa global tem várias divisões e fabricas, localizados em diferentes pontos do mundo e que, por sua vez, têm comunicação em tempo real entre si, criando assim um CPS de escala mundial.

Os CPS podem ser divididos em várias propriedades ou camadas:

- **Camada Física** – Os CPS integram computação com processos físicos. Estes tipos de processos não têm nenhuma componente digital e podem integrar sistemas mecânicos ou humanos. Estes processos físicos podem ser compostos por várias atividades simultâneas o que poderá trazer desafios na modelação do sistema[24].

- **Camada de Sensores e Actuadores** – Os CPS usam vários sensores para monitorizar e recolher informações sobre o ambiente físico e usa actuadores para interagir com o mundo físico, juntando o mundo "ciber" com a camada física.

- **Camada de Controlo** – CPS fornece mecanismos de controlo utilizando vários componentes e decisões lógicas. O sistema deve ler as medidas fornecidas a partir de sensores de entrada, processar os dados e determinar o comportamento correto para alterar o estado real. Uma ação de saída deve refletir-se utilizando os actuadores para transformar o ambiente físico. Estes apelidam-se de sistemas de ciclo fechado

[25]. Como consequência das mudanças de condições no ambiente físico, a abordagem de controlo aplicada deve ter um comportamento adaptativo e preditivo, respondendo às mudanças de condições e antecipando transformações nos processos físicos. Portanto, desta forma, o sistema herda a capacidade de se reorganizar e reconfigurar dinamicamente.

- **Camada de Rede** – Os CPS podem trocar informações entre sistemas, comunicar através da Internet ou conectar-se com outros dispositivos e sensores utilizando canais de comunicação, tais como redes com ou sem fios, *Bluetooth*, *Global Positioning System* (GPS), etc[25].

O NIST[5], em [21], abordou a questão dos CPS explicando que estes vão além do design convencional de produtos, sistemas e aplicações que são produzidos, com a ausência de uma interconetividade significativa. O NIST apresentou ainda as seguintes razões para tal:

- A importância da combinação entre o "ciber" e o físico e a conexão entre si. Geralmente um CPS envolve sensor, computação e atuador, ou seja, envolve *Information Technology* (IT) tradicional na passagem de informação dos sensores para o processamento desses dados em computação. Envolve, ainda, *Operational Technology* (OT) tradicional para o controlo e atuadores. A combinação destas duas tecnologias juntamente com as restrições temporais associadas são as características principais dos CPS.

- Os CPS precisam de uma metodologia para conseguir garantir a interoperabilidade, gerir a evolução e lidar com efeitos emergentes. É uma questão de extrema relevância, visto que em CPS de grande escala, como uma rede inteligente ou uma cidade inteligente, muitos dos subsistemas são da responsabilidade de diferentes fabricantes.

- Os CPS podem ser aplicados numa situação que não seja aquela para que foram concebidos. Por exemplo, um telemóvel num carro pode ser usado como sensor de tráfego ou a informação de uso de energia num eletrodoméstico pode diagnosticar falhas no equipamento.

2.2.2 Caso Exemplo: ABS

Um dos melhores casos para explicar o conceito de Sistema Ciber-Fisico é quando aplicado a uma situação da vida real, neste caso num *Anti-lock Braking System* (ABS) que o seu conceito tem por base um CPS. O ABS é um sistema que encapsula vários componentes (referencia para a figura) com o objetivo de uma condução mais segura:

[5] *National Institute of Standards and Technology*

- Sensores de Velocidade, que registam a velocidade da roda de um carro (camada de Sensores e Atuadores).

- Unidade de Controlo que, consoante a velocidade obtida pelo sensor, faz atuar o travão com mais ou menos intensidade (camada de Controlo).

- Travão, que irá diminuir a velocidade de rotação da roda. Este tem um atuador para que seja aplicada uma intensidade de travagem consoante a velocidade (camada Física).

O sistema vai monitorizando a velocidade de cada roda através dos seus sensores. Numa situação de travagem repentina, a força aplicada pelo condutor na travagem pode ser maior que a força que o pneu pode suportar causando desta forma uma instabilidade do automóvel podendo provocar um grave acidente. Nesse momento a unidade de controlo recebe, dos sensores, a informação de velocidade e envia, assim, aos atuadores a intensidade de travagem necessária para que se mantenha a segurança do condutor e dos passageiros.

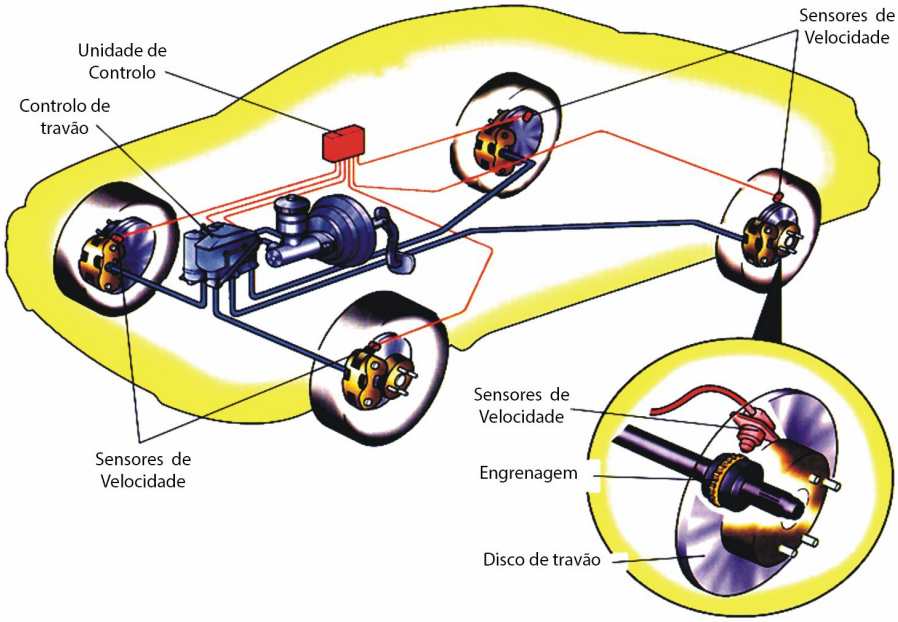


Figura 2.1: Esquema de componentes do sistema ABS - Adaptado de [26].

No sistema ABS é perfeitamente visível na Figura 2.1 as suas várias camadas (acima mencionadas), constituintes de um CPS e a também como a parte computacional (Unidade de Controlo) interage com a parte física (Travão) do sistema.

2.3 Optical Character Recognition

A replica de funções humanas, como a leitura, por parte de máquinas era um sonho antigo, sendo que, com o evoluir da tecnologia, nos últimos 70 anos, este sonho tornou-se uma realidade. O reconhecimento de caracteres transformou-se numa das aplicações mais bem-sucedidas da tecnologia no campo do reconhecimento de padrões e inteligência artificial. São inúmeras as aplicações comerciais, industriais e logísticas para a utilização desta tecnologia.

2.3.1 Introdução ao OCR

Optical Character Recognition (OCR) é uma técnica de conversão, do documento impresso ou manuscrito, em texto legível por uma máquina que, pode ser modificado, tem ganho popularidade ao longo dos últimos anos. Este tem replicado a função humana de ler texto a partir de imagens, reduzindo assim a intervenção humana obtendo-se, desta forma, um melhor desempenho e precisão, sendo ainda, também, um processo de classificação de padrões digitais obtidos a partir de uma imagem. No entanto, o OCR é um problema complexo devido à variedade de línguas e caligrafia, por exemplo. O desempenho do OCR depende, principalmente, da qualidade da entrada dada[27].

O OCR pertence a uma vasta família de técnicas que realizam identificação automática, desde reconhecimento de fala, passando pelas fitas magnéticas presentes nos cartões de crédito, e, até, ao reconhecimento visual, no qual se encontram presentes a leitura de códigos de barras e ainda OCR.

O reconhecimento de caracteres óticos é necessário quando a informação deve ser legível tanto para o humano como para uma máquina. Em comparação com as outras técnicas de identificação automática, o reconhecimento de caracteres óticos é único na medida em que não requer o controlo do processo que produz a informação, ou seja, o texto a ser reconhecido não necessita de ser controlado no momento da sua produção para que seja reconhecível por um OCR, ao contrário do código de barras, por exemplo.

2.3.1.1 *Background* Histórico

As origens do reconhecimento de caracteres remontam a 1870 onde Charles R. Carey inventou um *scanner* de retina que era, na verdade, um sistema de transmissão de imagem usando um mosaico de foto-células. Mais tarde, mais precisamente em 1885, Paul Nipkow inventou o Disco de Nipkow apresentando-se um avanço incrível para a televisão moderna e para os aparelhos de leitura. A primeira geração do OCR foi percetível com a primeira instalação de um aparelho que convertia relatórios de vendas para cartões perfurados de forma a serem introduzidos num computador. Esta instalação foi feita na redação do *Reader's Digest*, em 1954.

Na década seguinte começaram a ser comercializados aparelhos de OCR semelhantes

ao anterior. Sensivelmente a meio década de 1960, a *IBM* apresentou o primeiro leitor de caracteres, o *IBM 1287*, capaz de reconhecer caracteres escritos à mão. Nessa mesma altura a *Toshiba* criou um sistema capaz de organizar cartas através da leitura do código postal nelas inserido. Ainda neste período, foram criadas as primeiras fontes *standard* para facilitar o reconhecimento de caracteres por parte destes aparelhos que vinham a ser desenvolvidos.

No fim da década seguinte surgiram alguns desafios como documentos com má qualidade de impressão ou, então, caracteres escritos à mão de diferentes tipos e, ainda, a incerteza de como chegar a dispositivos de alto desempenho e baixo custo de fabrico. Nesta altura ainda era acessível para os dispositivos existentes pois os documentos impressos eram *standard*, mas com o aparecimento dos primeiros computadores pessoais e impressoras a laser o nível de complexidade cresceu apresentando desafios cada vez maiores. Desde então que se tem vindo a desenvolver aparelhos e abordagens diferenciadas de reconhecimento de caracteres o que faz com que, atualmente, seja possível reconhecer [28].

2.3.1.2 Tipos e Fases do OCR

Existem dois tipos de OCR, um deles acontece quando o reconhecimento é feito a partir de um documento acabado e impresso e é chamado de *offline*; por outro lado, *online* refere-se à capacidade da máquina reconhecer os caracteres à medida que estes são desenhados. Fazendo uma revisão de literatura, Marne *et al.* em [27], apontam para 5 fases principais no algoritmo do OCR em que cada fase tem como *input* o *output* da fase anterior:

- **Digitalização** – Nesta fase é captada uma imagem digital do documento em questão, geralmente recorrendo a um *scanner*. Em seguida a imagem é passada para uma escala de cinzentos, o que geralmente conhecemos como imagem a preto e branco. Mais tarde é transformada em imagem binária, contendo apenas pixeis pretos ou brancos consoante a intensidade do cinzento do mesmo pixel da imagem anterior e, por fim, é feita uma segmentação de localização da zona onde se encontram os caracteres a ser reconhecidos.

- **Pré-Processamento** – Nesta fase todos os defeitos e irregularidades da imagem são corrigidos. Esta primeira parte pode dificultar o reconhecimento, sendo que, desta forma, são feitas várias passagens pela imagem, pixel a pixel, usando os processos de dilatação e erosão de modo a preencher falhas e "buracos" nos caracteres e, também, suavizar os contornos dos mesmos.

- **Segmentação** – A imagem é segmentada caractere por caractere dividindo a imagem em várias imagens de dimensão inferior, contendo apenas um caractere. Esta segmentação é feita com base nas características (a Forma geralmente) do caractere.

- **Extração de características** – O objetivo desta fase é, como o próprio nome indica, a extração das características essências de cada símbolo, geralmente características diagonais, características de transição, *zoning*, características de estatística, entre outras. A seleção de um método e a seleção das características está altamente relacionado com eficácia de reconhecimento do sistema. Associada a esta fase está a classificação do caractere, ou seja, o próprio reconhecimento em si do caractere em questão tendo como base as características anteriormente extraídas.

- **Pós-Processamento** - Nesta fase, os caracteres são organizados e agrupados em palavras e, posteriormente, em *strings*. Finalmente é feita uma deteção de erros de reconhecimento, geralmente percorrendo cada palavra e verificando-se a existência da mesma no dicionário. Desta forma, caso não se encontre a palavra, a mesma é substituída pela palavra encontrada que lhe seja mais semelhante, obtendo-se, assim um texto editável resultante do OCR.

2.3.2 Abordagens tradicionais

2.3.2.1 *Pattern Matching*

Aquando da fase de extração de características, anteriormente mencionada, uma das formas de reconhecer o caractere é usando o *Pattern Matching* [29], inicialmente o caractere é isolado e binarizado, ou seja, com os pixeis pretos representados pelo número 1 e os brancos pelo número 0 como está demonstrado na Figura 2.2.

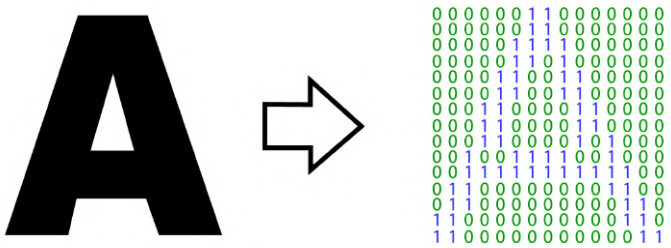


Figura 2.2: Binarização do caractere 'A' - Adaptado de [29].

Posteriormente, o reconhecimento é baseado na correspondência da matriz, gerada pela binarização, com um modelo já existente desse caractere. A matriz é dividida em faixas e cada faixa subdividida em setores. Essa divisão segue uma sequência: primeiramente, é encontrado o centro da matriz e o raio da circunferência, que é encontrado através da distância entre o centro e o pixel pertencente à matriz mais afastado. A distância é dada por: $D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Em seguida, o raio é dividido pelo número de faixas necessárias, sendo que cada faixa corresponde ao anel correspondente. Depois o círculo é dividido em setores de igual tamanho. Finalmente é criada uma matriz faixa-sector calculando número de '1' que se encontra em cada interceção entre setor e faixa como está demonstrado na Figura 2.3.

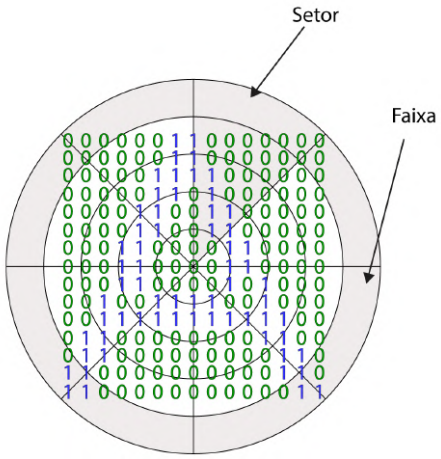


Figura 2.3: Exemplo de implementação de faixas e setores na Matriz - Adaptado de [29].

O processo de reconhecimento é feito pela comparação entre a matriz faixa-setor e uma matriz previamente construída. Se as matrizes forem iguais ou praticamente iguais sabemos de que caractere se trata. Esta técnica é bastante eficaz para texto escrito à máquina e quando a máquina que fará o reconhecimento tem um *dataset* de matrizes para a fonte ser reconhecida, caso contrário, se usar o *dataset* de outra fonte, terá um maior número de erros no reconhecimento. Por vezes se os caracteres a ser reconhecidos não estiverem na orientação correta, o método também falhará. O método tem, portanto, bastantes limitações.

2.3.2.2 Características Estruturais

Como a técnica anterior, esta diferencia-se das outras na fase de extração de características e, consoante as características extraídas, obtém-se uma técnica distinta:

- **Projeções** – Extraindo as projeções verticais e horizontais Rahman *et al.* em [30] usam-nas em forma de histogramas e, assim, é feita a comparação com os histogramas das projeções de todos os caracteres da biblioteca.

- **Contorno** – Kim e Kwon, em [31], apresentam uma técnica que consiste em extrair o contorno de cada caractere e criar um gráfico de "altura" do contorno (figura – tirar a figura dos. H do artigo) e, em seguida, comparar com os gráficos de contorno de cada caractere da biblioteca.

- **Cruzamento de Contornos** – No caractere segmentado são "desenhadas" linhas aleatórias verticais e horizontais a atravessar o caractere. Nessas linhas são contabilizadas as vezes em que essa mesma linha atravessa o contorno do caractere. E, por fim, como em todos as outras técnicas, é feita a comparação com a mesma característica dos caracteres da biblioteca [32].

- **Área periférica do caractere** - Este método considera não só o próprio caractere, mas também o segmento da imagem inicial e, onde se encontra o caractere, a imagem é dividida em X faixas tanto na vertical e na horizontal. Em seguida é medida a distância (em pixeis) entre o limite da imagem até ao ponto mais próximo do contorno do caractere. Por fim, a distância anteriormente apresentada, é comparada com a distância da faixa equivalente dos caracteres da biblioteca [32].

2.3.3 Falhas nas Abordagens Tradicionais

Todas as abordagens faladas anteriormente são excelentes em documentos digitalizados e pouco complexos, mas tem falhas quando:

- **Várias Fontes** – Caso um documento tenha várias fontes, o sistema apresentará uma taxa de acerto menor comparado com um documento só com uma fonte, porque todos os caracteres são comparados com a biblioteca de uma fonte, e aqueles caracteres que são de outra fonte, provavelmente, não vão ser reconhecidos. Por outro lado, o sistema pode usar várias bibliotecas de fontes diferentes, sendo que, assim, o processo de reconhecimento será mais complexo e demorado.

- **Fraca qualidade do documento** – Os sistemas OCR com abordagens tradicionais tendem a ter mais dificuldades e menores taxas de reconhecimento corretos quando aplicados a documentos com fraca qualidade ou imagens desfocadas, que fazem com que os contornos dos caracteres estejam mal definidos.

- **Escrita manual** – Algumas técnicas têm a capacidade de fazer reconhecimento de escrita manual, desde que esta escrita seja feita de forma cuidada e perfeitamente legível, caso contrário praticamente todas as abordagens tradicionais tem bastantes falhas.

2.3.4 *Machine Learning/Deep Learning*

Para combater a pouca evolução e as limitações das abordagens tradicionais e com a constante evolução da computação foram aparecendo sistemas de OCR baseadas em inteligência artificial usando *Machine Learning* (ML) e *Deep Learning* (DL). Estes fazem aumentar a taxa de acerto mesmo quando aplicados a documentos em que as abordagens tradicionais teriam bastantes dificuldades e, ainda, diminuem substancialmente o tempo de execução.

Sabemos, à partida, que o OCR se divide em duas componentes principais: a identificação das áreas de texto e o reconhecimento dos caracteres. Com o uso de DL podemos fazer ambos os processos. Para fazer a deteção das regiões com texto podemos usar métodos de DL como:

- **R-CNN** – A ideia principal do método de *Region-based Convolutional Neural Network* (R-CNN) é composta por dois passos. Em primeiro lugar é feita uma *selective search*[6], que identifica várias regiões contornadas (ou *Region of Interest* (ROI)) que possam ser áreas de texto. Seguidamente, extrai características de *Convolutional Neural Network* (CNN) de cada ROI independentemente para serem classificadas e, caso sejam efetivamente áreas de texto, apenas esse ROI segue para a fase de reconhecimento[33].

- **SSD** – *Single Shot MultiBox Detector*(SSD) é uma técnica geralmente usada para deteção de objetos em tempo real e baseia-se numa única rede. Não tem necessidade de fazer uma *selective search*, em vez disso, usa retângulos de diferentes tamanhos e, em seguida, ajusta os tamanhos dos retângulos como parte da previsão. Os diferentes retângulos são alcançados pelas últimas camadas da rede responsáveis pela previsão. Estes são cada vez mais pequenos e a previsão final é a união de todas as previsões anteriores[34].

Estas técnicas são especializadas em deteção de objetos e têm algumas falhas no reconhecimento de caracteres. Desta forma, para fazer o próprio reconhecimento dos caracteres são utilizados:

- **CRNN** - *Convolutional Recurrent Neural Network*(CRNN) pode ser considerado uma junção de uma CNN e de uma *Recurrent Neural Networks* (RNN) e funciona em 3 camadas[35] visíveis na Figura 2.4, sendo elas:

  - ***Convolutional Layer*** – Nesta camada é feita uma extração de uma sequência de características de cada imagem de *input*.

  - ***Recurrent Layer*** – Aqui, uma RNN é construída para fazer predições por cada *frame* da sequência de características que saiu da camada anterior.

  - ***Transcription Layer*** – Esta camada é adotada para traduzir as previsões feitas pela camada anterior numa sequência de caracteres e formando, assim, a palavra reconhecida.

[6] Algoritmo que fornece propostas de regiões da imagem que potencialmente contem objetos

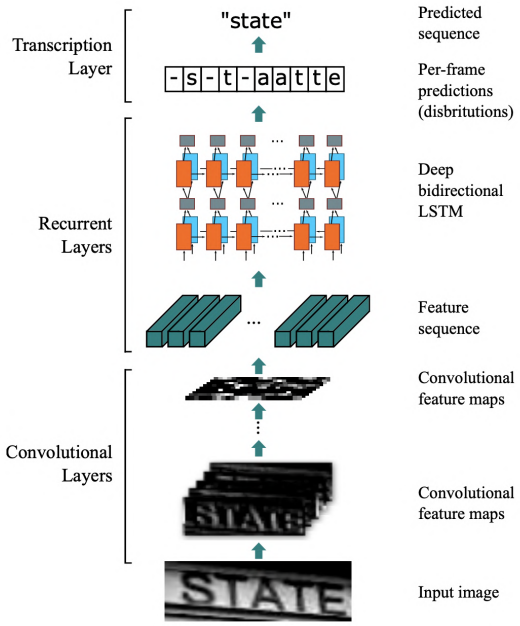


Figura 2.4: Esquema de camadas do modelo CRNN[35].

- **STN** – A técnica *Spatial Transformer Networks*(STN) é constituída por duas partes principais: a primeira engloba a rede de localização que pega na imagem a ser reconhecida e prevê N matrizes de transformação que são aplicadas a N grelhas idênticas formando N grelhas de amostragem. As grelhas de amostragem, que representam a segunda parte principal, são usadas de duas formas: (1) para calcular um retângulo contornado em volta de cada região de texto; (2) para realizar a amostragem da imagem de entrada com N grelhas de amostragem para extrair N regiões de texto. Cada uma dessas áreas de texto é usada numa rede de reconhecimento para realizar o próprio reconhecimento do texto. Todo o sistema é treinado *end-to-end* fornecendo apenas informação sobre as *labels* de texto de cada região do mesmo[36].

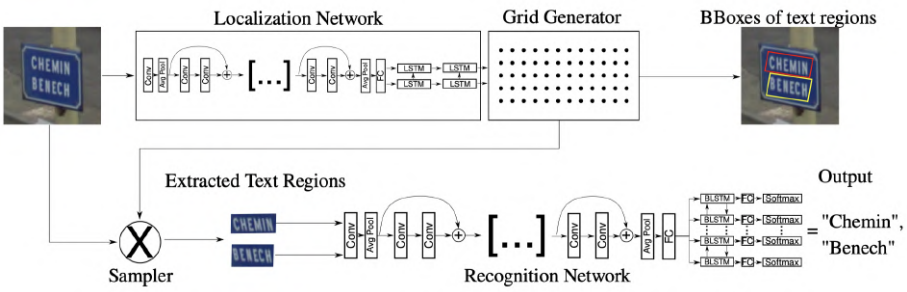


Figura 2.5: Esquema de camadas do modelo STN[37].

Existem ainda arquiteturas de sistemas que utilizam a mesma tecnologia tanto no processo de localização dos caracteres bem como no seu reconhecimento.

- ***Object Detection*** Este método é bastante comum em *Automatic License Plate Recognition* (ALPR), sistemas automáticos de reconhecimento de matriculas que consiste primeiro na localização da matricula em si, utilizando *You Only Look Once* (YOLO)[38]. O YOLO é um método para reconhecimento de objetos como um problema de regressão e probabilidades das classes do objeto. Neste caso os objetos serão os caracteres.No exemplo dos ALPR o primeiro passo do processo de reconhecimento, é a localização da matricula, então para tal o modelo YOLO é treinado com um *dataset* de matriculas e assim, ao fim de ser treinado, este será capaz de localizar uma matricula numa imagem data.Com a matricula localizada é feita uma segmentação, ou seja , cortar a imagem original deixando apenas a zona da matricula[39].

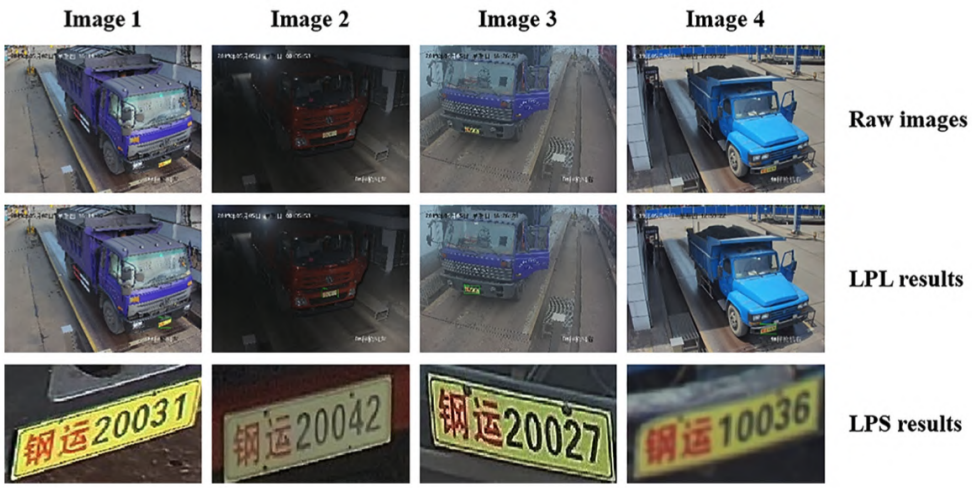


Figura 2.6: Resultados da Localização da matricula[39]. *Nota: LPL= licence plate localization; LPS= licence plate segmentation*

O passo seguinte será treinar novamente o modelo, mas desta vez com um *dataset* de caracteres(possíveis de ser encontrados na matricula), e assim , mostrando ao modelo a imagem já segmentada, este será capaz localizar e reconhecer cada caractere da matricula[39].

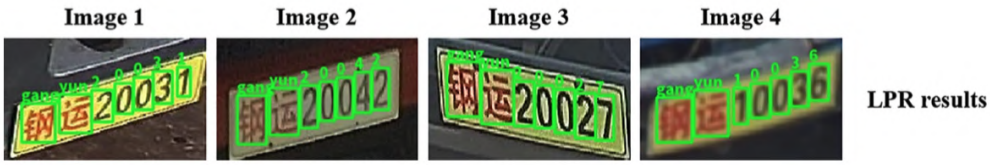


Figura 2.7: Resultados do Reconhecimento da matricula[39]. *Nota: LPR= licence plate recognition;*

Com toda a literatura revista anteriormente é fácil perceber que estas abordagens mais modernas, usando DL e ML, têm bastantes vantagens quando comparadas às abordagens mais tradicionais. Estas vantagens podem ser vistas de vários ângulos:

- Velocidade de execução.
- Taxa de acerto no reconhecimento com vários tipos de letra.
- Facilidade de implementação.
- Acerto em condições adversas como falta de luminosidade ou imagens de baixa resolução

3

Arquitetura do Sistema

Neste capítulo, do presente trabalho, encontrar-se-ão as informações relativas à arquitetura. Apresentar-se-ão, inicialmente, os requisitos funcionais e não funcionais identificados como base da arquitetura do sistema. Seguidamente a estrutura e o funcionamento do sistema.

3.1 Requisitos Funcionais e Não Funcionais

Inicialmente, para que se consiga chegar a uma solução para o problema enunciado no capítulo introdutório e, também, para que se perceba qual a melhor arquitetura para o sistema, foi necessário, primeiramente, perceber e descrever quais os requisitos funcionais e não funcionais do sistema a ser desenvolvido.

Para começar, é necessário compreender o que são estes requisitos. Assim, segundo o *IEEE Standard Glossary of Software Engineering* [40] o requisito define-se como:

1. Uma condição, ou capacidade, necessária ao utilizador para resolver um problema ou atingir um objetivo;

2. Uma condição, ou capacidade, que deve ser satisfeita ou possuída por um sistema, ou componente do sistema, para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos.

Resumidamente, estes consistem nos pedidos de um cliente e naquilo que o sistema deverá fazer para satisfazer as necessidades sentidas, que, neste caso, será a Opertri[4].

3.1.1 Requisitos Funcionais

Nesta categoria, os requisitos são o que o sistema deve fazer para satisfazer os pedidos do cliente.

Inicialmente, foram descritas algumas funcionalidades necessárias para a realização da aplicação, do ponto de vista do utilizador, por parte do cliente e também pelos orientadores do presente trabalho. Compreendeu-se que seria necessário capturar imagens, portanto o primeiro requisito será a capacidade de a aplicação usar a câmara do dispositivo para essa mesma finalidade. Seguidamente percebeu-se que também seria útil a capacidade de se utilizar uma imagem capturada à priori com a aplicação nativa da câmara. Para tal deve conceder-se, dentro da aplicação, a capacidade de abrir a galeria de fotografias do dispositivo e de escolha de uma imagem. Para uma utilização mais eficiente da aplicação pelo operador foram criados 3 requisitos, sendo que o primeiro, será a verificação do ID reconhecido para que, caso o ID reconhecido não se encontre no formato correto, esta o apresente como incorreto. Caso o formato do ID seja correto esta deve apresentá-lo ao utilizador para que este tome conhecimento, e o verifique. Se o ID reconhecido não estiver no formato correto , ou o utilizador verifique que não é o ID correto, deve ter a possibilidade de tentar um novo reconhecimento, ou então, introduzir o ID manualmente.

Requisito	Explicação
Tirar Fotografia	Capacidade de tirar fotografias dentro da aplicação.
Abrir a Galeria de fotos	Capacidade de abrir a galeria de fotografias e capacidade de escolha entre elas.
Verificação do Identificador (ID)	Capacidade de verificar se o ID tem um formato válido.
Apresentação do ID	Capacidade de mostrar o ID reconhecido, caso este seja válido.
Inserção Manual	Capacidade de uma inserção manual do ID, caso este não tenha sido reconhecido ou o reconhecimento não tenha sido eficaz.

Tabela 3.1: Lista de Requisitos Funcionais e respetiva explicação.

Como esta aplicação será utilizada por um operador na empresa, torna-se fundamental corresponder a estes requisitos para que se alcance o objetivo de rentabilizar e facilitar o trabalho ao operário designado para recolher os identificadores de cada contentor que chega ou sai do Porto da Horta.

3.1.2 Requisitos Não Funcionais

Estes referem-se aos requisitos técnicos do sistema, como características ou prevenção de erros. Para o sistema funcionar sem qualquer tipo de erro e de forma a que se comporte de maneira conducente à satisfação das necessidades do cliente, foram descritos os seguintes requisitos a serem alcançados:

Requisito	Explicação
Adaptabilidade	Capacidade da aplicação ser utilizada em qualquer dispositivo móvel, sem que as páginas apareçam desformatadas.
Envio do ID	Capacidade de estabelecer comunicação com o *Endpoint* no servidor da empresa para que seja possível o envio do ID.
Prevenção de erros humanos	Capacidade de apresentação de mensagens "pop-up" ao utilizador para que este confirme certo tipo de ações que poderão ser adotadas, prevenindo eventuais erros involuntários.

Tabela 3.2: Lista de Requisitos não Funcionais e respetiva explicação.

3.2 Estrutura

Com os requisitos já definidos começa a perceber-se os conceitos que vão formar o sistema e, igualmente, poder-se-á iniciar o desenho do mesmo. Primeiramente, descrever-se-á uma arquitetura dividida em três partes (*User*, Aplicação Móvel (APP) e *Endpoint* na Opertri) em que o reconhecimento seria feito diretamente na aplicação. No entanto, rapidamente se compreendeu que esta abordagem teria bastantes desvantagens relacionadas com a utilização do sistema como, por exemplo, a impossibilidade do sistema se adaptar a qualquer dispositivo móvel devido à falta de capacidade de processamento de alguns aparelhos sendo que, desta forma, não seria cumprido um dos pré-requisitos não funcional, comprometendo o desempenho da aplicação.

Assim, optou-se por adaptar um sistema constituído por quatro partes, sendo que cada uma delas apresenta uma série de funções e capacidades, identificados na representação seguinte, bem como linhas de comunicação entre as partes do sistema representadas na Figura 3.1.

- ***User*** - Esta representa o utilizador do sistema que, neste caso, apenas, vai interagir diretamente com a APP, enviando (tocando no ecrã) e recebendo (mostrando no ecrã) informação para a mesma.

- **APP** - Representa a aplicação móvel instalada num *smartphone* ou *tablet*. Apresenta-se como componente central deste sistema e, como tal, este tem a capacidade de

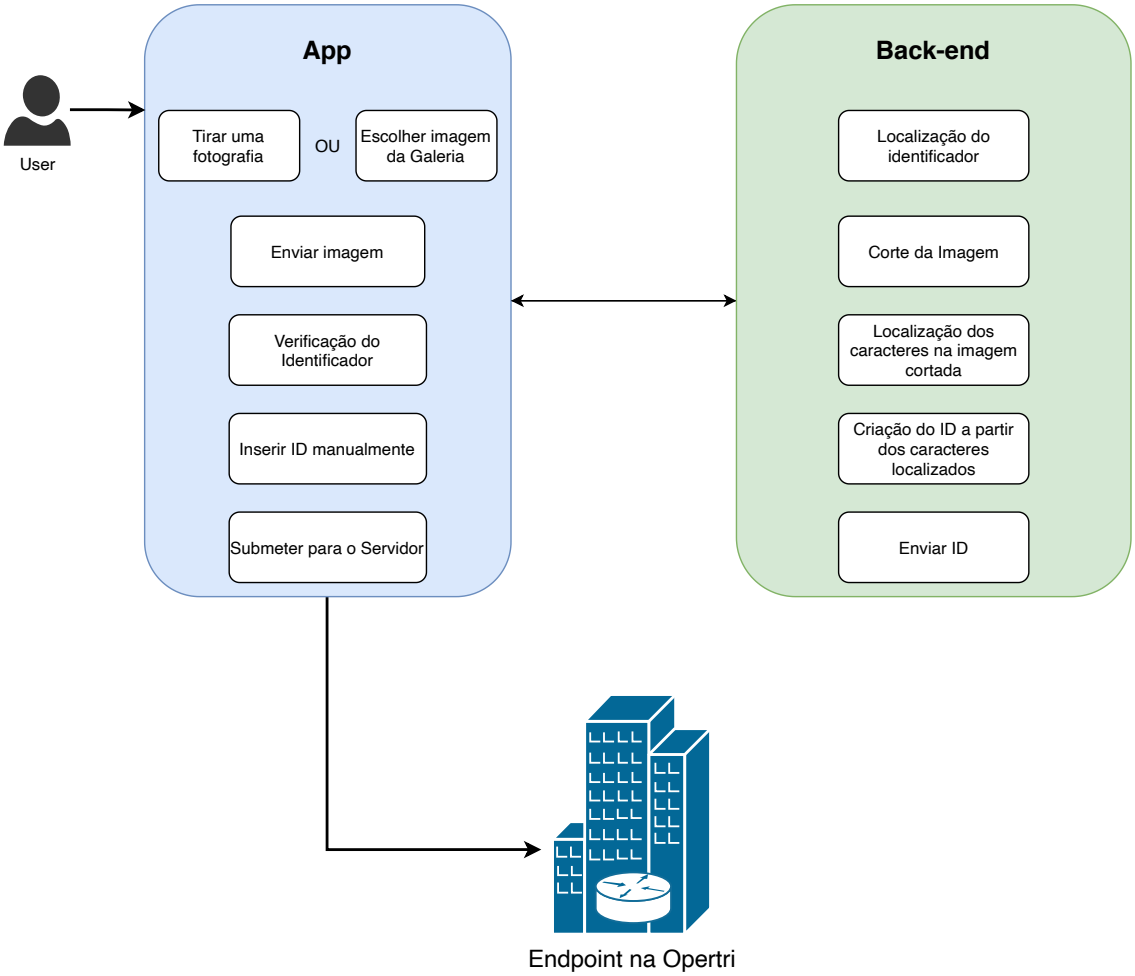


Figura 3.1: Arquitetura Conceptual do Sistema.

interagir com o utilizador, bem como de comunicar com os restantes componentes do sistema. Fora das capacidades comunicativas, apresenta, ainda, algumas funcionalidades gerais que servem para responder aos requisitos funcionais.

- ***Back-end*** - A parte do *Back-end* poderá ser implantada num servidor em *cloud* ou num servidor local dependendo da preferência do cliente. Destinar-se-á ao processo de localização e reconhecimento do identificador que, para além dessa tarefa, tem ainda a capacidade de comunicação com a APP, recebendo imagens e enviando identificadores.

- ***Endpoint* na Opertri** - Por fim, a componente *Endpoint* representa um *Endpoint* situado no servidor da empresa e tem a função única de receber os identificadores reconhecidos.

3.3 Funcionamento do Sistema

Após se compreender o funcionamento e utilidade dos requisitos do sistema e de se compreender os conceitos que fazem parte de cada um, far-se-á o plano estrutural do sistema e do seu funcionamento. Assim, primeiro, apresentar-se-á o diagrama de sequência na Figura 3.2 de todas as interações entre todos os componentes do sistema, incluindo o utilizador, e, posteriormente, a arquitetura da aplicação e do algoritmo de reconhecimento do ID.

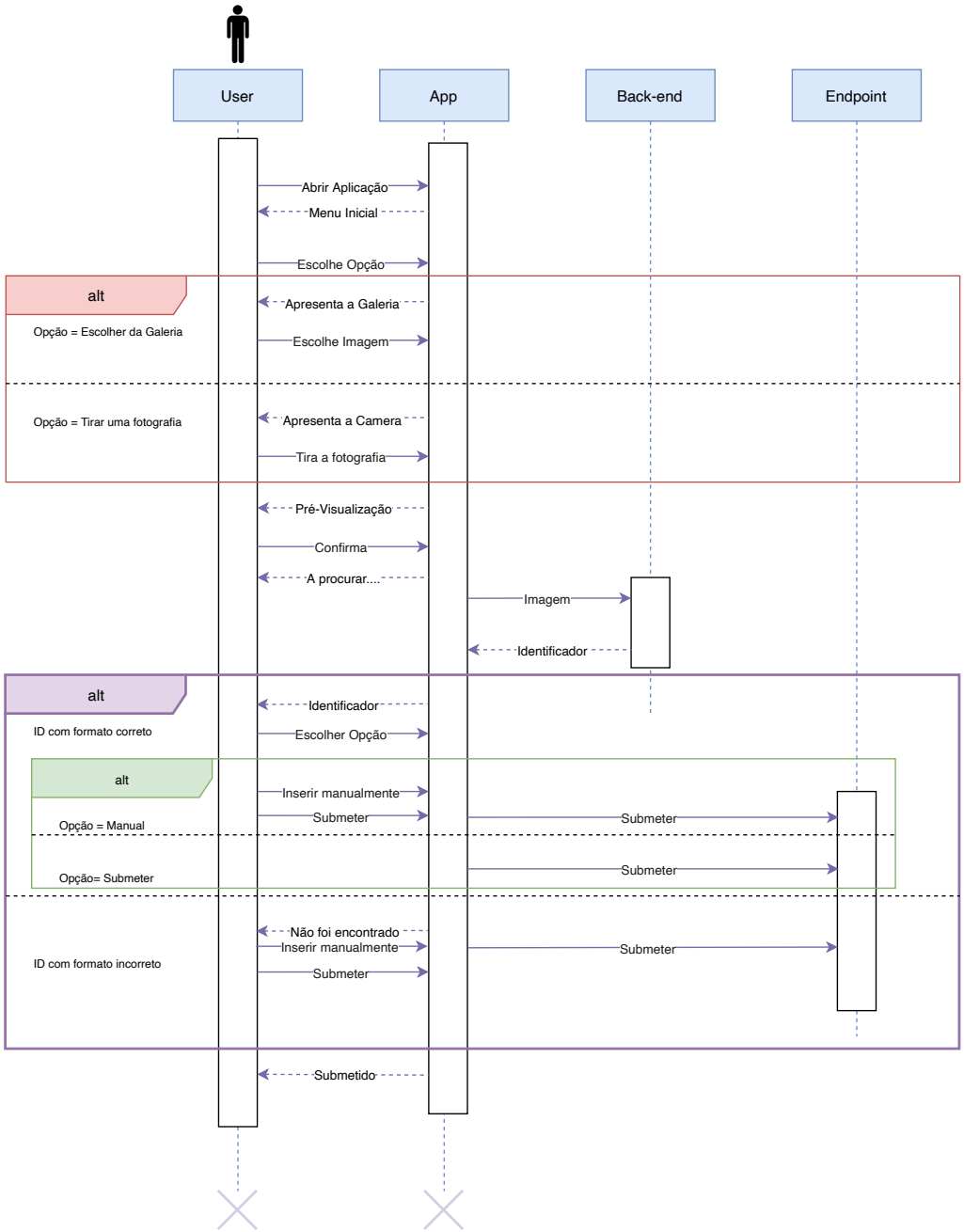


Figura 3.2: Diagrama de Sequência de interações entre as entidades dos sistema.

3.3.1 Funcionamento da Aplicação

Na figura 3.3 foi descrito o funcionamento da APP que ao abrir a aplicação o utilizador será direcionado, diretamente, para o menu principal. Nesta página principal, será permitido que se escolha entre "**Tirar uma fotografia**" ou "**Escolher da galeria de fotografias**". Ao escolher a opção "**Tirar uma fotografia**" aparecerá uma página com uma câmara, onde se poderá tirar a fotografia, cancelar a ação ou trocar entre as câmaras frontal ou traseira. Por outro lado, se a opção selecionada for "**Escolher da galeria de fotografias**", apresentar-se-á, primeiramente, uma página para permitir o acesso à galeria ou para cancelar o pedido sendo que a primeira opção abre a galeria nativa do dispositivo e, nesse momento, o utilizador poderá seleccionar a imagem que pretende utilizar.

De seguida, em qualquer das opções ("**Tirar uma fotografia**" ou "**Escolher da galeria de fotografias**") será apresentada uma pré-visualização da imagem, questionado-se o utilizador se pretende seguir com essa imagem. Em caso de resposta afirmativa a aplicação envia a imagem para o *Back-end* e apresenta uma página de espera. Quando recebe a resposta, esta surge em forma de identificador e faz uma verificação do mesmo.

Assim sendo, surgem duas opções no sistema: caso o identificador seja reconhecido no formato correto, será apresentado o ID reconhecido com as opções de inserção manual (caso o utilizador verifique que não foram reconhecidos os caracteres corretos) e com a opção de submeter. Por outro lado, caso o ID não esteja reconhecido de forma correta, será apresentada uma mensagem de erro com as opções de cancelar a operação ou de inserção manual. Nesta última surgirá uma caixa para o utilizador escrever o ID correto e, por fim, submeter. Ao fazer a submissão, o identificador, reconhecido ou inserido manualmente, será enviado para o *Endpoint* no servidor da empresa.

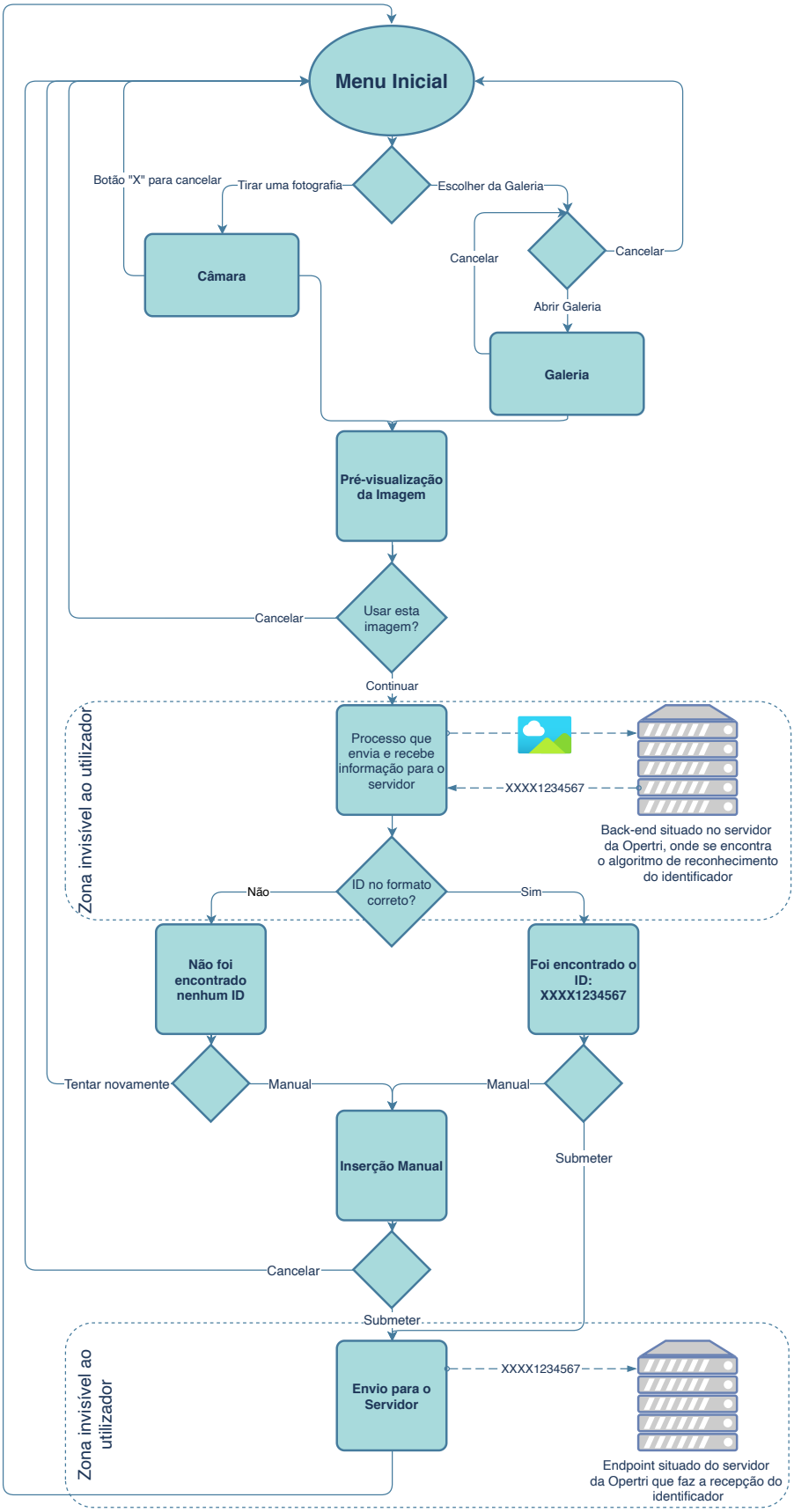


Figura 3.3: Fluxograma do funcionamento da Aplicação.

3.3.2 Funcionamento do *Back-end*

O Back-end será a parte do sistema invisível ao utilizador, correspondendo à parte do sistema onde a localização e reconhecimento do ID será efetuado. Este componente poderá ser implementado num servidor local na empresa ou, então, num servidor em cloud.

Este componente, ao estar ativo, procurará sempre receber uma imagem, vinda da aplicação. Aquando da receção da mesma guarda-a temporariamente. De seguida corre o algoritmo de localização de identificadores na imagem e, de todos os identificadores localizados, irá escolher aquele que estiver mais perto, ou seja, o pretendido caso o algoritmo localize mais que um ID.

De seguida, com a localização do ID, a imagem guardada anteriormente é cortada na área do ID localizado. Posteriormente, executar-se-á o algoritmo de reconhecimento do ID na imagem, já cortada. Deste algoritmo sai a localização e a letra/número de cada caractere localizado nesta nova imagem. Dependendo da orientação do ID, são agrupados os caracteres numa string da esquerda para a direita, caso a orientação seja horizontal e, caso seja vertical, os caracteres são agrupados de cima para baixo.

Por fim, o identificador será enviado para a aplicação. Se no processo de localização não existir nenhum ID localizado será enviada uma string vazia.Todo o processo foi descrito no Fluxograma da Figura 3.4

Com os requisitos mencionados anteriormente, e as especificações do problema desenhou-se a estrutura e consequente funcionamento de todo o sistema que seriam utilizados como por base do desenvolvimento de toda a solução.

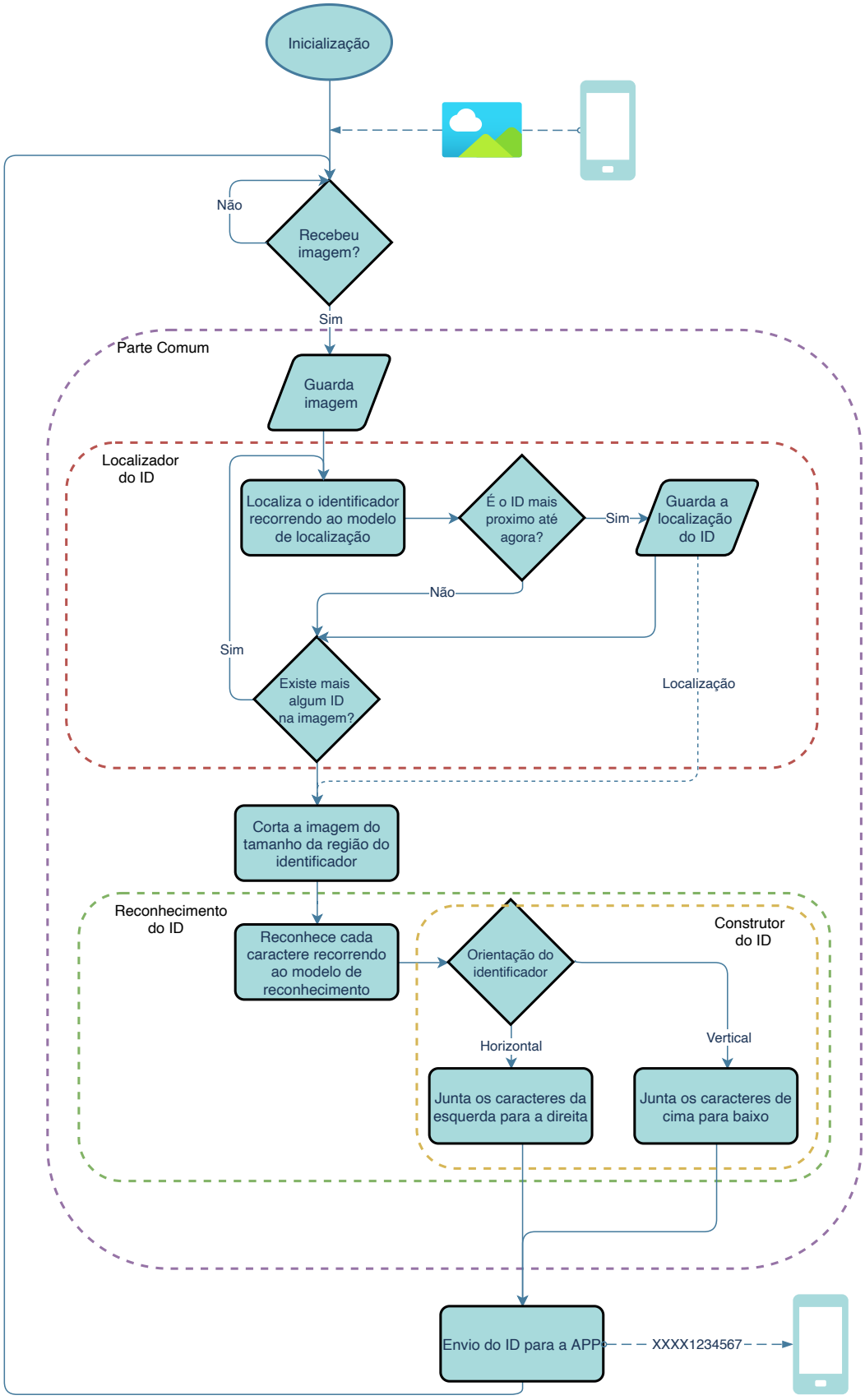


Figura 3.4: Fluxograma do funcionamento do *Back-end*.

4

Implementação

Neste capítulo, serão apresentados o método de implementação de cada componente do sistema e, também, as tecnologias utilizadas para a realização do mesmo. Inicialmente apresentar-se-ão algumas considerações onde se especificam alguns requisitos necessários ao bom funcionamento do sistema e, seguidamente, serão abordados os métodos de implementação tanto da aplicação como do *Back-end*.

4.1 Implementação da Aplicação

Antes do início do desenvolvimento da aplicação, realizou-se uma pesquisa acerca de quais seriam as melhores abordagens para iniciar o desenvolvimento e, também, sobre quais as abordagens que trariam não só os melhores resultados para o pretendido mas também as que dariam mais confiança para solucionar o problema principal da dissertação. Nesta pesquisa foram também consolidados alguns conhecimentos de *User Interface / User Experience* (UI/UX) de forma a que a aplicação tivesse um uso mais fácil e intuitivo.

Durante a pesquisa de uma tecnologia para desenvolver foram tidos em conta dois fatores:

- **Ser *Cross-platform*** – Capacidade de ser utilizada nos vários Sistemas Operativos (SO), *IOS* e *Android*.
- **Ser Nativa** – ou seja, a aplicação estar desenvolvida para cada SO e não apenas uma visualização de uma aplicação *web*. Tem ainda a facilidade de acesso ao hardware do dispositivo como por exemplo a câmara.

Considerando os fatores anteriores e os resultados da pesquisa foi seleccionado o *React Native* (RN) como plataforma de desenvolvimento da aplicação.

Criado pelo *Facebook*[41], o RN é uma ferramenta *open-source* de desenvolvimento de aplicações moveis usada tanto para *IOS*[42] como para *Android*[43]. Combina o melhor do desenvolvimento de aplicações nativo com o *React* e apresenta, ainda, uma biblioteca de *JavaScript* para desenvolvimento de interfaces gráficas de aplicações *web*.

Para além do desenvolvimento, a aplicação, para poder ser utilizada, precisa de ser implementada nos sistemas operativos. Para tal foi escolhido o *Expo*[44] como plataforma de desenvolvimento, teste e implementação da mesma. Esta traz uma série de vantagens para toda a implementação do sistema, como:

- **Fácil *setup*** – Em cerca de 5 minutos é possível criar uma aplicação simples.
- **Abrangência** – Capacidade de desenvolver uma aplicação *cross-platform* em qualquer SO (*Windows*, *MacOS* ou *Linux*), ao contrário da impossibilidade de, sem o *Expo*, desenvolver uma aplicação para *IOS* sem *MacOs*.
- **Pré-visualização em tempo real** – Facilidade para o desenvolvedor perceber as alterações feitas, em tempo real, utilizando diversos tipos de simuladores:
  - ***Snack*** - Caso seja necessário apenas testar um recurso do *Expo* ou alguma funcionalidade simples sem ser necessário criar um novo projeto;
  - ***Expo Client*** - Aplicação *Android* e *IOS* onde é possível pré-visualizar no próprio dispositivo as alterações feitas na aplicação;
  - **Emuladores** - Usando emuladores de *Android* ou *IOS* no próprio computador, em conjunto com o *Expo Client* .

Por fim, foi escolhido o *Visual Studio Code* [45] como editor principal de código, não só pela sua versatilidade e capacidade de processamento, mas, também, devido ao conhecimento da mesma.

4.1.1 Interface gráfica

Do ponto de vista visual do utilizador da aplicação, compreende-se que esta tem uma linearidade no aspeto geral da aplicação, assim, para tal ser possível, decidiu-se que os grafismos de todas as páginas da aplicação deveriam seguir sempre o mesmo aspeto. Desta forma, os elementos são um cabeçalho (*header*) com o nome da aplicação, uma imagem de fundo sempre igual, sendo que toda a informação apareceria num *modal*, ou seja, uma área retangular com o objetivo de centrar a atenção do utilizador nela mesma.

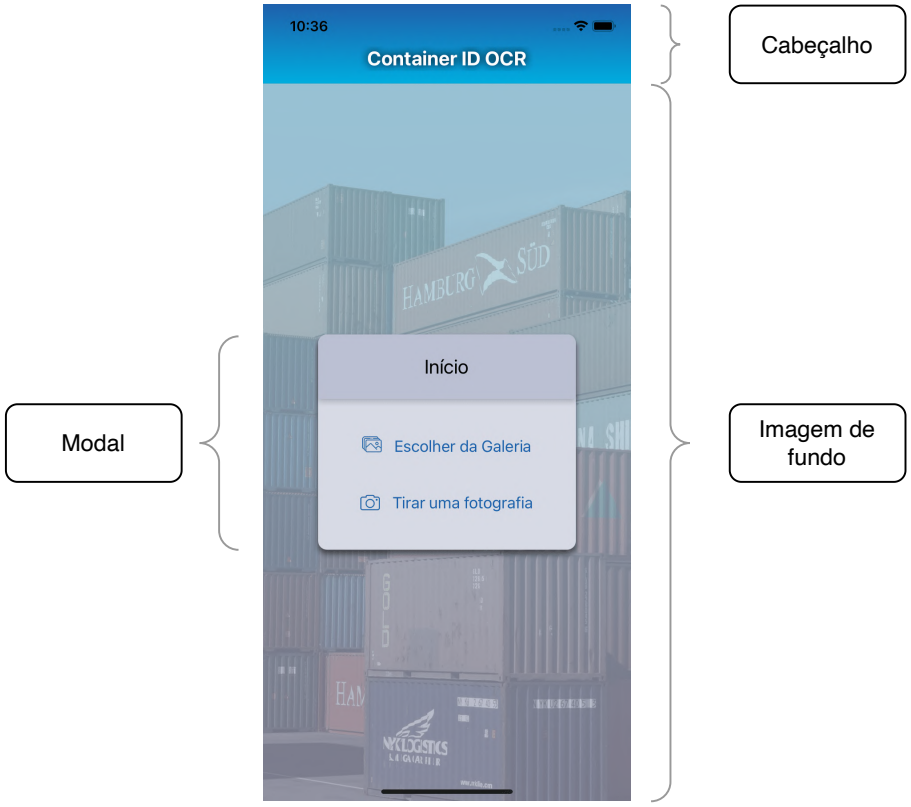


Figura 4.1: Descrição dos componentes da interface gráfica da aplicação.

A tecnologia RN tem por base componentes, nativos ou criados, que podem ser utilizados seguidos ou inseridos noutros componentes. Cada um desses apresenta as suas próprias características e, também, parâmetros para puderem ser utilizados em diferentes situações ao longo do desenvolvimento. Desta forma, e dando o exemplo da página do "**Menu Inicial**" que podemos verificar na Figura 4.1, são utilizados diversos tipos de componentes, dois dos quais criados com o objetivo de linearizar a interface - o Header e o Modal que, tal como os nomes indicam, são o cabeçalho e o retângulo a cinzento, já mencionado anteriormente. Para além deste existem ainda:

- ***View*** – O mais comum, que neste contexto auxilia bastante na organização da página.
- ***ImageBackground*** – Que tal como o nome indica serve, principalmente, para configurar o fundo.
- ***TouchableOpacity*** – Representa uma zona pressionável que, ao ser pressionada, aumenta a opacidade do esteja nessa zona. No caso desta e de outras páginas, este componente foi utilizado em substituição do componente *Button*, com o objetivo de manter a interface igual em qualquer dispositivo, visto que o componente *Button* utiliza o botão nativo do dispositivo, o que significa que de dispositivos com SO diferente resultariam interfaces diferentes.

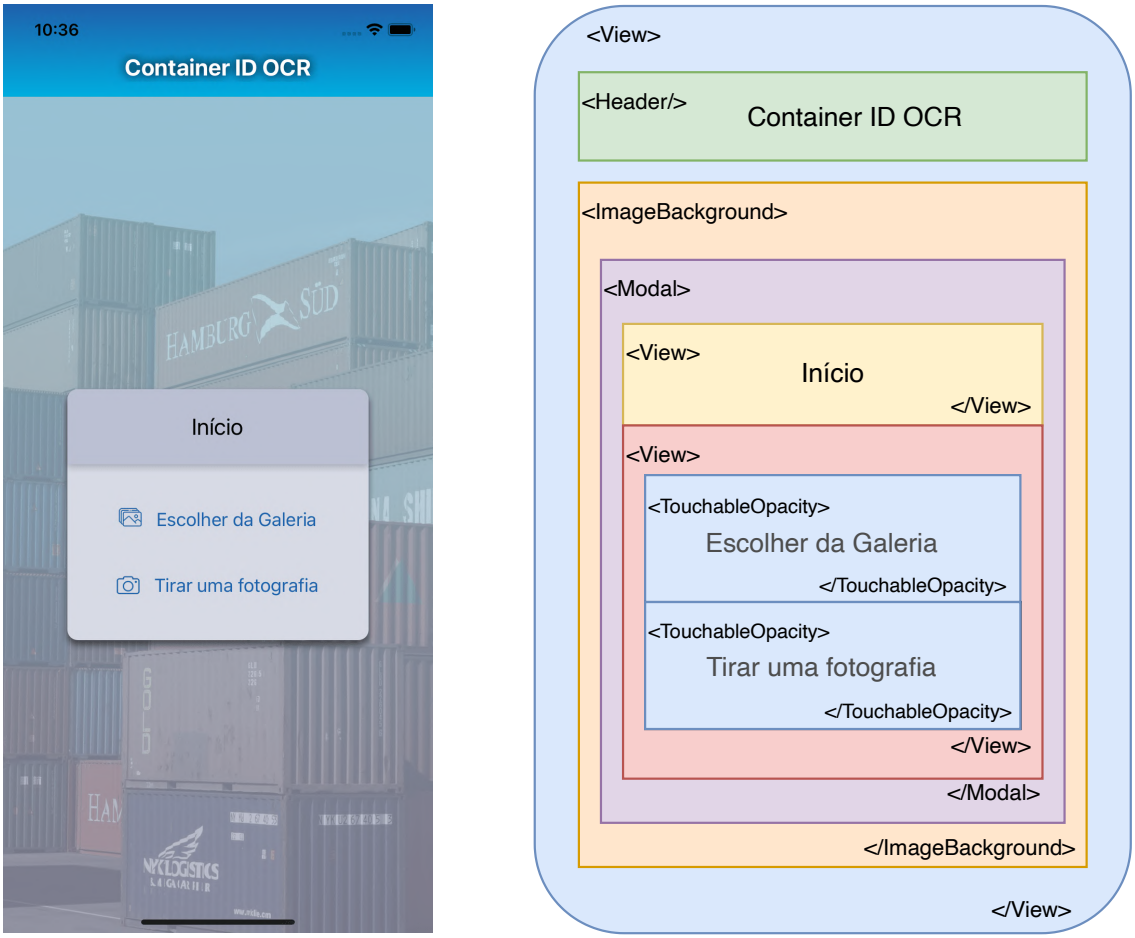


Figura 4.2: Exemplo de utilização dos componentes anteriormente descritos numa página da aplicação.

Para alem dos componentes anteriormente referidos, foram, ainda, utilizados os seguintes:

- ***Pressable*** – Também uma zona pressionável, como a componente *TouchableOpacity*, mas com um maior número de funcionalidades. Na página da câmara, por exemplo, este componente é utilizado no disparador para que, ao pressionar o botão, este diminua de tamanho criando um efeito de botão físico.

- ***Image*** – Ao contrário do *ImageBackground* que serve para configurar o fundo, este serve para apresentar uma imagem em qualquer parte do ecrã.

- ***TouchableWithoutFeedback*** – Zona pressionável, sem qualquer efeito visível. Neste caso é utilizado para manter a aplicação da forma mais natural possível. Aquando da utilização do teclado, o componente serve apenas para o esconder caso o utilizador pressione fora dele, sendo este o funcionamento natural da maior parte das aplicações móveis.

- ***ActivityIndicator*** – Este componente serve para mostrar ao utilizador que a aplicação está a executar uma ação. No caso desta, este aparece enquanto o *Back-end* está a fazer o reconhecimento do ID, ou seja, quando a aplicação se encontra à espera da resposta vinda do *Back-end*.

Por fim, foram, ainda, utilizados dois componentes não disponibilizados pelo RN, mas sim pelo *Expo*:

- ***expo-camera*** – Componente que consegue utilizar a câmara física do dispositivo, tanto a frontal como a traseira, dando uma visualização do que é captado pela mesma em tempo real, como se da aplicação da câmara nativa do dispositivo se tratasse.

- ***Expo-image-picker*** – Componente utilizado para visualizar a galeria de fotografias do dispositivo.

4.1.2 Métodos Utilizados

Para além de interface gráfica, na aplicação foram também implementadas uma série de operações necessárias para o funcionamento do sistema:

- **Tirar a fotografia com a câmara** – Como foi anteriormente descrito, foi utilizado o componente *expo-camera* para renderizar uma visualização da câmara, contudo seria ainda necessário capturar uma fotografia, assim, para tal foi utilizado o método ***takePictureAsync()*** que tira uma fotografia e a guarda.

- **Abrir a galeria** – Para abrir a galeria foi utilizado o método ***launchImageLibraryAsync()*** que abre a visualização da galeria de imagens do dispositivo. Este método foi parametrizado da seguinte forma:

  - Tipos de multimédia: *ImagePicker.MediaTypeOptions.Images*;
  - Permite edição: falso;
  - Qualidade: 1;

- **Comunicação com o *Back-end*** – Para enviar a imagem para o *Back-end* e, consequentemente, receber a resposta foi necessário criar uma ponte entre ambos. Para tal, foi utilizada a tecnologia *Fetch API* que cria uma interface *JavaScript* para aceder e manipular partes do *pipeline HTTP* como pedidos e respostas. Este método é parametrizado com o endereço do *Back-end* e também com as opções do pedido, sendo estas opções apresentadas na seguinte Figura 4.3.

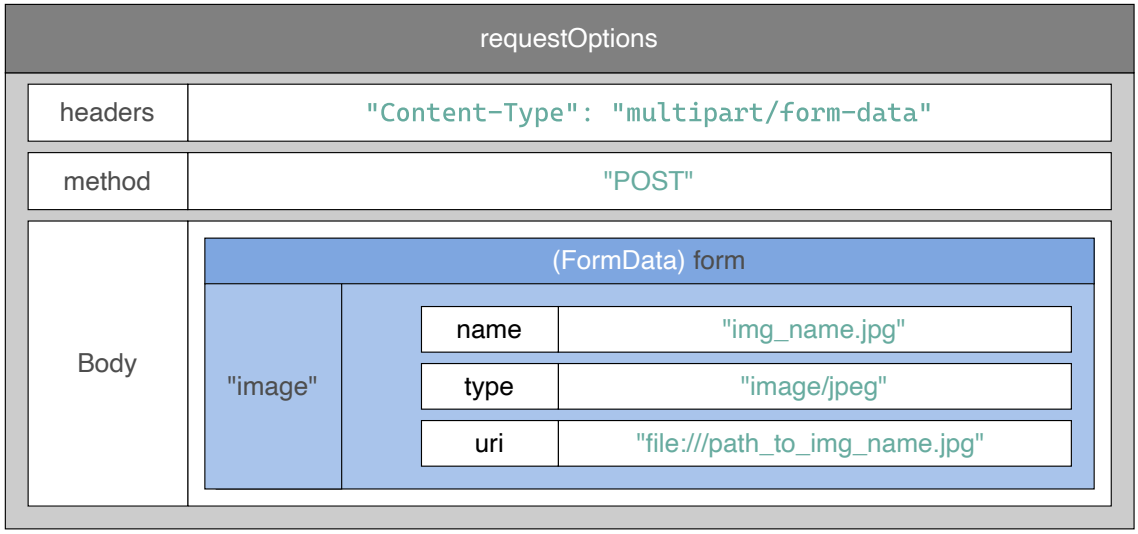


Figura 4.3: Formato do pedido onde é possível ver cada parte e os tipos de informação passada em cada uma dessas partes.

Este método prevê ainda a receção do ID como resposta do método, este ID surge no formato de string e seguirá para um método de verificação.

- **Verificação do ID** – Para verificar se o ID chegou à aplicação no formato correto, recorreu-se a este método que utilizou a tecnologia *Regex* (expressão regular) geralmente utilizada para identificar cadeias ou padrões de caracteres. A verificação percorre todos os caracteres do ID e verifica se os 4 primeiros são letras maiúsculas e se os 7 seguintes são dígitos. Caso não sejam verificadas as condições o ID recebido será eliminado e a aplicação apresentará uma página de erro.

4.1.3 Métodos de Segurança

Por questões de segurança, ao longo da utilização da aplicação serão pedidas várias permissões ao utilizador de modo a não ocorrerem situações de utilização indevida de recursos pertencentes ao dispositivo utilizado. Para tal foram utilizados os seguintes métodos:

Permissão	Método	Explicação
Câmara	***Camera.requestPermissionsAsync()***	- Pede ao utilizador permissão para que a aplicação possa utilizar a câmara.º
Galeria	***ImagePicker.requestMediaLibraryPermissionsAsync()***	- Pede ao utilizador permissão para que a aplicação utilize a galeria de fotografias.
Internet	• *Android: usesCleartextTraffic="true"* • *IOS:NSAllowsArbitraryLoads*	- Permite que a aplicação consiga comunicar com o *Back-end* através da *pipeline HTTP*.

Tabela 4.1: Métodos utilizados para garantir permissão para se usar Câmara , Galeria de fotos e comunicações via Internet.

4.2 Implementação do *Back-end*

Em desenvolvimento de *software* e, principalmente, em aplicações *web* ou móveis estes sistemas são, geralmente, e divididos em duas partes: o *Front-end* que será a interface com o utilizador que, normalmente, é de simples compreensão para o mesmo; e, o *Back-end* que, ao contrário do anterior, é invisível para o seu utilizador, correspondendo à parte onde se encontram os algoritmos e a manipulação de dados que posteriormente são apresentados ao utilizador através do *Front-end*. Desta definição de conceitos foi escolhido o nome de *Back-end* para este componente, visto que será a parte invisível ao utilizador, situada num servidor e onde se encontram os algoritmos e se faz a manipulação dos dados do sistema, como já retratado anteriormente.

Tal como para o desenvolvimento da aplicação, para o *Back-end* também foi feita uma breve pesquisa para que se compreendesse quais as tecnologias a usar para o próprio funcionamento deste componente e, também, para os algoritmos de localização e reconhecimento do ID a partir da imagem recebida da aplicação.

Para editor de código foi escolhido, novamente, o *Visual Studio Code*, como anteriormente, pelas suas funcionalidades e fácil utilização.

O *Python*[46] foi a linguagem de programação escolhida para o desenvolvimento deste componente por diversos fatores, como ser amigável para o utilizador, ter uma semântica bastante dinâmica e, o fator mais importante, por ser orientada a objetos, ideal para o treino e teste dos algoritmos de localização e reconhecimento utilizados.

Este componente foi dividido em três partes principais - um algoritmo de deteção, um algoritmo de reconhecimento e por fim uma parte comum onde são integrados os dois algoritmos anteriores com a comunicação com a aplicação.

4.2.1 Algoritmo de Localização do ID

Aquando da pesquisa inicial para este componente, foram selecionadas duas abordagens que poderiam ser utilizadas para executar a localização do ID na imagem recebida da aplicação. Foram ambas implementadas e testadas com o intuito de ser utilizada no projeto final aquela que trouxesse uma maior qualidade de deteção.

4.2.1.1 *EAST Model*

Os autores do *EAST: An Efficient and Accurate Scene Text Detector*[47] desenvolveram uma *Fully Convolutional Network* (FCN), apresentada na Figura 4.4, capaz de detetar texto de forma correta e eficaz num cenário natural. Inicialmente esta arquitetura foi selecionada pela sua eficácia em cenários naturais, como é o caso do propósito do sistema desenvolvido, e, também, por utilizar FCN que ao contrário das CNN tem uma densidade de camadas muito menor e que faz com o que tempo de execução seja menor.

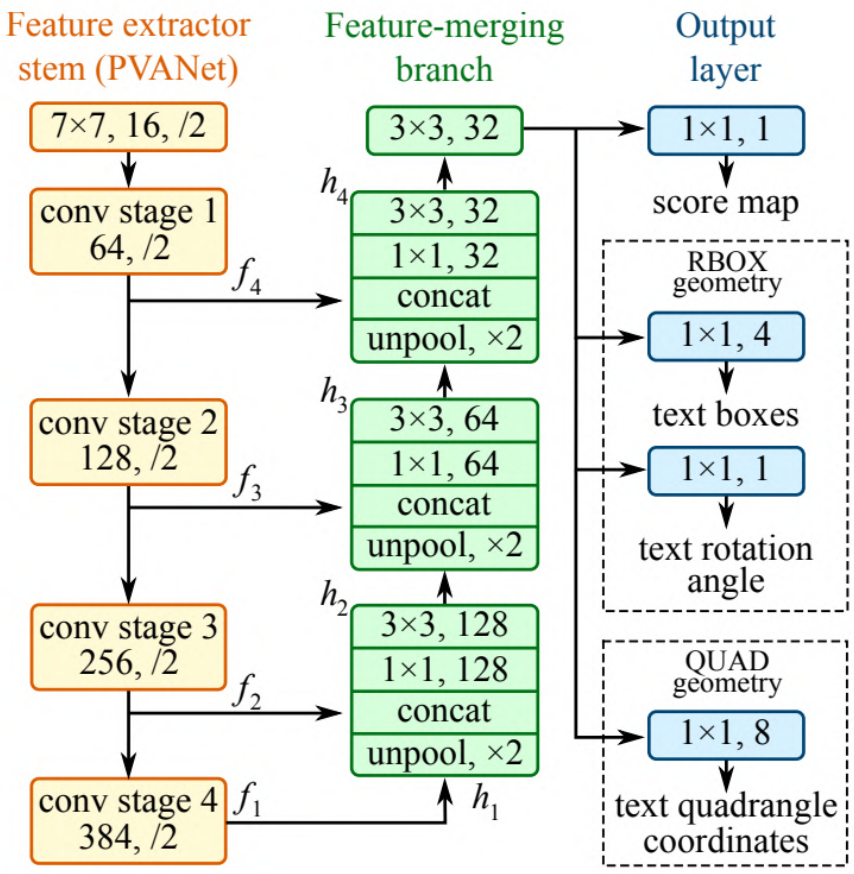


Figura 4.4: Estrutura da FCN desenvolvida pelos autores do artigo [47].

Este modelo foi previamente treinado e disponibilizado pelos autores na biblioteca *OpenCv*[48]. Esta biblioteca foi desenvolvida pela *Intel* com o objetivo de facultar aos desenvolvedores de software uma ferramenta de visão computacional. É usada para todos

os tipos de análise e manipulação de imagens e vídeos. Como resultado da predição, o modelo retorna um conjunto de *arrays*, cada um deles com a localização de cada ROI encontrada na imagem previamente submetida para predição. A localização das regiões preditas consiste nas coordenadas do vértice superior esquerdo e do vértice inferior direito de um retângulo.

Uma grande desvantagem deste modelo é dar uma grande lista de ROI's, ou seja, todas as palavras ou partes de texto localizados e muitas vezes, também, outras regiões que não tem texto, mas que o modelo reconhece. Este facto é bastante visível na Figura 4.5, e trouxe para o desenvolvimento do sistema a necessidade de uma funcionalidade para perceber qual as ROI's que realmente eram necessárias para o reconhecimento devido.

Figura 4.5: Exemplo de ROI's (a vermelho) produzidas pelo modelo.

Primeiramente foi necessário resolver a questão dos pequenos fragmentos localizados e que não contem nenhum texto. Assim, para tal, foi calculada a área de cada um das ROI's e, seguidamente, foram eliminadas as ROI's que não ultrapassassem 30% da área da ROI maior.

Tendo-se, apenas por base, as maiores ROI's surgiu o problema de muitas vezes o ID ser reconhecido em três ROI's separadas, uma reconhecendo as primeiras quatro letras, outra com os seis dígitos seguintes e outra apenas com o dígito de verificação. Foi, então, criado um método que juntava as ROI's que tivessem no mesmo plano horizontal.

Apesar de resolvidas todas estas questões, a percentagem de reconhecimentos ainda era bastante limitada.

4.2.1.2 *Object Detection*

A outra abordagem selecionada foi a utilização de um algoritmo de deteção de objetos treinado com um *dataset*, criado com o fim de localizar o ID de cada imagem. O primeiro passo do desenvolvimento desta abordagem foi, então, a criação de um *dataset*.

Para a criação do *dataset* foram utilizadas cerca 190 imagens, disponibilizadas pelo responsável da empresa de operações portuárias. Estas fotografias foram recolhidas pelo próprio aos contentores que chegaram e partiram do porto da Horta durante o período de uma semana. Foram, ainda, adicionadas cerca de 120 imagens retiradas do *Google Images*[49]. Para que o modelo conseguisse perceber onde está o/os ID(s) em cada imagem foi preciso etiquetar cada imagem com a localização de cada ID e a sua classe. Todo o *dataset* foi etiquetado com duas classes diferentes - "Horizontal" e "Vertical" - dependendo da orientação do ID.

A etiquetagem foi feita com recurso ao programa *LabelImg*[50], uma ferramenta para fazer anotações e que exporta para cada imagem um ficheiro *.txt* com as anotações de cada imagem. Estas anotações são a classe e a localização de cada objeto com o seguinte formato:

- <class id> <Xo/X> <Yo/Y> <W/X> <H/Y>

Onde:

- **class id** – Indice da classe a ser anotada (Horizontal – 0, Vertical – 1);
- **Xo** – Coordenada em X do centro da caixa;
- **Yo** - Coordenada em Y do centro da caixa;
- **W** – Largura da caixa;
- **H** – Altura da caixa;
- **X** – Largura da imagem;
- **Y** – Altura da imagem;

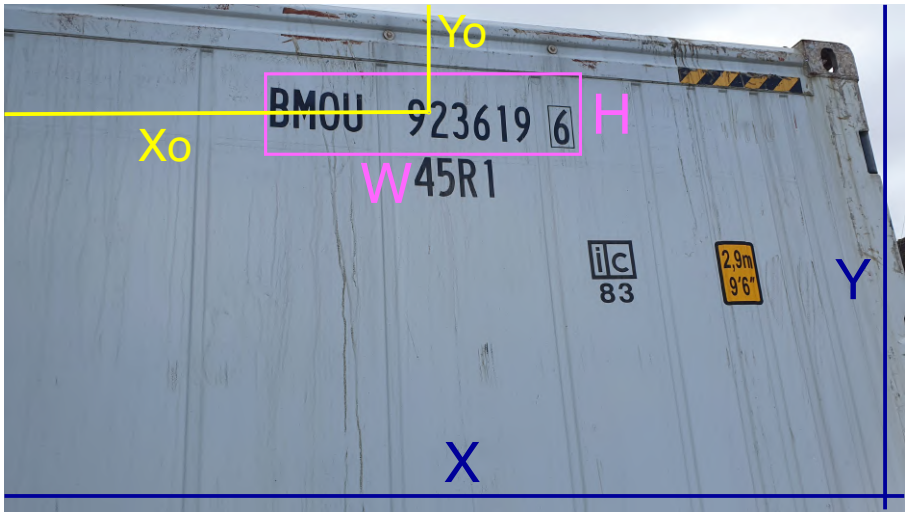


Figura 4.6: Exemplo da etiquetagem de uma imagem e a explicação de cada parâmetro da anotação.

O *dataset* no fim de estar devidamente etiquetado foi, então, dividido em duas partes: uma de treino com 80% das imagens e outra de teste com 20% das imagens.

Modelo

Com o *dataset* definido, centraram-se as atenções no algoritmo de deteção de objetos. Existe, atualmente, uma vasta gama dos mesmos, cada um deles baseados nas mais variadas tecnologias. A maior parte dos detetores baseados em CNN têm níveis de precisão bastante elevados, no entanto apresentam um tempo de predição igualmente elevado proveniente das suas arquiteturas longas e complexas. Para treinar, e usar, estes modelos são necessárias Graphics Processing Unit (GPU) potentes e, por sua vez, bastante dispendiosas.

Como neste sistema era fundamental, ao contrário dos anteriores, que fosse utilizada uma localização do ID o mais breve possível, selecionou-se para o desenvolvimento o modelo de deteção de objetos *YOLOv4*, sendo que mantem um nível de precisão alta e consegue obter velocidades de predição bastante mais elevadas.

O *YOLOv4* é a quarta versão do modelo *YOLO*, que se apresenta como um modelo de deteção de objetos, lançado no mês de Abril do ano de 2020 por *Alexey Bochkovsiy* [51]. O modelo *YOLO* foi desenvolvido para que pudesse ser utilizado em tempo real e tem, apenas, por base uma simples CNN que prevê múltiplas ROI calculando, simultaneamente, a probabilidade de cada classe para essa ROI. Este modelo está preparado para ser treinado com um *dataset* de imagens de um determinado objeto, para que quando a este modelo seja apresentada uma determinada imagem com o objeto em questão, a CNN já tenha os pesos alterados para identificar características relevantes desse objeto na imagem[38]. O modelo *YOLOv4* melhorou em relação à versão anterior, o *YOLOv3*, no que toca à velocidade de predição em funcionamento em 10% e, principalmente, na

precisão dos resultados com um aumento de 12%[51], estas comparações são visiveis na Figura 4.7.

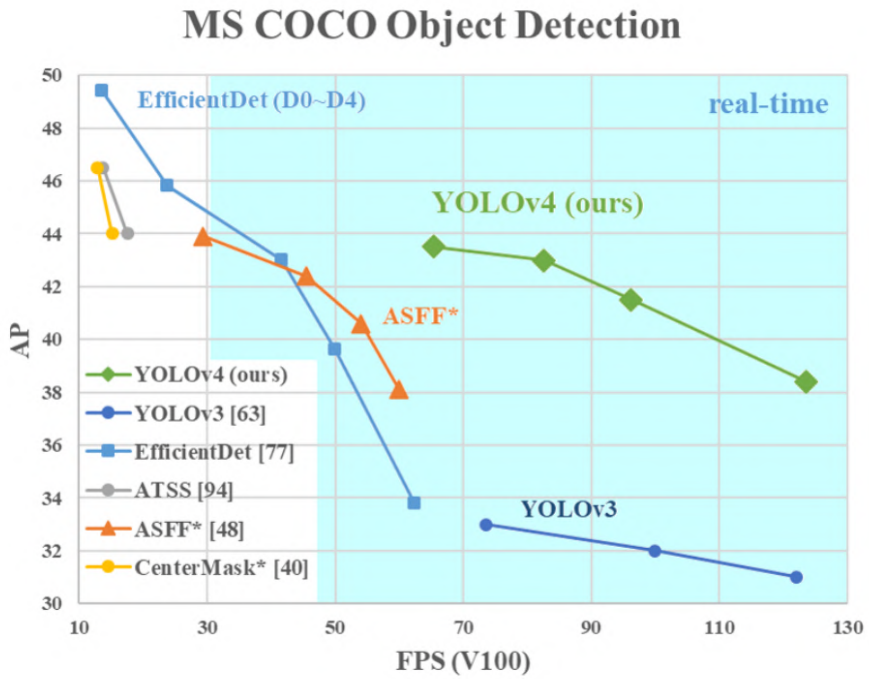


Figura 4.7: Comparação da velocidade de execução, e da precisão do modelo aprensentado pelos autores em comparação com outras versões ou outras abordagens [51].

Treino

Para executar o treino do modelo escolhido com o *dataset* anteriormente definido foi utilizada a plataforma *Google Colab*[52] que é um editor de *Jupyter notebooks*[1] no *browser* especialmente desenvolvido para ML e análise de dados, e que fornece acesso a recursos de computação, especialmente GPU. Esta revela-se bastante útil dado não ser necessário utilizar recursos da própria máquina, sendo que assim o treino pode ser um processo mais curto do que se fosse realizado utilizando os recursos da máquina.

Foi, ainda, utilizada a ferramenta *Darknet*[53], uma *Framework* de redes neuronais *open-source* escrita em linguagem C e tecnologia *CUDA*, o que faz com que o treino realizado do modelo e, posteriormente, a execução do modelo seja efetuada mais rapidamente permitindo, ainda, realizar ações computacionais na GPU.

Para a execução do treino foi necessário importar, para o ambiente do *Google Colab*, a ferramenta *Draknet*, o modelo *YOLOv4* e, ainda, o *dataset*.

[1] São documentos que podem conter código *Python* e tambem texto, figuras, equações. São documentos legíveis contendo descrições da análise e resultados, bem como documentos executáveis realizando ações e análise de dados.

```
# download and compile darknet_for_colab
!git clone https://github.com/quangnhat185/darknet_for_colab.git
%cd darknet_for_colab
# download YOLOv4 Model
!wget https://github.com/AlexeyAB/darknet/.../yolov4.conv.137
#download and unzip dataset
!wget --no-check-certificate "https://onedrive.<...>jqSCs" -O ts.zip
!unzip ts.zip
```

Listing 1: Codigo *Python* para importar *Darknet*, modelo e *dataset*.

Seguidamente, foi feita a configuração do treino com os seguintes valores:

Variável	Significado da variável	Valor
classes	Número de classes presentes no *dataset*	2
max_batches	Número máximo de iterações do treino	8000
batch	Número de imagens processadas numa iteração	64
subdivisions	Número de *mini_batches* num	16
	Valor de largura que a imagem terá depois de redimensionada para o tamanho da rede	416
height	Valor de altura que a imagem terá depois de redimensionada para o tamanho da rede	416
channels	Cada imagem será convertida em X canais (neste caso 3)	3
momentum	Quanto as iterações passadas afetam as seguintes	0.949
learning_rate	Quanto o modelo ajusta os pesos a cada iteração	0.001
steps	Em que iterações o *learning rate* vai multiplicar pelas *scales*	(6400,7200)
decay	Atualização dos pesos para características típicas	0.0005
scales	Quanto o *learning rate* vai multiplicar	(0.1,0,1)

Tabela 4.2: Parâmetros e, valores dos mesmos, de configuração inicial do treino do modelo.

Por fim, bastou iniciar o treino utilizando a ferramenta *Darknet*. No fim da execução do treino foram guardados o ficheiro de configuração da rede e, ainda, os pesos da rede na melhor iteração do treino:

4.2.1.3 Utilização do Modelo

Com o modelo já treinado foi possível aplicá-lo ao algoritmo de localização do ID, no método ***ID_locator***. Este começa por ler o modelo a partir dos pesos retornados pelo

```
#set the init values for training
!python yolov4_setup.py
#start training
!./darknet detector train data/yolov4.data cfg/yolov4_custom_train.cfg
        yolov4.conv.137  -map
```

Listing 2: Comandos de inicializar os valores e começar o treino.

treino e pela configuração da rede.

Já com a rede lida, é possível, a partir desse momento, fazer a localização do ID. Este inicia-se pela leitura da imagem utilizado o método ***imread()*** disponibilizado pela biblioteca *OpenCv*. Seguidamente é criado um *blob* a partir da imagem e inserido no input da rede. São lidas as camadas de output da rede que retornam a localização, a classe prevista, e o nível de confiança para cada um dos ID's encontrados na imagem.

Posteriormente, caso tenham sido detetados mais de que um ID, é feita uma verificação de qual deles é o que está mais próximo, ou seja, aquele cujo retângulo em volta do ID terá uma área maior.

Por fim, este método retorna a variável *result* que contem a classe resultante da predição e da localização do ID.

4.2.2 Algoritmo de Reconhecimento do ID

Tal como no algoritmo de localização, foram equacionadas duas abordagens diferentes, ambas testadas com o objetivo de verificar qual delas seria melhor para este sistema.

4.2.2.1 *Tesseract*

Tesseract é uma ferramenta de OCR *open-source* desenvolvida por *Hewlett-Packard* (HP) como um projeto de doutoramento de 1984 a 1995. Em 2005 a ferramenta foi tornada *open-source* pela HP. De 2006 a 2018 foi desenvolvida pela *Google* sendo que, atualmente, se encontra disponível no Github[54]. A ferramenta foi desenvolvida em C++.

Para a utilização no sistema, e como todo o *Back-end* estava a ser desenvolvido em *Python*, foi utilizada a biblioteca *pytesseract*[55], um *wrapper* do *Tesseract* para ser utilizado em *Python*.

Para a utilização desta tecnologia é necessário que a imagem que será aplicada tenha as letras pretas em fundo branco, ou seja, é necessário um método de pré-processamento de imagem aplicado à imagem do ID para que a biblioteca funcione corretamente.

O método de pré-processamento engloba uma serie de processos, entre eles:

- ***IMREAD_GREYSCALE*** – leitura da imagem em escala de cinzentos.

- ***medianBlur*** – técnica de filtragem utilizada para remover o ruido da imagem.
- ***treshold*** – técnica de binarização, neste caso usando o método de *Otsu*, esta binarização transforma cada pixel em preto ou em branco.
- ***morphologyEx(, cv2.MORPH_OPEN,)*** – transformação morfológica de abertura que consiste numa erosão seguida de uma dilatação.
- ***warpAffine*** – uma rotação orientada com o objetivo de ter o ID horizontal ou vertical caso a imagem tenha sido tirada de um ângulo que não seja favorável ao reconhecimento.

Mesmo com o pré-processamento da imagem a ser reconhecida, o método era ainda bastante fraco para esta utilização, pois o *Tesseract* foi desenvolvido para fazer reconhecimento de texto em documentos ou faturas, ou seja, em condições quase perfeitas, o que não é o caso deste sistema pois os próprios contentores não são lisos o que cria sempre sombras e cria, consequentemente, riscas verticais quando a imagem é tirada. Outro fator deve-se ao seu uso regular, sendo que apresentam defeitos que dificultam o reconhecimento.

4.2.2.2 Object Detection

No reconhecimento, tal como na deteção, a segunda abordagem testada, foi a que teve um melhor resultado e, mais uma vez, foi utilizado um algoritmo de deteção de objetos com *YOLOv4*. No entanto, foi utilizado um *dataset* diferente.

Para a preparação do *dataset* foram utilizadas as mesmas imagens que no *dataset* do algoritmo de reconhecimento. Foram recortados e guardados em imagens todos os ID's contidos nessa imagem.

Foi feita uma etiquetagem, novamente com o *LabelImg*, mas desta vez foram utilizadas 36 classes diferentes, 26 letras e 10 dígitos. Este *dataset* foi também divido em partes de 80% para treino e 20% para teste

O processo de utilização do modelo tem uma fase inicial semelhante ao de localização. Este método ***id_Recognizer*** inicia-se pela leitura do modelo a partir dos pesos retornados pelo treino e ainda pela configuração da rede. Seguidamente faz a leitura da imagem, a criação do *blob*, que é posteriormente inserido como *input* na rede. Como output da rede apresenta-se a localização, a classe e a confiança da predição de cada letra.

Este método tem como parâmetro de entrada a classe vinda do algoritmo de localização, necessária para fazer a construção do ID, ou seja, com o auxílio do método *id_maker* é inicialmente verificado se o ID se encontra na horizontal ou na vertical. Caso esteja na vertical, as localizações vindas do output da rede são organizadas de forma que o ID comece na letra mais acima e acabe na mais abaixo, caso seja horizontal o processo é semelhante,

mas o ID será lido da esquerda para a direita. Este método retorna uma string com o ID para o método principal.

4.2.3 Método Principal

De modo que o *Back-end* receba imagens da aplicação e possa responder com o ID reconhecido foi criado uma *RESTful API*, que fornecerá um meio de comunicação onde o *Back-end* estará à espera de receber uma informação proveniente da APP. Para a criação deste *API* foi utilizado o *Flask*[56] uma *microframework web* escrito em *Python* que tem como objetivo o desenvolvimento de aplicações *web*.

Nesta *API* foi definido um *endpoint* para receber *HTTP POST requests* provenientes da APP e que, no seu conteúdo, tem uma imagem. Este *endpoint* ficará sempre a espera de receber uma imagem enquanto o *Back-end* estiver ativo.

```
1 app = Flask(__name__)
2
3 @ app.route("/API", methods=["POST"])
4 def process_image():
5     file = request.files['image']
6     resultado = engine(file)
7     return resultado
```

Listing 3: Criação da *API* com a ferramenta *Flask* e inicialização do *endpoint "/API"* do tipo *POST*, seguido da gestão da informação recebida no *endpoint*.

Quando o *endpoint* recebe uma mensagem que contem uma imagem no formato correto passa para o método ***engine()*** que faz a gestão dos dois algoritmos falados visivel na Figura 4.8. O método tem como parâmetro o ficheiro recebido pela *API* no *endpoint*. Este ficheiro é consultado, com auxílio da biblioteca *werkzeug*[57] (uma biblioteca de ferramentas necessárias para o desenvolvimento de *API*'s), o nome do ficheiro e este é apresentado na linha de comandos onde o *Back-end* foi ativado. Assim, surge uma mensagem referindo que o ficheiro com o nome consultado foi recebido. Seguidamente, este método guarda temporariamente a imagem, e é chamado o método de localização do ID que retornará à localização do ID e a sua *label*, neste caso a orientação do identificador. Com esta localização a imagem original é cortada, ficando uma imagem com o ID, sendo que esta será utilizada, seguidamente, no algoritmo de reconhecimento do ID. Este último retornará uma *String* com o ID que será enviada para a APP como resposta ao *request* feito.

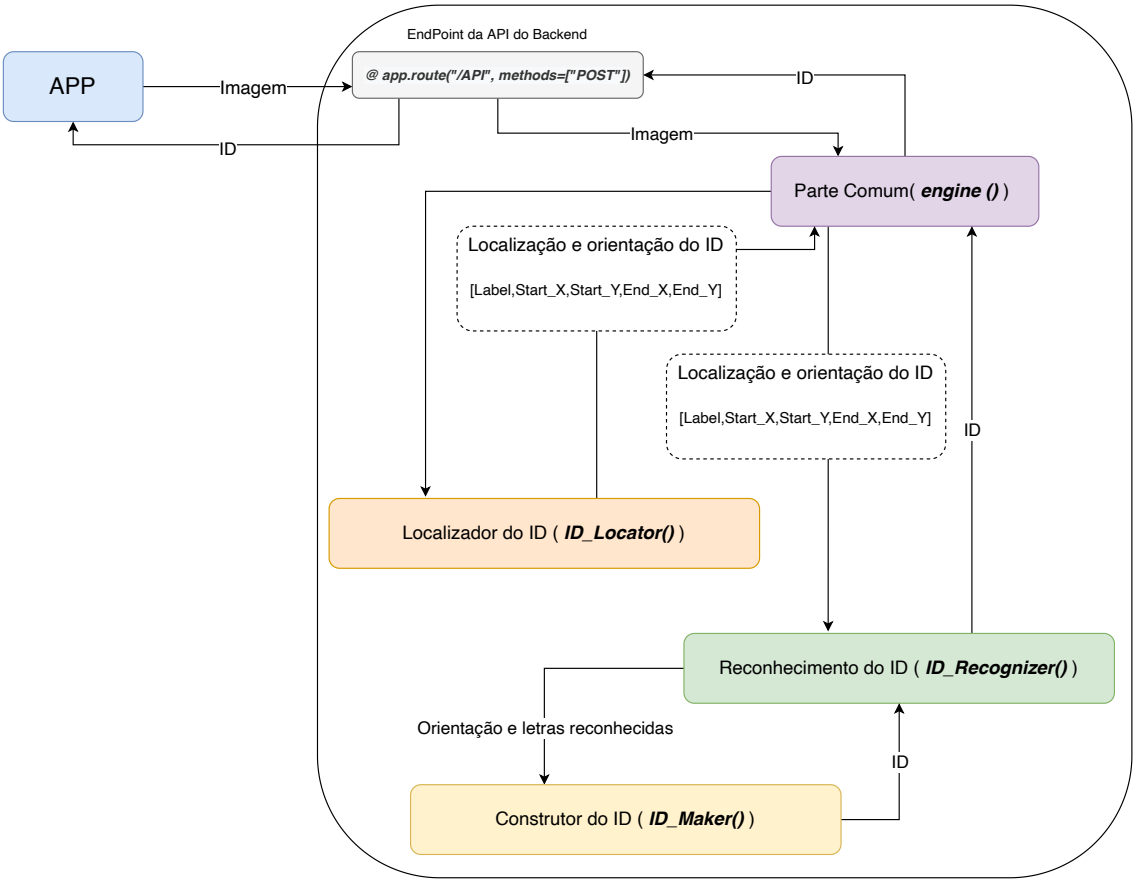


Figura 4.8: Diagrama de funcionamento dos métodos implementados no *Backend*.

5

Testes e Validação

Este capítulo apresenta uma avaliação ao sistema desenvolvido feita através de testes aos componentes do sistema.

5.1 Modelo de localização do ID

Nesta secção será apresentada a análise dos testes realizados ao modelo utilizado, mais especificamente o treino do modelo, a execução da deteção do ID e, ainda, os tempos de deteção. Por fim, será feita a comparação entre o modelo utilizado e o *EAST Model* aquando da implementação do mesmo com o objetivo de se perceber qual o melhor modelo para o sistema.

5.1.1 Treino

Para satisfazer o requisito da localização do ID numa imagem, foi utilizado o modelo de deteção de objetos *YOLOv4*, treinado no *Google Colab*. O *dataset* utilizado para treinar o modelo foi construído com base em fotografias disponibilizadas pelo responsável da empresa e outras disponíveis no *Google Images*.

O *dataset*, completo, conta com 309 imagens de contentores etiquetados com a localização e classe de cada ID, contendo 377 anotações diferentes. O *dataset* foi dividido em dois conjuntos, 80% das imagens para o conjunto de treino e os restantes 20% para o conjunto de teste.

O treino foi realizado com recurso *Google Colab Pro*, versão *premium* do *Google Colab*. Utilizou-se esta versão paga com o objetivo de diminuir o tempo de treino, devido à melhor qualidade das *GPU*'s disponibilizadas. Neste treino foi utilizada a *GPU Nvidia Tesla V100* com 32 GB de memoria.

Durante a realização do treino foram monitorizados 2 fatores:

- ***Loss*** – Soma de todas as perdas das 16 divisões de cada um dos 64 *batches* utilizados por cada época de treino[58].
- ***mAP*** – *mean Average Precison* mede quão corretas são as predições no conjunto de teste. É calculado na iteração 1000 e depois é calculado de 100 em 100 iterações[59].

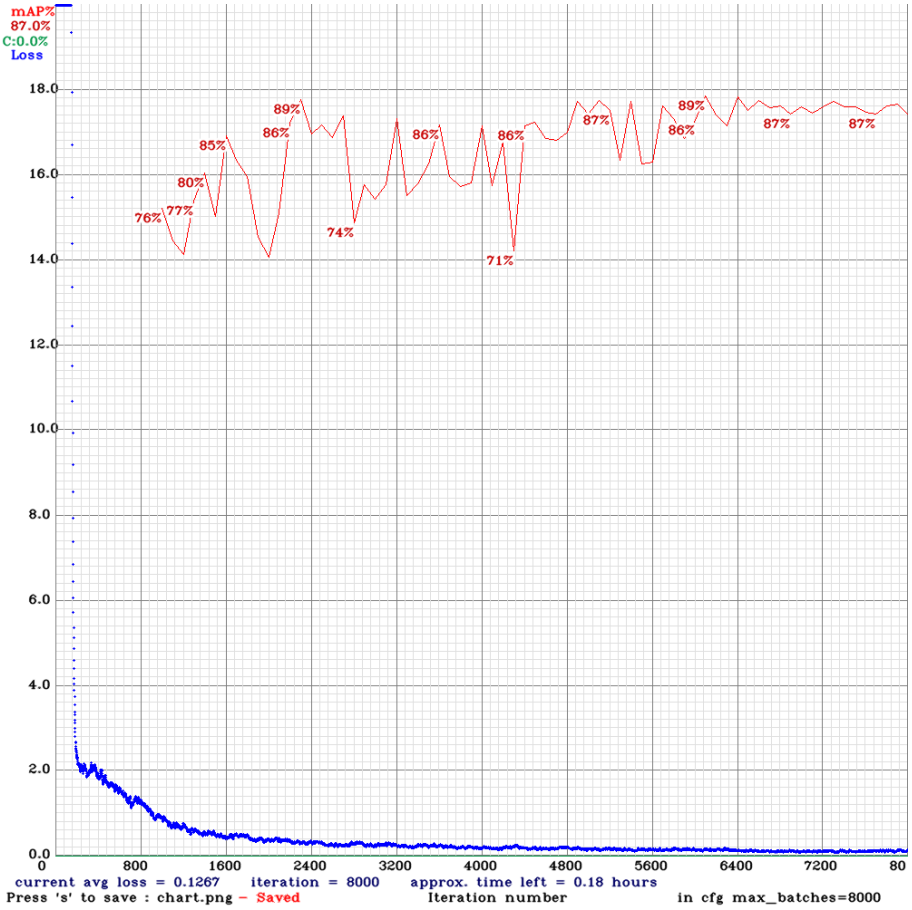


Figura 5.1: Evolução das métricas *Loss* e *mAp* ao longo do treino.

Na parametrização do treino foram selecionadas 8000 épocas, sendo que, uma época consiste na passagem pelo modelo de todos os exemplos do *dataset*. Contudo, ao analisar o gráfico da Figura 5.1, é possível verificar que o fator *mAP* foi bastante variável até à iteração 6400 e que, a partir desse ponto, começou a estabilizar até as 8000 iterações. Também é possível verificar um decréscimo acentuado da curva de *Loss* nas primeiras 500 iterações, como era espectável, sendo que continuou a aproximar-se de 0 até ao fim do treino.

Findado o treino, foi verificado que este teve, na sua melhor iteração, um *mAP* de 89.20%. Verificou-se, ainda, que na sua última iteração o treino teve um *Loss* de 0.129591 e que no geral teve um *average Loss* de 0.126748. Os dados retirados das métricas analisadas

ao longo e no fim do treino são relevantes para se perceber o sucesso do mesmo, uma percentagem alta de *mAP* revela que grande parte das predições feitas no conjunto de teste foram corretas, já o baixo valor de *Loss* mostra que não existiram grandes perdas em grande parte das épocas.

5.1.2 Localização do ID

Para além do *dataset* criado para realizar o treino, foi ainda criado um *dataset* para efetuar a validação do modelo. Este continha 74 imagens, devidamente etiquetadas, que eram todas diferentes das contidas no *dataset* de treino com o objetivo de apresentar ao modelo imagens que nunca tinham sido visualizadas pelo mesmo. Com este processo de validação, foi possível fazer uma avaliação do modelo de deteção do ID utilizando algumas métricas de classificação.

- ***True Positive (TP)*** – Quando o modelo prevê que o objeto pertence a uma das classes e que, efetivamente, pertence a essa classe.
- ***True Negative (TN)*** – Quando o modelo prevê que o objeto não pertence a nenhuma das classes e que, efetivamente, não pertence a nenhuma.
- ***False Positive (FP)*** – Quando o modelo prevê que o objeto pertence a uma das classe mas na realidade não pertence à classe em questão.
- ***False Negative (FN)*** – Quando o modelo prevê que o objeto não pertence a nenhuma classe mas o objeto pertence a uma das classes.

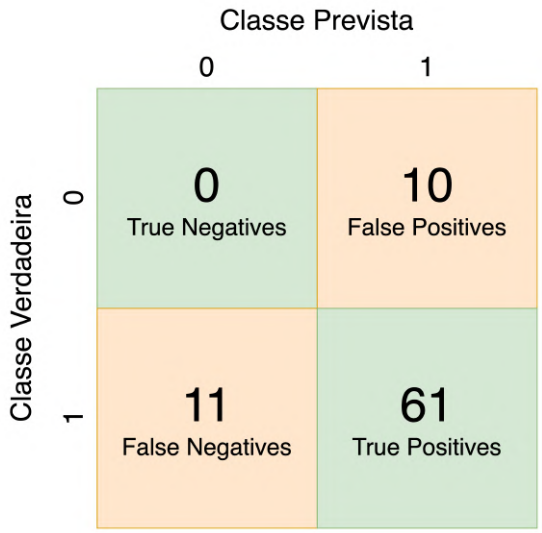


Figura 5.2: Resultado do teste feito como o conjunto de *dataset* no modelo de localização do ID.

Com as métricas anteriores calculadas e representadas na Figura 5.2 foi possível calcular outras métricas mais detalhadas e que são capazes de demonstrar a qualidade do modelo e do treino efetuado ao mesmo.

- ***Precision*** – Descreve a razão de exemplos relevantes (*True Positives*) ao longo de todas as predições efetuadas[59].

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

- ***Recall*** – Descreve a razão de exemplos que se previu pertencerem à mesma classe[59].

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

- ***F1 Score*** – Representa a precisão de um modelo num determinado *dataset*, relaciona *Precision* com *Recall*[59].

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Chegou-se aos seguintes resultados:

Precision	*Recall*	*F1 Score*
0,85915493	0,847222222	0,853146853

Tabela 5.1: Resultados das métricas *Precision*, *Recall* e *F1 Score* no teste ao modelo de localização.

As métricas anteriormente apresentadas dão-nos a qualidade da deteção em termos da classe. Já na qualidade da localização do ID, ou seja, como resultado do modelo uma área retangular que contem o ID, esta deverá ser o mais aproximada possível do que aquela que foi anotada aquando da criação do *dataset*. Para quantificar esta qualidade foram utilizadas 2 métricas disponibilizadas pela ferramenta anteriormente falada, a *Darknet*.

- ***Intersection over Union* (IoU)** – É uma métrica calculada utilizando a área da caixa prevista pelo modelo e a área da caixa anotada no modelo. Serve para medir a precisão da localização ID.

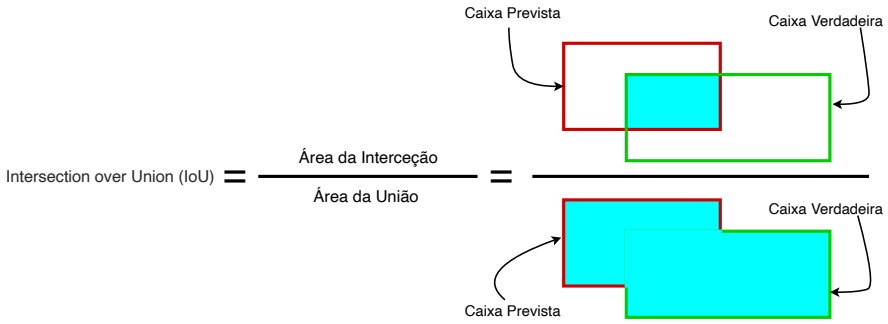


Figura 5.3: Descrição gráfica do calculo da métrica IoU.

- ***mAP@0,5*** – *mean Average Precision* com *treshold* de IoU de 50%, uma métrica disponibilizada pelo algoritmo de validação e que consiste na área abaixo da curva *Precision/Recall* calculados com um *treshold* de IoU de 0.5. Ou seja, são calculados os *True Positive*, *False Positive* e *False Negative* tendo em consideração o IoU seguindo o exemplo da figura seguinte. Caso a deteção não seja o mais preciso possível ela não vai ser um *True Positive*.

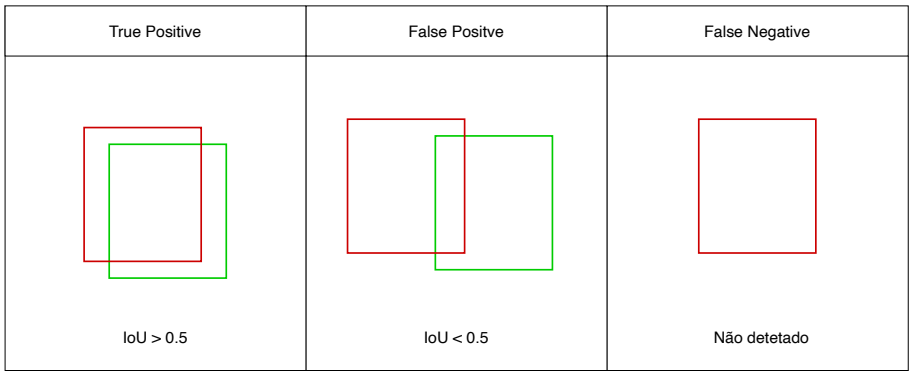


Figura 5.4: Definição de *True Positive* e *False Positive* em relação ao IoU - a vermelho a área real e a ver a área prevista pelo modelo.

O algoritmo de validação deu como output um *mAP@0.5* de 0.870312 ou 87.03%, resultando assim num valor médio de IoU de 71.99%.

Após este teste quantitativo utilizando as métricas de classificação, foi, ainda, feito um teste qualitativo utilizando apenas o modelo, parametrizando o *input* com imagens e verificando a deteção retornada pelo modelo. Chegou-se à conclusão de que a componente que tem mais importância na localização, por parte do modelo, é a resolução da imagem, ou seja, quanto menor for a resolução da imagem menor será a precisão do modelo, e até, em certos casos que não conseguiu localizar nenhum ID.

Teste	Imagem	Tempo	Teste	Imagem	Tempo
1	69.jpg	1,522	6	130.jpg	1,696
2	110.jpg	1,848	7	259.jpg	1,684
3	84.jpg	1,693	8	61.jpg	1,520
4	98.jpg	1,604	9	113.jpg	1,811
5	81.jpg	1,487	10	72.jpg	1,559

Tabela 5.2: Resultados do tempo de execução de 10 imagens com semelhante resolução.

Foi ainda testada a duração da deteção por parte do modelo, este teste foi realizado com o modelo a correr num *Apple MacBook Pro* com uma *GPU Intel Iris Plus 645*. Foram realizados 10 testes separados, com imagens diferentes, mas com a mesma resolução (2268×4032). O teste conta apenas o tempo de execução do modelo, ou seja, desde o momento em que é iniciado até que retorna a localização.

Com o teste anterior foi obtido um valor médio em segundos de 1,642278 (± 0,2), que nos garante a rapidez necessária para o seu uso diário.

5.1.3 Comparação de resultados com o *EAST Model*

Foi feita outra avaliação, inicialmente com o objetivo de perceber qual o melhor modelo a utilizar no sistema. Esta avaliação foi de caracter qualitativo, sendo que o modelo de deteção de objetos *YOLOv4* tem apenas um défice de acerto em imagens de reduzida resolução e a abordagem com o *EAST Model* tem um défice não só em imagens de baixa resolução, mas também em outras situações diversas, como quando o ID se encontra na vertical ou quando existe mais caracteres relativamente perto do ID. Assim, é visível na seguinte figura 5.5 algumas das falhas.

Foi, ainda, feita uma comparação de tempos de execução do *EAST Model* com os tempos anteriores, utilizando-se as mesmas imagens e as mesmas condições da avaliação feita ao modelo de deteção de objetos *YOLOv4*.

Teste	Imagem	Tempo	Teste	Imagem	Tempo
1	69.jpg	3,852	6	130.jpg	4,029
2	110.jpg	4,040	7	259.jpg	2,340
3	84.jpg	3,582	8	61.jpg	4,246
4	98.jpg	4,179	9	113.jpg	3,710
5	81.jpg	4,164	10	72.jpg	4,629

Tabela 5.3: Resultados do tempo de execução do *EAST Model* de 10 imagens com semelhante resolução.

Este teste resultou numa duração media em segundos de 3,8770767 (± 1,5), que é um aumento bastante significativo da duração quando comparado como o modelo utilizado no sistema.

(a) (b)

(c)

Figura 5.5: Em (a) temos uma localização correta de ambos os modelos, em (b o *EAST Model*) faz uma tripla predição e por fim em (c) a vermelho não localiza apenas o ID.Em que:Vermelho - *EAST Model*, Azul - *YOLOv4*, Verde - Área correta.

Com todos os resultados qualitativos e quantitativos relativos ao modelo de localização utilizado e também ao *EAST Model* é percetível é confirmada como uma melhoria a escolha do modelo utilizado ao invés do *EAST Model*.

5.2 Modelo de Reconhecimento do ID

Nesta secção será apresentada a análise aos testes realizados de acordo com o modelo utilizado, mais especificamente no treino do modelo, na execução do reconhecimento do ID e nos tempos de deteção.

5.2.1 Treino

Tal como para a localização do ID, foi utilizado, para o reconhecimento, um modelo de deteção de objetos *YOLOV4*, treinado no *Google Colab Pro*, utilizando uma *GPU Nvidia Tesla V100* com 31 GB de memoria. Apenas o *dataset* utilizado foi diferente pois o propósito neste caso seria localizar os caracteres e não a posição do ID, mas com uma estrutura semelhante, neste caso foram utilizadas 308 imagens de ID's de contentores devidamente etiquetados com a localização e a classe de cada caractere, contando o *dataset* com 4210 anotações no total.

Foram novamente monitorizadas as métricas *Loss* e*mAP*, e é possível verificar na figura seguinte que, ao contrário do treino do modelo de localização, este apresenta uma estabilização do indicador*mAP* a partir da iteração 4800. Contudo a curva de *Loss* tem um decréscimo muito menos acentuado devido à complexidade do *dataset*, ou seja, quanto mais classes tiver o *dataset* maior vai ser o nível de *Loss* no final do treino[60].

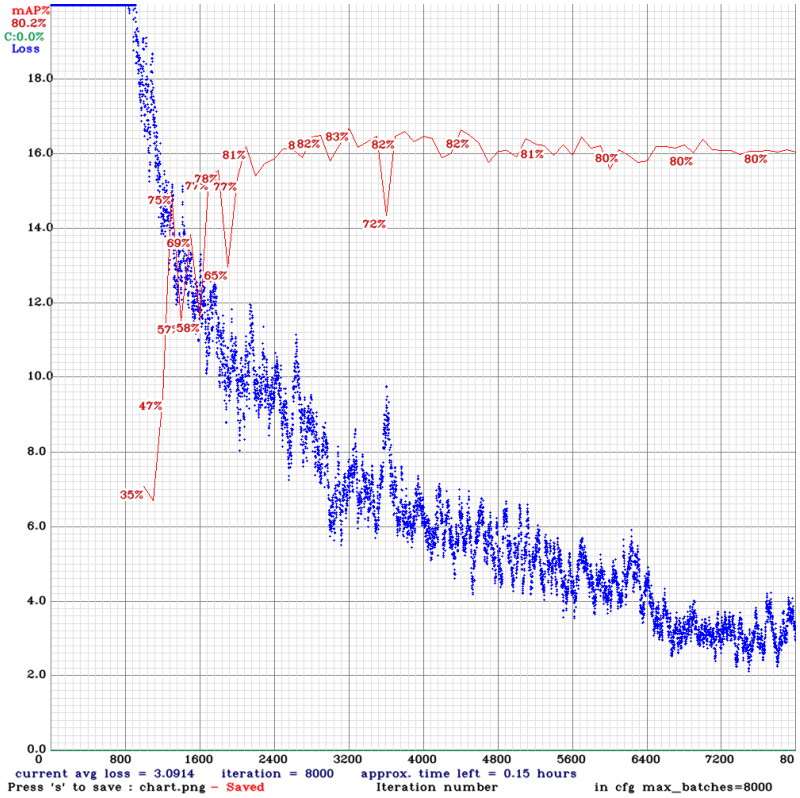


Figura 5.6: Evolução das métricas *Loss* e *mAp* ao longo do treino.

Foi verificado que a melhor iteração onde foi calculado o*mAP*, ocorreu na iteração 3200 com um valor de*mAP* de 82.11%, sendo que o treino finalizou com um valor de *Loss* de 4.025177 e um average *Loss* (valor medio de *Loss* ao longo do treino) de 3.091402.

5.2.2 Localização dos caracteres do ID

Novamente, tal como no modelo de localização do ID, foi criado um conjunto de validação com cerca de 98 imagens devidamente anotadas no formato correto. Foi aplicado a esse conjunto um teste de validação ao qual se obteve os seguintes resultados:

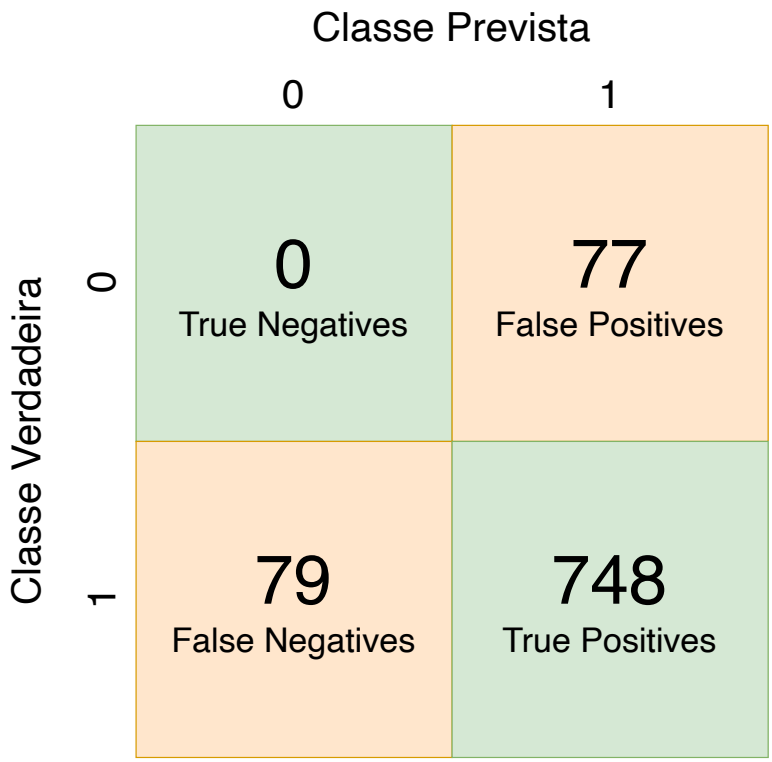


Figura 5.7: Resultado do teste feito como o conjunto de validação modelo de reconhecimento do ID.

Precision	*Recall*	*F1 Score*	*Average IoU*	*mAP@0.5*
90,67%	90,45%	90,56%	72,89 %	80,16 %

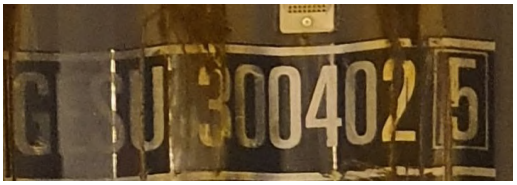
Tabela 5.4: Resultados das métricas calculadas aquando da utilização do conjunto de validação no modelo de reconhecimento do ID.

Tal como no anterior, feito um teste de caracter qualitativo onde foi possível verificar que a localização de cada caracter se encontra, no geral, bem definida pela previsão do modelo. E, de forma semelhante ao modelo de localização do ID, neste verificou-se que em imagens de resolução inferior a qualidade da previsão será, significativamente, inferior. Foi ainda verificado que devido à deterioração dos contentores poderiam existir algumas falhas, como se pode verificar na figura 5.8.

(a) Resultado: MUTU 7496960

(b) Resultado: BIT 223335

(c) Resultado: GESU 292478



(d) Resultado : U 3004025

Figura 5.8: Alguns exemplos de reconhecimentos defeituosos devido á deterioração dos contentores.

Nas mesmas condições do teste de duração da execução do modelo anterior, foram testados os tempos de execução deste com 10 imagens que apresentavam resoluções semelhantes. Percebeu-se que como se trata de imagens com um tamanho (em memória) inferior, o tempo de execução também apresentaria um valor inferior, comprovado nos ensaios realizados onde se obteve uma média de 0,8795 (± 0,2) segundos.

5.3 Aplicação

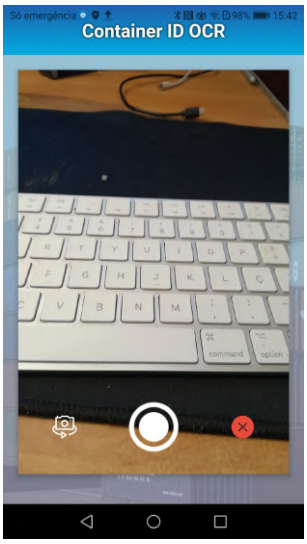
Todos os testes à aplicação e ao sistema foram realizados de quatro formas diferentes com recurso tanto a simuladores/emuladores de dispositivos móveis bem como a dispositivos físicos.

- **Emulador *Android*** – Utilizando o *Android Studio*[61] foi criado um emulador de um dispositivo *Google Pixel 4* com o SO *Android* 10.0
- **Simulador *IOS*** – Utilizando o *Xcode*[62] foi criado um simulador de dispositivo *iPhone 11 Pro* com o SO *IOS 14.5*
- ***Iphone 11*** – Dispositivo físico com um processador *A13 Bionic*, 4GB de RAM e SO *IOS* 15.0.
- ***Huawei P8 lite 2017*** – Dispositivo físico com um processador *Kirin 655*, 3 GB de RAM e SO *Android* 8.0.

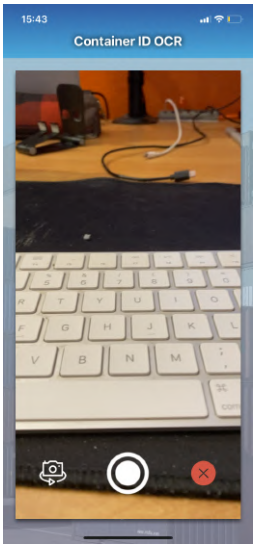
5.3.1 Verificação dos requisitos Funcionais e não Funcionais

No final da implementação da aplicação, foram verificados todos os requisitos funcionais e não funcionais do sistema. Dentro dos requisitos funcionais são apresentados, de seguida, os testes aos requisitos e os seus resultados.

- **Tirar fotografia** – A aplicação consegue efetivamente apresentar uma pré-visualização da câmara e tirar fotografias.



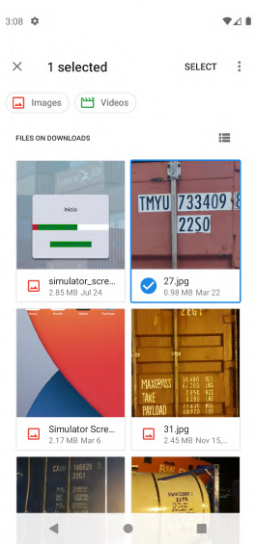
(a) *Android(Huawei)*



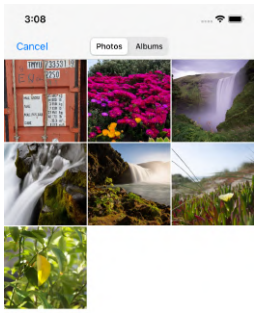
(b) *IOS(iPhone)*

Figura 5.9: Apresentação da Câmara.

- **Abrir Galeria de fotos** – Quando o utilizador escolhe a opção de escolher da galeria e seleciona "**Abrir Galeria**" a aplicação abre a galeria de fotografias nativa do dispositivo e permite ao utilizador escolher uma imagem.



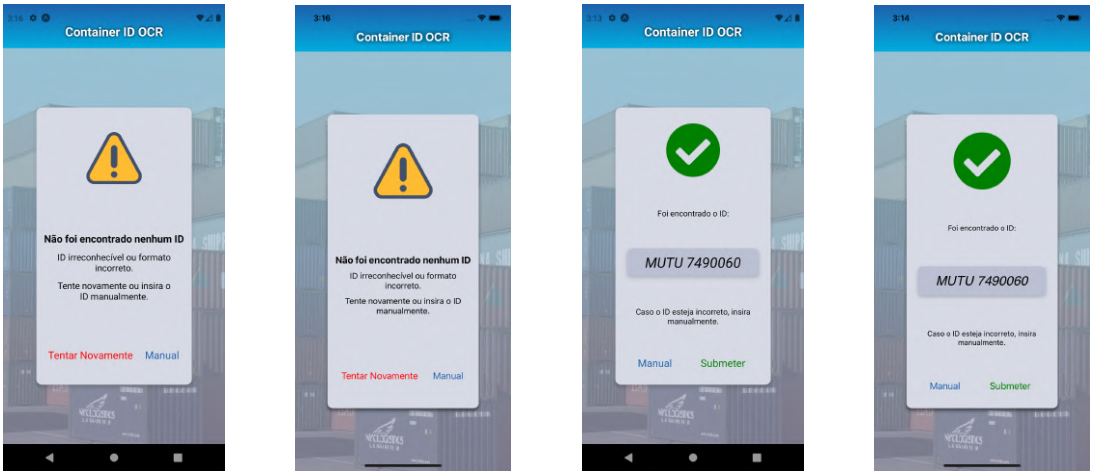
(a) emulador *Android*



(b) emulador *IOS*

Figura 5.10: Apresentação da Galeria.

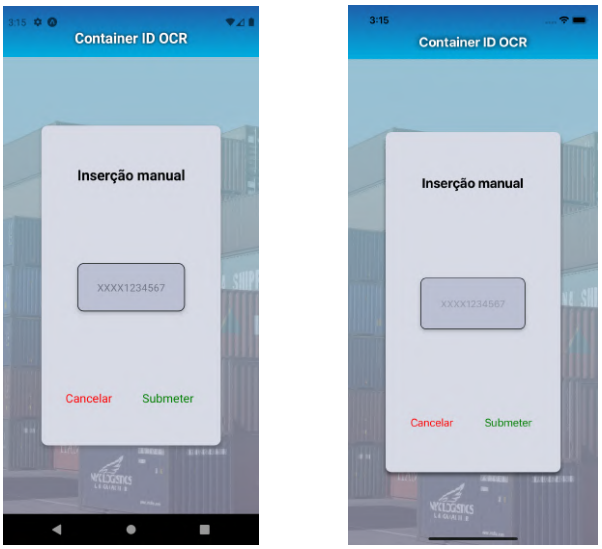
- **Verificação e Apresentação do ID** – Aquando da receção da resposta vinda do *Back-end* o algoritmo de verificação faz, efetivamente, a verificação e apresenta o ID. Caso não se verifiquem as condições necessárias, apresenta a página de ID não encontrado.



(a) *Android* (b) *IOS* (c) *Android* (d) *IOS*

Figura 5.11: Apresentação do ID quando este é reconhecido com a forma correta, ou pagina informativa de não localização do ID.

- **Inserção Manual** – Caso o utilizador verifique que o ID não está correto ou que a aplicação não o deteta ou reconhece, pode escolher a opção "Manual" e aparecerá uma caixa de texto para inserir o ID.

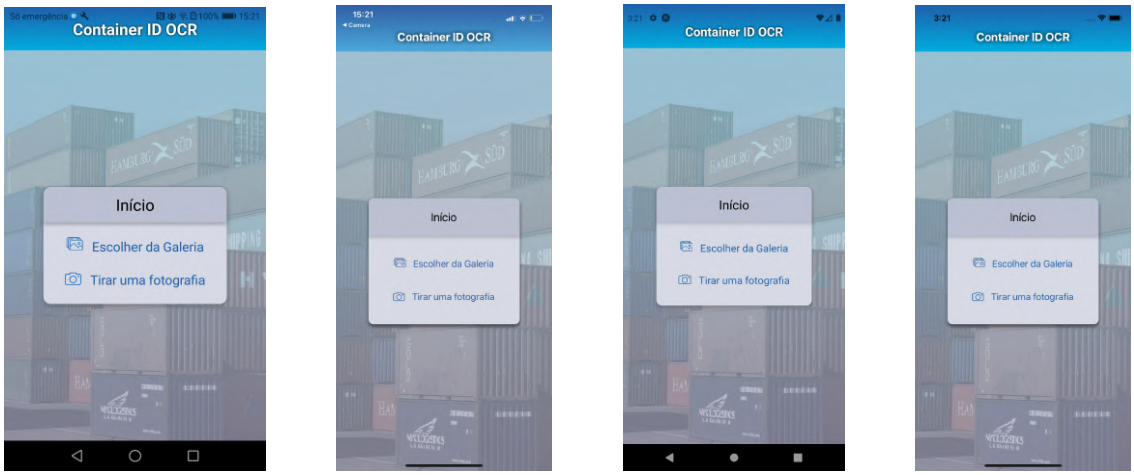


(a) emulador *Android* (b) emulador *IOS*

Figura 5.12: Inserção Manual.

Em relação aos requisitos não funcionais.

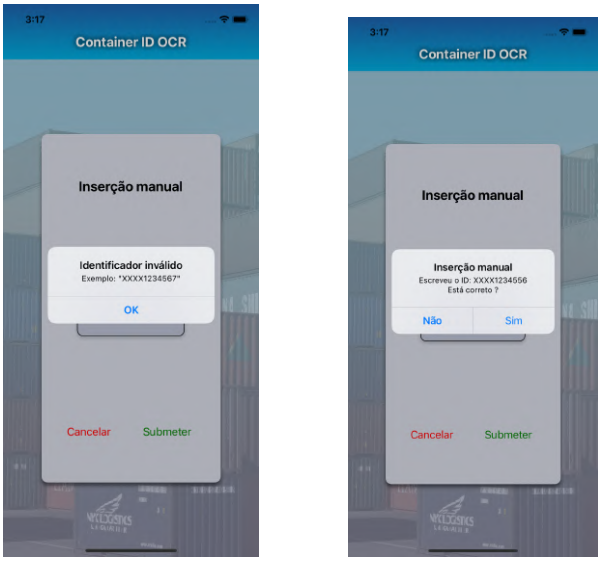
- **Adaptabilidade** – É possível verificar na figura seguinte que, mesmo em quatro dispositivos diferentes, a aparência da aplicação é bastante semelhante.
- **Envio do ID** – A aplicação é capaz de estabelecer comunicação com o Back-end, sendo capaz de enviar a imagem e de receber o ID.



(a) *Huawei* (b) *iPhone* (c) emulador *Android* (d) emualdor *IOS*

Figura 5.13: Apresentação do menu inicial em cada uma das opções utilizadas para testar a aplicação.

- **Prevenção de erros humanos** – Para certo tipo de ações, na aplicação, desenvolveram-se mecanismos como *pop-up's*, ou automatismos. Por exemplo, quando, na inserção manual, o utilizador coloca um caractere que não é nem letra nem número é automaticamente apagado.



(a) ID inválido (b) Confirmação

Figura 5.14: Dois exemplos de avisos com vista a evitar erros.

6

Conclusão

Através da pesquisa inicial, sobre a problema que esta dissertação pretende resolver, percebeu-se que existem algumas soluções que poderiam ser aplicadas, mas nenhuma delas seria acessível em termos monetários e logísticos para que fossem, efetivamente, aplicadas em portos de pequena dimensão, como o Porto da Horta. Portanto, nesta dissertação, pretendeu-se desenvolver não só uma solução eficaz como, também, acessível.

Foi desenvolvida uma arquitetura do sistema que visava a eficiência do mesmo, não utilizando a solução de desenvolver todo o sistema no dispositivo móvel, fazendo-se assim a separação da aplicação e do *Back-end*, deixando este para um servidor físico ou em *cloud* onde se teria mais capacidade de processamento e, por sua vez, mais rapidez de execução. Assim, foi possível não sobrecarregar o dispositivo com os métodos de processamento, métodos que, em dispositivos mais antigos ou com menos capacidade de processamento, seriam bastante demorados ou, em alguns casos, impossíveis.

Foi atingido pelo autor o principal objetivo, desenvolver um protótipo de uma aplicação móvel capaz de fazer a deteção e reconhecimento do ID de um contentor através de uma imagem ou fotografia, sendo apenas necessário, para o seu uso, um dispositivo móvel com acesso à internet e uma câmara.

Toda a aplicação foi desenvolvida com o propósito de uma utilização intuitiva por parte do operador, com um design simples e claro. Para chegar a esse objetivo foi desenvolvido e testado o próprio design para chegar à solução final. Já no *Back-end*, dado que é invisível para o utilizador, este foi desenvolvido e testado inúmeras vezes, com o objetivo de se chegar a um algoritmo equilibrado em rapidez de execução e taxa de acerto na localização e reconhecimento do ID.

Foram apresentados, no capítulo anterior, os testes realizados a todo o sistema, os componentes que o constituem e os resultados obtidos. Lá, é perfeitamente visível os bons resultados em termos de tempo, eficácia de reconhecimento e, ainda, usabilidade da aplicação. No entanto, podem, ainda, ser feitas algumas críticas ao sistema:

- Havendo ainda a existência de muitos dispositivos mais antigos, a qualidade das câmaras é mais baixa, sendo que isto traz como consequência a menor resolução da imagem e por sua vez a diminuição da taxa de localização e reconhecimento do ID.
- Tanto o modelo de localização como o de reconhecimento utilizaram *datasets*que em grande parte foram imagens capturadas no local, no entanto, algumas foram retiradas do *Google Images* com uma resolução inferior o que fez com que a taxa de acerto não fosse maior. Com um *dataset* maior e com imagens de maior resolução, a taxa iria aumentar melhorando os resultados do sistema.

É interessante, ainda, perceber que a solução resolve o problema desta dissertação e pode ser implementada no porto da Horta, podendo também ser implantada, não só nos outros portos nacionais com menor capacidade financeira para adquirir sistemas dispendiosos, mas também em empresas de logística de contentores ou em estações ferroviárias de mercadorias onde existe também a entrada de grandes quantidades destes contentores. Visto de um ponto mais geral, o sistema poderia ser adaptado para outro tipo de identificadores, como matrículas de automóveis.

Por fim, resta um sentimento sucesso no desenvolvimento da solução com os resultados esperados e, ainda, a confirmação de que todos os requisitos inicias foram cumpridos rigorosamente. A solução, sendo implementada no porto da Horta, deverá promover um avanço tecnológico do mesmo e, ainda, a simplificação e eficiência deste processo logístico.

Contudo, o sistema poderia ainda receber novas funcionalidades para tornar a utilidade do sistema cada vez maior, e consequentemente uma maior facilidade de utilização da aplicação por parte dos operários da Opertri. Uma das funcionalidades passaria pela integração do sistema desenvolvido com a Janela Única Portuária (JUP), onde a Opertri descarrega os documentos oficiais de carga e descarga de navios, nos quais referem quais contentores devem ser descarregados/carregados em cada porto por onde passa o navio. Esta funcionalidade consistiria no acesso aos documentos e extração da lista de ID's dos contentores a ser descarregados/carregados aquando da chegada do navio, para posteriormente aquando da leitura de um ID na aplicação, este poder ser comparado com aqueles presentes na lista e assim diminuir o tempo de reconhecimento e também o tempo de verificação do ID.

Outro aspeto a ser melhorado seria a criação de *datasets* mais extensos e treinar novamente os modelos com esses *datasets* e estudar os benefícios dos modelos treinados com estes novos *datasets* em termos de tempo de execução e de taxa de acerto comparado com os que foram utilizados no sistema. Caso esse estudo obtivesses resultados positivos fazer a alteração do mesmo com o objetivo de melhorar a eficiência e acerto dos reconhecimentos efetuados pela aplicação.

Bibliografia

[1] *Logistics industry to be worth $15.5tn by 2023*. consultado: 2021-11-05. Nov. de 2016. URL: https://www.cips.org/supply-management/news/2016/november/logistics-industry-forecast-to-be-worth-155tn-by-2023/.

[2] *SIM European Logistics Warehousing the future*. consultado: 2021-11-05. Dez. de 2017. URL: https://www.savillsim.com/research/sim-european-logistics-warehousing-the-future.aspx#.

[3] *BoxCatcher STS OCR: INCREASING CRANE PERFORMANCE AT MPET - Camco*. https://www.camco.be/3577-2/. consultado: 2021-11-05.

[4] *Opertri*. consultado: 2021-11-05. URL: http://opertri.pt.

[5] *Container Identification Number*. consultado: 2021-11-05. URL: https://www.bic-code.org/identification-number.

[6] J.-P. Rodrigue. "Transportation and the spatial structure". Em: *The Geography of Transport Systems* (2016), pp. 49–94. DOI: 10.4324/9781315618159-2.

[7] C. J. Bartodziej. *The concept industry 4.0: an empirical analysis of technologies and applications in production logistics*. Springer Gabler, 2017.

[8] Y. Sheffi e P. Klaus. "Logistics at Large: Jumping the Barriers of the Logistics Function". Em: (1997).

[9] M. Amr, M. Ezzat e S. Kassem. "Logistics 4.0: Definition and Historical Background". Em: *2019 Novel Intelligent and Leading Emerging Sciences Conference (NILES)* (2019). DOI: 10.1109/niles.2019.8909314.

[10] L. D. Galindo. "The Challenges of Logistics 4.0 for the Supply Chain Management and the Information Technology". Tese de mestrado. Norwegian University of Science e Technology, 2016.

[11] H. Kagermann, W. Wahlster e J. Helbig. *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0 – Securing the Future of German Manufacturing Industry*. Rel. téc. München: acatech – National Academy of Science e Engineering, 2013.

[12] H. Lasi et al. "Industry 4.0". Em: *Business & Information Systems Engineering* 6.4 (2014), pp. 239–242. DOI: 10.1007/s12599-014-0334-4.

[13] "Impact of the Fourth Industrial Revolution on Supply Chains". Em: *World Economic Forum Annual Meeting*. 2017.

[14] K. Wang. "Logistics 4.0 Solution-New Challenges and Opportunities". Em: *Proceedings of the 6th International Workshop of Advanced Manufacturing and Automation*. Atlantis Press, 2016, pp. 68–74. DOI: https://doi.org/10.2991/iwama-16.2016.13.

[15] L. Barreto, A. Amaral e T. Pereira. "Industry 4.0 implications in logistics: an overview". Em: *Procedia Manufacturing* 13 (dez. de 2017), pp. 1245–1252. DOI: 10.1016/j.promfg.2017.09.045.

[16] M. B. Harald v. Heynitz. "Part 2". Em: *The Factory of the Future - Industry 4.0 The challenges of tomorrow*. KPMG AG Wirtschaftsprüfungsgesellschaft, 2016.

[17] M. D.
bibinitperiod Company. Em: *Industry 4.0 How to navigate digitization of the manufacturing sector*. McKinsey Digital, 2015.

[18] J. Macaulay, L. Buckalew e G. Chung. Em: *INTERNET OF THINGS IN LOGISTICS - A collaborative report by DHL and Cisco on implications and use cases for the logistics industry*. DHL Customer Solutions & Innovation, 2015.

[19] S. Schrauf e P. Berttram. Em: *Industry 4.0 - How digitization makes the supply chain more efficient, agile, and customer-focused*. PwC Strategy&, 2016.

[20] J. Bunch et al. "Review of Existing Literature and Deployment Tracking Surveys : Decision Factors Influencing ITS Adoption". Em: 2012.

[21] E. Griffor et al. "Framework for Cyber-Physical Systems: Volume 1, Overview". Em: (jun. de 2017). DOI: 10.6028/NIST.SP.1500-201.

[22] E. A. Lee. "Cyber Physical Systems: Design Challenges". Em: *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. 2008, pp. 363–369. DOI: 10.1109/ISORC.2008.25.

[23] E. Geisberger e M. Broy. "Living in a networked world. Integrated research agenda Cyber-Physical Systems". Em: (mar. de 2015).

[24] P. Derler, E. A. Lee e A. Sangiovanni Vincentelli. "Modeling Cyber–Physical Systems". Em: *Proceedings of the IEEE* 100.1 (2012), pp. 13–28. DOI: 10.1109/JPROC.2011.2160929.

[25] F. Hu. *Cyber-physical systems: integrated computing and engineering design*. CRC Press, 2014.

[26] A. A. Aly et al. "An antilock-braking systems (ABS) control: A technical review". Em: *Intelligent control and Automation* 2.03 (2011), p. 186.

[27] M. G. Marne et al. "Identification of Optimal Optical Character Recognition (OCR) Engine for Proposed System". Em: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, pp. 1–4. DOI: 10.1109/ICCUBEA.2018.8697487.

[28] L. Eikvil. "Optical character recognition". Em: *citeseer.ist.psu.edu* 26 (1993).

[29] F. Mohammad et al. "Optical character recognition implementation using pattern matching". Em: *International Journal of Computer Science and Information Technologies* 5.2 (2014), pp. 2088–2090.

[30] C. A. Rahman, W. Badawy e A. Radmanesh. "A real time vehicle's license plate recognition system". Em: *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.* 2003, pp. 163–166. DOI: 10.1109/AVSS.2003.1217917.

[31] Min-Ki Kim e Young-Bin Kwon. "Multi-font and multi-size character recognition based on the sampling and quantization of an unwrapped contour". Em: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 3. 1996, 170–174 vol.3. DOI: 10.1109/ICPR.1996.546816.

[32] S. Wang e H. Lee. "A Cascade Framework for a Real-Time Statistical Plate Recognition System". Em: *IEEE Transactions on Information Forensics and Security* 2.2 (2007), pp. 267–282. DOI: 10.1109/TIFS.2007.897251.

[33] R. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[34] W. Liu et al. "Ssd: Single shot multibox detector". Em: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[35] B. Shi, X. Bai e C. Yao. "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition". Em: *IEEE transactions on pattern analysis and machine intelligence* 39.11 (2016), pp. 2298–2304.

[36] C. Bartz, H. Yang e C. Meinel. "STN-OCR: A single neural network for text detection and text recognition". Em: *arXiv preprint arXiv:1707.08831* (2017).

[37] C. Bartz, H. Yang e C. Meinel. "SEE: towards semi-supervised end-to-end scene text recognition". Em: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[38] J. Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[39] T. Tao et al. "Object Detection-Based License Plate Localization and Recognition in Complex Environments". Em: *Transportation Research Record* 2674.12 (2020), pp. 212–223.

[40] I. of Electrical e E. Engineers. "IEEE Standard Glossary of Software Engineering Terminology". Em: *IEEE Std 610.12-1990* 1.1 (1990), pp. 1–84.

[41] Meta. *Facebook*. consultado: 2021-11-05. URL: https://www.facebook.com/.

[42] Apple. *iOS*. consultado: 2021-11-05. URL: https://developer.apple.com/ios/.

[43] Android. *What is Android*. consultado: 2021-11-05. URL: https://www.android.com/what-is-android/.

[44] Expo. consultado: 2021-11-05. URL: https://expo.dev/.

[45] Microsoft. *Visual Studio Code - Code Editing. Redefined*. Abr. de 2016. URL: https://code.visualstudio.com/.

[46] G. Van Rossum e F. L. Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[47] X. Zhou et al. *EAST: An Efficient and Accurate Scene Text Detector*. 2017. arXiv: 1704.03155 [cs.CV].

[48] G. Bradski. "The OpenCV Library". Em: *Dr. Dobb's Journal of Software Tools* (2000).

[49] Google. consultado: 2021-11-05. 2021. URL: https://www.google.com/imghp?hl=en.

[50] T. LIN. *labelImg: LabelImg is a graphical image annotation tool and label object bounding boxes in images*. consultado: 2021-11-05. 2021. URL: https://github.com/tzutalin/labelImg.

[51] A. Bochkovskiy, C.-Y. Wang e H.-Y. M. Liao. "Yolov4: Optimal speed and accuracy of object detection". Em: *arXiv preprint arXiv:2004.10934* (2020).

[52] Google. *Google Colaboratory*. consultado: 2021-11-05. 2021. URL: https://colab.research.google.com.

[53] J. Redmon. *Darknet: Open Source Neural Networks in C*. consultado: 2021-11-05. 2021. URL: https://pjreddie.com/darknet/.

[54] Tesseract-Ocr. *tesseract-ocr/tesseract: Tesseract Open Source OCR Engine*. consultado: 2021-11-05. URL: https://github.com/tesseract-ocr/tesseract.

[55] Pytesseract. *pytesseract*. consultado: 2021-11-05. URL: https://pypi.org/project/pytesseract/.

[56] Flask. *Welcome to Flask*. consultado: 2021-11-05. URL: http://flask.palletsprojects.com/.

[57] Werkzeug. *Werkzeug*. consultado: 2021-11-05. URL: https://werkzeug.palletsprojects.com/en/2.0.x/.

[58] J. Brownlee. *Loss and Loss Functions for Training Deep Learning Neural Networks*. consultado: 2021-11-05. Out. de 2019. URL: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.

[59] S. Yohanandan. *mAP (mean Average Precision) might confuse you!* consultado: 2021-11-05. Jun. de 2020. URL: https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2.

[60] J. Brownlee. *Impact of Dataset Size on Deep Learning Model Skill And Performance Estimates*. Ago. de 2020. URL: https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/.

[61] Android. *Download Android Studio and SDK tools : Android Developers*. consultado: 2021-11-05. URL: https://developer.android.com/studio.

[62] Apple. *Xcode*. consultado: 2021-11-05. URL: https://developer.apple.com/xcode/.