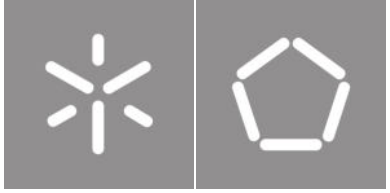


Universidade do Minho
Escola de Engenharia

Renata Araújo Fernandes Martins

Deteção e Reconhecimento de utilizadores específicos em tarefas de longa duração

Outubro de 2023



Universidade do Minho
Escola de Engenharia

Renata Araújo Fernandes Martins

Deteção e Reconhecimento de utilizadores específicos em tarefas de longa duração

Dissertação de Mestrado
Mestrado em Engenharia Eletrónica Industrial e Computadores
⟨Controlo, Automação e Robótica⟩
Trabalho efetuado sob a orientação de:
Prof. Doutor Fernando Ribeiro

Outubro de 2023

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositoriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Creative Commons Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional
CC BY-NC-SA 4.0

https://creativecommons.org/licenses/by-nc-sa/4.0/deed.pt

Agradecimentos

O término e a entrega desta dissertação representa o fim de um ciclo. Foram 5 anos intensos e desafiantes que não conseguiria ter superado sem a ajuda de diversas pessoas que de formas diferentes me ajudaram e apoiaram incondicionalmente e que não poderia deixar de agradecer.

Por todo o amor, apoio e esforços que fizeram por mim ao longo de toda a minha vida, agradeço aos meus pais, Armanda Fernandes e Artur Martins.

Ao meu orientador Fernando Ribeiro e ao professor Gil Lopes, um enorme obrigada, pela oportunidade e por terem confiado em mim. Por toda a ajuda, apoio e por me terem permitido desenvolver o meu trabalho no Laboratório de Automação e Robótica. Com isto em consideração, agradeço também a todos os meus colegas e amigos LARianos por toda a paciência que sempre tiveram para comigo e acima de tudo por todo o apoio e amizade.

Às minhas amigas do coração Lages, Limões, Bia, Kaka e Isa, por terem estado sempre comigo e terem sido para mim um pilar, obrigada! Por todas as vezes que me receberam com um abraço, que ouviram todas as minhas lamentações transformadas em áudios de 5 minutos, por todos os festejos, obrigada!

A todos os meus amigos que durante este longo ciclo estiveram a meu lado, Duarte, Mendes, Bruno, Pedro, Barros, Diogo, Leandro, e às minhas amigas e colegas de equipa Rafa, Capela, Barros, Vieira, Bibs, Gabi, Cátia e Nanda, obrigada!

Um agradecimento especial ao Tiago Ribeiro, a pessoa que todos os dias lidou com uma Renata diferente e que sempre me ajudou, orientou e aconselhou. Por ter confiado em mim de uma forma que nem eu confiava e por me ter ajudado a evoluir enquanto engenheira e enquanto pessoa. Obrigada Tiago por não teres exigido menos de mim do que aquilo que eu poderia dar.

Um agradecimento especial do fundo do meu coração e com todo o amor ao Rodrigo, a pessoa que compartilha comigo todos os momentos, que nunca desiste e que é o meu exemplo e força todos os dias. Por todo o amor, compreensão e paciência quando dizia que ia trabalhar à noite, por todos os dias que desculpou os meus atrasos. Acima de tudo por ter estado em todos os momentos bons mas principalmente nos mais difíceis.

Por último, aos meus avós, Emília e Manel que sem eles nada disto teria acontecido. Por terem sempre colocado as vossas expectativas e sonhos em mim! Por me terem ajudado a conquistar tudo o que conquistei e sempre terem estado a meu lado para me dar colo e chocolates. Nunca irão existir palavras suficientes para vos agradecer e espero que consigam ver onde cheguei.

Resumo

Deteção e Reconhecimento de utilizadores específicos em tarefas de longa duração

Com os avanços existentes na área da robótica, os robôs estão cada vez mais presentes no dia-a-dia do ser humano. Da mesma forma que existem pessoas a trabalhar nas fábricas, nos hospitais e em lares, os robôs também já começam a fazer parte destes ambientes. Em consequência disto, a necessidade de interação Homem-Robô está a aumentar e a tornar-se mais complexa. O robô CHARMIE (Collaborative Home / Healthcare Assistant Robot da Minho Industrial Electronics) é um robô antropomórfico que realiza tarefas em configurações não convencionais usando algoritmos de *Machine Learning*, que toma decisões com base no meio, condições e pessoas que o rodeiam. Este robô tem uma componente social e competitiva pelo que esta dissertação está dividida em duas abordagens.

Na base do problema de deteção e reconhecimento facial estão três módulos nomeadamente, deteção facial onde é utilizada uma solução com *MediaPipe* que procura faces prevendo as coordenadas, que permitem isolar as caras detetadas ao detetar todos os rostos exibidos ao usar uma câmara. Para o segundo módulo é usada uma rede neuronal Inception-ResNet que foi treinada para um dado *dataset* e funciona como extrator de características faciais. Isso gera os vetores característicos de pessoas que serão usados como entrada para um classificador com *Support Vector Machine*, um modelo de *Machine Learning* que irá prever a qual das identidades esses vetores correspondem, realizando a tarefa de reconhecimento facial, ou seja o terceiro módulo.

Foi também desenvolvido um extrator de características humanas, como idade, género e etnia, de forma a ser possível saber as características do utilizador e adaptar o comportamento do robô de acordo com as mesmas. De forma extra e para garantir algum tipo de segurança, foi desenvolvido um módulo onde o objetivo passa por detetar através de imagens de profundidade validando se na imagem está uma pessoa ou se alguém a utilizar uma fotografia de forma a tentar enganar o sistema.

A par disto, em contexto de equipa com o objetivo de competir na maior competição de robótica do mundo, RoboCup@Home foi também desenvolvida através de ROS2 uma solução para o robô CHARMIE. Este robô tem de ser capaz de navegar, falar e ouvir, detetar faces e pessoas, para conseguir relacionar-se com as mesmas. Foram para isso desenvolvidas as tarefas onde o CHARMIE participa, de forma a que o robô tivesse sucesso e conseguisse o máximo de pontos possíveis na respetiva prova.

Palavras-chave: Deep Learning, Machine Learning, Deteção Facial, Reconhecimento Facial, Extração de Características, Robôs de Serviços

Abstract

Detection and Recognition of specific human users in long-running tasks

With advances in robotics, robots are increasingly present in people's daily lives. Just as there are people working in factories, hospitals and homes, robots are also becoming part of these environments. As a result, the need for human-robot interaction is increasing and becoming more complex. The CHARMIE robot (Collaborative Home / Healthcare Assistant Robot from Minho Industrial Electronics) is an anthropomorphic robot that performs tasks in unconventional settings using *Machine Learning* algorithms, and able the robot to make decisions based on the environment, conditions and people around it. This robot has a social and competitive component, which is why this dissertation is divided into two approaches.

There are three modules at the base of the face detection and recognition problem, namely: a) Face detection, where a solution is used that searches for faces predicting the coordinates that allow the detected faces to be isolated, and detecting all the faces displayed when using a camera. b) The Inception-ResNet neural network, which has been trained for a given *dataset*, acts as an extractor of facial features. c) The second module generates the characteristic vectors of people that will be used as input for an Support Vector Machine classifier, a *Machine Learning* model that will predict which of the identities these vectors correspond to, performing the facial recognition task.

In addition, an extractor of human characteristics was also developed, such as age, gender and ethnicity, so that it would be possible to know the user's characteristics and adapt the robot's behavior accordingly. As an extra measure and to guarantee some kind of security, a module was developed where the aim was to detect, through depth images, whether or not the images were actually people.

In addition, in a team context with the aim of competing in the world's largest robotics competition, RoboCup@Home, a solution for the CHARMIE robot was also developed using ROS2. In this context, the robot needs to be able to navigate, speak and listen, detect faces and people, and be able to relate to them. The tasks in which CHARMIE participates were developed so that the robot succeeds and achieves as many points as possible.

Keywords: Deep Learning, Machine Learning, Face Detection, Face Recognition, Feature Extraction, Service Robots

Índice

Índice de Figuras xii

Índice de Tabelas xvi

Índice de Listagens xvii

Siglas xviii

1 Introdução 1

1.1 Motivação . . . 3
1.2 Objetivos . . . 3
1.3 Contribuições Científicas . . . 5
1.4 Estrutura do Documento . . . 6

2 Revisão Literária 8

2.1 *Machine Learning* . . . 8
2.1.1 Aprendizagem supervisionada . . . 9
2.1.2 *Random Forest* . . . 10
2.1.3 Aprendizagem não-supervisionada . . . 11
2.1.4 Aprendizagem de reforço . . . 11
2.1.5 Avaliação de modelos de *Machine Learning* . . . 12
2.1.6 *Support Vector Machine* - SVM . . . 13
2.2 Deep Learning . . . 14
2.2.1 Redes Neuronais Artificiais ANNs . . . 15
2.2.2 Redes Neuronais Convolucionais CNNs . . . 16
2.3 Arquiteturas das CNN . . . 20
2.3.1 LeNet-5 . . . 20
2.3.2 AlexNet . . . 20
2.3.3 VGG-Net . . . 21
2.3.4 Google Net ou Inception V1 . . . 22
2.3.5 ResNet . . . 23

2.3.6 Inception ResNet V1 e V2 . . . 25
2.4 ROS/ROS2 . . . 26
2.4.1 Tipos de Comunicação no ROS . . . 26
2.5 Deteção e Reconhecimento Facial . . . 28
2.5.1 Deteção facial . . . 28
2.5.2 Imagens de Profundidade na Deteção Facial . . . 33
2.5.3 Reconhecimento facial . . . 34

3 Métodos e Metodologias **37**
3.1 Especificações do Sistema e Ferramentas . . . 37
3.1.1 *Hardware* . . . 39
3.1.2 Desenvolvimento de Peças em 3D para o robô CHARMIE . . . 41
3.2 Abordagem com Redes Neuronais para Ambiente Social . . . 45
3.2.1 Deteção Facial . . . 47
3.2.2 Imagens de profundidade . . . 48
3.2.3 Criação do *Dataset* . . . 49
3.2.4 Inception ResNet . . . 53
3.2.5 Classificação . . . 55
3.2.6 Visualização dos Dados . . . 56
3.3 Abordagem usada no RoboCup@Home . . . 57
3.3.1 RoboCup . . . 58
3.3.2 Desenvolvimento da Prova Rececionista . . . 61
3.3.3 Desenvolvimento da Prova *Stickler for the Rules* . . . 66
3.4 Reconhecimento de outras características . . . 67
3.4.1 Género . . . 68
3.4.2 Idade . . . 69
3.4.3 Etnia . . . 69
3.4.4 Altura . . . 70

4 Resultados **71**
4.1 Abordagem com Redes Neuronais para Ambiente Social . . . 71
4.1.1 Deteção Facial . . . 71
4.1.2 Imagens de profundidade . . . 75
4.1.3 Modelos Inception Resnet . . . 79
4.1.4 Classificador SVM . . . 81
4.1.5 Visualização e análise de *embeddings* de faces utilizando PCA e t-SNE . . . 86
4.1.6 Aplicação no robô CHARMIE com recurso a ROS2 . . . 88
4.2 Abordagem usada no RoboCup@Home . . . 91

4.2.1 Prova Rececionista e *Stickler for the Rules* . . . 91
4.2.2 Módulos relativos à prova do Rececionista . . . 97
4.2.3 Módulos relativos à prova *Stickler for the Rules* . . . 99
4.3 Extração de Caraterísticas . . . 99

5 Conclusões **103**
5.1 Conclusões . . . 103
5.2 Trabalho Futuro . . . 105

Bibliografia **106**

Anexos

I Anexos **110**
I.1 Arquitetura Inception ResNet V1 . . . 110
I.2 Arquitetura Inception ResNet V2 . . . 111

II Anexos **112**
II.1 Fluxogramas da Abordagem Competitiva . . . 112
II.1.1 Fluxograma da prova Rececionista . . . 113
II.1.2 Fluxograma do trabalho desenvolvido na prova Rececionista (parte 1 e 2 . . . 114
II.1.3 Fluxograma da prova *Stickler for the Rules* . . . 116
II.1.4 Fluxograma da trabalho desenvolvido na prova *Stickler for the Rules* . . . 117

III Anexos **118**
III.1 Desenhos peças 3D . . . 118
III.1.1 Caixa para Botão de Emergência . . . 118
III.1.2 Suporte para o Comando de Emergência . . . 119
III.1.3 Calhas Laterais e Tampa . . . 120
III.1.4 Suporte para o sensor TOF inferior . . . 121
III.1.5 Suporte para o sensor TOF superior . . . 122
III.1.6 Suporte para a bússola CMPS inferior . . . 123
III.1.7 Suporte para a bússola CMPS superior . . . 124
III.1.8 Proteção Traseira . . . 125
III.1.9 Proteção Frontal . . . 126
III.1.10 Proteção dos Fios das Baterias . . . 127
III.1.11 Ombros e Suporte do Microfone . . . 128

Índice de Figuras

1 Robô Collaborative Home Assistant Robot by Minho Industrial Electronics (CHARMIE), versão Junho de 2023 . . . 6

2 Relação entre Inteligência Artificial (IA), *Machine Learning (ML)* e *Deep Learning (DL)* . . 8

3 Categorias de *ML* . . . 9

4 Representação da técnica *bagging* . . . 11

5 Representação de uma Support Vector Machine (SVM) . . . 13

6 Neurónio [16] . . . 14

7 Rede Neuronal Profunda Deep Neural Network (DNN) . . . 16

8 Arquitetura CNN . . . 17

9 Exemplo do que acontece na camada convolucional . . . 18

10 Exemplos de agrupamento máximo e médio respetivamente . . . 19

11 Configuração da VGG-16 e VGG-19 . . . 22

12 Esquemático de Inception-V1 com redução de dimensão . . . 23

13 Esquemático de Inception-V2 . . . 23

14 Esquemático de Inception-V3 . . . 24

15 Bloco residual de um modelo de ResNet . . . 24

16 Comunicações em Robot Operating System (ROS)2 . . . 27

17 Pirâmide de imagens . . . 30

18 Fase P-Net . . . 30

19 Fase R-Net . . . 31

20 Interseção e união entre duas caixas delimitadoras . . . 31

21 IoU - Comparação de performance . . . 32

22 Fase O-Net . . . 32

23 Protótipo do robô CHARMIE utilizado no RoboCup@Home de 2023: a) Vista frontal b) Vista traseira . . . 40

24 Posição em altura mínima possível . . . 40

25 Câmara Intel® RealSense Depth™ D455 . . . 41

26 Caixa para o botão de emergência . . . 42

27 Suporte para o comando no robô CHARMIE 42
28 Calha e tampa no robô CHARMIE 43
29 Suportes das bússolas CMPS 12: a) Atuador linear para subida/descida do tronco, b) Atuador linear para inclinação do tronco 44
30 Proteção traseira no robô CHARMIE 44
31 Proteção dos cabos que ligam as baterias ao robô 45
32 Ombro e suporte para microfone 45
33 Fluxograma da abordagem social 46
34 Imagens para a) treino (dimensão original 160x160 px) e b) teste (dimensão original 1280x720 px) 49
35 Diretórios relativos à criação do *dataset* 50
36 Exemplo de um diretório 50
37 Exemplo de imagens de profundidade para treino 51
38 Transformações de rotação a) -16.33º b) -7.58º c) 2.33º d) 4.56º 52
39 Transformações de rotação a) 0.4 b) 1.1 c) 1.8 d) 2.4 53
40 Transformação horizontal a) original b) flip 53
41 Mapa da casa com indicação da área da prova do Rececionista [47] 59
42 Descrição dos pontos referentes à tarefa Rececionista [48] 60
43 Mapa para a prova *Stickler for the rules* [47] 60
44 Descrição dos pontos referentes à tarefa *Stickler for the Rules* [48] 61
45 Resultados obtidos para a imagem [49] a) MediaPipe, b)Multi-task Cascaded Convolutional Neural Network (MTCNN) 72
46 Resultados obtidos para a imagem [50] a)MediaPipe, b) MTCNN 72
47 Resultado após *resize* da imagem [50] 73
48 Comparação do *streams* com a câmara do computador a) MediaPipe, b) MTCNN 73
49 Comparação do *streams* com câmara Intel *RealSense Depth* D455 a) MediaPipe, b) MTCNN 74
50 Resultados obtidos com MediaPipe (50(a) e 50(b)) e MTCNN (50(c) e 50(d)) para diferentes condições de iluminação 75
51 Resultados obtidos com *MediaPipe* (a), b) c) e d)) e MTCNN (e), f), g) h)) para diferentes poses 76
52 Output na ausência de faces 77
53 Matrizes de confusão para dados de teste em imagens de profundidade 77
54 *Output* com pessoa real e tentativa de *spoofing* 78
55 *Output* do modelo em treino 79
56 Treino de um modelo com *overfit* 79
57 Treino de um modelo instável 80
58 *Output* do classificador SVM 81

59 Ficheiros *pickles* . . . 81
60 Resultados com o modelo FaceNet para a) um utilizador e b) dois utilizadores . . . 83
61 Resultados com o modelo 2 da arquitetura *Inception Resnet* V1 para a) um utilizador e b) dois utilizadores . . . 84
62 Resultado com o modelo 2 com um conhecido e um desconhecido . . . 84
63 Resultado com o modelo 2 com dois conhecidos e um desconhecido . . . 85
64 Matrizes de confusão dos melhores modelos utilizados a) *FaceNet* e b) modelo 2 *Inception ResNet* V1 . . . 85
65 Análise do PCA com o modelo a) *FaceNet* e b) *Inception ResNet* V1 . . . 87
66 Análise do t-SNE com o modelo a) *FaceNet* e b) *Inception ResNet* V1 . . . 88
67 Resultado obtido com o modelo 2 com pessoa presente . . . 88
68 *Output* do módulo *Speaker* . . . 88
69 Resultado com o modelo 2 com duas pessoas conhecidas . . . 89
70 *Output* do módulo *Speaker* . . . 89
71 Resultado com o modelo 2 com um conhecido e um desconhecido . . . 89
72 *Output* do módulo *Speaker* . . . 89
73 Resultado com o modelo 2 com três pessoas conhecidas . . . 90
74 *Output* do módulo *Speaker* . . . 90
75 *Output* quando a "Renata Martins"é detetada . . . 90
76 Comparação da *accuracy* quando os utilizadores estão diferentes comparativamente ao dataset . . . 90
77 Informação dada pelo convidado 1, nome e bebida favorita . . . 92
78 Informação dada pelo convidado 2, nome e bebida favorita . . . 92
79 Informação retida pelo robô e transmitida na apresentação dos elementos presentes na sala 93
80 Imagem referente ao inicio da prova do Rececionista . . . 94
81 Imagem referente ao fim da prova do Rececionista . . . 94
82 Imagem referente à navegação do robô onde o pescoço orienta-se para baixo . . . 95
83 Deteção de pessoa em espaço proibido: a) dentro ou b) fora . . . 95
84 Sinal luminoso referente ao inicio da prova associado a premir o *start button* . . . 96
85 Informação transmitida aos convidados para saberem quando responder . . . 96
86 Imagem inicial com face não visível . . . 97
87 Imagem após o ajuste do pescoço . . . 97
88 Imagem inicial com face está visível mas não centrada . . . 98
89 Imagem após o ajuste do pescoço . . . 98
90 Imagem capturada do convidado 1 . . . 98
91 *Output* da posição das pessoas detetadas na casa . . . 99
92 *Output* da posição das pessoas detetadas, a) fora da casa e b) dentro da casa . . . 100
93 Informação do género, idade, etnia, altura e cor dos olhos do convidado 1 . . . 100

94 Imagem com as faces detetadas 101
95 Resultados relativos à etnia 101
96 Resultados relativos ao género 101
97 Resultados relativos à idade 102

Índice de Tabelas

1 Módulos utilizados e respetivas versões 38

2 Comparação dos tempos na utilização das bibliotecas *MediaPipe* e MTCNN 73
3 Comparação das soluções *random forest* e Convolutional Neural Network (CNN) 77
4 *Accuracy* obtida nos 4 modelos (100 imagens por classe) 82
5 *Accuracy* obtida nos 4 modelos (50 imagens por classe) 82
6 *Accuracy* obtida nos 4 modelos (10 imagens por classe) 82
7 *Accuracy* obtida nos 4 modelos (6 imagens por classe) 82
8 Tempo médio para treino do classificador SVM 83
9 Comparação da *accuracy* obtida entre o utilizador nas mesmas/diferentes condições do *dataset* 91

Índice de Listagens

3.1 Definição da função de deteção facial com MediaPipe 47
3.2 Definição da função de deteção facial com MTCNN 47
3.3 Extração de características Histogram of Oriented Gradients (HOG) e criação e treino do classificador 48
3.4 Treino do modelo 48
3.5 Código relativo à transformação de imagens 52
3.6 Método de *EarlyStopping* e registo dos modelos 55
3.7 Implementação da SVM 56
3.8 Gráficos para visualização dos resultados 56
3.9 Visualização de resultados (PCA) 57
3.10 Visualização de resultados (t-SNE) 57
3.11 Excerto da função *init* com as declarações de *publishers* e *subscribers* 61
3.12 Exemplo de código para ajuste do ruído previamente à ação de ouvir 62
3.13 Extrato do dicionário relativo a alguns dos nomes 62
3.14 Registo do nome e bebida 63
3.15 Utilização do tópico dos *speakers* 63
3.16 Definição de coordenada 64
3.17 Chamada da função de navegação 64
3.18 Função de navegação do robô para coordenada específica 64
3.19 Procura de landmarks numa face 65
3.20 Recolha da imagem 65
3.21 Definição das coordenadas das várias divisões 67
3.22 Previsão do género 68
3.23 Previsão da idade 69
3.24 Previsão da etnia 69

Siglas

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CHARMIE	Collaborative Home Assistant Robot by Minho Industrial Electronics
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
FN	Falso Negativo
FP	Falso Positivo
fps	frames por segundo
HOG	Histogram of Oriented Gradients
IA	Inteligência Artificial
IoU	Intersection over Union
ML	Machine Learning
MTCNN	Multi-task Cascaded Convolutional Neural Network
NMS	Non maximum suppression
PCA	Principal Component Analysis
RGB	Red Green Blue
ROS	Robot Operating System

SGD Stochastic Gradient Descent
SVM Support Vector Machine

t-SNE t-distributed Stochastic Neighbor Embedding

VN Verdadeiro Negativo
VP Verdadeiro Positivo

Capítulo 1

Introdução

A visão é um dos sentidos mais evoluídos e importantes no ser humano. Graças a ele podemos ver a existência de estímulos positivos ou negativos à nossa volta com alto nível de precisão, especialmente à luz do dia o que permite ajudar o ser humano a tomar decisões, em tempo real ou não.

O sentido da visão não é um processo tão simples quanto parece, não chega apenas capturar a imagem é também necessário interpretar os parâmetros, como a distância, a forma, a cor e até mesmo o movimento. Ao nível do cérebro, estes processos precisam de um processamento que ocorre em diferentes regiões do mesmo. Neste sentido, destaca-se o papel do córtex visual do cérebro, sendo capaz de receber, compreender e processar informação captada pela retina.

O reconhecimento facial é uma categoria da biometria, sendo a biometria uma tecnologia utilizada como meio de comparação e verificação de uma característica humana com informação armazenada anteriormente tendo como objetivo principal permitir acesso à identificação de um individuo.

Atualmente, a robótica está cada vez mais integrada na sociedade humana e o seu papel tem sido cada vez mais relevante e fundamental no avanço da sociedade. Os robôs têm desempenhado tarefas mais complexas e tornaram-se capazes de agir e tomar decisões nos mais variados contextos. Nos últimos anos, os robôs têm sido integrados nas mais variadas áreas, seja na indústria, no auxílio e na prestação de serviços a humanos em instalações domésticas e de saúde, na investigação/exploração do espaço, etc. Algumas tarefas realizadas por estes robôs exigem que estes sejam autónomos e capazes de tomar decisões, sendo capazes de analisar o ambiente externo e atuar de acordo com este, caso contrário podem não conseguir atingir o seu objetivo e até causar danos no ambiente onde estão inseridos.

O objetivo desta dissertação passa por replicar partes do sistema visual do ser humano e permitir a uma máquina identificar e processar objetos em imagens e vídeos da mesma forma que os humanos.

Para prestar um serviço ou assistência personalizada é de extrema importância que o robô de serviços consiga detetar e reconhecer utilizadores aos quais já tenha sido apresentado. É então necessário abordar

temas como visão por computador e visão robótica. Visão por computador tem como principal objetivo compreender o conteúdo de vídeos e imagens estáticas e, a partir disso, obter informações úteis para solucionar problemas. Já visão robótica refere-se à capacidade de um robô compreender o ambiente em que se encontra e utilizar essa informação para realizar tarefas através de algoritmos de visão, usando IA ou programação clássica de acordo com D. Kragic e M. Vincze [1].

Os casos de uso da visão por computador incluem reconhecimento de imagens, classificação de imagens e vídeos. Abaixo encontram-se algumas utilizações mais populares da visão por computador:

- Na medicina - com o processamento de imagens médicas com a extração rápida de dados vitais de imagens para ajudar a diagnosticar adequadamente determinados problemas num paciente, incluindo a deteção rápida de tumores e do endurecimento das artérias. Embora a visão por computador não forneça diagnósticos, esta é uma parte inestimável das técnicas modernas de diagnóstico médico, reforçando o que os médicos diagnosticam e, cada vez mais, fornecendo informações que não teriam sido obtidas de outra forma.

- Veículos autónomos - podem ser totalmente baseados em soluções de visão por computador ou ter as suas operações muito aprimoradas. As aplicações mais comuns e que já estão em funcionamento incluem sistemas maioritariamente de aviso antecipado nos carros.

- Indústria - a manufatura usa soluções de visão por computador para auxiliar nos seus processos tais como controlo de qualidade em que os sistemas de visão por computador verificam peças e produtos finais na procura de defeitos.

As técnicas tradicionais de visão por computador e processamento de imagens são usadas há décadas em inúmeras aplicações e trabalhos de investigação. No entanto, com o avanço das técnicas modernas de IA, o *DL* e as redes neuronais artificias permitem uma maior precisão e avanços na computação de alto desempenho, o que possibilita uma precisão sobre-humana.

ML é um subconjunto de IA que se foca no estudo de algoritmos que melhoram autonomamente através de dados. Esses algoritmos são usados em diversas aplicações, da visão por computador às otimizações matemáticas. Os algoritmos de ML são necessários quando é difícil ou inviável desenvolver algoritmos convencionais para executar uma tarefa necessária. O *DL* é um subconjunto de *ML* que utiliza um nível hierárquico de redes neuronais artificiais, com múltiplas camadas ocultas, para realizar o processo de *ML*. As redes neuronais artificiais são construídas como o cérebro humano, com "nós" de neurónios conectados entre si, formando algo semelhante a uma teia. Enquanto os programas tradicionais criam análises com dados de maneira linear, a função hierárquica dos sistemas de *DL* permite que as máquinas processem dados com uma abordagem não linear.

Esta dissertação propõe abordar o problema de ser capaz de, através de uma imagem, extrair informação válida para que um robô possa interagir com o ser humano. Nesse sentido, o trabalho irá focar o desenvolvimento de um sistema de perceção, baseado em técnicas de *DL*, que permitam identificar um utilizador e ainda detetar características como a sua idade, género, altura e etnia.

1.1 Motivação

Este projeto de dissertação está inserido no projeto *CHARMIE* [2], um robô antropomórfico colaborativo de cuidados de saúde e assistente doméstico capaz de executar tarefas de serviços genéricas em ambientes não padronizados de cuidados de saúde e ambiente doméstico. O objetivo social do projeto CHARMIE é ajudar em lares familiares, lares de idosos, e instalações de cuidados de saúde. O objetivo científico do CHARMIE é ser capaz de executar um vasto conjunto de tarefas baseado unicamente em soluções de *ML* que permitam ao robô aprender a executar e melhorar tarefas através da observação e de interações de ensaio e erro com o ambiente. Com isto, os seres humanos ficam livres de realizar tarefas monótonas e repetitivas, sendo desta forma possível focarem-se em tarefas onde a sua presença é realmente imprescindível. Para além disto, o projeto CHARMIE possui uma vertente competitiva no RoboCup@Home (Campeonato do Mundo de Robótica para robôs de serviços), [3].

Deste modo, a principal motivação consiste na exploração de técnicas de *DL* aplicadas ao reconhecimento facial e identificação de características humanas. O objetivo passa por integrar robôs em ambientes hospitalares, em casas e lares, realizar tarefas tais como receber e dar indicações a pessoas ou até realizar tarefas mais duras e indesejáveis. A utilização dos robôs para auxilio e ajuda do homem poderá estar a evoluir para um novo patamar por os robôs poderão estar no futuro a executar tarefas mais delicadas e mais complexas.

Tudo isto é possível graças às melhorias e evoluções tecnológicas ao nível de câmaras, sensores, e ferramentas de controlo robótico, avanços na IA e ainda o desenvolvimento no que toca a técnicas de *DL*. Os sistemas de reconhecimento facial captam uma imagem recebida de um dispositivo com câmara de uma forma bi ou tridimensional, dependendo das características do dispositivo. Estes comparam a imagem recebida com as imagens presentes na base de dados. Devido à facilidade, praticidade e segurança, a biometria facial deixou de ser ficção e passou a integrar diversas aplicações com diferentes finalidades. Esta tecnologia, para além das aplicações anteriormente mencionadas, tem sido aplicada no *check-in* nos aeroportos e em sistemas de vídeos em comércios, tornando-se uma tecnologia fundamental na área de segurança, para detetar pessoas suspeitas e procuradas. Para além disto, é também utilizada como fator de autenticação, para adicionar segurança extra em qualquer processo de *login*.

1.2 Objetivos

Os objetivos desta dissertação dividem-se em duas abordagens distintas. A primeira é uma abordagem social onde se pretende desenvolver um sistema de deteção e reconhecimento de utilizadores humanos específicos em tarefas de longa duração para ser implementado no CHARMIE. Ou seja, pretende-se que o CHARMIE seja capaz de reconhecer pessoas previamente definidas e também aprender novas identidades para ambientes sociais e/ou domésticos e a segunda abordagem é uma abordagem competitiva, onde o objetivo passa por desenvolver um sistema de deteção e reconhecimento de pessoas para

a realização de tarefas na competição RoboCup@Home. Embora os objetivos sejam semelhantes, o seu desenvolvimento é bastante diferente. Num contexto competitivo o tempo das provas é limitado e as condições são variáveis pelo que é necessário desenvolver uma solução que vá de encontro às características das provas e dos espaços onde decorrem. Num contexto social o tempo não é tão limitado e espera-se uma solução mais aprimorada pelo que, serão utilizadas redes neuronais convolucionais e um conjunto de imagens RGB obtidas a partir da câmara de profundidade Intel RealSense D455.

O RoboCup@Home, é uma competição robótica internacional que visa desenvolver tecnologia aplicada à robótica de serviços e assistência de pessoas com alta relevância para futuras aplicações pessoais domésticas. A eficiência e precisão nas tarefas do dia-a-dia do CHARMIE são aferidas no RoboCup@Home. A equipa já participou em anos anteriores no RoboCup@Home, nomeadamente no ano de 2010 com um robô anterior ao CHARMIE, nomeadamente o robô *MARy*, sendo que todo o trabalho desenvolvido e que continua em desenvolvimento teve como objetivo participar no Campeonato Mundial de Robótica em Bordeaux, França em Julho de 2023, [4], bem como estar apto e com mais capacidades para futuros RoboCup@Home.

De acordo com as duas abordagens, as tarefas a realizar foram as seguintes:

- Estudo e análise do *Hardware*, nomeadamente o robô CHARMIE e a câmara a utilizar
- Desenvolvimento de Peças em 3D para o robô CHARMIE
- Estudo do regulamento da competição RoboCup e das tarefas em competição
- Estudo do estado da arte de tecnologias e conceitos relativos à deteção e reconhecimento facial
- Estudo de algoritmos para deteção e reconhecimento de utilizadores humanos:
  - Estudo de diversas bibliotecas relativas à deteção e reconhecimento facial
  - Estudo de diversas arquiteturas de redes neuronais relativas à deteção e reconhecimento facial
- Desenvolvimento relativamente à deteção e reconhecimento facial de utilizadores previamente definidos:
  - Criação do *dataset* e algoritmo que seja capaz de aprender um novo utilizador
  - Implementação de um sistema de deteção facial em imagens RGB e imagens de profundidade
  - Implementação e treino de CNN
  - Implementação de um classificador SVM
- Desenvolvimento do Reconhecimento de género, raça, altura, previsão da idade

- Desenvolvimento das provas Rececionista e *Stickler for the Rules* do RoboCup@Home
- Testes e melhorias aos algoritmos implementados no robô CHARMIE e análise dos resultados obtidos
- Participação no Festival Nacional de Robótica na liga *@Home*
- Participação no *RoboCup* 2023 na liga *RoboCup@Home*

1.3 Contribuições Científicas

Esta dissertação e todo o trabalho implementado foi desenvolvido no Laboratório de Automação e Robótica (LAR) do departamento de Eletrónica Industrial da Universidade do Minho. Esta dissertação decorreu no âmbito do projeto CHARMIE (Collaborative Home Assistant Robot by Minho Industrial Electronics), que tal como foi mencionado anteriormente é um robô de serviços doméstico e de saúde e um robô de assistência capaz de executar tarefas em ambientes não padronizados. O objetivo principal deste robô é realizar tarefas diárias e ajudar humanos que possuam algum tipo de dificuldade na execução dessas mesmas tarefas, para além de poder apoiar humanos em tarefas algumas delas monótonas e repetitivas permitindo aliviar a carga de trabalho e a dependência de humanos nas tarefas. Alguns exemplos de tarefas são abrir portas e armários, manipulação e transporte de objetos, estabelecer diálogos, ajuda na preparação de refeições, na verificação de pessoas, ou seja se as mesmas estão bem e não precisam de ajuda, entre outras.

Para além disto, existe também um segundo objetivo que tal como mencionado anteriormente consiste na participação do CHARMIE no RoboCup@Home ou seja, a realização de serviços domésticos multiusos dentro de uma casa, onde o objetivo passa por realizar tarefas domésticas e demonstrar capacidades e ações inovadoras como arrumar as compras, colocar a louça na máquina, verificar se todas as regras de uma casa são cumpridas e até receber convidados numa festa.

Em [5], [6], [7] e [8] estão os trabalhos apresentados pelas equipas que no RoboCup@Home de 2019 nas ligas OPL e DSPL ficaram nos lugares cimeiros.

A figura 1, mostra o robô CHARMIE, utilizado na dissertação bem como no RoboCup@Home de 2023.

O manipulador móvel CHARMIE segue um design onde o objetivo é proporcionar a melhor interação possível entre homem-robô, onde não prevaleça medo ou estranheza entre os seus futuros utilizadores. Para isto, as entradas do robô são semelhantes aos sentidos humanos, como a visão, tato, audição e possui ainda sensores (LiDAR). A sua forma e os seus graus de liberdade são semelhantes ao corpo humano, exceto na locomoção que se trata de uma base omnidirecional de quatro rodas. O seu tronco é capaz de se baixar e subir e ainda inclinar-se permitindo aceder a locais que numa situação normal um robô teria mais dificuldades, algo que é considerado inovador tendo em conta os robôs que anteriormente foram desenvolvidos e participaram na competição.



Figura 1: Robô CHARMIE, versão Junho de 2023

A par de tudo isto, o robô deve ser capaz de interagir com o seu ambiente da mesma forma que um ser humano interage, falar, ouvir, manipular objetos e transportá-los, entre outras coisas e para isso estão a ser usados diferentes algoritmos de *ML*

1.4 Estrutura do Documento

Este documento está dividido em 5 partes: Introdução, Revisão Literária, Métodos e Metodologias, Resultados obtidos e Conclusões.

No capítulo **1 Introdução**, é apresentada uma breve descrição do trabalho, bem como a motivação para o seu desenvolvimento, sendo ainda delineados os objetivos necessários para alcançar o resultado previsto e as contribuições científicas do projeto.

Ao longo do capítulo **2 Revisão Literária**, é apresentado o estudo do estado da arte. Os aspetos teóricos essenciais a esta dissertação são detalhados, sendo inicialmente abordado o tema de *ML* e os diferentes tipos de aprendizagem, seguido por um estudo sobre as Redes Neuronais, onde são abordados desde os conceitos anteriores e básicos até as arquiteturas mais complexas utilizadas atualmente, onde são destacadas as redes neuronais convolucionais. É também abordada a importância de imagens de profundidade num problema de deteção e reconhecimento, sendo de seguida introduzidos métodos utilizados para a deteção e reconhecimento da face humana, demonstrando como a introdução da *IA* melhorou a precisão na deteção e reconhecimento facial.

O capítulo **3 Métodos e Metodologias** começa com uma explicação das ferramentas utilizadas bem como das especificações do sistema. É analisado o *hardware* utilizado bem como aquele que foi desenvolvido a par desta dissertação. Depois é analisada a abordagem com redes neuronais para um

ambiente social, com todos os módulos desenvolvidos, sendo o mesmo feito relativamente à abordagem competitiva onde é feita uma explicação da competição RoboCup e da modalidade RoboCup@Home, bem como das diversas provas e do trabalho efetuado para o desenvolvimento das provas.

Em seguida, o capítulo **4 Resultados** compara os resultados obtidos entre os algoritmos do capítulo anterior para as duas abordagens em causa. Tanto no caso da abordagem com redes neuronais para um ambiente social como para a abordagem competitiva, estes incluem a deteção de rostos humanos em fotos e vídeos, o processamento em tempo real, a aquisição e pré-processamento de imagens individuais (para criar um conjunto de dados). Na abordagem social acresce ainda os resultados do treino das redes neuronais usando diferentes hiperparâmetros e os resultados do classificador na tarefa de reconhecimento facial. Na abordagem competitiva são ainda apresentados os resultados obtidos tendo em conta os algoritmos desenvolvimentos. Finalmente, todas essas partes são interligadas, mostrando o resultado da dissertação ao juntar os diversos módulos no robô CHARMIE.

O capítulo **5 Conclusões** apresenta os resultados obtidos tendo em conta os objetivos bem como sugestões de trabalho futuro.

Capítulo 2

Revisão Literária

Neste capítulo serão abordados diversos tópicos necessários para o desenvolvimento desta dissertação e que serão abordados ao longo de todo o documento. Serão abordados os vários campos da IA e redes neuronais que estão associados à deteção e reconhecimento facial, (ver figura 2).

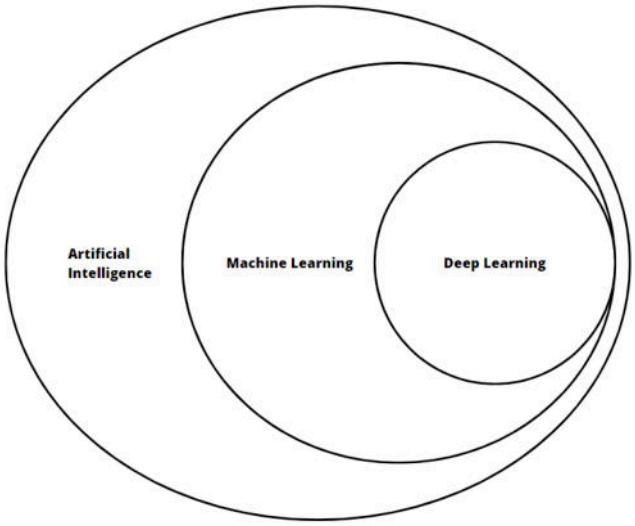


Figura 2: Relação entre IA, *ML* e *DL*

2.1 *Machine Learning*

A IA é a capacidade de uma máquina reproduzir competências semelhantes às humanas como é o caso do raciocínio, a aprendizagem, o planeamento e a criatividade. IA é o estudo científico dos algoritmos

e modelos estatísticos que os sistemas informáticos utilizam para realizar um tarefa específica sem ser "explicitamente programada"como refere Arthur Samuel [9]. *ML* pode ser dividida em três ramos (figura 3): aprendizagem supervisionada, aprendizagem não-supervisionada e aprendizagem de reforço, que podem ser respetivamente divididos em sub-ramos específicos. O tipo de algoritmo utilizado depende do tipo de problema a resolver, do número de variáveis e do tipo de modelo que melhor se adequa, entre outros.

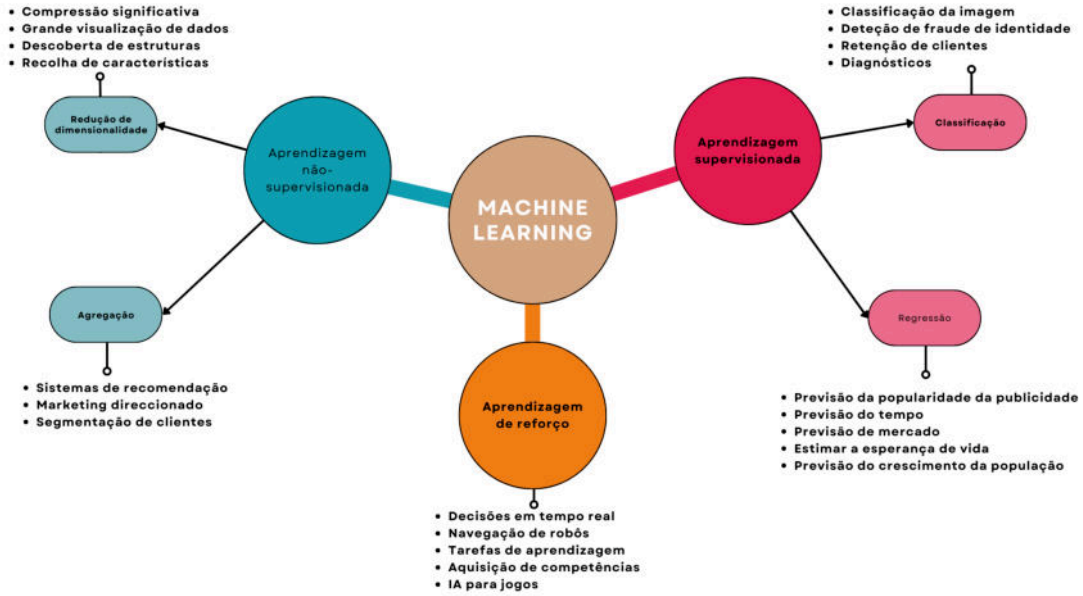


Figura 3: Categorias de *ML*

2.1.1 Aprendizagem supervisionada

A aprendizagem supervisionada [10] consiste no mapeamento de uma entrada para uma saída com base num exemplo de pares entrada-saída. São usados *datasets* categorizados, que incluem um conjunto de amostras de formação que consiste num par tanto do objeto de entrada como do valor de saída desejado, para treinar algoritmos de classificação de dados ou previsão de resultados com precisão. À medida que os dados de entrada são inseridos no modelo, o modelo ajusta os seus pesos até que tenha sido ajustado adequadamente. Isso ocorre como parte do processo de validação cruzada para garantir que o erro do modelo é mínimo. Este é um algoritmo onde é necessária assistência externa e é daí que provém a sua nomenclatura.

Tal como é possível verificar na imagem acima, a aprendizagem supervisionada pode ainda ser agrupada em: classificação e regressão [11].

2.1.1.1 Classificação

É uma tarefa de aprendizagem supervisionada onde a saída tem categorias definidos (valor discreto). É o processo de encontrar ou descobrir um modelo ou função que ajude a separar os dados em múltiplas

classes categóricas, ou seja, valores discretos. Na classificação, os dados são categorizados sob diferentes etiquetas de acordo com alguns parâmetros indicados na entrada e, em seguida, as etiquetas são previstas para os dados. O processo de classificação trata dos problemas em que os dados podem ser divididos em etiquetas binárias ou múltiplas etiquetas discretas. Na classificação binária, o modelo prevê 0 ou 1; sim ou não, mas no caso da classificação multi-classe, o modelo prevê mais do que uma classe.

2.1.1.2 Regressão

É o processo de encontrar um modelo ou função para distinguir os dados onde o resultado é um valor real contínuo, em vez de usar classes ou valores discretos. O objetivo aqui é prever um valor o mais próximo possível do valor real de saída do nosso modelo e depois a avaliação é feita calculando o valor do erro. Quanto menor for o erro, maior é a precisão do nosso modelo de regressão.

2.1.2 *Random Forest*

Um algoritmo de *Random Forest* é um algoritmo de ML de aprendizagem supervisionada que é usado tanto para problemas de classificação como de regressão.

Uma das principais características do *Random Forest* é a sua capacidade de lidar com dados de alta dimensionalidade, bem como com *datasets* grandes. De acordo com *Breiman* em [12], *Random Forest* é um algoritmo que é construído a partir de algoritmos de árvore de decisão. Em vez de depender de uma árvore de decisão, um *Random Forest* pega na previsão de cada árvore e com base nos votos da maioria das previsões, prevê o resultado final.

Cada árvore individual num *Random Forest* é construída usando uma técnica conhecida como *bootstrap aggregating*, ou *bagging*, representada na figura 4. Cada árvore é treinada com uma amostra aleatória com substituição dos dados de treino original. Além disso, durante o processo de construção da árvore, em cada nó apenas um subconjunto aleatório de variáveis é considerado para dividir os dados, tornando as árvores independentes umas das outras.

Após a construção das árvores, o *Random Forest* realiza as previsões agregando as previsões individuais de cada árvore. Para problemas de classificação, a classe mais frequente entre as previsões de todas as árvores é escolhida como a previsão final. Para problemas de regressão, a média das previsões de todas as árvores é tomada como a previsão final.

Uma vantagem significativa do *Random Forest* é a sua capacidade de lidar com *overfitting*. Devido à natureza aleatória da seleção de amostras e variáveis, este algoritmo reduz a tendência de sobre ajuste aos dados de treino. Além disso, o *Random Forest* é capaz de lidar com dados ausentes e valores discrepantes, sem a necessidade de pré-processamento extensivo.

Para além do mais, o *Random Forest* fornece uma medida de importância das variáveis, permitindo a identificação das características mais relevantes para o problema em questão. Isso pode ser útil na extração de *insights* e tomada de decisões.

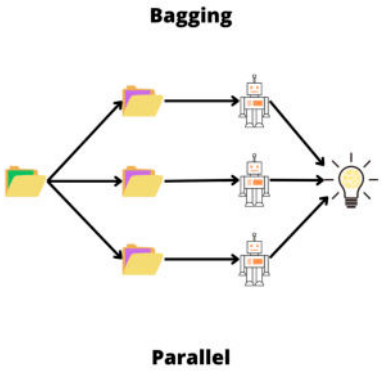


Figura 4: Representação da técnica *bagging*

2.1.3 Aprendizagem não-supervisionada

É chamada de aprendizagem não supervisionada porque ao contrário da aprendizagem supervisionada, não há respostas e não há ninguém a ensinar. A aprendizagem não supervisionada analisa e agrupa conjuntos de dados não categorizados utilizando algoritmos de *ML*. Estes algoritmos encontram padrões e dados ocultos sem qualquer intervenção humana, ou seja, não sendo por isso fornecidas saídas ao nosso modelo. O modelo de formação tem apenas valores de parâmetros de entrada e descobre os grupos ou padrões por si. É principalmente utilizado para *clustering* e redução de dimensionalidade.

2.1.3.1 *Clustering*

É uma técnica de aprendizagem não supervisionada, que agrupa conjuntos de dados não categorizados, sendo definido como uma maneira de agrupar os pontos de dados em *clusters* diferentes, consistindo em pontos de dados semelhantes. Os objetos com as possíveis semelhanças permanecem num grupo que tem menos ou nenhuma semelhança com outro grupo.

2.1.3.2 Redução de dimensionalidade

Este tipo de aprendizagem não supervisionada é usado para reduzir a quantidade de variáveis ou dimensões num conjunto de dados. Isto pode ser útil para visualização de dados ou para reduzir a complexidade de modelos de ML.

2.1.4 Aprendizagem de reforço

Aprendizagem de reforço trata de tomar as medidas adequadas para maximizar a recompensa numa situação particular, e é utilizada para encontrar o melhor comportamento ou caminho possível numa situação específica. A aprendizagem de reforço difere da aprendizagem supervisionada de uma forma que, na aprendizagem supervisionada, os dados têm a chave de resposta, pelo que o modelo é treinado

com a resposta correta em si, enquanto que na aprendizagem de reforço não há resposta, mas o agente de reforço decide o que fazer para executar a tarefa dada. Na ausência de um conjunto de dados de formação, é obrigado a aprender com os dados que for adquirindo ao longo do tempo.

2.1.5 Avaliação de modelos de *Machine Learning*

Na aprendizagem supervisionada, os classificadores são algoritmos que têm a tarefa de atribuir categorias a instâncias de dados com base num conjunto de características ou atributos. Para avaliar o desempenho desses classificadores, é comum comparar as categorias reais (também conhecidas como rótulos de referência) com as categorias previstas pelos classificadores para as mesmas instâncias de dados.

Avaliar o desempenho de um classificador é importante para determinar o quão efetivamente ele está a realizar a tarefa de classificação. A comparação entre as etiquetas reais e previstas permite medir a precisão do classificador e determinar quão bem ele está a generalizar o conhecimento dos dados de treino para novas instâncias de teste.

Estas etiquetas previstas pelos classificadores são normalmente designadas como positivas ou negativas, e são designadas pela sua notação [13]:

- Verdadeiro Positivo (VP) quando o classificador prevê corretamente uma etiqueta positiva.
- Falso Positivo (FP) quando o classificador prevê incorretamente uma etiqueta positiva.
- Verdadeiro Negativo (VN) quando o classificador prevê corretamente uma etiqueta negativa.
- Falso Negativo (FN) quando o classificador prevê incorretamente uma etiqueta negativa.

Estas notações permitem calcular diferentes métricas, como *Accuracy* (Exatidão), *Recall*, Precisão e *F1-Score*, variando entre 0 e 1. Abaixo estão as equações de cada métrica mencionada.

$$\text{Exatidão} = \frac{VP + VN}{VP + FP + VN + FN} \tag{2.1}$$

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{2.2}$$

$$\textit{Recall} = \frac{VP}{VP + FN} \tag{2.3}$$

$$\textit{F1-Score} = \frac{2 * Precisao * Recall}{Precisao + Recall} \tag{2.4}$$

Exatidão é a proporção de instâncias corretamente classificadas em relação ao total de instâncias. Por outro lado, precisão é a proporção de instâncias corretamente classificadas como positivas em relação ao total de instâncias classificadas como positivas. *Recall* (também conhecido como sensibilidade) é a

proporção de instâncias positivas corretamente classificadas em relação ao total de instâncias positivas. O *recall* é útil quando o objetivo é minimizar os falsos negativos. Por último, *F1-Score* pode ser interpretado como uma métrica que combina precisão e *recall* numa única medida, sendo a média harmónica entre os dois.

Estas métricas são importantes para analisar e avaliar o desempenho do modelo classificador e podem ser obtidas através de uma matriz de confusão, que consiste numa representação visual através da qual é possível avaliar quais as classes que estão a ser identificadas corretamente e quais estão a ser confundidas.

2.1.6 *Support Vector Machine* - SVM

Support Vector Machine [14] é um dos algoritmos de aprendizagem supervisionada que é usado para problemas de classificação e regressão. O principal objetivo do algoritmo SVM é encontrar o hiperplano ideal num espaço N-dimensional que possa separar os pontos em diferentes classes no espaço de feição. O hiperplano tenta que a margem entre os pontos mais próximos de diferentes classes seja a máxima possível. A dimensão do hiperplano depende do número de características. Os vetores de suporte são pontos de dados que estão mais próximos do hiperplano e influenciam a posição e a orientação do hiperplano. Usando esses vetores de suporte, maximizamos a margem do classificador. A exclusão dos vetores de suporte altera a posição do hiperplano. Esses são os pontos que ajudam a construir o SVM, na figura 5.

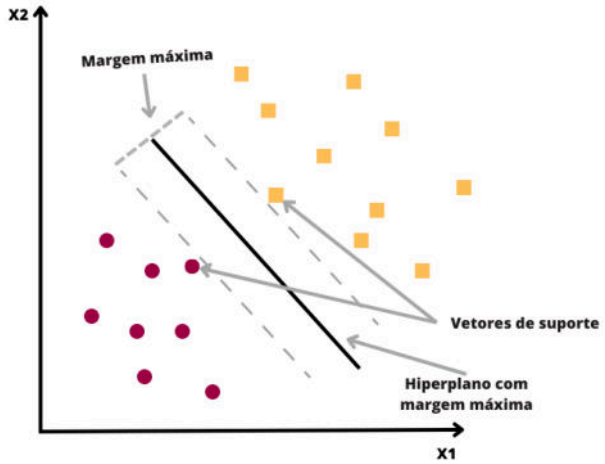


Figura 5: Representação de uma SVM

No contexto do reconhecimento facial, o SVM tem sido utilizado para enfrentar os desafios inerentes a essa tarefa, como variações na iluminação, expressões faciais, ângulos de visão e até mesmo oclusões parciais. Comparativamente a outras técnicas de classificação, como redes neuronais profundas, o SVM possui vantagens significativas, principalmente quando o tamanho do *datset* de treino é limitado e não exige uma elevada capacidade computacional.

Tal como é referido em [15] uma metodologia comum é treinar uma rede neuronal, como uma CNN, para atuar como um *facial feature extrator*. As redes neuronais convolucionais têm demonstrado grande sucesso na extração de características relevantes de imagens, aprendendo representações hierárquicas que capturam informações discriminativas dos rostos. Ao utilizar uma CNN pré-treinada num grande *dataset*, podemos aproveitar esse conhecimento prévio e transferir as capacidades de extração de características para o nosso problema específico de reconhecimento facial.

Uma vez que a CNN tenha sido treinada como um *facial feature extrator*, são utilizadas as representações aprendidas pelos neurónios na camada final da rede como entradas para o SVM. A transformação das imagens faciais num espaço de características mais compacto e discriminativo permite que o SVM realize a tarefa de classificação. O SVM é capaz de lidar com problemas de classificação binária e multi-classe, tornando-o adequado para o reconhecimento facial, onde múltiplas identidades precisam de ser classificadas.

2.2 Deep Learning

DL é um ramo de ML que se baseia em redes neuronais artificiais uma vez que as redes neuronais foram concebidas para simular o comportamento do cérebro humano na resolução de problemas, quer pela sua estrutura, quer pelo seu método de tratamento da informação. O cérebro é composto por uma rede de neurónios que processam informação, sendo também identificados como neurónios, (figura 6).

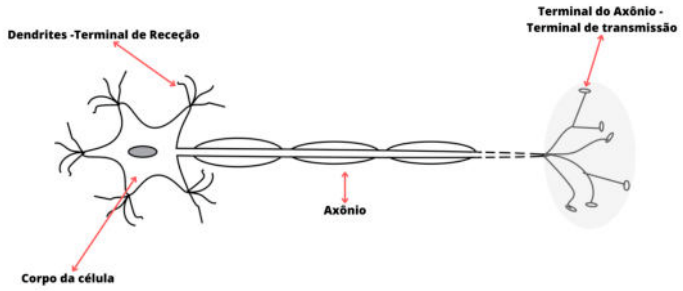


Figura 6: Neurónio [16]

DL é uma forma de aprender representações a partir de dados pois este sub-campo enfatiza a aprendizagem através de sucessivas camadas de representações, gradualmente mais significativas. Estas representações sucessivas formam camadas como um todo, as chamadas redes neuronais, que são a estrutura básica do DL. É um modelo matemático simples do cérebro que é utilizado para processar capacidades não lineares, relações entre entradas e saídas em paralelo tal como um cérebro humano faz em cada instância.

O conceito de rede neuronal começou com o trabalho do fisiologista, *Warren McCulloch*, e do matemático, *Walter Pitts*, quando em 1943 eles modelaram uma rede neuronal simples usando circuitos elétricos para descrever como os neurónios no cérebro poderiam funcionar [17].

2.2.1 Redes Neuronais Artificiais ANNs

Uma Artificial Neural Network (ANN) é um modelo computacional capaz de executar tarefas como previsão, classificação, tomada de decisões, etc. É constituída por neurónios artificiais, sendo estes neurónios uma cópia dos neurónios do cérebro humano. Os neurónios no cérebro passam os sinais para realizar as ações, da mesma forma que os neurónios artificiais ligam-se numa rede neuronal para executar as tarefas. A ligação entre os neurónios artificiais é chamada peso.

Uma rede neuronal é constituída por três camadas, sendo a primeira a de "entrada"onde os neurónios enviam informação para a segunda camada, sendo esta denominada "oculta". A camada oculta realiza os cálculos sobre os dados de entrada e transfere a saída para a última camada, a camada de saída.

O neurónio artificial recebe um ou mais sinais de entrada, *inputs*, processa essa informação e produz uma resposta, *output*. Geralmente aos *inputs* é aplicado um peso, e os sinais resultantes são posteriormente somados. À soma resultante é aplicada uma função não linear, chamada de função de transferência, sendo assim gerado o *output*. Uma rede neuronal artificial tem a capacidade de aprender com a informação que lhe é fornecida e de melhorar a sua performance durante o processo de aprendizagem - aprende por um processo iterativo de ajuste das chamadas forças sinápticas (correspondentes a pesos). A cada iteração do processo de aprendizagem, é capaz de aperfeiçoar a sua representação e compreensão do problema em questão, ou seja, aprende por treino e experiência segundo certas regras pré-definidas. Ao conjunto de regras pré-definidas pelo qual se faz a alteração dos pesos dá-se o nome de algoritmo de aprendizagem e existem diversos algoritmos de aprendizagem que diferem entre si, tanto na forma como os pesos são corrigidos, bem como da estrutura da própria rede.

O cérebro humano contém uma rede densamente interligada de aproximadamente 10^{11}-10^{12} neurónios, onde cada neurónio está ligado em média a 10^{4}-10^{5} outros neurónios. Assim, em média, o cérebro humano leva aproximadamente 10^{-1}s para tomar decisões surpreendentemente complexas. Os sistemas ANN são motivados a capturar este tipo de cálculo altamente paralelo com base em representações distribuídas. Geralmente, as ANNs são construídas a partir de um conjunto densamente interligado de unidades simples, onde cada unidade usa um número de entradas com valor real e produz uma única saída com valor real.

2.2.1.1 Redes Neuronais Profundas DNNs

O *DL* é uma área especifica do *ML* que trabalha com DNN que tal como representado na figura 7, são redes neuronais mas com múltiplas camadas ocultas. As DNN têm a mesma estrutura que as ANN, mas com mais camadas ocultas, ou seja os dados passam por um processo de reconhecimento de padrões em mais camadas. Para uma rede neuronal se qualificar como profunda, esta deve ser constituída por pelo menos duas camadas ocultas. Cada camada de neurónios treina sobre um conjunto distinto de características com base na saída da camada anterior. As DNN são redes *Feedforward*, uma vez que o fluxo de informação é passado de camada a camada, desde a entrada até à saída, sem que haja *feedback* entre as camadas. Quanto mais os dados avançam na rede, mais complexas são as características que os

neurónios podem reconhecer uma vez que agregam e recombinam características das camadas anteriores tratando-se de uma hierarquia de características, sendo esta de complexidade e abstração crescentes, o que torna as DNN capazes de manipular *datasets* de grandes dimensões e com uma quantidade grande de parâmetros que passam por funções não lineares.

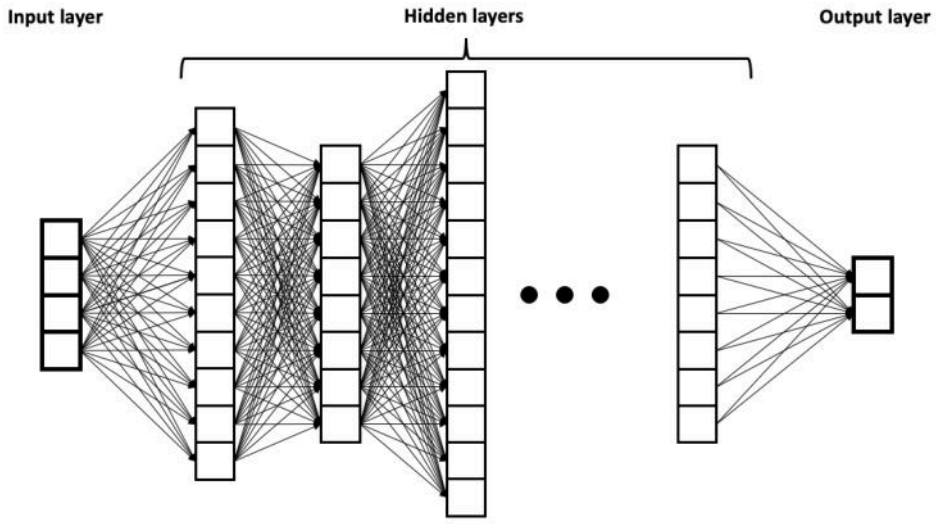


Figura 7: Rede Neuronal Profunda DNN

2.2.2 Redes Neuronais Convolucionais CNNs

Uma rede neuronal convolucional, ou CNN, desenvolvida por *Yann LeCun* em 1998 [18], é uma rede neuronal de *DL* projetada para processar matrizes estruturadas de dados, como imagens. O desenho de uma CNN é motivado pela descoberta de um mecanismo visual, o córtex visual, no cérebro. As CNN são amplamente utilizadas na visão por computador nomeadamente na classificação de imagens uma vez que estas são muito boas em captar padrões na imagem de entrada, como linhas, gradientes, círculos ou até mesmo olhos e rostos. É essa propriedade que torna as redes neuronais convolucionais tão poderosas para a visão por computador.

Cada pixel de uma imagem é considerado um neurónio de entrada e o seu valor é dado pela sua intensidade, por exemplo, a base de dados MNIST [19], de dígitos manuscritos é adequada para a maioria das formas de ANN, devido à sua pequena dimensionalidade de imagem de apenas 28×28 *pixels*. Com este conjunto de dados, um único neurónio na primeira camada oculta contém 784 pesos (28×28), o que é controlável para a maioria das formas de ANN. Estando as camadas totalmente ligadas, as DNNs ligam todos os neurónios de uma camada à próxima, o que faz com que os *pixels* em vários pontos de uma imagem sejam tratados da mesma forma. Dada a estrutura espacial de uma imagem, os *pixels* num determinado local têm características muito semelhantes e tendo isto em conta é possível mapear localizações na imagem, facilitando o processamento e melhorando a recolha de características.

Quando comparado com outros algoritmos de classificação, a CNN requer menos pré-processamento, sendo a CNN um método supervisionado que consiste numa camada de entrada e saída, bem como em múltiplas camadas ocultas, tal como mostra a figura 8. Neste método, as camadas dentro da CNN são constituídas por neurónios organizados em três dimensões nomeadamente altura, largura e profundidade [19] e estas camadas estão geralmente divididas em três tipos: camadas convolutivas, camadas de agrupamento e camadas totalmente ligadas [20].

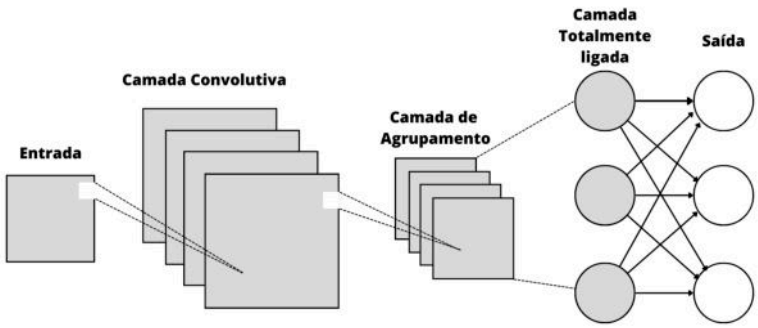


Figura 8: Arquitetura CNN

2.2.2.1 Camada Convolutiva

A operação matemática de convolução consiste no deslocamento de uma função sobre outra e a medição do integral da sua multiplicação pontual. É também conhecida como um detetor de feições de uma CNN onde aplica um *Kernel* de convolução à entrada e retorna um mapa de feição como saída [21].

A camada convolutiva é a primeira camada usada para extrair os vários recursos das imagens de entrada e a operação matemática de convolução é realizada entre a imagem de entrada e um filtro de um determinado tamanho MxM, que ao deslizar sobre a imagem de entrada, o produto de ponto é levado entre o filtro e as partes da imagem de entrada em relação ao tamanho do filtro (MxM).

A saída denominada como o mapa de feição dá informações sobre a imagem como os cantos e bordas sendo que mais tarde esse mapa de feição e é alimentado para outras camadas para aprender vários outros recursos da imagem de entrada.

A camada de convolução na CNN passa o resultado para a próxima camada uma vez que se aplica a operação de convolução na entrada. As camadas convolucionais na CNN são benéficas pois garantem que a relação espacial entre os *pixels* esteja intacta.

No exemplo acima para uma dada imagem de entrada e um filtro de dimensão 3x3 resulta uma determinada saída. Para cada conjunto de 3x3 *pixels* na imagem de entrada, o filtro efetua uma multiplicação

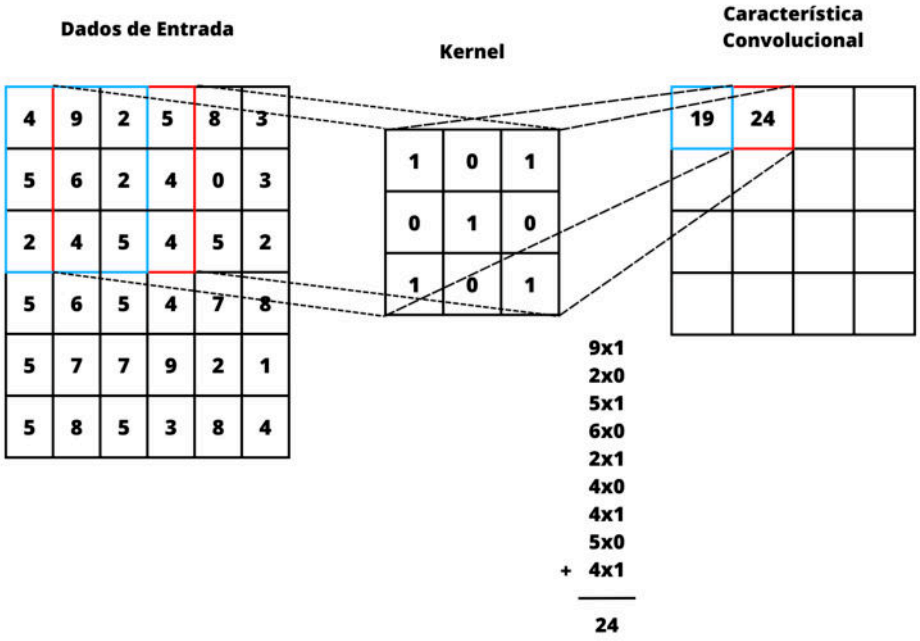


Figura 9: Exemplo do que acontece na camada convolucional

no sentido dos elementos à medida que desliza através de cada um dos conjuntos. Quando atinge o limite à direita, como descrito na figura 9, o filtro desce uma linha e continua a interação através da imagem de entrada, sendo o resultado desta multiplicação a imagem de saída.

As camadas convolutivas podem reduzir a complexidade do modelo através da otimização da sua produção e é através de hiperparâmetros que ocorre a dita otimização, uma vez que estes ditam o arranjo espacial e o tamanho do volume de saída de uma camada convolucional [22]. A seguir estão alguns dos hiperparâmetros mais importantes:

- Tamanho do filtro: É geralmente espacialmente pequeno e possui três dimensões - largura, altura e canais de cor.

- Profundidade de saída: Controla o número de neurónios na camada convolucional que estão ligados à mesma região no volume de entrada.

- Passo: Define o ritmo de deslizamento do filtro por aplicação. A profundidade do volume de saída é inversamente proporcional ao valor do passo.

- Preenchimento zero: Preenche com zeros certos *pixels*, é o processo de adicionar zeros simetricamente à matriz de entrada e é usado principalmente quando as dimensões da matriz de entrada precisam ser preservadas na matriz de saída.

2.2.2.2 Camada de Agrupamento

A camada de agrupamento é geralmente incorporada entre camadas convolucionais sucessivas de forma a reduzir progressivamente o tamanho espacial da representação de dados e, desta forma, reduzir os custos computacionais e a complexidade do modelo.

Isso é realizado diminuindo as ligações entre as camadas e operando independentemente em cada mapa de feição. Dependendo do método usado, existem vários tipos de operações de agrupamento onde basicamente são resumidas as características geradas por uma camada de convolução.

No agrupamento máximo, o maior elemento é retirado do mapa de feição e o agrupamento médio calcula a média dos elementos numa secção da imagem de tamanho predefinido. A soma total dos elementos na secção predefinida é calculada num agrupamento de somas.

Esta camada também utiliza filtros, semelhantes à camada convolutiva, mas em vez de realizar a soma ponderada executa funções aritméticas em todos os mapas de ativação anteriores. Os métodos mais comuns da camada de agrupamento são o agrupamento máximo, que procura o valor máximo de todos os *pixels* dentro do conjunto, e o agrupamento médio, em que o valor médio é calculado

Este modelo CNN generaliza as características extraídas pela camada de convolução e ajuda as redes a reconhecer as características de forma independente e com isto os cálculos também são reduzidos numa rede.

Na figura 10, encontra-se um exemplo de agrupamento máximo (*Max Polling*) e agrupamento médio (*Average Polling*), onde cada matriz 2x2 é combinada num único valor, nomeadamente no valor máximo e no valor médio de acordo com o tipo de agrupamento.

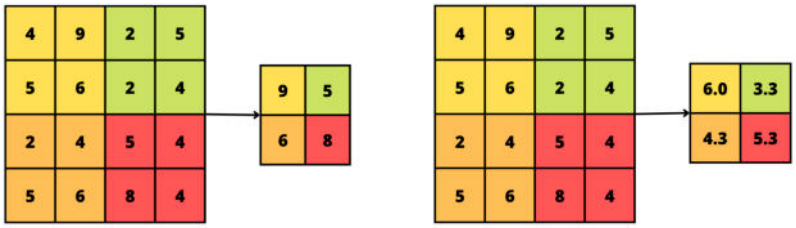


Figura 10: Exemplos de agrupamento máximo e médio respetivamente

2.2.2.3 Camada totalmente ligada

As camadas totalmente ligadas serão utilizadas no final da rede após a extração das características desejadas através de convoluções e agrupamento (que ocorre nas camadas anteriores). Essa camada atua como camada de saída para a rede e tem a dimensão do volume de saída como [1 x 1 x N] onde N é o número de classes de saída a serem avaliadas. As camadas totalmente ligadas têm parâmetros gerais da camada de rede neuronal e hiperparâmetros. Cada nó de uma camada totalmente ligada está diretamente ligado a cada nó da camada anterior e da camada seguinte. Num modelo básico da CNN, as

características geradas pela última camada de convolução representam uma parte da imagem de entrada, uma vez que o seu campo recetivo não integra toda a dimensão espacial da imagem.

2.3 Arquiteturas das CNN

Nesta secção é apresentada uma visão geral de algumas estruturas CNN que foram importantes para o desenvolvimento desta área.

2.3.1 LeNet-5

A primeira arquitetura notória de uma CNN tem o nome LeNet-5, foi concebida em 1990 por *Yann LeCun* [23], sendo que LeCun foi um pioneiro deste tipo de redes. Sendo a primeira arquitetura estrutural para este tipo de redes, LeNet-5 introduziu o conceito de redes convolucionais, incorporando o mapeamento espacial numa imagem, onde extraiu as características presentes através de filtros designados. A arquitetura foi criada com várias camadas, nomeadamente:

- Camada de entrada no início da rede - considerada como imagem inicial
- Camada Convolucional - criada a partir das convoluções da entrada inicial, onde foram aplicados 5x5 filtros com passo de 1
- Camada Média de Agrupamento - criada por sub-amostragem com um filtro 2x2 com passo de 2, reduzindo o tamanho espacial da imagem inicial
- Camada totalmente ligada - ordena os dados recebidos da rede, prevendo a entrada

Embora o método tenha provado ser eficiente, sendo reconhecido pela sua inovação na altura, o seu elevado custo computacional exigiu que o modelo fosse treinado durante 2 a 3 dias até que pudesse prever em conformidade um conjunto de dados simples. Combinado com a falta de dados de formação para problemas da vida real, esta abordagem provou ser insustentável para fins melhores do que o reconhecimento de características simples em imagens pequenas.

2.3.2 AlexNet

Enquanto LeNet é o que inicia a era das Redes Neuronais Convolucionais (CNN), AlexNet em 2012 [24] e é quem inicia a era da CNN usada para a classificação ImageNet.

A arquitetura AlexNet é semelhante à LeNet-5, mas tem um maior número de camadas e AlexNet introduz a função de ativação ReLU e LRN na mistura. O ReLU tornou-se tão popular que quase todas as arquiteturas da CNN desenvolvidas após a AlexNet usaram o ReLU nas camadas ocultas, abandonando

o uso da função de ativação tanh no LeNet-5. Tem uma composição simples comparando com as arquiteturas mais atuais. O design da arquitetura AlexNet é composto por 5 camadas enroladas, camadas de agrupamento máximo, camadas de abandono e 3 camadas totalmente ligadas.

Com o avanço do *hardware* moderno a AlexNet pode ser treinada com 60 milhões de parâmetros e tornou-se a vencedora da competição *ImageNet Large Scale Visual Recognition Challenge* em 2012. O ImageNet tornou-se um *dataset* de referência no desenvolvimento de arquiteturas CNN e um subconjunto dele (ILSVRC) consiste em várias imagens com 1000 classes.

2.3.3 VGG-Net

VGG-Net foi proposto por *Karen Simonyan e Andrew Zisserman* em 2014 [25]. O VGG foi desenvolvido para aumentar a profundidade de CNN, a fim de aumentar o desempenho do modelo, sendo esta arquitetura a base de modelos inovadores de reconhecimento de objetos. Desenvolvido como uma rede neuronal profunda, é uma das arquiteturas de reconhecimento de imagem mais populares.

O modelo VGG, ou VGGNet, suporta 16 camadas (também é referido como VGG16), é um modelo de rede neuronal convolucional. O conceito do modelo VGG19 (também VGGNet-19) é o mesmo que o VGG16, com a diferença de que este suporta 19 camadas. Isso significa que o VGG19 tem três camadas convolucionais a mais do que o VGG16, como é possível ver na figura 11.

A arquitetura do VGG está descrita abaixo:

- Entrada: O VGGNet tem um tamanho de entrada de imagem de 224×224. Para a competição ImageNet, os criadores do modelo cortaram o *patch* central 224×224 em cada imagem para manter o tamanho de entrada da imagem consistente.

- Camadas Convolucionais: As camadas convolucionais do VGG aproveitam um campo recetivo mínimo, ou seja 3×3, o menor tamanho possível que ainda captura para cima/para baixo e esquerda/direita. Além disso existem também filtros de convolução 1×1 que atuam como uma transformação linear da entrada. Isto é seguido por uma unidade ReLU, que é uma enorme inovação da AlexNet que reduz o tempo de treino. ReLU significa função de ativação de unidade linear retificada. É uma função linear por partes que produz a entrada se for positiva; caso contrário a saída será zero. O passo de convolução é fixada em 1 pixel para manter a resolução espacial preservada após a convolução

- Camadas ocultas: Todas as camadas ocultas na rede VGG usam ReLU. O VGG geralmente não aproveita a Normalização de Resposta Local (LRN), pois aumenta o consumo de memória e o tempo de treino. Além disso, não faz melhorias na precisão geral

- Camadas totalmente ligadas: O VGGNet tem três camadas totalmente ligadas, sendo que das três camadas, as duas primeiras têm 4096 canais cada, e a terceira tem 1000 canais sendo 1 para cada classe

A VGG-Net participou no ILSVRC em 2014 no mesmo conjunto de dados que a AlexNet, onde obteve uma taxa de erro top-5 de apenas 7,3%, colocando em 2º lugar na classificação e 1º na localização (*ImageNet Large Scale Visual Recognition Competition*) 2014 (ILSVRC2014).



Figura 11: Configuração da VGG-16 e VGG-19

2.3.4 Google Net ou Inception V1

O Google Net (ou Inception V1) foi proposto por uma pesquisa no Google (com a colaboração de várias universidades) em 2014 [26]. Esta arquitetura, que possui 22 camadas (27 se forem incluídas as camadas de agrupamento), foi a vencedora no desafio de classificação de imagens ILSVRC 2014, onde VGG-Net ficou em segundo lugar como foi mencionado anteriormente. A arquitetura GoogleNet é muito diferente das arquiteturas de geração anteriores, como AlexNet. São usados muitos tipos diferentes de métodos, como convolução 1×1 e agrupamento médio global o que permite criar uma arquitetura mais profunda. Uma convolução 1×1 simplesmente mapeia um pixel de entrada com todos os seus respetivos canais para um pixel de saída. A convolução 1×1 pode ser usada como um módulo de redução de dimensionalidade para reduzir a computação.

Apesar da excelente classificação em ILSVRC em 2014, existiam alguns problemas que condicionavam a precisão do modelo. Em Inception V1, (ver figura 12) eram usadas convoluções 5x5 o que faz com que as dimensões de entrada diminuam por uma grande margem. Isso faz com que a rede neuronal use alguma diminuição de precisão. A razão por trás disto é suscetível à perda de informação se a dimensão de entrada diminuir drasticamente. Na arquitetura Inception V2, figura 13, a convolução 5×5 é substituída por duas convoluções 3×3. Isso também diminui o tempo computacional porque uma convolução 5×5 é 2,78 mais cara do que uma convolução 3×3. Assim, o uso de duas camadas 3×3 em vez de 5×5 aumenta o desempenho da arquitetura.

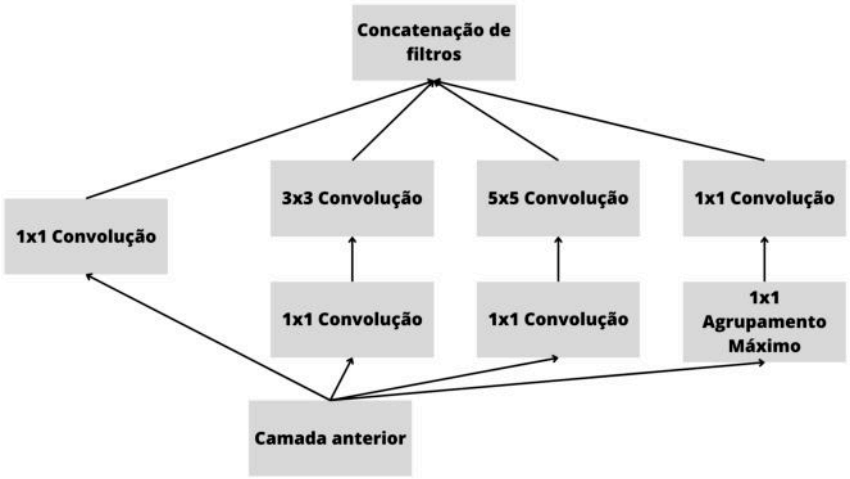


Figura 12: Esquemático de Inception-V1 com redução de dimensão

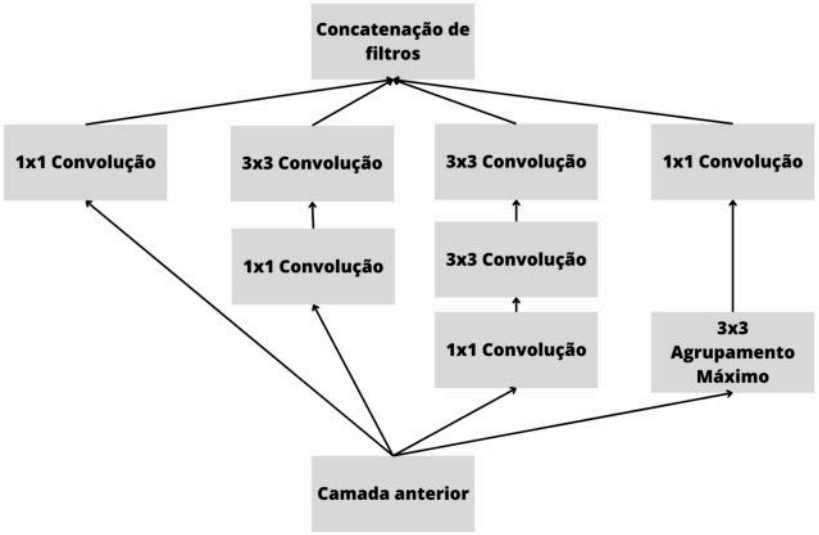


Figura 13: Esquemático de Inception-V2

Inception V3, como descrito na figura 14 é semelhante e contém os recurso de V2 com algumas alterações e adições, nomeadamente o uso de convolução 7x7.

2.3.5 ResNet

Após a primeira arquitetura baseada em CNN (AlexNet) que venceu a competição ImageNet 2012, cada arquitetura vencedora subsequente usa mais camadas numa rede neuronal profunda para reduzir a taxa de erro. Isso funciona para um número menor de camadas, mas quando o número de camadas é aumentado, há um problema comum no *Deep Learning* associado ao chamado gradiente de desaparecimento/explosão, aumentando a taxa de erro de treino e teste.

A ResNet foi proposta em 2015 por elementos da Microsoft *Research* e introduziu uma nova arquitetura

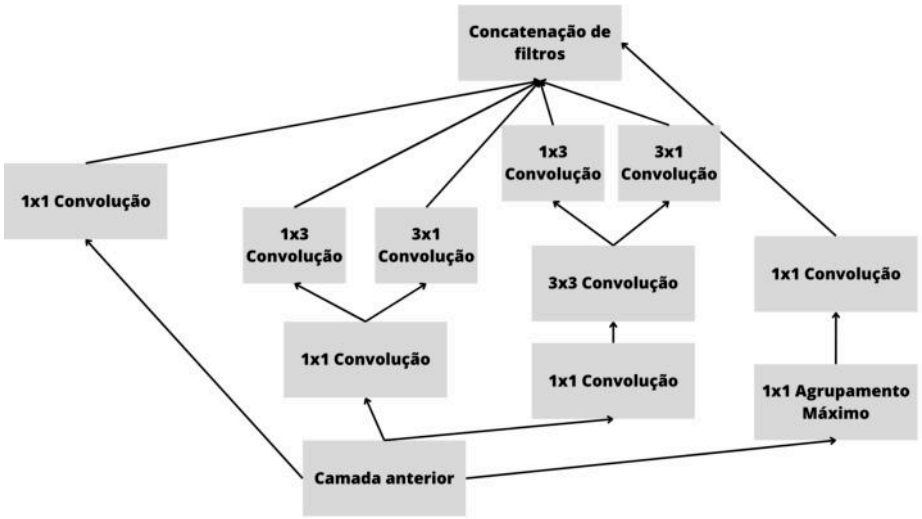


Figura 14: Esquemático de Inception-V3

chamada Residual *Network* [27]. A fim de resolver o problema do gradiente de desaparecimento/explosão, esta arquitetura introduziu o conceito chamado blocos residuais. Nessa rede, é usada uma técnica denominada saltar ligações, com isto são ligadas ativações de uma camada a outras camadas saltando algumas camadas entre elas formando um bloco residual. As ResNets são feitas empilhando esses blocos residuais juntos como descrito na figura 15.

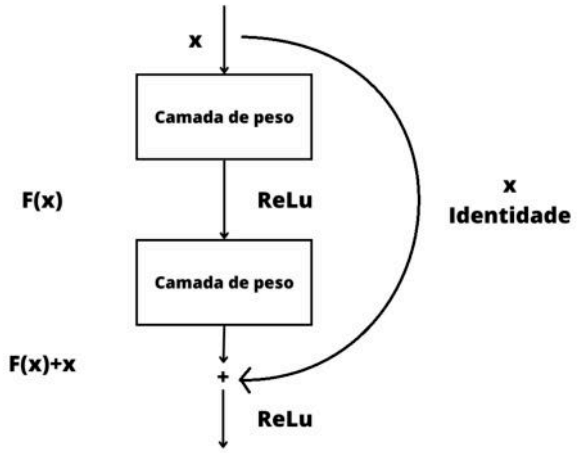


Figura 15: Bloco residual de um modelo de ResNet

A abordagem por trás desta rede é o facto de em vez de as camadas aprenderem o mapeamento subjacente, ser permitido que a rede se ajuste ao mapeamento residual.

ResNet usa uma arquitetura de rede simples de 34 camadas inspirada no VGG-19, na qual a ligação de atalho é adicionada. Essas ligações de atalho convertem a arquitetura numa rede residual.

2.3.6 Inception ResNet V1 e V2

Proposto por Christian Szegedy, Sergey Ioffe e Vincent Vanhoucke [28] a *Inception ResNet* V1 e a *Inception ResNet* V2 são variantes das arquiteturas Inception, que combinam múltiplas escalas espaciais de convolução, com os princípios introduzidos pela arquitetura ResNet, que lida com o desafio de treinar redes neuronais profundas. Ambas as arquiteturas têm como objetivo melhorar o desempenho e a eficiência das CNN em tarefas de visão por computador, mas possuem diferenças na sua estrutura e abordagem.

A *Inception ResNet* V1 é uma arquitetura de rede neuronal convolucional que combina os princípios do módulo *Inception* com a ideia de resíduos introduzida pela *ResNet*. Esta foi desenvolvida para melhorar o desempenho e a eficiência das CNN em tarefas de visão por computador, como classificação de imagens. A estrutura da *Inception ResNet* V1 é composta por blocos *Inception ResNet*, que incorporam diferentes tamanhos de filtro e caminhos de convolução para capturar informações em várias escalas espaciais. Esses blocos permitem a extração de características ricas e discriminativas, enquanto que a utilização de resíduos ajuda a lidar com o desafio de treinar redes neuronais profundas. Na *Inception ResNet* V1, (ver anexo I.1), os blocos são compostos por uma mistura de operações de convolução 1x1, 3x3 e 5x5, além de *max pooling*, que são aplicadas paralelamente. Essas operações são projetadas para capturar informações em diferentes escalas espaciais e permitir a extração de características complexas. A ideia central é utilizar filtros de diferentes tamanhos para capturar detalhes finos e características em diferentes contextos.

Em seguida foi proposta a *Inception ResNet* V2, (ver anexo I.2) como uma extensão da *Inception ResNet* V1, projetada para melhorar ainda mais o desempenho e a eficiência. Assim como a versão anterior, ela utiliza blocos *Inception ResNet* para extrair características em diferentes escalas espaciais. No entanto, a *Inception ResNet* V2 introduz melhorias na estrutura dos blocos, tais como a adição de convoluções separáveis em camadas intermediárias para reduzir o número de parâmetros e o consumo computacional. Além disso, a *Inception ResNet* V2 foi pré-treinada em grandes *datasets*, como o ImageNet, para aprender representações gerais de características.

A *Inception ResNet* V2 aprimora a estrutura dos blocos em relação à versão anterior. Ela introduz a utilização de convoluções separáveis que são uma combinação de convoluções espaciais e convoluções em canais separados. Essas convoluções separáveis são aplicadas em camadas intermediárias dos blocos, reduzindo o número de parâmetros e o consumo computacional, mantendo um desempenho eficiente. Além disso, a *Inception ResNet* V2 utiliza ativações lineares nas camadas de convolução 1x1 para promover a independência das ativações e facilitar o treino.

Outra diferença significativa é a introdução dos chamados "atalhos"(shortcuts) na *Inception ResNet* V2. Esses atalhos são ligações diretas que saltam um ou mais blocos, permitindo que as informações fluam em caminhos alternativos dentro da rede. Essa técnica é inspirada na ideia de resíduos introduzida pela *ResNet*, onde os atalhos ajudam a mitigar os desafios de treinar redes neuronais profundas, permitindo um fluxo de informações mais direto e preservando gradientes.

2.4 ROS/ROS2

O ROS é um *framework* de software de código aberto muito utilizado para o desenvolvimento de sistemas robóticos. Foi introduzido pela *Willow Garage* em 2007 [29] e tem como objetivo fornecer uma plataforma flexível e robusta para a criação de robôs mais avançados e eficientes.

O ROS segue uma abordagem modular e distribuída, permitindo que sejam construídos sistemas robóticos complexos com mais facilidade e a sua arquitetura é baseada em nós (*nodes*) independentes que podem comunicar entre si através de mensagens, serviços e tópicos. Isso promove a reutilização de código e a modularidade, permitindo que diferentes partes de um sistema sejam desenvolvidas e testadas separadamente. Uma das principais vantagens do ROS é a grande variedade de bibliotecas e ferramentas, uma vez que o ROS fornece uma coleção abrangente de pacotes que abordam diversas funcionalidades necessárias no desenvolvimento de robôs, como o controlo de *hardware*, perceção sensorial, planeamento de movimento e interação com o ambiente. Além disso, o sistema de gestão de pacotes do ROS facilita a instalação e partilha de código.

No entanto, devido à necessidade de melhorias e de outros recursos, em 2015, o ROS2 foi projetado para dar resposta a sistemas robóticos mais complexos e de maior escala. Uma das principais melhorias do ROS2 está na arquitetura e na comunicação. Enquanto o ROS original usava um sistema de transporte específico, o ROS2 adota um *middleware* de comunicação chamado *Data Distribution Service (DDS)*. O DDS oferece comunicação em tempo real e suporte para troca de mensagens assíncrona, permitindo que os componentes do sistema robótico comuniquem entre si de maneira eficiente e confiável. Para além disto, o ROS2 introduziu melhorias relativamente às linguagens de programação suportada, como por exemplo C++, Python, Java e JavaScript, proporcionando flexibilidade aos programadores.

Outra melhoria importante do ROS2 é a sua arquitetura flexível em camadas (*layers*), permitindo que diferentes implementações de funcionalidades, como *middleware* de comunicação e drivers de *hardware*, sejam personalizadas e substituídas de forma modular. Isso proporciona maior flexibilidade e adaptabilidade em diferentes contextos de aplicação [30]. Além disto, outra ferramenta de ROS2 é o RViz que consiste num poderoso visualizador 3D que permite a representação gráfica do ambiente, dos sensores e do estado do robô, facilitando o planeamento de movimento e a interação com o ambiente.

2.4.1 Tipos de Comunicação no ROS

A comunicação é um aspeto fundamental no desenvolvimento de sistemas robóticos. Ambos os *frameworks* suportam diferentes métodos de comunicação, nomeadamente modelo de publicação e assinatura (*publisher e subscriber*), serviços e ações (ver figura 16).

No modelo de publicação e assinatura, os componentes do sistema são divididos em dois papéis principais: publicadores (*publishers*) e assinantes (*subscribers*). Os *publishers* são responsáveis por enviar mensagens contendo informações sobre um determinado tópico. Os *subscribers*, por sua vez, inscrevem-se para receber as mensagens relacionadas a um ou mais tópicos específicos. Isso promove

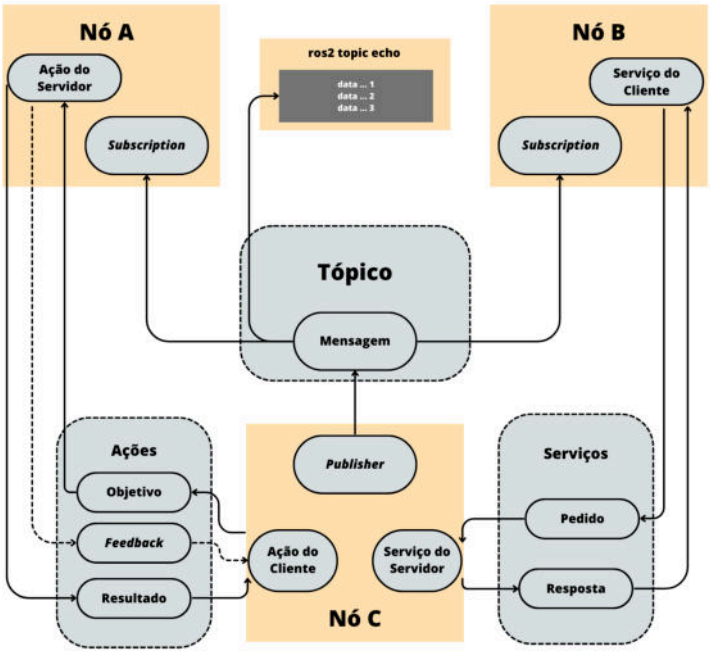


Figura 16: Comunicações em ROS2

a modularidade e a flexibilidade do sistema, já que os componentes não precisam estar cientes uns dos outros para se comunicarem. O modelo de publicação e assinatura em ROS2 é realizado por meio do DDS (Data Distribution Service) como mencionado anteriormente. O DDS fornece um *middleware* de comunicação que permite a troca de mensagens entre os nós. Os nós podem publicar mensagens num tópico DDS, e os nós interessados podem inscrever-se para receber essas mensagens.

A comunicação através de serviços é usada quando um nó precisa de solicitar a execução de uma tarefa específica a outro nó e aguardar a resposta. Nesse modelo, o nó que oferece o serviço é chamado de provedor (*service provider*) e o nó que solicita o serviço é chamado de cliente (*client*).

No ROS, os serviços são definidos usando o conceito de *Service Description Language* (SDL). O provedor disponibiliza um serviço com uma estrutura de mensagem específica, e o cliente envia uma solicitação para o provedor, que processa a solicitação e retorna uma resposta. No ROS2, os serviços são definidos usando a IDL (*Interface Definition Language*), permitindo que os nós se comuniquem de forma síncrona através de chamadas de serviço. A comunicação por serviços é adequada para operações que exigem uma resposta específica do provedor e quando é necessário estabelecer um fluxo de comunicação bidirecional entre os nós.

A comunicação através de ações é semelhante à comunicação por serviços, mas permite que ações mais complexas sejam executadas de forma assíncrona. Nesse modelo, o nó cliente envia uma solicitação de ação para o provedor, que pode ser uma tarefa que leva algum tempo para ser concluída. O provedor executa a ação e pode enviar atualizações periódicas ao cliente durante o processo. No ROS, a comunicação por ações é baseada num protocolo cliente-servidor. O cliente envia um objetivo (*goal*) ao provedor

e pode receber *feedback* sobre o progresso da ação em andamento. Quando a ação é concluída, o provedor envia um resultado de volta ao cliente. No ROS2, o modelo de ação foi redesenhado para melhorar a fiabilidade e a flexibilidade. A comunicação por ações é baseada em tópicos especiais de ação, que permitem o envio de metas, *feedbacks* e resultados entre os nós envolvidos. A comunicação por ações é adequada para tarefas que requerem interações complexas, *feedback* em tempo real e monitorização de progresso. Por exemplo, um robô que precisa de navegar até um local específico pode usar ações para enviar metas de navegação ao provedor e receber *feedback* contínuo sobre o progresso da navegação.

2.5 Deteção e Reconhecimento Facial

O reconhecimento facial é uma técnica de processamento de imagem que permite identificar ou verificar a identidade de uma pessoa a partir de uma imagem ou vídeo da sua face. Isso é feito por meio de um processo que inclui a deteção de faces na imagem, a extração de características faciais únicas da face e a comparação dessas características com uma base de dados de referência.

2.5.1 Deteção facial

A deteção facial é uma etapa importante do processo de reconhecimento facial, pois envolve a localização das faces presentes na imagem. Isso pode ser feito de diversas maneiras, nomeadamente utilizando redes neuronais profundas e algoritmos de deteção, tais como CNNs.

2.5.1.1 *MediaPipe*

MediaPipe é um conjunto de ferramentas de código aberto desenvolvido pela Google para construção de aplicações de processamento de imagem e áudio em tempo real. Ele inclui uma variedade de componentes pré-construídos, como modelos de redes neuronais treinadas, que podem ser facilmente integrados em aplicações para realizar tarefas comuns de processamento de imagem como a deteção facial.

A deteção facial com *MediaPipe* é realizada por um componente chamado *Face Detection Calculator*, que utiliza uma rede neuronal treinada para detetar e localizar rostos em imagens em tempo real. A rede foi treinada para detetar rostos em diferentes orientações, tamanhos e condições de iluminação e pode ser facilmente integrada em aplicações de processamento de imagem através de uma API simples.

Relativamente ao conceito por trás de *MediaPipe* tendo em conta o artigo [31] este consiste em três elementos principais, um sistema para inferência a partir de dados sensoriais (imagem), um conjunto de ferramentas para avaliação do desempenho e também componentes reutilizáveis para inferência e processamento.

2.5.1.1.1 Arquitetura do *MediaPipe*

O *MediaPipe* permite que um programador crie um protótipo de um fluxo de forma incremental. Um fluxo de visão é definido como um gráfico direcionado de componentes, em que cada componente é um nó (normalmente conhecido como "Calculadora"). No gráfico, as calculadoras estão ligadas por "*Streams*"de dados. Cada fluxo representa uma série temporal de "Pacotes"de dados. Em conjunto, as calculadoras e os fluxos definem um gráfico de fluxo de dados, sendo que os pacotes que fluem através do gráfico são agrupados pelos seus carimbos de data/hora dentro da série temporal. Cada fluxo de entrada mantém a sua própria fila para permitir que o nó recetor processe os pacotes ao seu próprio ritmo. O fluxo pode ser refinado de forma incremental, inserindo ou substituindo calculadoras em qualquer parte do gráfico e os programadores podem também definir calculadoras personalizadas. Enquanto o gráfico executa calculadoras em paralelo, cada calculadora executa no máximo um processo (*thread*) de cada vez. Esta restrição garante que as calculadoras personalizadas podem ser definidas sem conhecimentos profundos em programação *multithread*.

2.5.1.2 MTCNN

MTCNN é um método de deteção facial que utiliza redes neuronais para detetar e localizar rostos em imagens, desenvolvido por *Kaipeng Zhang, Zhanpeng Zhang e Ying Wu* em 2016 [32], e é amplamente utilizado em aplicações de visão por computador devido à sua precisão e velocidade.

O MTCNN consiste em três redes neuronais treinadas em forma de cascata: uma rede para detetar regiões candidatas a rostos (P-Net), uma rede para refinar a localização dessas regiões (R-Net) e uma rede para realizar a deteção final dos rostos (O-Net). Cada rede é treinada para realizar uma tarefa específica e a saída da rede anterior é utilizada como entrada da próxima.

Antes das imagens serem utilizadas em qualquer uma das três fases, a imagem é redimensionada em diferentes escalas num formato de pirâmide de imagens como na figura 17.

2.5.1.2.1 Fase 1: P-Net

Inicialmente é usada uma rede totalmente convolucional (FCN). A diferença entre uma CNN e uma FCN é que uma rede totalmente convolucional, que não usa uma camada densa como parte da arquitetura. Esta rede proposta é utilizada para obter janelas candidatas e os respetivos vetores de regressão de caixa de delimitação. A regressão de caixas delimitadoras é uma técnica popular para prever a localização de caixas quando o objetivo é detetar um objeto de alguma classe pré-definida, neste caso, faces. Após a obtenção dos vetores de caixa de delimitação é feito algum refinamento para combinar regiões sobrepostas. O resultado final desta fase consiste no conjunto de janelas de candidatos após o refinamento, para reduzir o volume de candidatos, figura 18.



Figura 17: Pirâmide de imagens

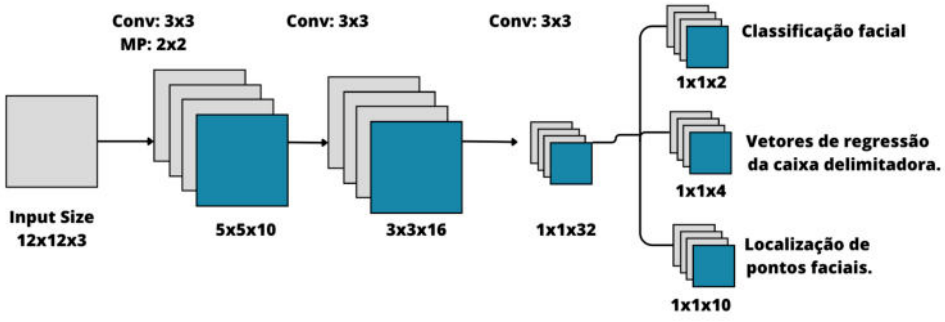


Figura 18: Fase P-Net

2.5.1.2.2 Fase 2: R-Net

Todos os candidatos da P-Net são alimentados na R-Net, figura 19. Nesta fase já é utilizada uma CNN e não uma FCN como no caso anterior, pois há uma camada densa no último estágio da arquitetura de rede.

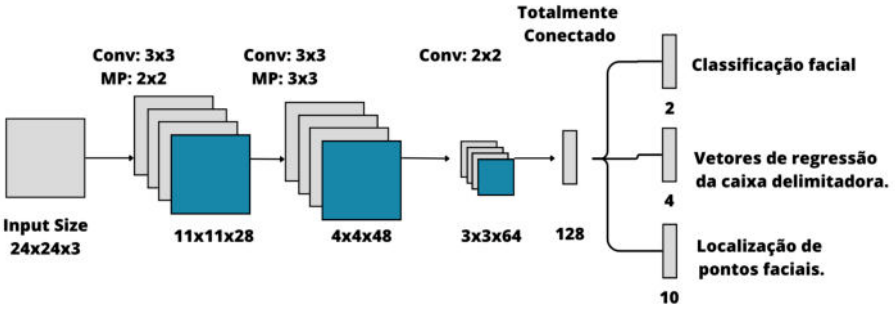


Figura 19: Fase R-Net

O R-Net reduz ainda mais o número de candidatos, realiza a calibração com regressão de caixa delimitadora e emprega supressão não máxima, Non maximum suppression (NMS), para fundir candidatos sobrepostos. Esta técnica é utilizada quando existem várias propostas do mesmo objeto, neste contexto um rosto, escolhendo o candidato com a melhor perspetiva e filtrando os restantes. A pontuação dos candidatos é calculada pela métrica Intersection over Union (IoU), utilizada para quantificar a percentagem de sobreposição entre duas previsões de caixas delimitadoras. Para calcular o IoU, são precisa duas duas regiões delimitadas, uma região "A"e outra região "B", e o IoU é definido como a razão entre a área da interseção dessas duas regiões e a área da união dessas duas regiões como descrito na figura 20.

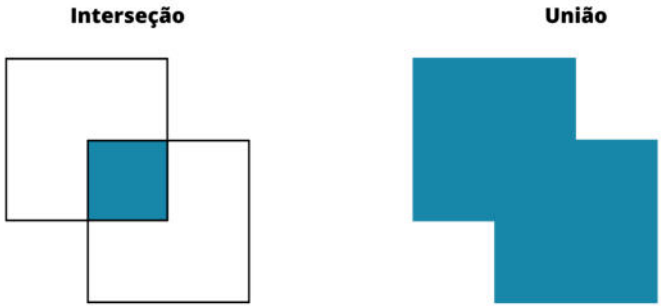


Figura 20: Interseção e união entre duas caixas delimitadoras

Na figura 21 está presente uma comparação da performance para diferentes configurações de IoU.

2.5.1.2.3 Fase 3: O-Net

Esta fase é semelhante ao R-Net, mas a O-Net tem como objetivo descrever o rosto com mais detalhes e produzir as posições dos cinco marcos faciais (ver figura 22). Esta estrutura produz três resultados: classificação binária de rostos, vetor de regressão de caixa delimitadora de 4 elementos e vetor de localização

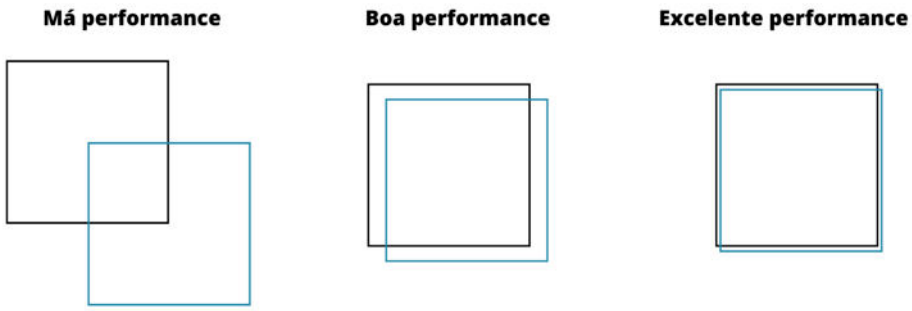


Figura 21: IoU - Comparação de performance

de pontos de referência faciais de 10 elementos.

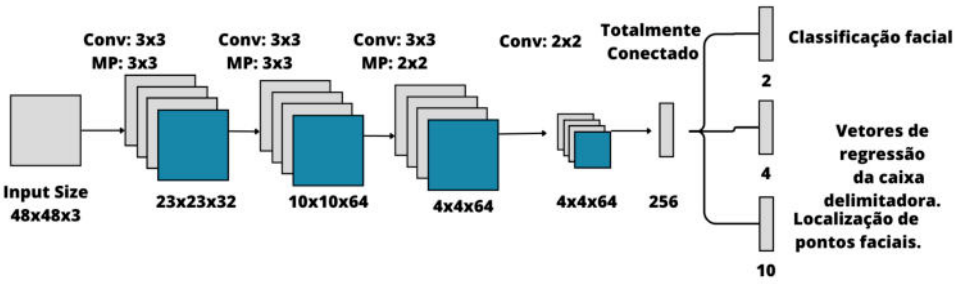


Figura 22: Fase O-Net

2.5.1.3 Metodologia do Treino de MTCNN

Classificação facial é um problema de classificação binária, que normalmente recorre ao *cross-entropy loss*, onde o principal objetivo é verificar se uma determinada imagem tem efetivamente uma face ou não. Isto é dado pela equação 2.5, onde: - ρ_i é a probabilidade que indica se a rede é um rosto ou não - y^{det} é o rótulo com a resposta.

$$L_i^{det} = -(y_i^{det} log(\rho_i) + (1 - y_i^{det})(1 - log(\rho_i))) \tag{2.5}$$

Regressão da caixa delimitadora e NMS foram utilizados para analisar a diferença do valor previsto com o valor real através de perda euclidiana, equação 2.6.

$$L_i^{box} = ||(\hat{y}_i^{box} - y_i^{box})||_2^2 \tag{2.6}$$

Onde: - $\hat{y}_i^{box}$ corresponde ao objetivo obtido da rede - y_i^{box} é a coordenada que corresponde à verdade

Assim como a regressão da caixa limitadora, para a localização de pontos faciais, a métrica utilizada é a distância euclidiana, equação 2.7.

$$L_i^{landmark} = ||(\hat{y}_i^{landmark} - y_i^{landmark})||_2^2 \tag{2.7}$$

2.5.2 Imagens de Profundidade na Deteção Facial

A deteção facial é uma técnica fundamental em várias aplicações nomeadamente em sistemas de reconhecimento facial. No entanto outros métodos tradicionais como o uso de cascatas de Haar, MediaPipe e MTCNN, podem ser enganados por ataques de *spoofing*. *Spoofing* pode ser descrito como ataques de apresentação que consiste numa técnica onde alguém tenta enganar sistemas de reconhecimento facial utilizando fotografias, máscaras, vídeos ou outros meios de apresentação não genuínos. Esses ataques representam uma ameaça significativa para a segurança e fiabilidade dos sistemas baseados em reconhecimento facial.

Para lidar com esse desafio, torna-se necessário integrar a componente de imagens de profundidade na deteção facial. A importância de realizar a verificação utilizando imagens de profundidade reside no facto de que essas imagens fornecem informação tridimensional da face, permitindo a deteção de características que são exclusivas de um rosto real e dificilmente replicáveis por apresentações falsas. A análise da geometria facial tridimensional pode identificar diferenças pequenas entre uma face real e uma representação falsa, como a falta de profundidade ou detalhes tridimensionais ausentes nas fotografias. Têm surgido abordagens promissoras que exploram o poder de aprendizagem das redes neuronais convolucionais e dos algoritmos de *Random Forest*.

Uma das vantagens de utilizar redes neuronais convolucionais é a capacidade de aprender padrões complexos e discriminativos diretamente dos dados de imagem. Por exemplo, os autores *D. Wen, H. Han e A. Jaino* [33] propõe uma abordagem que utiliza uma CNN para analisar distorções de imagens causadas por ataques de *spoofing*, como máscaras ou fotografias. A rede é treinada para classificar as imagens como genuínas ou falsas, com base nas características aprendidas durante o treino.

Outra técnica que tem sido aplicada com sucesso na deteção de *spoofing* facial é a *Random Forest*. Tal como mencionado anteriormente *Random Forest* é um algoritmo de aprendizagem supervisionada que combina múltiplas árvores de decisão para realizar classificação. No contexto da deteção de *spoofing*, a *Random Forest* tem sido utilizada para identificar padrões discriminativos extraídos de imagens de profundidade. Por exemplo, *M. Dindorkar et al.* [34] propõem um método de deteção de *spoofing* facial aprimorado que utiliza um classificador *Random Forest* combinado com a fusão de características de luminância e crominância. A extração de características é realizada nas imagens faciais e as características resultantes são combinadas e alimentadas no classificador *Random Forest* para realizar a classificação entre faces genuínas e ataques de *spoofing*. O estudo demonstra uma melhoria na precisão da deteção de *spoofing* em comparação com abordagens anteriores.

Além disso, abordagens combinadas também têm sido exploradas, aproveitando os pontos fortes tanto das redes neuronais convolucionais como do algoritmo *Random Forest.C.S. Fahn, C.P. Lee e M.L. Wu* [35] abordam a deteção de *spoofing* facial e exploram o uso de algoritmos de ML, como *Random Forests* e redes neuronais convolucionais (CNNs), para melhorar a precisão da deteção de ataques de *spoofing*.

Em resumo, a deteção de *spoofing* facial tem saído beneficiada com o uso de redes neuronais convolucionais e algoritmos de *Random Forest*. A utilização destas redes permite a extração automática de características discriminativas diretamente dos dados de imagem, enquanto a *Random Forest* aproveita a capacidade de combinação de múltiplas árvores de decisão para a classificação. Essas abordagens têm sido aplicadas em diferentes cenários como análise de distorção de imagem, características de profundidade e combinação de modalidades. Os resultados promissores obtidos por estas abordagens indicam um avanço significativo na deteção de ataques de *spoofing* facial e contribuem para fortalecer a segurança dos sistemas de reconhecimento facial.

2.5.3 Reconhecimento facial

Dando seguimento, ao tópico abordado acima, após as faces serem detetadas na imagem, são identificadas características faciais e extraídos pontos de referência, ou características, de uma imagem do rosto do humano. De acordo com *W. Zhao et al.* [36], há duas abordagens principais para o problema da extração de características. O primeiro baseia-se na extração geométrica e de características estruturais faciais como as formas dos olhos, nariz, boca e distância entre estes. Estes métodos não são afetados por informações irrelevantes nas imagens, mas são sensíveis à imprevisibilidade da aparência facial e das condições ambientais. No segundo método que é a abordagem holística, as características de toda a imagem são extraídas. Uma vez que as características são globais em toda a imagem, elementos irrelevantes como o fundo e o cabelo podem afetar os vetores das características e causar resultados do reconhecimento errados.

2.5.3.1 Trabalhos Relacionados

As primeiras pesquisas sobre reconhecimento facial tiveram início em 1964, quando *Woodrow Bledsoe*, matemático e cientista da computação, criou uma metodologia para ensinar um computador a reconhecer até 10 faces [37].

Com o passar do tempo, o reconhecimento facial sendo um tipo de classificação multi-classe beneficiou muito de técnicas de DL, tendo como base as arquiteturas CNN. Em [38] os autores discutem a evolução das arquiteturas de rede e as funções de perda utilizadas no treino de redes neuronais para reconhecimento facial. As funções de perda são categorizadas em perdas baseadas em distâncias euclidianas, perdas baseadas em margens de ângulo/cosseno, perdas de *Softmax* e as suas variações. A função de perda *Softmax* é usada para classificar a entrada em várias classes. No reconhecimento facial ela é usada para classificar uma imagem de um rosto num conjunto de classes, cada uma representando uma pessoa diferente. A perda de *Softmax* usa a função exponencial para converter as saídas da rede neuronal em probabilidades e em seguida calcula a *cross-entropy* entre as probabilidades previstas e as classes verdadeiras. A *cross-entropy* é uma medida da diferença entre as distribuições de probabilidade previstas e as classes verdadeiras. O objetivo da perda de *Softmax* é minimizar a *cross-entropy* durante o treino, ajustando os pesos da rede neuronal para melhorar o desempenho na tarefa de classificação.

Na perda baseada em distância Euclidiana as características extraídas dos rostos são normalizadas e as distâncias Euclidianas entre elas são calculadas, por outras palavras, perda é definida como a diferença entre as distâncias Euclidianas entre os rostos correspondentes e uma margem pré-definida. O objetivo da perda é minimizar a distância entre os rostos correspondentes e maximizar a distância entre rostos não correspondentes. A perda contrastiva é uma função de perda usada num conjunto com a perda baseada em distância Euclidiana ajuda a melhorar a separabilidade das características extraídas dos rostos, garantindo que as características de rostos semelhantes estejam mais próximas entre si e que as características de rostos diferentes estejam mais distantes.

As perdas baseadas em margens de ângulo/cosseno são semelhantes à perda baseada em distância Euclidiana e à perda contrastiva, mas diferem na maneira como calculam a perda pois baseiam-se na geometria do espaço vetorial em que as características dos rostos residem. Em vez de calcular a distância Euclidiana, essas perdas medem a diferença angular ou a diferença de cosseno entre as características dos rostos.

A perda tripla *(Triplet Loss)* é uma técnica de aprendizagem por pares que visa aprender uma representação de características discriminativas de rostos, garantindo que rostos semelhantes estejam próximos no espaço de características e rostos diferentes estejam distantes. A técnica usa três imagens: a imagem âncora, a imagem positiva e a imagem negativa. A imagem âncora é uma imagem de um rosto específico, a imagem positiva é outra imagem da mesma pessoa e a imagem negativa é uma imagem de uma pessoa diferente. O objetivo é aprender uma representação de características que garanta que a distância entre as características do rosto âncora e do rosto positivo seja menor do que a distância entre as características do rosto âncora e do rosto negativo, por uma margem pré-definida. A vantagem da perda tripla é que ela aprende uma representação de características discriminativas de rostos, que podem ser usadas para identificar indivíduos em imagens.

Contudo, concluiu-se que a perda contrastiva e a perda tripla ocasionalmente encontram instabilidade de formação devido a terem de selecionar amostras de formação. A perda do centro e as suas variantes são boas escolhas para reduzir esta instabilidade uma vez que é usada a função de perda de *Softmax*, mas ao invés de serem ajustados os pesos da rede neuronal para minimizar apenas a perda de *Softmax*, ela adiciona outra perda, chamada de *center loss*, que minimiza a distância entre as características do rosto e o centro do *cluster* correspondente. O centro do *cluster* é uma média móvel das características de todos os exemplos de treino pertencentes a um determinado rótulo.

Em [38] é demonstrado que as perdas baseadas em margens de ângulo/cosseno, obtém os melhores resultados com uma boa base de dados mas é vulnerável ao ruído e torna-se pior do que *center loss* e *Softmax* na região de alto ruído.

Posto isto, as técnicas de deteção e reconhecimento facial baseadas em DL atingiram desempenhos bons em comparação com os métodos tradicionais e uma contribuição chave foi a criação de grandes *datasets*, uma vez que a existência de bases de dados suficientemente grande é um dos principais requisitos de uma rede neuronal profunda.

No inicio dos trabalhos de reconhecimento facial eram utilizadas bases de dados privadas como é o

exemplo da base de dados *FaceNet* [39] que foi treinada com 200 milhões de imagens pertencentes a 3 milhões de classes. Hoje em dia existem para além desta, outras bases de dados públicas que são tidas como referência, nomeadamente: *VGGFace2* [40] onde o *dataset* contém 3.31 milhões de imagens de 9131 pessoas (uma média de 362,6 imagens por pessoa) e *CASIA-WebFace* que possui 500.000 imagens de 10.000 pessoas [41].

Capítulo 3

Métodos e Metodologias

Neste capítulo são apresentadas as metodologias principais do sistema onde inicialmente são abordadas as ferramentas utilizadas neste projeto bem como algumas especificações que formam o sistema em estudo. É também explicado o *hardware* utilizado e necessário neste projeto nomeadamente o robô CHARMIE e a câmara, bem como o *hardware* desenvolvido a par do desenvolvimento desta dissertação. São então explicadas e analisadas as duas abordagens, primeiramente a abordagem com redes neuronais para um ambiente social, que se traduz para os objetivos do CHARMIE num ambiente real, seja ele doméstico ou de saúde e onde se pretende que o robô seja capaz de se relacionar com os seus utilizadores e cumprir as tarefas pedidas. A segunda abordagem foi desenvolvida para um contexto competitivo, que consistiu na participação na liga *RoboCup@Home* no *RoboCup* 2023. São também descritas as provas em que participou.

3.1 Especificações do Sistema e Ferramentas

O computador utilizado neste projeto foi um Lenovo IdeaPad 5 Pro 14ITL6. Este computador contém 4 *cores* Intel Core e um processador i7-1165G7 @ 2.80GHz 2.80 GHz. Possui também duas placas gráficas, usando a interna (Intel Iris Xe) nos recursos básicos de processamento e uma gráfica Nvidia (GeForce MX 450) dedicada para a execução de processos mais exigentes nomeadamente em processamento de imagem. A imagem proveniente da câmara do computador tem uma resolução de 1280x720 *pixels* e 30 frames por segundo (fps). Foi utilizado o *Windows* 11 para desenvolver os diferentes módulos desta dissertação foi utilizado o *Windows* 11, e o *Ubuntu* 20.04 para aplicar o trabalho desenvolvido no robô. A câmara presente no robô CHARMIE é a câmara de profundidade *Intel Depth RealSense* D455. A linguagem utilizada neste projeto foi o *Python* e os IDE foram o *PyCharm* e o *Visual Studio Code*, o primeiro no desenvolvimento dos módulos no *Windows* e o segundo no *Ubuntu*. Os principais módulos

utilizados para a captura e processamento de imagens foram o*pyrealsense2* para ser possível trabalhar com a câmara Intel Realsente Depth D455 e a biblioteca *OpenCV*. Para manipulação de matrizes, a principal biblioteca utilizada foi a *Numpy*. Todos os dados utilizados foram armazenados localmente, incluindo imagens, modelos treinados e gráficos de visualização de dados. Para aceder a esses diretórios e caminhos de ficheiros, foi utilizado o módulo *OS*. O módulo *Scikit-learn* foi também utilizado e é uma biblioteca de ML que suporta aprendizagem supervisionada e não supervisionada. Fornece uma gama de ferramentas para extração de dados, pré-processamento de dados e algoritmos de ML. O módulo *Pickle* foi também utilizado para guardar e carregar estruturas de dados complexas. Para visualização de dados foi utilizado o módulo *Matplotlib* e *Seaborn* que consiste numa biblioteca construída sobre o *Matplotlib*. Estas servem para gerar gráficos e tabelas sendo que o segundo permite criar gráficos de forma mais atraente e informativa. Por fim, o módulo *Keras* é uma API concebida para desenvolver e treinar modelos de DL, que pode ser executado em várias estruturas de ML, incluindo o *TensorFlow*. As versões utilizadas para cada módulo estão mencionadas na tabela 1. Para além disto, de forma a ser possível incorporar os diversos módulos no robô, foi utilizado o *ROS2* e a versão *foxy* do mesmo.

Módulo utilizado	Versão
Python	3.9
Pyrealsense2	2.54.1.5217
OpenCV	4.8.0.76
NumPy	1.24.3
Scikit-Learn	1.3
Matplotlib	3.7.2
Seaborn	0.12.2
Keras	2.13.1
Tensorflow	2.13

Tabela 1: Módulos utilizados e respetivas versões

Tendo sido desenvolvidas duas abordagens diferentes (abordagem para ambiente social e ambiente competitivo) esta secção do documento está dividida nessas duas partes, apesar de ambas abordarem o mesmo tema, deteção e reconhecimento facial em tarefas de longa duração. Na abordagem para um ambiente social o projeto está dividido em 5 fases: a captura e processamento de imagens, criação do *dataset*, desenvolvimento e treino do modelo, treino do classificador SVM e testes finais.

Na abordagem para um ambiente competitivo, o projeto está dividido em 4 fases: deteção facial, recolha e processamento de imagens, desenvolvimento do algoritmo de extração de características da pessoa e testes finais.

3.1.1 *Hardware*

Para o desenvolvimento da dissertação foi utilizado o robô CHARMIE e a câmara Intel® RealSense Depth™ D455. Contudo e uma vez que o CHARMIE é um robô em desenvolvimento, a par desta dissertação foi necessário desenvolver algumas peças em 3D, que apesar de não estar diretamente ligado com o tema da dissertação, foi uma contribuição para projeto. Nas próximas secções serão abordados e explicados em pormenor os elementos de *hardware* bem como o trabalho desenvolvido.

3.1.1.1 CHARMIE

O robô móvel CHARMIE tem um desenho inspirado no ser humano para desta forma melhorar a interação homem-robô, (ver figura 23). A base possui quatro rodas omnidirecionais colocadas a 90° para melhorar a sua capacidade de locomoção. Cada roda possui um sistema de suspensão independente para garantir o contacto contínuo com o chão, e evitar a perde de tração.

Para aumentar o espaço de trabalho de manipulação do robô, foi desenvolvido um tronco articulado com dois graus de liberdade. Este tronco permite que o robô se baixe num movimento linear e dobre o seu tronco para a frente. Também no tronco do robô está presente um LiDAR que tem um papel importante no mapeamento e deteção de obstáculos.

O objetivo planeado para o robô CHARMIE é possuir dois braços com seis graus de liberdade cada um, nomeadamente flexão do ombro abdução do ombro, rotação interna do ombro, rotação externa do ombro, flexão do cotovelo, elevação do cotovelo e flexão do pulso. Contudo os mesmos ainda estão em desenvolvimento e não puderam ser utilizados nesta dissertação. A par disto existe também um tabuleiro na parte frontal do robô para servir de apoio sempre que for necessário o CHARMIE transportar algo.

O pescoço do robô possui dois graus de liberdade em série, controlados por motores, para permitir a sua flexão e rotação (um movimento equivale a dizer que sim com a cabeça e outro a dizer que não). O CHARMIE possui ainda uma câmara, um microfone, duas colunas de som e uma cara para que seja possível uma interação mais amigável com humanos.

Na parte traseira do robô existe um suporte para pousar o computador e é na parte inferior estão as baterias que alimentam o robô.

Tudo isto resulta num peso total do robô de 60 kg e uma altura que varia entre 128 cm e 142 cm. Para além do movimento para cima/para baixo, o robô também dobra o seu tronco para a frente, atingindo uma altura mínima de 92 cm. O seu tronco e braços pode passar por baixo de estruturas como mesas até 80 cm (ver figura 24, permitindo que o robô alcance objetos do solo que estão 60 cm afastados da plataforma do robô [4].



(a) (b)

Figura 23: Protótipo do robô CHARMIE utilizado no RoboCup@Home de 2023: a) Vista frontal b) Vista traseira

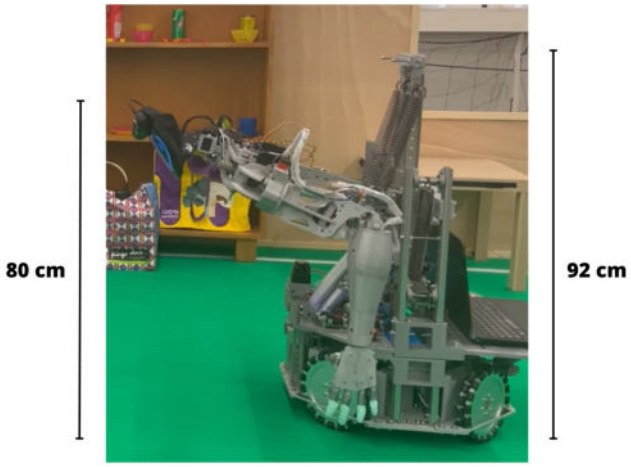


Figura 24: Posição em altura mínima possível

3.1.1.2 Câmara *Intel RealSense Depth D455*

A câmara Intel® RealSense Depth™ D455, (ver figura 25), é uma câmara de profundidade baseada em estéreo, equipada com um alcance longo e grande precisão [42].

A câmara D455 tem um alcance longo, uma precisão aprimorada como descrito abaixo e é de fácil colocação. A profundidade estéreo funciona numa ampla variedade de condições de iluminação.



Figura 25: Câmara Intel® RealSense Depth™ D455

- Resolução de imagem Red Green Blue (RGB) : 1280x720p - 30fps
- Resolução de imagem em profundidade : 1280x720p - até 90fps
- Distância mínima em profundidade : 52cm
- Distância máxima em profundidade : 6m
- *Depth Accuracy*: <2% a 4m
- RGB - Ângulo do campo de visão (Horizontal) : 90°
- RGB - Ângulo do campo de visão (Vertical) : 65°
- Profundidade - Ângulo do campo de visão (Horizontal) : 87°
- Profundidade - Ângulo do campo de visão (Vertical) : 58°

3.1.2 Desenvolvimento de Peças em 3D para o robô CHARMIE

Esta secção apresenta as peças em 3D que foram projetadas especificamente para o protótipo de robô desenvolvido. Este trabalho contribuiu para a construção do robô e é parte integrante do projeto CHARMIE. A escolha do *Fusion 360* como ferramenta de modelagem deveu-se à sua ampla gama de recursos e à sua capacidade de integração com outros softwares de engenharia, permitindo uma abordagem abrangente na conceção e construção do robô. Todos os desenhos referentes às peças desenvolvidas encontram-se no anexo III.

3.1.2.1 Caixa para Botão de Emergência

Tendo em conta os objetivos, este robô necessita de botões de emergência uma vez que estará próximo de pessoas e poderá ter algum comportamento de risco, que coloque a segurança em causa.

Este botão de emergência fará com que o robô desligue imediatamente todos os seus sistemas. O CHARMIE, tem três botões de emergência um do lado esquerdo, um do lado direito e outro remoto de forma a ser facilmente acessível em qualquer situação de urgência, independentemente da posição do robô. Foi necessário desenvolver uma caixa que, para além de suportar os botões, protege e guarda todas as suas ligações elétricas mantendo apenas à vista os respetivos botões. Esta peça encaixa no suporte de aço que está na parte traseira do robô e foi ajustada de forma a não ter qualquer folga. Esta peça é presa ao suporte em aço através de parafusos como é possível verificar na figura 26.

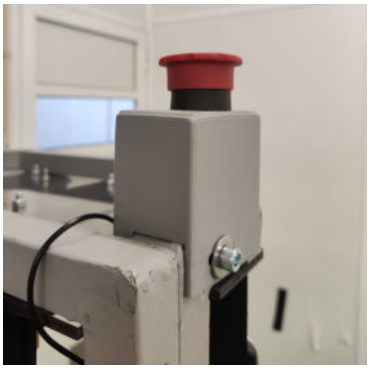


Figura 26: Caixa para o botão de emergência

3.1.2.2 Suporte para o comando

Para além dos botões de emergência existe também um comando que pode ser utilizado para desligar e/ou ligar o robô à distância. Foi desenvolvida uma peça para que o comando nunca se perder e para ter um acesso fácil e rápido. Na figura 27 é possível verificar a sua aplicação no robô.

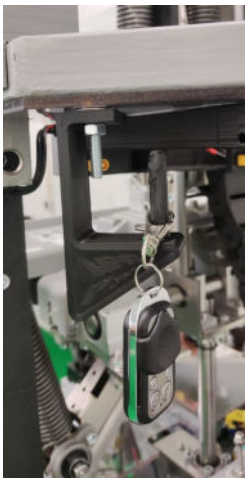


Figura 27: Suporte para o comando no robô CHARMIE

3.1.2.3 Calhas

Por motivos estéticos e para melhor organização da cablagem do robô, foram desenvolvidas calhas que levam os cabos da parte superior do robô para a inferior, e vice-versa. Estas calham permitem manipular o CHARMIE com maior segurança uma vez que as calhas e tampas têm os cantos arredondados.

Nas figuras 23 é possível ver no robô CHARMIE, onde foram colocadas as calhas e de que forma cumprem o seu objetivo. Na figura 28 é possível ver de forma mais aproximada uma das calhas com a respetiva tampa.



Figura 28: Calha e tampa no robô CHARMIE

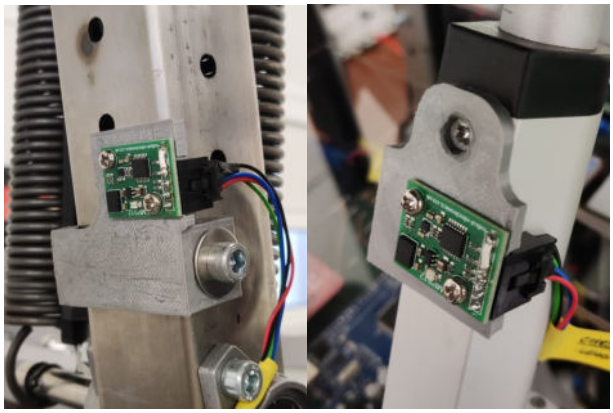
3.1.2.4 Suporte dos sensores TOF e bússolas CMPS 12

O robô tem a capacidade de se baixar e levantar bem como de se inclinar. Para um melhor controlo destes movimentos, inicialmente implementaram-se 2 sensores TOF (*Time of flight*), um em cada atuador linear de forma a saber o ângulo de inclinação e a distância do movimento. Por motivos práticos os TOF acabaram por ser substituídos por bússolas CMPS 12, pelo que foi então necessário desenvolver novos suportes para as novas peças, (ver figura 29).

3.1.2.5 Proteção frontal e traseira do robô

Como é possível ver na imagem 23(a), na frente do robô há uma proteção aplicada na base do robô composta por duas partes simétricas para proteção das placas eletrónicas com um segundo objetivo de esconder o interior do robô e não permitir o acesso a qualquer pessoa.

A proteção traseira possuiu mais especificações, uma vez que é nesta peça que se encontram os botões que controlam todos os módulos do robô. Possuiu ainda um espaço para o *display* que permite visualizar a tensão e corrente instantâneas.



(a) (b)

Figura 29: Suportes das bússolas CMPS 12: a) Atuador linear para subida/descida do tronco, b) Atuador linear para inclinação do tronco

Como se vê na figura 30, a proteção traseira é composta por 3 módulos: um à esquerda, um no centro e outro à direita. As proteções esquerda e direita possuem os botões acima descrito. A proteção central, contém o botão que liga o robô e o *display* acima mencionado.

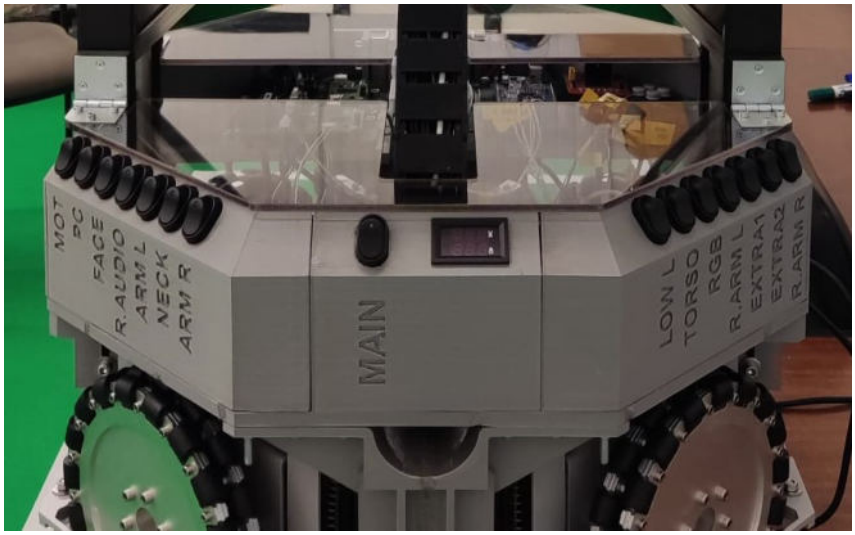


Figura 30: Proteção traseira no robô CHARMIE

3.1.2.6 Proteção dos fios das baterias

Tendo a necessidade de esconder toda a cablagem do robô por restrições da competição, foram desenvolvidas duas peças, para adaptar ao suportes das baterias. Na figura 31 é possível verificar a respetiva proteção e a invisibilidade da cablagem.

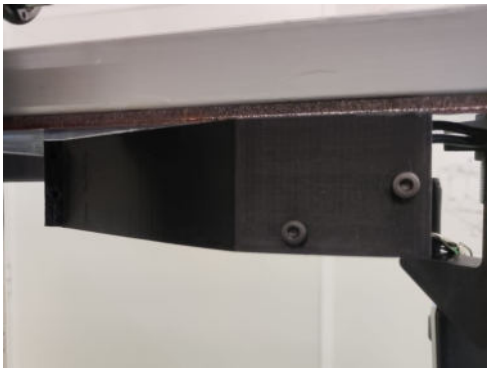


Figura 31: Proteção dos cabos que ligam as baterias ao robô

3.1.2.7 Ombros/Suporte do microfone

Uma vez que aquando da participação do CHARMIE no RoboCup@Home os braços ainda não tinham sido produzidos, foi decidido conceber peças que apesar da ausência dos braços, dessem ao tronco do robô um aspeto mais humano. Foram desenhados dois ombros sendo que o ombro direito foi ajustado para ser possível colocar nele o microfone do CHARMIE. Este microfone é usado como forma do CHARMIE ouvir comandos do utilizador pelo que este deve estar o mais visível e próximo da cabeça possível. Na figura 32 é possível ver o ombro e microfone aplicados à estrutura do robô.

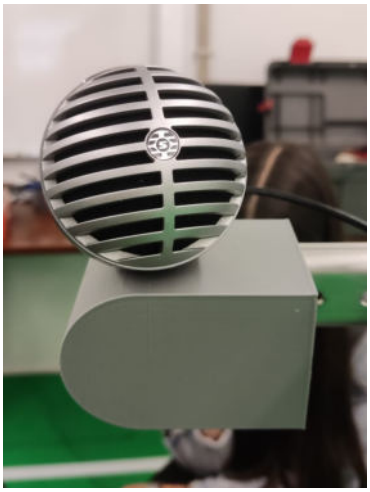


Figura 32: Ombro e suporte para microfone

3.2 Abordagem com Redes Neuronais para Ambiente Social

Considerando que o CHARMIE foi desenvolvido para tarefas em ambiente domésticos e de cuidados de saúde, a abordagem com redes neuronais para um ambiente social consiste numa solução para deteção e reconhecimento de utilizadores específicos em tarefas de longa duração. O robô deverá interpretar um utilizador como sendo o seu utilizador principal e independentemente das tarefas e ações

feitas, ser sempre capaz de reconhecer para além das restantes pessoas o seu utilizador principal. Para esta solução algumas questões devem ser previamente preparadas e serão analisadas nesta secção do documento. Na base deste sistema estão 3 partes fulcrais: deteção facial, extração das características faciais e classificação das mesmas, representado no fluxograma apresentado na figura 33.

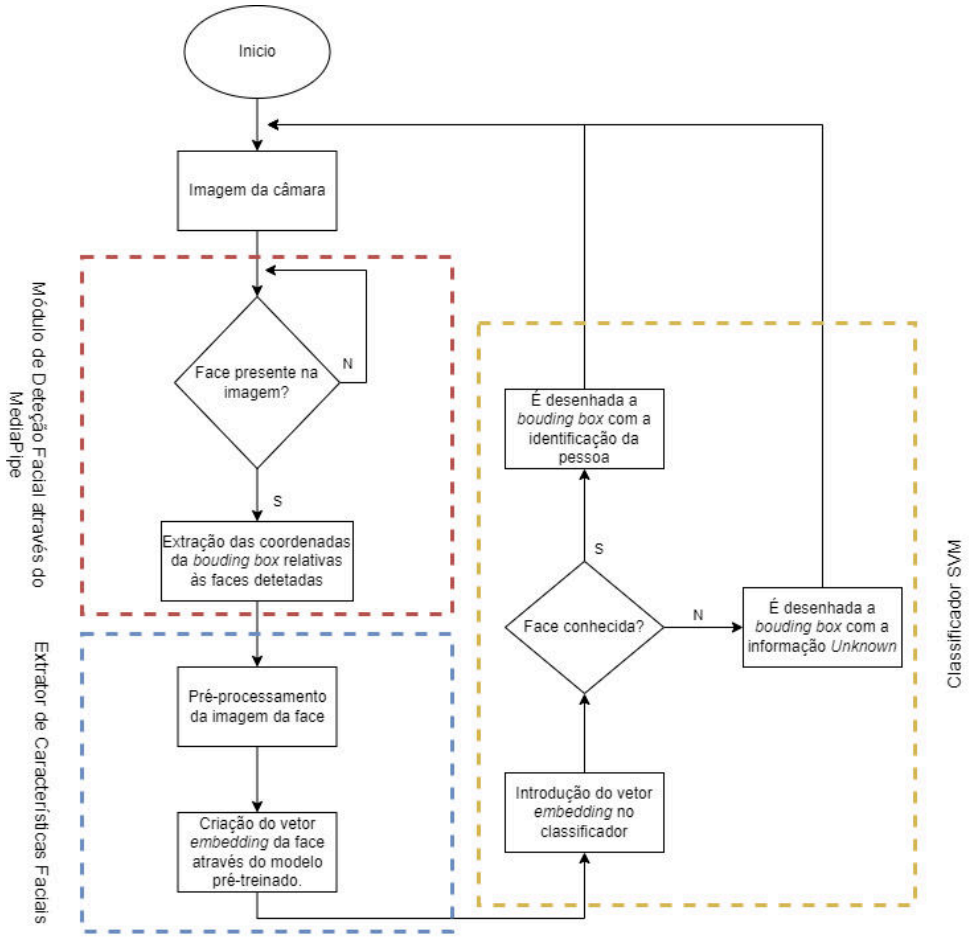


Figura 33: Fluxograma da abordagem social

Para a criação de uma solução de reconhecimento facial é necessário que o *dataset* seja feito com um maior controlo e especificidade, como por exemplo o número de fotos retiradas e as condições onde as mesmas são tiradas. Para além disto, são utilizados recursos de *ML*, sendo que para a construção das redes neuronais de reconhecimento facial e processo de treino foram utilizadas as bibliotecas *TensorFlow* juntamente com *Keras* [43]. Ambas são plataformas de código aberto que permitem a implementação de soluções de ML e contêm várias soluções integradas que permitem implementar redes neuronais profundas através da criação de camadas, especificar as dimensões dos filtros e outros parâmetros relevantes. Após o treino e teste das redes neuronais com as imagens recolhidas, são extraídas as características e posteriormente classificadas tendo em conta a identificação dos elementos presentes. De uma forma a melhorar o sistema de reconhecimento de longa duração são também analisadas outras características como por exemplo género, idade, etnia, altura.

3.2.1 Deteção Facial

Um ponto fulcral num problema de reconhecimento facial é a deteção facial e a criação de um *dataset* tal como mencionado anteriormente. Foram estudadas e analisadas três soluções, nomeadamente *Haar Cascade*, *MediaPipe* e MTCNN. Tendo em consideração o estudo feito, foram utilizadas e comparadas apenas as soluções *MediaPipe* e MTCNN devido à sua eficácia e robustez comparativamente à solução *Haar Cascade*.

3.2.1.1 *MediaPipe*

A biblioteca *MediaPipe* fornece uma solução para deteção facial, permitindo a localização precisa dos rostos na imagem capturada. Com base nessas informações, é recortada uma imagem apenas da face e a mesma é guardada num diretório pré-definido. Para a definição da função de deteção facial com recurso à biblioteca *MediaPipe* foi necessário ter em consideração o valor atribuído à variável correspondente ao valor mínimo de confiança, neste caso 0.5, para que a deteção de rosto seja considerada bem sucedida, como exposto em 3.1.

```
1 mp_face_detection = mp.solutions.face_detection.FaceDetection(
      min_detection_confidence=0.5)
2 rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
3 results = mp_face_detection.process(rgb_frame)
```

Listagem 3.1: Definição da função de deteção facial com MediaPipe

3.2.1.2 MTCNN

A MTCNN consiste num método de deteção facial que usa redes neuronais para detetar e localizar rostos em imagens, tal como referido em 2.5.1.2. Este método utiliza uma abordagem de cascata de redes neuronais convolucionais que trabalham em conjunto para detetar regiões faciais em múltiplas etapas. A principal característica do MTCNN é a sua capacidade de detetar faces em diferentes tamanhos e orientações, tornando-o robusto para várias aplicações.

Relativamente à definição da função de deteção facial associada a esta solução MTCNN (ver listagem 3.2), foram consideradas três variáveis, sendo a primeira o tamanho mínimo das faces (em *pixels*) que devem ser detetadas, a segunda os limiares utilizados no algoritmo NMS e a última corresponde ao fator escalar para o método das pirâmides.

```
1 mtcnn_face_detector = mtcnn.MTCNN(min_face_size=80,steps_threshold
      =[0.6,0.7,0.7], scale_factor=0.709)
```

Listagem 3.2: Definição da função de deteção facial com MTCNN

3.2.2 Imagens de profundidade

Para além da deteção facial é necessário verificar nas imagens de profundidade se se trata realmente de uma pessoa ou de uma tentativa de *spoofing* para garantir a segurança e fiabilidade da informação. Assim para além de uma análise às imagens RGB foram também desenvolvidas duas soluções para analisar as imagens de profundidade obtidas da câmara, nomeadamente uma CNN e um classificador *random forest* com características HOG tal como referido em 2.5.2. Para ambas as soluções são usadas imagens com dimensão 64x64 px. Este foi o tamanho escolhido tendo em conta os resultados obtidos para as diferentes dimensões testadas.

3.2.2.1 Classificador *Random Forest* com Características HOG

Relativamente à utilização de um classificador *random forest* com características HOG, o processo inclui o carregamento das imagens com pessoas e sem pessoas, a extração das características HOG de todas as imagens, a criação do *dataset* de treino e teste, o treino do classificador *random forest* e, por fim, a avaliação dos resultados obtidos do *dataset* de teste através de uma matriz de confusão. Na listagem 3.3, é possível ver a parte do código referente à extração das características das imagens e também o treino do classificador *random forest*.

```
1 hog_features = hog(img, orientations=9, pixels_per_cell=(8, 8),
      cells_per_block=(2, 2), visualize=False)
2 classifier = RandomForestClassifier(n_estimators=100, random_state=42)
3 classifier.fit(x_train, y_train)
```

Listagem 3.3: Extração de características HOG e criação e treino do classificador

3.2.2.2 Rede Neural Convolucional

Para além desta solução foi também utilizada uma rede neuronal convolucional sendo esta composta por camadas de convolução e *pooling*, seguidas por camadas densas. O processo inclui o carregamento das imagens com e sem pessoas, a criação das categorias, o redimensionamento e normalização das imagens, a criação do modelo CNN, a compilação e treino do modelo (visível na listagem 3.4), a avaliação do *dataset* de teste e a geração da matriz de confusão.

```
1 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['
      accuracy'])
2 model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(
      X_test, y_test))
```

Listagem 3.4: Treino do modelo

3.2.3 Criação do *Dataset*

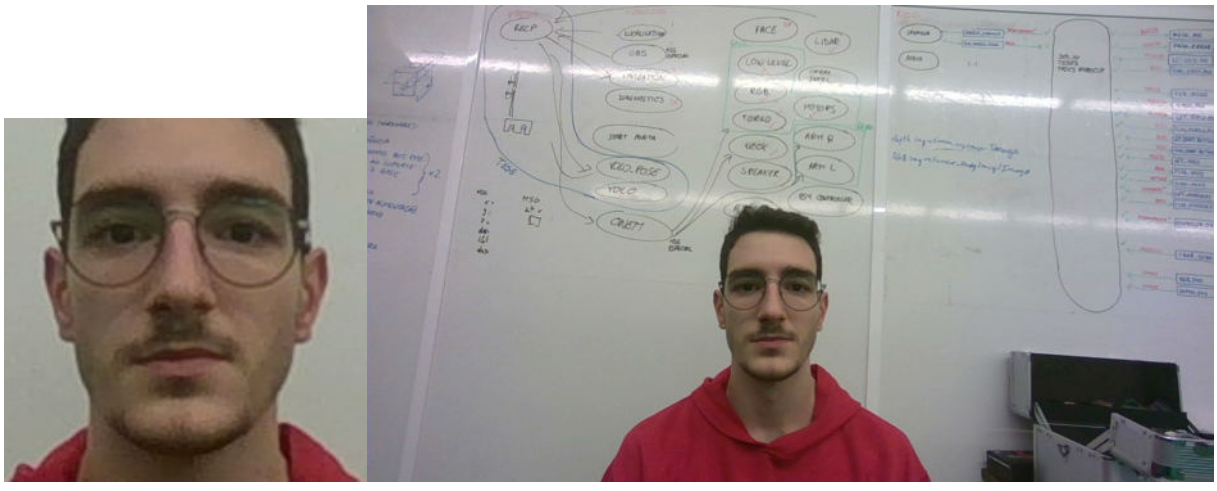
Uma das etapas fundamentais para o desenvolvimento de sistemas de reconhecimento facial é a criação de um *dataset* robusto e representativo, uma vez que a qualidade do *dataset* é essencial para treinar e avaliar algoritmos de reconhecimento facial. O *dataset* consiste num conjunto de imagens de faces que abrangem diversas variações de iluminação, expressões faciais, diferentes distâncias e ângulos de visão. Quanto mais abrangentes e diversificadas forem as imagens, melhor será a capacidade do sistema de reconhecimento facial em lidar com cenários do mundo real e melhorar a sua precisão.

Para responder aos problemas deste projeto foi necessário criar dois *datasets*, sendo que a descrição dos *datasets* e respetivas características está descrita mais abaixo.

Importar realçar que, para a criação dos *datasets* foram utilizadas imagens e identidades de alunos da Universidade do Minho sendo maioritariamente elementos do Laboratório de Automação e Robótica, tendo todos os elementos consentido o uso da sua identidade e imagens.

A criação do *dataset* para treinar o modelo é composta por um conjunto de passos: inicialmente é fornecido o nome do utilizador sendo toda a estrutura da base de dados organizada e dependente deste nome. São utilizados diretórios previamente definidos para guardar as imagens de treino, teste e validação.

São adquiridas e guardadas 1000 imagens de cada rosto, sendo estas armazenadas da seguinte forma. Para cada pessoa, são guardadas 1000 imagens RGB originais no diretório A, e as respetivas imagens originais de profundidade no diretório B. Seguindo as percentagens de 80% das imagens para treino, 10% para teste e 10% para validação, estas imagens são guardadas nos diretórios C,D e E, respetivamente. Importa realçar que as imagens constantes nestes três diretórios já foram processadas com o intuito de conter apenas a face da pessoa (ver figura 34.



(a) (b)

Figura 34: Imagens para a) treino (dimensão original 160x160 px) e b) teste (dimensão original 1280x720 px)

A distribuição de imagens entre os diretórios é uma prática comum para avaliar e otimizar o desempenho do modelo de reconhecimento facial. Nas imagens 35 e 36 é possível ver um exemplo onde estão estes diretórios criados e o nome das imagens guardadas.

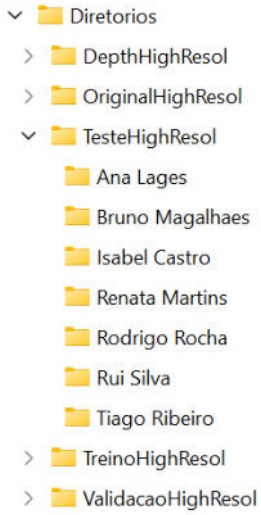


Figura 35: Diretórios relativos à criação do *dataset*

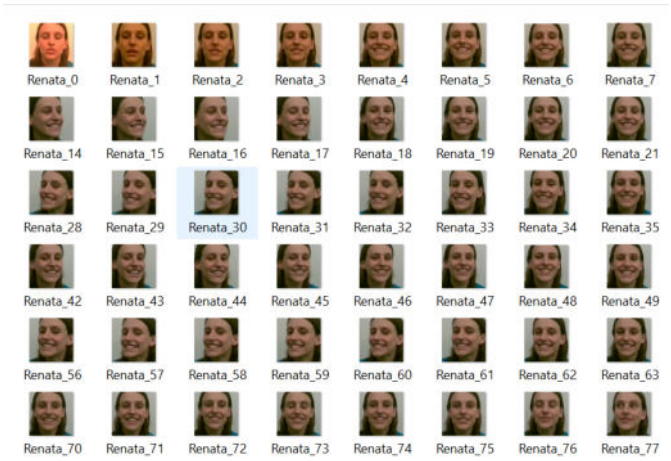
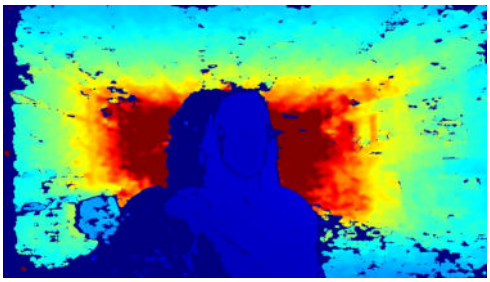


Figura 36: Exemplo de um diretório

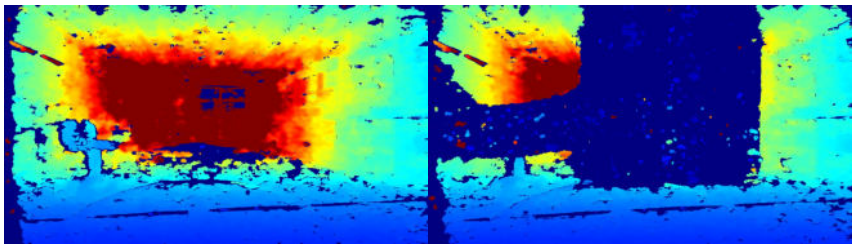
De salientar que existem alguns pormenores a ter em consideração nomeadamente a utilização de óculos, algum tipo de penteado, ou alguma outra característica dinâmica. É importante na criação do *dataset* ter estas variações em consideração para aquando a utilização do *dataset*, o mesmo esteja adequado para as diferentes condições possíveis, de forma a conseguir reconhecer uma pessoa independentemente se está de óculos ou não, ou se o penteado corresponde ao do *dataset*. Desta forma, o *dataset* será mais robusto e é uma forma de garantir o seu correto funcionamento.

Para evitar técnicas de *spoofing*, foi criado um outro *dataset* apenas com imagens de profundidade contendo 2000 imagens com uma pessoa em diversas poses (ver figura 37(a)) e outras 2000 imagens sem

qualquer pessoa real presente (ver figura 37(b)) mas com possíveis tentativas de *spoofing* nomeadamente a mostragem de telemóveis, *tablets*, molduras, etc (ver figura 37(c)).



(a)



(b) (c)

Figura 37: Exemplo de imagens de profundidade para treino

3.2.3.1 *Data Augmentation*

Quando se trabalha com DL um problema frequente é a falta de dados disponíveis para realizar o treino do modelo. Uma grande quantidade de dados é a chave para assegurar desempenhos bastante bons na previsão de modelos e é aqui que se insere o conceito de *data augmentation*.

Data augmentation é uma técnica usada em ML e visão por computador para gerar novos dados a partir dos dados de treino existentes através de uma série de transformações aplicadas às imagens. A ideia por trás de *data augmentation* é aumentar a quantidade de dados de treino disponíveis para o modelo, o que ajuda a reduzir o *overfitting* e melhora a capacidade do modelo de generalizar para novas imagens. Além disso *data augmentation* pode ajudar a tornar o modelo mais robusto a variações nas imagens, como diferenças de iluminação, ângulo ou escala.

A biblioteca *ImageDataGenerator* do *Keras* é uma das ferramentas mais comuns para aplicar *data augmentation*, permitindo que os utilizadores definam uma variedade de transformações que podem ser aplicadas durante o treino do modelo. Neste caso, foram utilizadas técnicas como rotação, brilho e zoom (ver listagem 3.5).

3.2.3.1.1 Rotação

A transformação *rotation_range* é uma técnica de *data augmentation* usada para rodar as imagens aleatoriamente durante o treino. A transformação roda as imagens num ângulo aleatório onde neste caso

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=20,
                                   brightness_range=[0.4,2.5],
                                   horizontal_flip=True
                                   )
```

Listagem 3.5: Código relativo à transformação de imagens

foi definido +/-20 graus como se pode ver na figura 38.

Ao treinar um modelo com imagens rodadas aleatoriamente, o modelo pode ser mais robusto e capaz de reconhecer independentemente da orientação. Por exemplo, se o modelo só for treinado com imagens de rostos voltados para a frente, pode não ser capaz de reconhecer rostos que estejam inclinados ou de lado e ao usar esta transformação, o modelo pode ser treinado com imagens de rostos em várias orientações.

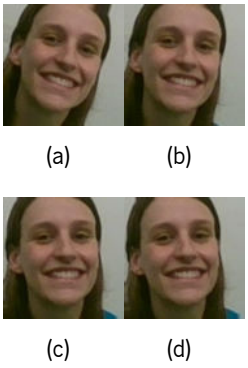


Figura 38: Transformações de rotação a) -16.33° b) -7.58° c) 2.33° d) 4.56°

3.2.3.1.2 Brilho

A transformação *brightness_range* altera o brilho das imagens durante o treino, ajustando o brilho dentro de uma escala, neste caso [0.4, 2.5], como por exemplo na figura 39. Os valores escolhidos tiveram como base abranger uma gama de imagens que pudessem aparecer num contexto real. Isto ajuda a tornar o modelo mais robusto em relação às variações de brilho.

3.2.3.1.3 Inversão Horizontal

Ao aplicar a transformação de inversão horizontal numa imagem esta é invertida em relação a um eixo vertical como se fosse colocada num espelho, como na figura 40. Se houver um objeto à direita da imagem original, ele será movido para a esquerda na imagem espelhada.



(a) (b)



(c) (d)

Figura 39: Transformações de rotação a) 0.4 b) 1.1 c) 1.8 d) 2.4



(a) (b)

Figura 40: Transformação horizontal a) original b) flip

3.2.4 Inception ResNet

As principais redes neuronais utilizadas para a extração de características de rostos foram o *Inception ResNet* versão 1 e 2, sendo que este modelo resulta de uma combinação de duas metodologias diferentes de DL, previamente analisadas em 2.3.6: ligações residuais introduzidas pela *ResNet* e os módulos *Inception* introduzidos com a *GoogleNet*. O modelo proposto foi treinado localmente utilizando métodos de aprendizagem supervisionada.

Nas redes neuronais tradicionais uma camada alimenta a seguinte com dados, enquanto que com as ligações residuais a informação é enviada diretamente de uma camada para outra mais profunda. Estes blocos melhoram dois domínios distintos: em primeiro lugar o tempo de treino, dado que as ligações residuais podem saltar camadas sem treino e em segundo lugar a perda de informação quando se efetua a descida do gradiente, uma vez que as redes mais profundas têm dificuldade em ser exatas sem se adaptarem excessivamente ou trabalharem em tarefas mais simples.

As implementações de ambas as versões desta arquitetura de rede neuronal foram utilizadas a partir

de [44] e a implementação da CNN é importada para o treino/inferência através de um *script python*.

3.2.4.1 Treino do Modelo

Para o treino do modelo são utilizados vários argumentos de entrada para vários fins como o carregamento do *dataset*, a modificação dos hiperparâmetros para controlar o processo de aprendizagem e a definição do ficheiro de armazenamento de informações.

3.2.4.1.1 Definição de Hiperparâmetros

De forma a escolher o melhor modelo para o problema em causa foram utilizadas diferentes definições de entrada de hiperparâmetros. Os hiperparâmetros são parâmetros cruciais que definem o comportamento e o desempenho de um modelo de DL e estes são definidos antes do processo de treino e podem influenciar significativamente a convergência, a generalização e a eficácia geral do modelo.

As dimensões da imagem são 160 *pixels* tanto em largura como em altura, e esta escolha deve-se à procura de um equilíbrio entre a eficiência computacional e a preservação das características visuais essenciais.

O tamanho do *batch*, o número de épocas e o *learning rate* foram utilizados de uma forma dinâmica para encontrar o melhor resultado possível e esta abordagem visa afinar a convergência do modelo e evitar o excesso de ajuste durante a otimização.

Foram introduzidas camadas personalizadas como camadas convolucionais, *max-pooling*, normalização de lotes, *dropout* e camadas densas, com o intuito de melhorar a capacidade de abstração de características do modelo, permitindo-lhe captar padrões intrínsecos nos dados. Camadas de *dropout* com taxas variáveis e camadas de normalização em lote foram introduzidas para evitar o ajuste excessivo, estabilizar o processo de treino e garantir uma melhor generalização.

Para além dos testes anteriores foram utilizados dois otimizadores para testar o correto funcionamento dos modelos nomeadamente: Adam e Stochastic Gradient Descent (SGD). A função de perda utilizada para o nosso problema de classificação multi-classe foi *categorical crossentropy* uma vez que esta função de perda é particularmente adequada para problemas em que cada amostra de entrada pertence exatamente a uma classe de entre várias classes possíveis.

3.2.4.1.2 Método de Aprendizagem

A rede neuronal utiliza *Softmax* com "*Cross Entropy*"como função de perda de treino num lote de entrada. A função *Softmax* calcula a matriz de probabilidades de saída das classes no *dataset* e quanto maior for a probabilidade de uma classe, mais o modelo considera que o objeto de entrada representa essa classe, sendo que a soma das probabilidades das classes é sempre 1.

A "*Cross Entropy*"mede a diferença entre as probabilidades de saída previstas e as probabilidades reais (categorias). Quanto mais a distribuição prevista se desvia da distribuição real, maior será o valor da *Cross Entropy*.

3.2.4.1.3 Registo dos modelos

O treino do modelo desempenha um papel crítico no desenvolvimento de algoritmos de DL, permitindo a sintonização dos pesos da rede neuronal para otimizar o seu desempenho. Para controlar este processo é utilizada uma técnica conhecida como *Early Stopping*, através da classe *EarlyStopping* disponibilizada pela biblioteca *Keras* sendo esta uma técnica que monitoriza uma métrica de interesse, que neste caso corresponde à *accuracy* nos dados de validação e interrompe o treino caso essa métrica não melhore durante um número predefinido de épocas. Uma vez terminado o treino o modelo é guardado no formato H5 (*Hierarchical Data Format*), um formato amplamente utilizado para armazenamento de dados hierárquicos (ver listagem 3.6). Um ficheiro de texto é utilizado para guardar toda a informação essencial sobre o modelo treinado, pois esta abordagem ajuda a manter um registo claro dos resultados obtidos em diferentes testes e configurações de modelo.

```
1 early_stopping = EarlyStopping(monitor='val_accuracy', patience=20,
      restore_best_weights=True,verbose=1)
2 model.save(f'IR V2/Modelos/IR_V2_SGD_Intel_{batch_size}_0.001_{epochs}
      _dropout_0.65_{i}.h5')
```

Listagem 3.6: Método de *EarlyStopping* e registo dos modelos

3.2.5 Classificação

Apesar de um dos objetivos deste projeto ser a utilização de um modelo *Inception ResNet* como classificador, foi facilmente foi percetível que uma das dificuldades ao usar redes neuronais (neste caso a arquitetura *Inception ResNet*) surge quando é acrescentada uma nova pessoa ao *dataset*, pois isto implica fazer um novo treino para o novo *dataset* e o tempo que está associado a esta ação é demasiado elevado. Os resultados obtidos deste modelo na tarefa de classificação foram tão insatisfatórios que não foram incluídos neste documento. Posto isto para além de utilizar o modelo como classificador foi estudada outra abordagem como forma de resolver este problema, nomeadamente utilizar o modelo *Inception ResNet* apenas para codificar as imagens num vetor de características (*embedding*). Estes vetores de características representam as imagens num espaço N-dimensional, onde N é o número de características extraídas pelo modelo *Inception ResNet*. O modelo é utilizado como *facial feature extrator* no problema de reconhecimento facial e este é depois utilizado para treinar um classificador SVM com um *kernel* linear. Foi também utilizado o modelo *FaceNet* da biblioteca *Keras* [45] previamente treinado para fazer comparação dos resultados obtidos através do classificador SVM.

As SVM produziram uma precisão significativa com um menor poder computacional e o objetivo do algoritmo de SVM é encontrar um hiperplano num espaço N-dimensional que classifique distintamente os dados.

O modelo utilizado foi a implementação SVM da biblioteca *Scikit-learn* (ver listagem 3.7) com a técnica de *kernel* linear. O *kernel* linear é utilizado quando os dados podem ser separados usando uma única linha, uma vez que o *dataset* tinha um grande número de características.

```
model = SVC(kernel='linear', probability=True)
model.fit(trainX, trainy)
```

Listagem 3.7: Implementação da SVM

Para treinar o classificador SVM é inicialmente preciso fazer a deteção facial para localizar e delimitar as regiões faciais e a estas imagens é aplicado o modelo previamente treinado para extrair os *embeddings* faciais a partir das faces detetadas.

Após calculados os *embeddings* é então utilizado o algoritmo SVM com *kernel* linear para criar um classificador capaz de distinguir diferentes indivíduos com base nas representações vetoriais das faces.

O *dataset* utilizado para treinar e testar o modelo contém imagens de faces associadas às suas classes correspondentes e foram retiradas do *dataset* criado para treinar o modelo. Através deste *dataset*, o modelo SVM é treinado para associar as representações de face às suas respetivas classes.

3.2.6 Visualização dos Dados

A visualização dos dados é uma etapa crucial no processo de reconhecimento facial com algoritmos de ML pois esta etapa permite entender melhor o comportamento dos dados, identificar padrões e tendências, para além de auxiliar na análise dos resultados.

3.2.6.1 Gráficos

Uma maneira eficaz de visualizar o desempenho dos modelos ao longo do treino é através de gráficos, onde as métricas chave como a perda (*loss*) e a exatidão (*accuracy*) são representadas em relação ao número de épocas (*epochs*) de treino. Na listagem 3.8 está representada a forma de obter os valores de *loss* e *accuracy* em treino e teste de um modelo.

```
loss=history.history['loss']
val_loss = history.history['val_loss']
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(len(loss))
```

Listagem 3.8: Gráficos para visualização dos resultados

3.2.6.2 *Principal Component Analysis*

O Principal Component Analysis (PCA) é uma técnica de redução de dimensionalidade que transforma os dados originais num novo espaço de menor dimensão, mantendo as principais informações dos dados. Nesse novo espaço as coordenadas representam os componentes principais, que são combinações lineares dos atributos originais.

No contexto do reconhecimento facial, o PCA pode ser aplicado aos *embeddings* (representações) das faces extraídas por um modelo de DL pois ao projetar os *embeddings* num espaço tridimensional, é possível visualizar a distribuição das classes de faces. Cada classe é representada por pontos no espaço e pode-se observar se as classes estão bem separadas ou se há alguma sobreposição. Cada classe é representada por uma cor diferente e pontos próximos indicam que as faces daquelas classes têm *embeddings* similares, já pontos distantes indicam que as classes têm características distintas.

```
1 pca = PCA(n_components=3)
2 pca_result = pca.fit_transform(trainX)
```

Listagem 3.9: Visualização de resultados (PCA)

3.2.6.3 *t-distribuited Stochastic Neighbor Embedding*

O t-distributed Stochastic Neighbor Embedding (t-SNE) é uma técnica de redução de dimensionalidade não linear que preserva as relações de proximidade entre os dados. Diferente do PCA, o t-SNE é mais adequado para visualizar dados em espaços de alta dimensão e capturar padrões complexos.

No contexto do reconhecimento facial o t-SNE pode ser aplicado aos *embeddings* das faces, projetando-as num espaço bidimensional. Esta projeção permite visualizar as classes de faces num gráfico de dispersão, onde cada ponto representa uma face e as cores indicam a classe correspondente. A distribuição das classes no gráfico revela informações sobre a similaridade das faces pertencentes a cada classe sendo que é possível identificar agrupamentos ou padrões que podem indicar a presença de subgrupos dentro das classes.

```
1 tsne = TSNE(n_components=2, random_state=42)
2 tsne_result = tsne.fit_transform(trainX)
```

Listagem 3.10: Visualização de resultados (t-SNE)

3.3 Abordagem usada no RoboCup@Home

O robô CHARMIE para além da sua componente social tem também a componente competitiva e esta componente é necessária devido às diferentes características e especificidades do trabalho desenvolvido

para um ambiente social e para um ambiente competitivo. Algumas dessas diferenças são nomeadamente o desenvolvimento de *software* adequado tendo em conta as tarefas a desenvolver, o tempo disponível para cada prova que é bastante reduzido comparativamente ao tempo disponível para uma tarefa num contexto social e para além disto não necessita de ser tão robusto. No contexto do RoboCup@Home onde um robô participa em provas que envolvem interação com seres humanos, a deteção de pessoas e o reconhecimento facial desempenham um papel fundamental. Nesta secção do documento serão explicados todos os processos executados para o funcionamento do robô em contexto de competição nomeadamente o funcionamento do RoboCup e da competição em que o CHARMIE participa e é ainda feita uma explicação das provas associadas ao tema desenvolvido ao longo deste trabalho.

3.3.1 RoboCup

O *RoboCup* é a maior competição de robótica e inteligência artificial do mundo, foi proposto em 1993 por *Hiroaki Kitano* para estimular a investigação na área da robótica e inteligência artificial. O *RoboCup* nasceu com o desafio louco e muito complexo de em 2050 realizar um jogo entre as equipas campeãs do mundo de futebol em humanos e robôs [46].

Organizado em torno de ligas competitivas que representam domínios de investigação científica, o evento reúne milhares de investigadores em torno de uma ambição comum: partilhar conhecimentos e fazer avançar o progresso técnico e científico. Ao longo dos anos foram surgindo novos desafios e criadas novas ligas nomeadamente *Rescue*, *@Home*, etc.

3.3.1.1 RoboCup@Home

A liga *RoboCup@Home* tem como objetivo desenvolver uma tecnologia de robôs de serviço e de assistência com grande relevância para futuras aplicações domésticas pessoais. Trata-se da maior competição internacional anual para robôs de serviço autónomos e faz parte da iniciativa *RoboCup*. São realizados um conjunto de testes de referência para avaliar as capacidades e o desempenho dos robôs num ambiente doméstico realista e não normalizado. A atenção centra-se nos seguintes domínios, mas não se limita a eles: Interação e cooperação entre humanos e robôs, navegação e mapeamento em ambientes dinâmicos, visão por computador e reconhecimento de objetos em condições de luz natural, manipulação de objetos, comportamentos adaptativos, integração de comportamentos, inteligência ambiental, normalização e integração de sistemas.

A competição do *RoboCup@Home* apresenta três ligas e consiste numa série de testes que os robôs têm de resolver. A liga em que o CHARMIE participa é a OPL - *Open Platform League* que não possui quaisquer restrições de *hardware* e oferece o mais elevado grau de liberdade na condução da investigação. A avaliação do desempenho baseia-se numa pontuação definida nas regras do concurso e na avaliação por um júri onde os critérios se baseiam em: incluir interação homem-robô, ser socialmente relevantes, ser direcionados/orientados para aplicações, ser cientificamente estimulantes, ser fáceis de montar e de

baixo custo, ser simples e ter regras auto-explicativas, ser interessantes de observar e demorar pouco tempo na realização das tarefas.

Na edição de 2023, cada sub-liga do *RoboCup@Home* é dividido em 3 fases nomeadamente o Estágio 1, Estágio 2 e as finais. Do estágio 1 fazem parte as seguintes provas: *Carry My Luggage, General Purpose Service Robot (GPSR)*, Rececionista, *Serve Breakfast* e *Storing Groceries*. No estágio 2 as provas são: *Clean the Table, Enhanced General Purpose Service Robot (EGPSR), Restaurant* e *Stickler for the Rules*. Na final o objetivo é demonstrar as capacidades do robô relativas ao tema de preparar uma refeição.

3.3.1.2 Explicação das provas

O CHARMIE participou em todas as provas do RoboCup@Home com exceção das provas onde os braços eram uma necessidade. A par desta dissertação foi também desenvolvido o *software* relativo à prova Rececionista e *Stickler for the rules*.

Na prova Rececionista o objetivo é o robô comportar-se como um rececionista de uma festa e fazer as apresentações entre o anfitrião da festa e todos os convidados. Esta prova implica a interação entre humano e robô uma vez que o CHARMIE tem de falar com os convidados e também compreender aquilo que lhe é dito, para que exista uma recolha de informação tanto de imagem através da câmara como de som através do microfone, para além de ser ainda necessária a deteção de pessoas. Na figura 41 está o mapa da casa para a prova desta edição.

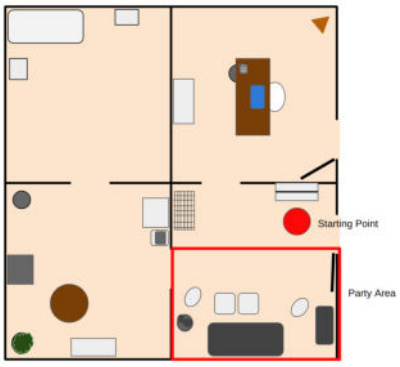


Figura 41: Mapa da casa com indicação da área da prova do Rececionista [47]

De uma forma mais explícita a tarefa do rececionista começa com o robô dentro da casa à espera que o convidado bata à porta. Nessa altura o robô deve deslocar-se até à mesma para receber o convidado e ao recebê-lo deve apresentar-se e pedir as suas informações, neste caso o seu nome e bebida favorita. Após o diálogo terminado e todas as informações obtidas, o robô encaminha o convidado até à zona dos sofás onde procede à apresentação do convidado ao anfitrião e vice-versa. De salientar que as informações do anfitrião são previamente cedidas ao robô. Após convidar a sentar o convidado, o próximo convidado bate à porta e é repetido o processo. No momento de apresentações, para além de apresentar os convidados um ao outro e apresentar o anfitrião ao mais recente convidado, é feita também uma descrição do primeiro

convidado ao segundo, sendo que nesta descrição está incluído apresentar quatro características como por exemplo género, idade, etnia, altura, cor dos olhos etc.

Todos os processos que formam a tarefa do Rececionista e que permitem às equipas participantes do *RoboCup@Home* obter pontos encontram-se na figura 42.

Action	Score
Main Goal	
Introduce a new guest to every other guest and offer a seat	*2×250*
Look at the person talking	*50*
Look at the person the robot is introducing the guest to	*50*
Look in the direction of navigation	*50*
Continue with wrong name or drink	*2×−50*
Persistent inappropriate gaze - away from conversational partner	*2×−50*
Persistent gaze not in the direction of the navigation while moving	*−50*
Bonus Rewards	
Open the entrance door for a guest	*2×100*
Describe the first guest to the second guest	*150*
Deus Ex Machina	
Alternative HRI	*2×−75*
Not recognizing people	*2×−200*
Special Penalties & Bonuses	
Not attending (see sec. 3.9.1)	*−500*
Total Score (excluding special penalties & standard bonuses)	*1000*

Figura 42: Descrição dos pontos referentes à tarefa Rececionista [48]

Relativamente à prova *Stickler for the Rules* o objetivo é que o robô garanta que as regras da casa são cumpridas e respeitadas. O robô precisa de identificar quais os convidados que estão a desrespeitar as regras, comunicar aos mesmos a regra que estão a quebrar, a ação que devem tomar para seguirem as regras e confirmar que o convidado respeitou a ação indicada. As regras da casa são: não pode estar nenhum convidado no espaço proibido (nesta edição da da competição o espaço proibida foi o quarto), não pode ninguém estar com sapatos dentro da casa, todos os convidados devem ter um copo na mão e não pode existir lixo no chão.

Nesta prova é então necessário que haja interação entre o robô e os convidados, que haja deteção e classificação de objetos, deteção de pessoas e reconhecimento de ações. Na figura 43 representa o mapa da casa, especificando a zona proibida.

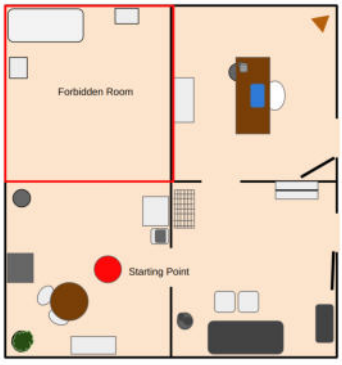


Figura 43: Mapa para a prova *Stickler for the rules* [47]

Explicando mais detalhadamente, o robô começa numa localização da casa sendo-lhe previamente dito qual o espaço proibido da casa que tal como mencionado anteriormente e percetível no mapa acima, nesta edição tratou-se do quarto. De seguida e dependente da estratégia de cada equipa, começa a fase de procurar por regras quebradas sendo que para cada regra quebrada existem ações associadas. No caso de alguém estar de sapatos dentro da casa o robô deverá informar o convidado que terá de ir até à entrada da casa e retirar os sapatos. Caso algum convidado esteja no espaço proibido deverá ser-lhe dito para sair do espaço. Se for encontrado lixo no chão, o robô deverá ser capaz de encontrar a pessoa mais próxima do lixo e pedir-lhe para o colocar no caixote do lixo. Neste caso existiam dois caixotes do lixo na casa pelo que o robô deveria encaminhar o convidado para um deles. Por fim, no caso de detetar algum convidado sem uma bebida na mão, o robô deverá acompanhar o convidado até ao espaço onde pode pegar na sua bebida, que nesta edição é no escritório.

Todos os processos que formam a tarefa do *Stickler for the Rules* e que permitem às equipas participantes do RoboCup@Home obter pontos encontram-se na figura 44.

Action	Score
Regular Rewards	
Identify a guest breaking a house rule (indicating the rule by voice or log).	*4×100*
Making eye-contact, politely clarify to the guest what action he should take.	*4×100*
Confirm that the guest is following the rule.	*4×200*
Bonus Rewards	
Making eye-contact, politely clarify to the guest what rule is being broken.	*4×100*
Regular Penalties	
Talking to a guest about a rule they are not breaking	*4×−100*
Deus ex Machina Penalties	
A human directs the robot towards a guest who is breaking a rule	*4×−50*
A human tells the robot which rule is being broken	*4×−100*
Special Penalties & Bonuses	
Not attending (see sec. 3.9.1)	*−500*
Using alternative start signal (see sec. 3.7.8)	*−100*
Total Score (excluding special penalties & standard bonuses)	*2000*

Figura 44: Descrição dos pontos referentes à tarefa *Stickler for the Rules* [48]

3.3.2 Desenvolvimento da Prova Rececionista

No anexo II.1.1 é possível ver o fluxograma da prova do Rececionista acima descrita. Da mesma forma, no anexo II.1.2 é possível ver em detalhe a solução criada para a prova do Rececionista.

Todos os módulos explicados a seguir foram desenvolvidos com base na arquitetura ROS2 de forma a ser apenas necessário publicar e subscrever nos tópicos precisos.

No excerto de código abaixo, listagem 3.11, é possível ver exemplos de *publishers* e *subscribers* utilizados na prova Rececionista.

```
1 # Audio
2 self.audio_command_publisher = self.create_publisher(SpeechType, "
    audio_command", 10)
3 self.get_speech_subscriber = self.create_subscription(String, "get_speech",
    self.get_speech_callback, 10)
```

Listagem 3.11: Excerto da função *init* com as declarações de *publishers* e *subscribers*

O código foi desenvolvido recorrendo a *threads* uma vez que existia a necessidade de correr dois processos em simultâneo. Foi então necessário criar duas classes onde numa foram declarados todos os tópicos acima mencionados, variáveis e funções utilizadas, a outra consiste basicamente numa *main* onde é possível encontrar a máquina de estados criada para executar a tarefa.

3.3.2.1 Áudio

Tal como mencionado anteriormente na prova Rececionista existe um diálogo entre o robô e os convidados. A interação vocal dos humanos com o robô foi conseguida através de Reconhecimento Automático da Fala (Automatic Speech Recognition (ASR)), um *software* de código aberto chamado *Whisper*. ASR permite que os seres humanos usem as suas vozes para falar com um microfone/computador. *Whisper* é um sistema de reconhecimento automático de fala (ASR) treinado com 680.000 horas de dados supervisionados com múltiplas línguas e tarefas retirados da internet. O uso de um *dataset* tão grande e diversificado permite uma maior robustez para sotaques, ruído de fundo e linguagem técnica.

Para um melhor funcionamento do sistema de áudio do robô é necessário um ajuste do ruído para que o robô consiga distinguir o ruído de fundo da voz do humano e permite ainda ao robô determinar quando o humano terminou a sua fala (ver listagem 3.12).

```
1 calib = Bool()
2 calib.data = True
3 self.node.calibrate_ambient_noise_publisher.publish(calib)
```

Listagem 3.12: Exemplo de código para ajuste do ruído previamente à ação de ouvir

Na prova Rececionista é disponibilizada durante a competição uma lista de nomes que o robô terá de ser capaz de reconhecer. Assim é criado um dicionário que contém para cada nome um conjunto de palavras que se assemelham ao respetivo nome. Por exemplo para o nome "Adel"podem associar-se as palavras: "Adel", "Adele", "Avel". Este trabalho é necessário para que independentemente do sotaque ou forma de falar o robô consiga compreender aquilo que é dito (ver listagem 3.13).

```
1 names_dict = {
2     "Adel":     ["adel", "adele", "adam", "avel"],
3     "Angel":    ["angel", "anjo", "hell", "ahil", "anjou", "anjal",
4     "Robin":    ["robin", "harbin", "robyn", "hobby", "hobean", "hohvin", "
          robon", "corbin", "haubin", "horven"],
5     "Simone":   ["simone", "simon", "simo", "siman"]
6     }
```

Listagem 3.13: Extrato do dicionário relativo a alguns dos nomes

Para o módulo do áudio é necessário receber um nome e uma bebida por parte do convidado em forma de áudio. Quando o convidado termina de falar a frase é processada e interpretada e no caso de estar correta (ter sido dado um nome e uma bebida válida) a mesma é dividida, tal como é possível ver

na listagem 3.14. No caso de não ser válida o robô informa oralmente que a informação está incorreta e pede para repetir o processo até receber informação valida.

```
1 keyword_list= self.node.keywords.data.split(" ")
2 self.node.names.append(keyword_list[0])
3 self.node.drinks.append(keyword_list[1])
```

Listagem 3.14: Registo do nome e bebida

3.3.2.2 Voz

É também necessário que o robô fale com os seus utilizadores e para isto foram desenvolvidos dois programas. Inicialmente foi feito um sistema que funcionava online contudo devido às condicionantes do evento, onde a estabilidade e fiabilidade da ligação à internet é baixa, foi necessário desenvolver outra solução mas que funcionasse *offline*. A solução online foi desenvolvida através da biblioteca *python gTTs* que converte uma *string* de texto em fala a qual é emitida através das colunas de som do robô. Para a solução *offline* foi utilizada a biblioteca *python pyttsx3* que funciona da mesma forma que a biblioteca *gTTS* mas *offline*.

De salientar que a voz do robô é extremamente importante em todas as situações pois é necessário que o robô seja claro em tudo aquilo que pretende obter por parte do convidado e também esclarecer tudo aquilo que está a fazer. Isto permite que o convidado tenha todas as informações do seu lado facilitando a sua compreensão e o correto seguimento da prova. Para além disto, permite também informar todas as pessoas presentes do que está a acontecer e dos passos seguintes a executar dando segurança de que tudo se está a desenrolar da forma esperada e toda a gente está ciente do que vai acontecer.

Independentemente da língua é necessário enviar a *string* a ser dita pelo robô publicando-a no tópico. No tópico do *speaker* a língua predefinida é o inglês. Na listagem 3.15 está um exemplo de uma das frases ditas pelo CHARMIE.

```
1 self.node.speech_str.command = "What's your name and favourite drink?"
2 self.node.speaker_publisher.publish(self.node.speech_str)
```

Listagem 3.15: Utilização do tópico dos *speakers*

3.3.2.3 Navegação

Uma das questões fulcrais na prova é o robô movimentar-se pela casa, pelo que foram desenvolvidas soluções para o robô conseguir deslocar-se na casa sabendo onde se encontra, desviando-se de obstáculos e parando quando chega ao local desejado. Foi necessário nos dias de preparação das provas decidir os locais de paragem do robô como visível na listagem 3.16. A listagem 3.17 mostra um exemplo de código para movimentar o robô para determinada coordenada usando a função listado em 3.18.

```
self.sofa_coordinates = (-2.8, 2.8)
```

Listagem 3.16: Definição de coordenada

```
self.node.coordinates_to_navigation(self.node.door_coordinates, self.node.
    find_coordinates, False)
```

Listagem 3.17: Chamada da função de navegação

```
def coordinates_to_navigation(self, p1, p2, bool):
    nav = TarNavSDNL()
    nav.flag_not_obs = bool
    nav.move_target_coordinates.x = p1[0]
    nav.move_target_coordinates.y = p1[1]
    nav.rotate_target_coordinates.x = p2[0]
    nav.rotate_target_coordinates.y = p2[1]
    self.target_position_publisher.publish(nav)
```

Listagem 3.18: Função de navegação do robô para coordenada específica

3.3.2.4 Sistema de *debug* via *LEDs RGB* e *Buzzer*

As ferramentas de *debug* são essenciais, quer durante o desenvolvimento de código mas acima de tudo durante o decorrer das provas, onde toda a equipa tem de estar fora da casa onde a prova se realiza. Na prova do Rececionista há momentos fulcrais onde, para quem está de fora possa ter alguma ferramenta de *debug* visual para ajudar a perceber em que fase do código está. A utilização de *LEDs RGB* que se encontram debaixo da base do CHARMIE foram a melhor solução encontrada, por permitirem a utilização de diversas cores e modos, os quais são acionados tendo em conta as necessidades, funcionando como um método de *debug* externo.

Existe também um método de *dubug* interno auditivo que é o *buzzer*. O *buzzer* é ativado quando o botão de emergência é premido ou então quando a bateria está abaixo de um certo nível necessário para o movimento fluído do robô.

3.3.2.5 Deteção Facial

Na prova do rececionista a deteção facial é necessária nos dois momentos em que o robô está à espera do respetivo convidado, uma vez que a prova apenas avança no caso de serem detetadas as faces, pois o objetivo da prova passa necessariamente por apresentar os convidados e inclusive conseguir extrair características da foto de um deles.

Para a deteção facial foi utilizada a câmara e o pescoço, contudo sempre que o objetivo era o robô ter um diálogo com os novos convidados, este assumia uma posição (x, y) de (180°, 180°) permitindo-lhe olhar de frente para a pessoa. A imagem era obtida através da câmara onde a partir da função *found_landmarks* se detetavam os *keypoints* do convidado e eram tidos em consideração o x e y do nariz,

anca e ombro esquerdo. Foi utilizada a biblioteca *MediaPipe*, e um dos objetivos ao usar esta função era detetar uma pessoa independentemente da sua face estar ou não na imagem. O convidado pode ser de estatura alta, onde podem ser detetados apenas os seus *keypoints* dos ombros e/ou da cintura, mas continua a ser necessário ter em conta esta pessoa uma vez que se trata do convidado que está em frente ao robô.

A partir desta função é obtida então a informação do número de pessoas detetadas, os valores dos *keypoints* detetados e ainda as dimensões da imagem. Na listagem 3.19 encontra-se um exemplo de utilização da função para a imagem obtida da câmara.

```
num_faces, face_x, face_y, shoulder, hip, center_x, center_y = self.node.
    found_landmarks(cv2_img)
```

Listagem 3.19: Procura de landmarks numa face

Tal como mencionado anteriormente é importante obter uma foto com qualidade do convidado que não tenha cortes e que não seja obtida em movimento para evitar desfoque. Como existe a possibilidade dos convidados serem mais altos que o robô ou simplesmente a face do convidado não estar centrada na imagem, foi criada uma função que move o pescoço do robô até encontrar uma face centrada na imagem considerando o centro da face o nariz pois isto permite uma melhor interpretação da cara.

3.3.2.6 Recolha de Imagens

Durante o decorrer da prova, é necessário recolher uma imagem do convidado 1 para que desta fotografia sejam extraídas características e posteriormente apresentadas ao convidado 2. Contrariamente à abordagem social neste caso não é necessário recolher centenas de fotos e associá-las a diferentes diretórios, é apenas necessária uma imagem da face do utilizador. Após o robô questionar o convidado sobre o seu nome e a sua bebida favorita, é-lhe pedido que olhe para a câmara para captura uma imagem da sua face e guardá-la com o nome associado num diretório onde estão todas as imagens relativas as provas. Na listagem 3.20 é possível ver a forma como a imagem é obtida após a deteção facial.

```
1 cv2_img = self.node.br.imgmsg_to_cv2(self.node.image_color, "bgr8")
2 self.filename = self.node.names[1]
3 file_path = os.path.join(self.node.pasta_imagens_conhecidas, self.filename
      + '.png')
4 cv2.imwrite(file_path, cv2_img)
```

Listagem 3.20: Recolha da imagem

É a partir desta imagem guardada que são extraídas as características do convidado para serem posteriormente apresentadas pelo robô.

3.3.3 Desenvolvimento da Prova *Stickler for the Rules*

Atendendo a que esta foi a primeira participação neste desafio tão complexo como é a liga *RoboCup@Home*, esta equipa LAR@Home nem todos os módulos puderam ser desenvolvidos previamente à competição.

O *software* relativo à prova *Stickler for the Rules* foi desenvolvido apenas após as provas do estágio 1 terem terminado o que implicou uma análise e seleção criteriosa dos objetivos a desenvolver para esta prova. Assim, foi desenvolvido apenas a parte relativa à deteção de pessoas na sala proibida, uma vez que também estava associado à prova Rececionista no âmbito de deteção de pessoas.

No anexo II.1.3 está representado o fluxograma da prova *Stickler for the Rules*, e no anexo II.1.4 o fluxograma relativo ao trabalho desenvolvido.

A estrutura do código desenvolvido para esta prova é bastante similar ao código do Rececionista pois foram utilizadas *threads* e foi tido por base uma máquina de estados. Nas secções seguintes é possível encontrar o trabalho desenvolvido para cada módulo.

3.3.3.1 Voz

Tal como na prova do Rececionista um método de comunicação com os convidados da festa é a utilização da voz do robô e é desta forma que o mesmo consegue comunicar cada vez que há uma regra infringida. Apesar de o robô ter de ser capaz de perceber qual a regra que está a ser infringida e por quem, é possível (com penalização de pontos) existir comunicação com os convidados como forma de verificação. Contudo esta funcionalidade não foi utilizada uma vez que se optou por não confirmar para evitar a penalização. O módulo utilizado tal como na prova anterior foi o *offline* pelo que a solução utilizada está representada na secção 3.3.2.2.

3.3.3.2 Navegação

Nesta prova são utilizadas coordenadas previamente definidas para onde o robô deverá ir e observar toda a divisória onde se encontra. Tal como na prova do Rececionista foi utilizada a solução apresentada na secção 3.3.2.3

3.3.3.3 Deteção de Pessoas

Relativamente à deteção de pessoas foi necessário desenvolver uma solução que para além de detetar pessoas conseguisse perceber em que divisão da casa estavam e principalmente se estavam dentro da casa. As regras obrigam o robô estar na mesma divisão da pessoa uma vez que em casos reais o mesmo não poderia ver de uma divisão para outra através das paredes. Para além do desenvolvimento de uma solução que indicasse a posição do robô em relação à casa tornou-se necessário indicar também onde estão as pessoas na casa sendo que desta forma o robô estaria em condições para saber se efetivamente detetou alguém na sala proibida ou não.

Da mesma forma que o robô é capaz de detetar a posição da pessoa em relação à casa, é também capaz de saber se está no quarto e confirmar se a mesma já saiu ou não do espaço. As coordenadas das pessoas são comparadas com as coordenadas que delimitam cada divisão da casa e estão representadas na listagem 3.21.

```
1  self.house_center_coordinates = (1.45, 4.95)
2  self.house_left_bot_coordinates = (-4.05, 0.45)
3  self.house_left_upp_coordinates = (-4.05, 9.45)
4  self.house_right_bot_coordinates = (4.95, 0.45)
5  self.house_right_upp_coordinates = (4.95, 9.45)
6
7  self.house_left_bot_name = "Living Room"
8  self.house_left_upp_name = "Kitchen"
9  self.house_right_bot_name = "Office or Study"
10 self.house_right_upp_name = "Bedroom"
11
12 self.coordinates_to_divisions(self.house_center_coordinates, self.
       house_left_bot_coordinates, self.house_left_bot_name)
13 self.coordinates_to_divisions(self.house_center_coordinates, self.
       house_left_upp_coordinates, self.house_left_upp_name)
14 self.coordinates_to_divisions(self.house_center_coordinates, self.
       house_right_bot_coordinates, self.house_right_bot_name)
15 self.coordinates_to_divisions(self.house_center_coordinates, self.
       house_right_upp_coordinates, self.house_right_upp_name)
```

Listagem 3.21: Definição das coordenadas das várias divisões

Assim que uma pessoa é detetada é utilizada uma das soluções já referida anteriormente, nomeadamente seguir a face das pessoas. Isto deve-se à necessidade de acompanhar o movimento da pessoa para saber se a mesma respeitou ou não a regra. Assim que seja recebida a informação de que a pessoa se encontra noutra divisão, o robô informa isso mesmo.

3.3.3.4 Sistema de *debug* via *LEDs RGB* e *Buzzer*

Tal como na prova do Rececionista também nesta prova o objetivo do *buzzer* é o mesmo, contudo neste caso os *LEDs RGB* são utilizados de uma forma diferente. No caso de o robô detetar uma regra a ser infringida os *LEDs* piscam com luzes azul e vermelha (similar às luzes da polícia) como forma de demonstrar que foi detetada uma regra quebrada, assim quando a ação for respeitada é acionada a luz verde nos *LEDs*. Isto ajuda em termos visuais, tanto a equipa como os árbitros a terem uma confirmação de que o robô está efetivamente a detetar.

3.4 Reconhecimento de outras características

Tanto no contexto competitivo como no social é importante reconhecer características dos utilizadores. Neste projeto para além do reconhecimento facial, foi também estudada a recolha de características como

o género, a idade, a etnia e a altura. Este reconhecimento de outras características pode ajudar a validar a identidade de uma pessoa, por exemplo num contexto social, onde é necessário detetar e reconhecer utilizadores em tarefas de longa duração. Num contexto competitivo e como é possível observar na figura 42, há pontos associados à correta identificação e apresentação de características do primeiro convidado ao segundo.

Devido às limitações dos espaços onde esta funcionalidade pode vir a ser utilizada, foi desenvolvida uma solução *offline* para permitir que seja usada independentemente da situação. Para a deteção das características como o género e a etnia, foi utilizada a biblioteca *DeepFace* que consiste numa biblioteca desenvolvida em *Python* que utiliza redes neuronais convolucionais (CNNs) para realizar tarefas de reconhecimento facial, incluindo a deteção e análise de características faciais. No contexto do reconhecimento do género e da etnia, a biblioteca *DeepFace* implementa um modelo de ML pré-treinado denominado *VGGFace* que consiste numa rede neuronal profunda treinada com um *dataset* bastante grande, contendo imagens faciais de pessoas de diferentes géneros e etnias. Para obter uma solução *offline* foi necessário treinar previamente o modelo para que quando necessário fosse apenas utilizado o modelo treinado. Relativamente à idade foram utilizados pesos e parâmetros já treinados pelo que já se tratava de uma solução *offline*. Relativamente à altura, a mesma é obtida através de uma comparação com a altura do próprio CHARMIE e o movimento do pescoço para centrar com a face. No caso do movimento do pescoço ser para cima, a altura da pessoa é superior à altura do robô, o mesmo raciocínio pode ser usado quando o movimento for para baixo, onde o robô é mais alto que o utilizador. Os métodos e o trabalho desenvolvido relativamente a este tópico, bem como as diferentes necessidades e aplicabilidades associadas a cada característica são explicados de seguida.

3.4.1 Género

O objetivo passa por o robô conseguir identificar se a pessoa é do género feminino ou masculino. Numa situação competitiva, alguns dos nomes disponíveis para ser usado em prova por árbitro e voluntários são considerados "sem género". Através desta característica é possível o robô detetar na mesma o género da pessoa. O mesmo pode acontecer numa situação social, doméstico ou médico onde no caso de não ser dada previamente a informação dos utilizadores ao robô, este deve ser capaz de reconhecer a pessoa com quem está a lidar de forma a acompanhar as necessidades do utilizador, como será o caso em indicar qual a casa de banho a utilizar bem como adaptar a comunicação com o utilizador.

Para a deteção de género através do modelo previamente treinado e de uma imagem fornecida, é obtida a previsão do qual o género predominante, como representado na listagem 3.22.

```
1 predictions = self.model_loaded_gender.predict(expanded_img)
2 gender_predominante_index = np.argmax(predictions, axis=1)
3 gender_predominante = ['Female', 'Male'][gender_predominante_index[0]]
```

Listagem 3.22: Previsão do género

3.4.2 Idade

Para além do género, a idade é uma característica fundamental no que toca a interação robô-humano. O robô deve ser capaz de reconhecer a idade dos seus utilizadores, onde neste caso foram considerados os respetivos intervalos de idades ((0 - 3), (4 - 7), (8 - 14), (15 - 22), (23 - 35), (36 - 45), (46 - 59), (60 - 100)). Num ambiente doméstico/médico ou até mesmo num ambiente competitivo, para além da necessidade de se saber as características do utilizador, o conhecimento da idade da pessoa com que o robô está a lidar é importante no sentido em que todo o seu comportamento é ajustado consoante a idade. No caso de se tratar de uma criança ou uma pessoa mais idosa, todos os movimentos e ações devem ser mais controlados/pensados e lentos uma vez que as crianças são pessoas mais inesperadas e os idosos poderão ser pessoas com mobilidade reduzida, dificuldade de visão ou até mesmo audição o que pode exigir que o CHARMIE fale mais alto ou até com uma linguagem mais trivial. Para o reconhecimento desta característica foram utilizados modelos pré-treinados de forma a através das imagens recolhidas ser- capaz de analisar o intervalo de idades correspondente (ver listagem 3.23).

```
1 blob = cv2.dnn.blobFromImage(face, 1.0, (227, 227), self.MODEL_MEAN_VALUES,
      swapRB=False)
2 self.ageNet.setInput(blob)
3 agePreds = self.ageNet.forward()
4 age_index = agePreds[0].argmax()
5 age= self.ageList[age_index]
```

Listagem 3.23: Previsão da idade

3.4.3 Etnia

O reconhecimento da etnia vêm principalmente pela componente competitiva onde o robô estará a participar com pessoas e robôs de todo o lado do mundo. Trata-se então de uma especificidade que será analisada e valorizada em contexto competitivo. Contudo, também num ambiente social é importante para o robô reconhecer a etnia da pessoa com quem está a falar, uma vez que em termos médicos por exemplo, os seres humanos têm necessidades diferentes, medicações diferentes e inclusive problemas de saúde distintos tendo em conta a sua etnia (ver listagem 3.24).

```
1 predictions = self.model_loaded_race.predict(expanded_img)
2 race_predominante_index = np.argmax(predictions, axis=1)
3 race_predominante = ['Asian', 'Indian', 'Black or African Descendent', '
    White', 'Middle Eastern', 'Latino Hispanic'][race_predominante_index[0]]
```

Listagem 3.24: Previsão da etnia

3.4.4 Altura

Relativamente à altura a análise é feita tendo em conta a altura do CHARMIE. O objetivo é que o robô percecione se a pessoa que tem à sua frente é mais alta ou mais baixa que o próprio robô. A aquisição da altura da pessoa já foi previamente descrita. A deteção da face e o consequente ajuste da altura do robô é importante no sentido em que o objetivo do CHARMIE é o seu contacto com pessoas, para isto e para tornar mais humana a comunicação, é importante que o CHARMIE seja capaz de encarar o utilizador aquando da comunicação com o mesmo.

Capítulo 4

Resultados

Neste capítulo são apresentados e demonstrados todos os resultados dos métodos acima descritos, estando estes divididos tal como no capítulo anterior, em duas abordagens.

4.1 Abordagem com Redes Neuronais para Ambiente Social

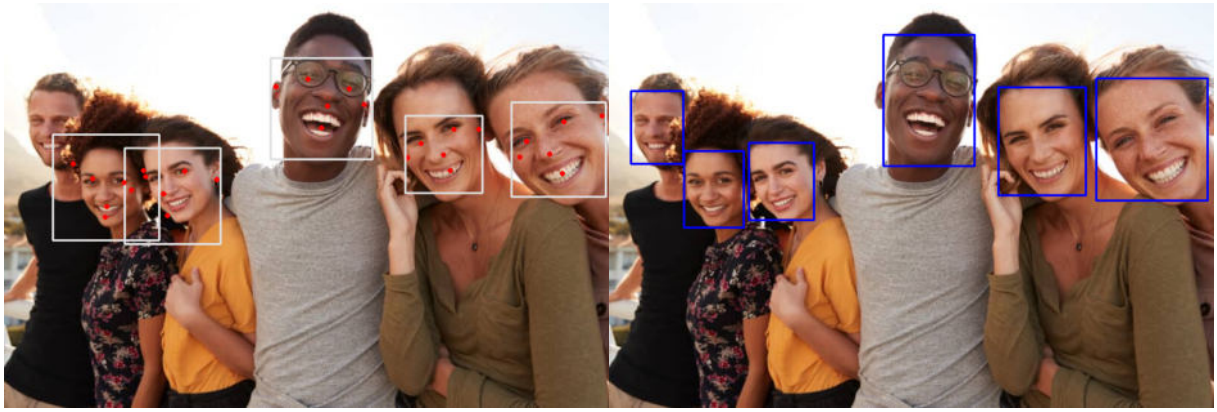
Os objetivos iniciais desta dissertação refletem-se na abordagem social deste projeto e para isto foram testadas diversas componentes de forma a obter um resultado final adequado e ajustável para o robô CHARMIE.

4.1.1 Deteção Facial

A deteção facial é relevante e foi desenvolvida dada a sua importância quando associada a um problema de reconhecimento facial. Abaixo encontram-se os resultados relativos à biblioteca MTCNN e *MediaPipe* aplicadas em variadas situações.

4.1.1.1 Teste em Imagens

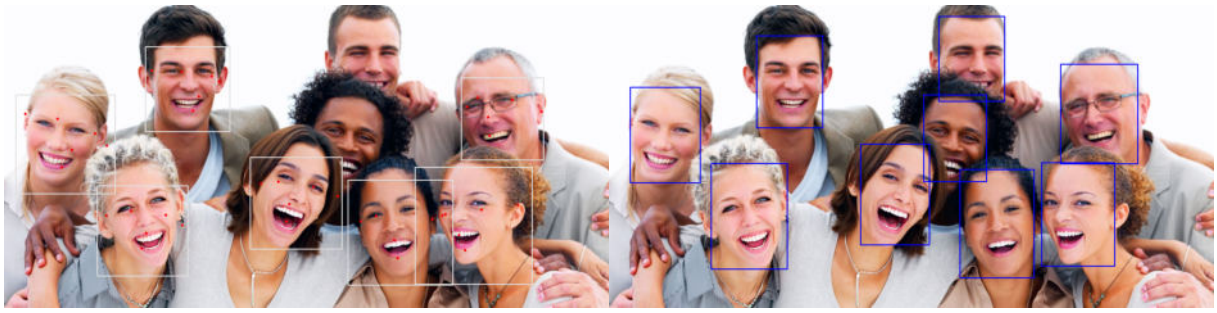
A deteção facial foi testada em diversas imagens com diferentes situações como diferente número de pessoas, diferentes dimensões das faces e diferentes condições de iluminação, sendo que são demonstrados os resultados de duas delas (ver figuras 45 e 46. Na figura 45(a) encontra-se o resultado obtido quando utilizado o *MediaPipe*, onde é possível verificar que a face mais distante não foi detetada uma vez que está parcialmente ocultada. Contrariamente a isto, quando utilizado o MTCNN todas as faces foram detetadas como se pode ver na figura 45(b).



(a) (b)

Figura 45: Resultados obtidos para a imagem [49] a) MediaPipe, b)MTCNN

Na figura 46 os resultados são similares aos anteriores. Quando utilizado o *MediaPipe* as faces que não estão completamente visíveis não são detetadas, quando usado o MTCNN todas são bem detetadas.



(a) (b)

Figura 46: Resultados obtidos para a imagem [50] a)MediaPipe, b) MTCNN

Contudo este resultado obtido através da biblioteca *MediaPipe* muda quando alterada a dimensão da imagem. A figura 47 mostra a alteração do resultado obtido onde todas as faces foram detetadas após um simples *resize* (não proporcional às duas dimensões) da imagem. Esta solução não é transversal para todas as situações contudo denota alguma possibilidade de melhoria dos resultados quando alterada a imagem.

Além da *accuracy* da deteção facial a questão do tempo é também importante. O processo MTCNN é computacionalmente mais pesado do que o *MediaPipe* o que se traduz num processo mais lento. Num processo de reconhecimento facial pretende-se que todos os passos intermédios para obter o resultado final não demorem muito tempo a processar, pelo que o tempo que o processo demora é um fator importante na escolha do método.

Na tabela 2 é possível verificar os tempos necessários para obter os resultados representados na figura 46 para as duas bibliotecas em estudo.



Figura 47: Resultado após *resize* da imagem [50]

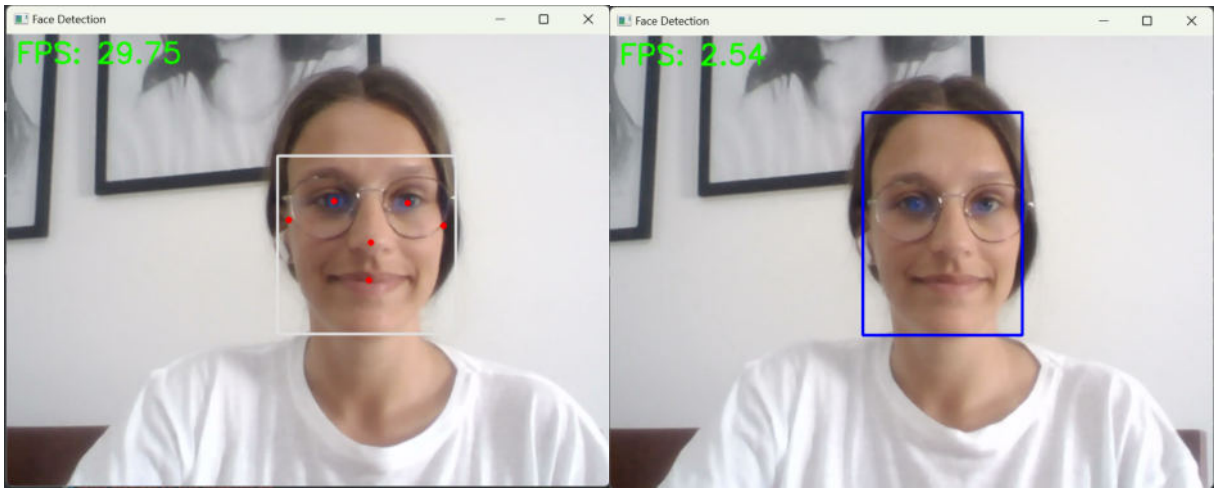
	MediaPipe	MTCNN
Tempo	0.015 segundos	1.47 segundos

Tabela 2: Comparação dos tempos na utilização das bibliotecas *MediaPipe* e MTCNN

4.1.1.2 Testes em *stream* de vídeo

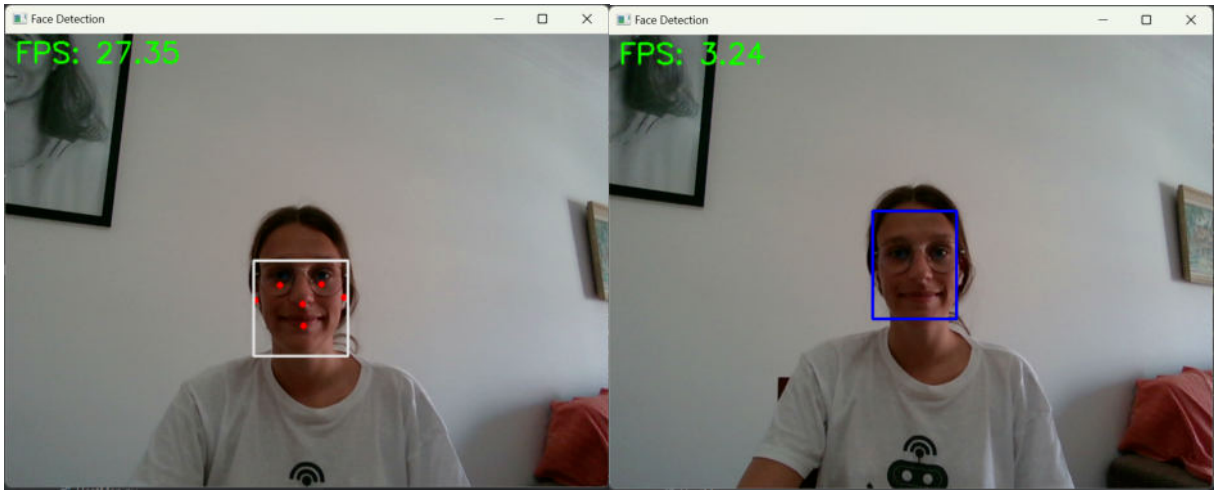
Os mesmos testes realizados em imagens foram também aplicados à deteção facial num *stream* de vídeo a 30 fps com uma resolução de xxx recorrendo à câmara do computador e também à câmara Intel *RealSense Depth* D455.

O primeiro teste realizado foi o número de fps obtidas quando utilizada cada uma das bibliotecas em análise. Já foi mencionado anteriormente que o MTCNN é um processo mais pesado e consecutivamente mais lento que o *MediaPipe* e isto traduziu-se em cerca de 1 a 4 fps obtidas quando utilizada a biblioteca MTCNN e 26 a 30 fps obtidas quando usado o *MediaPipe*, como é possível verificar nas imagens 48 e 49.



(a) (b)

Figura 48: Comparação do *streams* com a câmara do computador a) MediaPipe, b) MTCNN



(a) (b)

Figura 49: Comparação do *streams* com câmara Intel *RealSense Depth* D455 a) MediaPipe, b) MTCNN

Foi também testado o correto funcionamento do módulo de deteção facial para diferentes condições de iluminação, como se vê figura 50, e também com diferentes poses da face (ver figura 51), novamente usando as duas câmaras disponíveis.

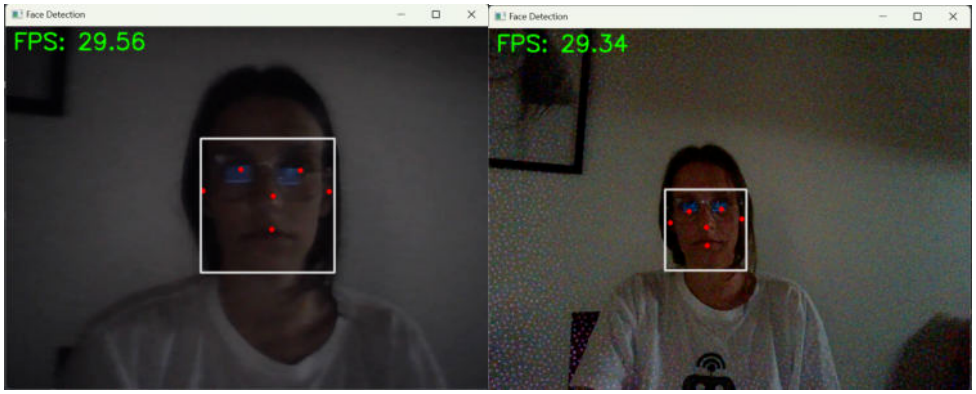
4.1.1.3 *Dataset*

Para além dos testes efetuados à correta deteção facial uma das principais aplicações deste módulo de deteção é a criação dos *datasets*. Os *datasets* em causa são utilizados para problemas de reconhecimento facial e para isto é imperativo que todas as imagens obtidas tenham faces na imagem. Como não é possível garantia a existência de uma face quando as imagens estão a ser retiradas, é utilizada a biblioteca *MediaPipe* na deteção facial de forma a garantir que quando uma face não está a ser detetada, não sejam guardadas imagens e seja dado um aviso ao utilizador que não está a ser detetada nenhuma face como está representado na figura 52.

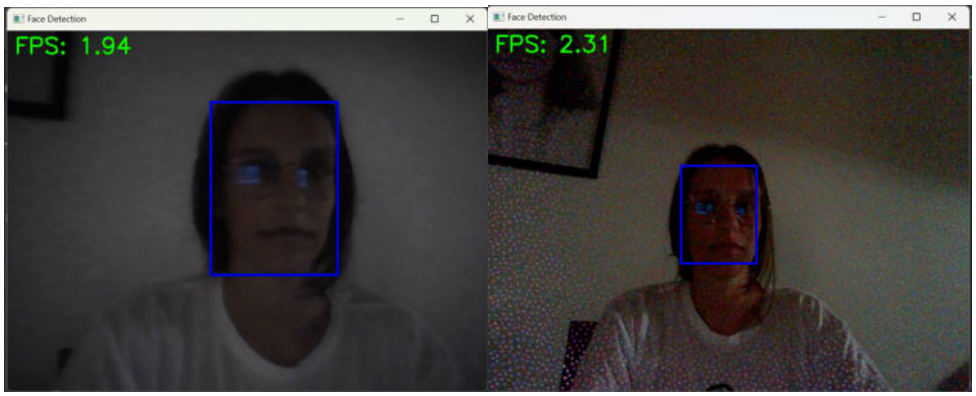
Para além das vantagens mencionadas acima relativas ao uso da biblioteca *MediaPipe* ao invés do uso da biblioteca MTCNN, o tempo necessário para executar a tarefa de criar o *dataset* é também um ponto relevante. Para obter um mesmo número de fotos em condições iguais, sem existir qualquer pausa para momentos sem faces detetadas ou qualquer interrupção do processo, com a biblioteca *MediaPipe* o processo demorou 72 segundos, enquanto que com o MTCNN o mesmo processo demorou 513 segundos.

4.1.1.4 Conclusões

Relativamente às soluções desenvolvidas para a deteção facial ambas as implementações foram simples de utilizar e implementar para imagens obtidas de *stream* de vídeo. Os testes foram realizados em diferentes situações nomeadamente variação do número de faces, pose, iluminação e foi demonstrada eficiência de ambos os algoritmos testados.



(a) Câmara do computador (b) Câmara Intel RealSense Depth D455



(c) Câmara do Computador (d) Câmara Intel RealSense Depth D455

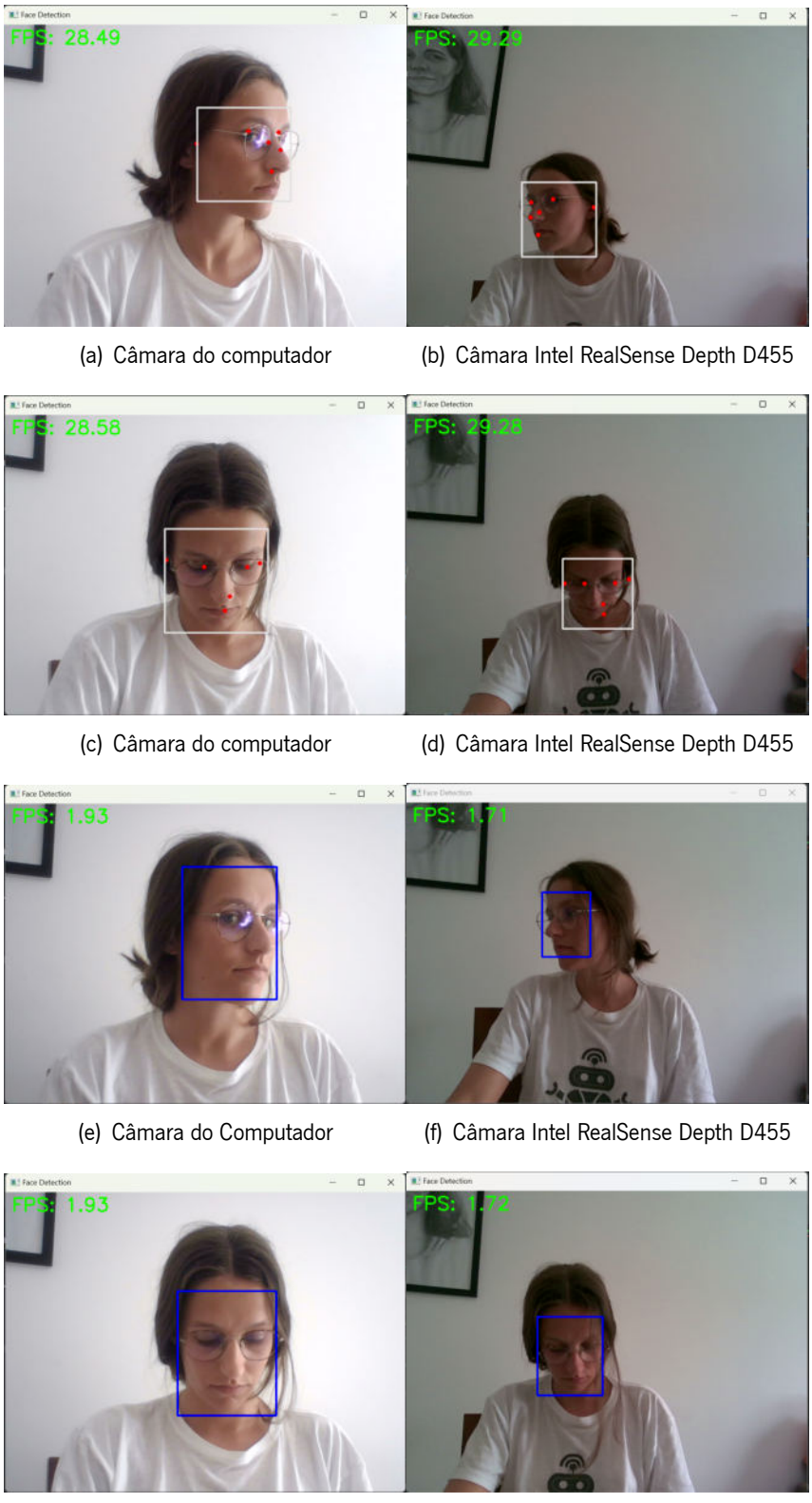
Figura 50: Resultados obtidos com MediaPipe (50(a) e 50(b)) e MTCNN (50(c) e 50(d)) para diferentes condições de iluminação

O *MediaPipe* provou ser uma solução que para uma aplicação em *stream* de vídeo demonstrou mais sucesso, tendo obtido 27 a 30 fps o que tornou o modelo fiável em configurações *stream* de vídeo. Foram analisadas as situações onde esta solução falhou nomeadamente caras desfocadas, mais distantes da câmara ou que não estivessem totalmente visíveis e estas condições revelaram-se difíceis para a estrutura do detetor porque os pontos de referência faciais não estavam expostos ou estavam em más condições de deteção. Contudo apesar de não ser aplicável para todas as imagens percebeu-se que há possibilidades de melhoria dos resultados.

Outra conclusão é que, quando se constrói um *dataset* para reconhecimento facial as amostras são normalmente o rosto recortado da identidade da classe, com o mínimo de fundo possível. A biblioteca *MediaPipe* foi também utilizada para isto recortando o rosto das pessoas a inserir no *dataset* personalizado.

4.1.2 Imagens de profundidade

Tal como mencionado anteriormente ser possível analisar imagens de profundidade e detetar faces nas mesmas é fundamental em questões de segurança. Ao desenvolver este projeto, constatou-se que



(a) Câmara do computador (b) Câmara Intel RealSense Depth D455

(c) Câmara do computador (d) Câmara Intel RealSense Depth D455

(e) Câmara do Computador (f) Câmara Intel RealSense Depth D455

(g) Câmara do Computador (h) Câmara Intel RealSense Depth D455

Figura 51: Resultados obtidos com *MediaPipe* (a), b) c) e d)) e MTCNN (e), f), g) h)) para diferentes poses

```
No face detected. Waiting for a face or press 'q' to quit.
```

Figura 52: Output na ausência de faces

a deteção de faces em imagens de profundidade poderá ser utilizada em diversos casos, contudo a necessidade desta segurança acrescida torna-se mais relevante numas situações do que noutras. Estando a solução desenvolvida, poderá partir do utilizador ou de soluções específicas solicitadas adicionar ou não este módulo.

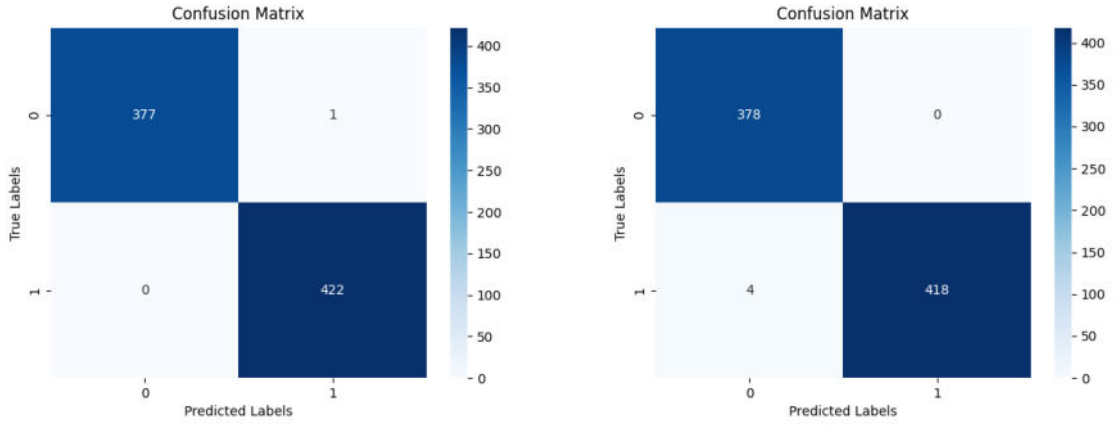
A verificação de faces em imagens de profundidade é analisada de forma a estar apta para ser adicionado em qualquer outro módulo, seja na criação do *dataset* ou na tarefa de reconhecimento facial.

Foram desenvolvidas duas soluções utilizando CNN e *Random Forest*. Na tabela 3 encontram-se os tempos e a *accuracy* obtida para cada uma das soluções analisadas tendo em consideração que foi utilizado o mesmo *dataset*.

	Random Forest	CNN
Tempo	57 segundos	120 segundos
Accuracy	99.5%	99.875%

Tabela 3: Comparação das soluções *random forest* e CNN

Na figura 53 estão representadas as matrizes de confusão obtidas para cada uma das soluções.



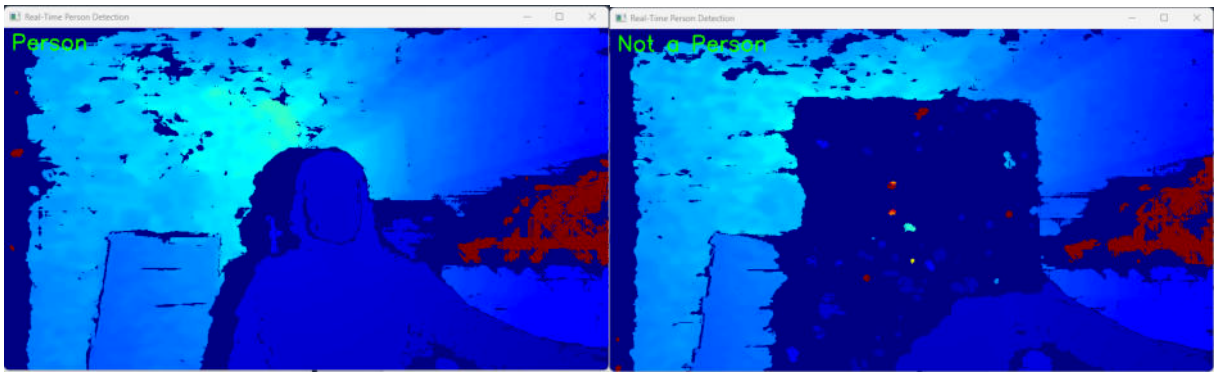
(a) CNN (b) *Random Forest*

Figura 53: Matrizes de confusão para dados de teste em imagens de profundidade

4.1.2.1 Testes em *stream* de vídeo

Foram testadas as duas soluções em *stream* de vídeo através da câmara Intel *RealSense Depth* D455, contudo apesar dos resultados acima mencionados, quando testado o modelo CNN os resultados eram bastante instáveis contrariamente aos obtidos quando utilizado o modelo *Random Forest*. Na figura 54

encontram-se dois exemplos do resultado obtido quando está uma pessoa presente na imagem ou uma tentativa de *spoofing*.



(a) (b)

Figura 54: *Output* com pessoa real e tentativa de *spoofing*

4.1.2.2 Conclusões

As soluções desenvolvidas foram ajustadas e alteradas tendo em conta o *dataset* previamente criado, contudo foi necessário fazer alguns ajustes e variações de hiperpârametros de forma a obter os melhores resultados possíveis e que foram expostos acima. À medida que um modelo era alterado, para além da análise à matriz de confusão era também testado com a câmara o resultado em *stream* de vídeo, para perceber como estava o modelo a comportar-se. Após os testes realizados o modelo *Random Forest* demonstrou uma *accuracy* melhor e principalmente um resultado mais estável.

Para explicar os resultados obtidos é necessário abordar a forma como estas soluções funcionam. O modelo *Random Forest* constrói árvores de decisão cada uma treinada num subconjunto de características que permitem que o modelo aproveite coletivamente os diversos aspetos das características de profundidade e capte relações complexas nos dados. As árvores de decisão proporcionam uma compreensão clara do impacto das características de profundidade individuais nas decisões de classificação. Em contrapartida a extração hierárquica de características das CNN pode não captar eficazmente os padrões de profundidade matizados que contribuem significativamente para a classificação. As imagens de profundidade com a sua representação tridimensional introduzem uma complexidade única na análise. Enquanto as CNNs destacam-se na aprendizagem de características hierárquicas a partir de dados 2D, a capacidade do modelo *Random Forest* para aproveitar a informação multidimensional de profundidade pode ser um fator decisivo para o seu melhor desempenho. O tamanho do *dataset* utilizado é também uma condicionante nos resultados obtidos pois as CNN exigem um conjunto de dados maior para aprenderem de forma eficaz as características e neste caso, o tamanho reduzido do *dataset* que foi utilizado é também um justificação para que o modelo *Random Forest* fornecesse classificações mais estáveis e fiáveis com os dados disponíveis.

4.1.3 Modelos Inception Resnet

Para o treino com a arquitetura *Inception ResNet* tanto na versão 1 como na 2, foram estudados hiperparâmetros importantes para afinar as redes de modo a obter a máxima *accuracy* e a mínima perda (*loss*). Estes hiperparâmetros incluíam o ajuste do tamanho do *batch*, o número de *epochs*, *learning rate* e *dropout*. Os modelos foram também avaliados para diferentes otimizadores, neste caso Adam e SGD. Para cada treino efetuado foram traçadas e analisadas 4 métricas separadas em 2 gráficos: *accuracy* em treino e teste e a *loss* em treino e teste. A figura 55 demonstra como é supervisionado o treino do modelo.

```
Epoch 1/100
88/88 [==============================] - 271s 3s/step - loss: 2.3185 - accuracy: 0.2000 - val_loss: 1.9394 - val_accuracy: 0.1529
Epoch 2/100
88/88 [==============================] - 279s 3s/step - loss: 2.0140 - accuracy: 0.2721 - val_loss: 1.8961 - val_accuracy: 0.2786
Epoch 3/100
88/88 [==============================] - 268s 3s/step - loss: 1.7479 - accuracy: 0.3445 - val_loss: 1.7707 - val_accuracy: 0.4700
Epoch 4/100
88/88 [==============================] - 242s 3s/step - loss: 1.5405 - accuracy: 0.4209 - val_loss: 1.4378 - val_accuracy: 0.6357
Epoch 5/100
88/88 [==============================] - 234s 3s/step - loss: 1.4114 - accuracy: 0.4618 - val_loss: 1.0416 - val_accuracy: 0.7386
```

Figura 55: *Output* do modelo em treino

Para treinar ambas as versões *Inception Resnet* em estudo v1 e v2 foi utilizado o *dataset* previamente criado. O maior problema encontrado durante os treino recaía sempre no *overfitting* devido ao reduzido tamanho do *dataset* como mostra a figura 56. Isto acontece quando a *accuracy* do treino é muito elevada, mas não consegue classificar corretamente as amostras externas (os dados de validação).

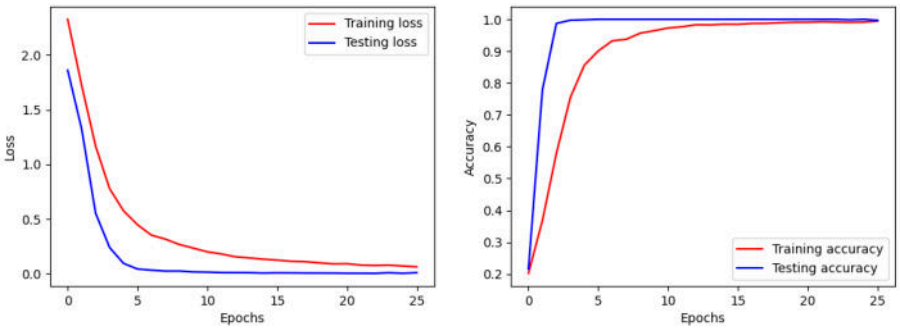


Figura 56: Treino de um modelo com *overfit*

Para além do *overfitting* outro problema recorrente foi a instabilidade do treino como demonstra a figura 57.

Tal como mencionado acima os testes realizados consistiram numa variação entre o otimizador Adam e SGD, na variação do *learning rate* entre 0.01 e 0.0001, bem como do valor do dropout, sendo que foram considerados os valores 0.3, 0.5, 0.65 e 0.8. O valor do *batch* foi também alterado entre 32 e 64 e os testes foram feitos de igual forma entre ambas as versões de forma a ser possível fazer uma comparação verdadeira dos resultados obtidos.

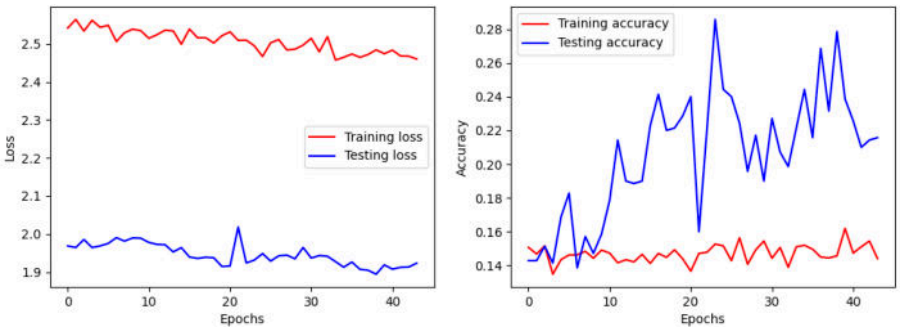


Figura 57: Treino de um modelo instável

4.1.3.1 Conclusões

Neste capítulo a CNN *Inception ResNet* foi treinada e validada com o *dataset* criado e foi realizado um estudo para otimizar os hiperparâmetros da rede com a ajuda dos gráficos, visualizando a convergência das topologias ao longo do tempo. A otimização desta CNN revelou-se um desafio, dado o número de gradientes e parâmetros apresentados na sua arquitetura.

Um dos maiores problemas enfrentados durante o treino foi o constante *overfitting*, mesmo com um baixo valor de *dropout*. Com os testes realizados foi claro que quando utilizado o otimizador Adam os resultados eram melhores comparativamente ao SGD. Relativamente aos valores de dropout, quando utilizados os valores de 0.3 e 0.8 os resultados eram muito instáveis, como se pode ver na figura 57 o que representou uma má capacidade de aprendizagem como será visto na secção seguinte. Apesar disto, relativamente ao valor do *batch* o seu efeito não trouxe uma conclusão concreta e o valor do *learning rate* quando aliado a certas definições de hiperpârametros e versão da arquitetura demonstrou uma tendência de melhores resultados com o valor 0.0001, contudo não se retirou uma conclusão clara relativamente a este parâmetro.

Contrariamente ao lido na literatura relativa a esta arquitetura, os melhores resultados foram obtidos com a arquitetura *Inception ResNet* V1. Um dos motivos que provocou estes resultados deve-se ao facto do modelo *Inception ResNet* V2 ser mais complexo, o que apesar de o tornar melhor para aprender torna também mais propenso a dar *overfit*, principalmente quando se possui um *dataset* pequeno, que é o caso. Os resultados obtidos levam a considerar que a versão 1 teve uma capacidade superior de lidar com as características do *dataset* em questão.

No geral esta topologia revelou-se muito difícil quando treinada com conjuntos de dados mais pequenos uma vez que o seu desempenho foi otimizado para conjuntos de dados maiores no que diz respeito à classificação.

Inicialmente o objetivo ao utilizar esta arquitetura passava por utilizar o modelo para classificação de faces, contudo ao analisar os resultados foi percetível que apesar da dificuldade em classificar corretamente as classes o modelo era capaz de extrair corretamente as características. Pelo que se passou para uma abordagem diferente onde o modelo era utilizado apenas para *facial feature extrator* utilizando um SVM para classificação.

4.1.4 Classificador SVM

O classificador SVM foi treinado com o modelo *FaceNet* da biblioteca *Keras* previamente treinado e também com um modelo da arquitetura *Inception ResNet*, tendo sido estes modelos testados com o *dataset* criado e mencionado na secção 3.2.3. Foi também obtida a matriz de confusão do teste verificando a classificação das classes. Foram consideradas 7 classes e para cada classe foram utilizadas 100 imagens de treino e 12 de teste (ver figura 58).

```
Loaded:  (700, 160, 160, 3) (700,) (84, 160, 160, 3) (84,)
```

Figura 58: *Output* do classificador SVM

Para treinar o SVM é necessário codificar num vetor *embedding* as características das imagens do *dataset*. Neste processo foram utilizados modelos pré-treinados bem como o modelo *FaceNet*.

Para uma análise e comparação dos resultados mais completa testaram-se diversos modelos que já foram anteriormente apresentados em 4.1.3. Foram analisados em concreto modelos para demonstrar a forma como um bom e um mau extrator de características afeta o desempenho do classificador SVM. Foi alterado o número de imagens dos *datasets* de forma a encontrar o número mínimo de imagens necessárias para que o modelo tenha um bom desempenho. Após o treino feito, os ficheiros gerados no formato *pickle* são guardados num dado diretório para serem posteriormente carregados e utilizados para categorizar dadas amostras (ver figura 59).

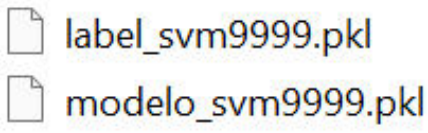


Figura 59: Ficheiros *pickles*

4.1.4.1 Resultados do SVM

Este secção pretende explicar e analisar os resultados para os diversos modelos tendo como base 6, 10, 50 e 100 imagens de treino por classe. Foram considerados 3 modelos bons e 1 mau de forma a, para além de comparar os modelos bons entre si, perceber o diferencial causado em ambos os tipos de modelos pelo número de fotos utilizado.

Na tabela 4 estão representados os dados obtidos para os modelos utilizando um *dataset* de treino com 100 imagens. Esta foi a configuração utilizada para realizar todos os testes, pelo que a comparação será com os valores abaixo expressos.

À medida que o número de imagens por classe diminuía, a *accuracy* em treino obtida pelo pior modelo, modelo 4, aumentava. Contudo este modelo não é considerado para testes devido aos maus resultados que ele possui. Relativamente aos restantes modelos pode ser afirmado que os melhores resultados são

Modelo Inception Resnet	Accuracy Treino (%)	Accuracy Teste (%)
Modelo 1	97.714	98.809
Modelo 2	97.143	100.000
Modelo 3	98.286	98.810
Modelo 4	31.857	26.190

Tabela 4: *Accuracy* obtida nos 4 modelos (100 imagens por classe)

Modelo Inception Resnet	Accuracy Treino (%)	Accuracy Teste (%)
Modelo 1	96.000	97.619
Modelo 2	96.000	97.619
Modelo 3	97.143	98.810
Modelo 4	33.429	22.619

Tabela 5: *Accuracy* obtida nos 4 modelos (50 imagens por classe)

Modelo Inception Resnet	Accuracy Treino (%)	Accuracy Teste (%)
Modelo 1	92.857	94.048
Modelo 2	97.143	98.810
Modelo 3	97.143	97.619
Modelo 4	40.000	26.190

Tabela 6: *Accuracy* obtida nos 4 modelos (10 imagens por classe)

para 100 imagens por classe (ver tabela 4), nomeadamente o modelo 2. Quando comparado modelo a modelo apesar de resultados similares entre o modelo 1, 2 e 3 com 50 imagens (ver tabela 5) e 10 imagens (ver tabela 6), o modelo 1 tem uma diminuição de *accuracy* à medida que o número de imagens diminui. O modelo 2 e 3 obtém resultados estáveis e bastante similares entre si apesar das diferentes configurações do *dataset*.

Foi também analisado o tempo médio necessário para treinar o classificador SVM com diferente número de imagens por classe. Na tabela 8 encontram-se os valores de tempo médio obtidos por número de imagens.

Os valores obtidos em segundos não são extremamente elevados pelo que manter o *dataset* com as 100 imagens não é prejudicial quando for necessário treinar novamente o modelo por se ter adicionado alguém ao *dataset*. Contudo se o número de classes adicionadas for elevado ou efetivamente for necessário treinar o classificador no mínimo tempo possível, não há problema em considerar por exemplo o *dataset* com 10 imagens por classe, uma vez que o tempo é bastante próximo do mínimo retirado e os

Modelo Inception Resnet	Accuracy Treino (%)	Accuracy Teste (%)
Modelo 1	88.095	92.857
Modelo 2	95.238	98.810
Modelo 3	92.857	97.619
Modelo 4	42.857	22.619

Tabela 7: *Accuracy* obtida nos 4 modelos (6 imagens por classe)

Número de imagens por classe	Tempo (s)
6	11
10	12
50	32
100	52

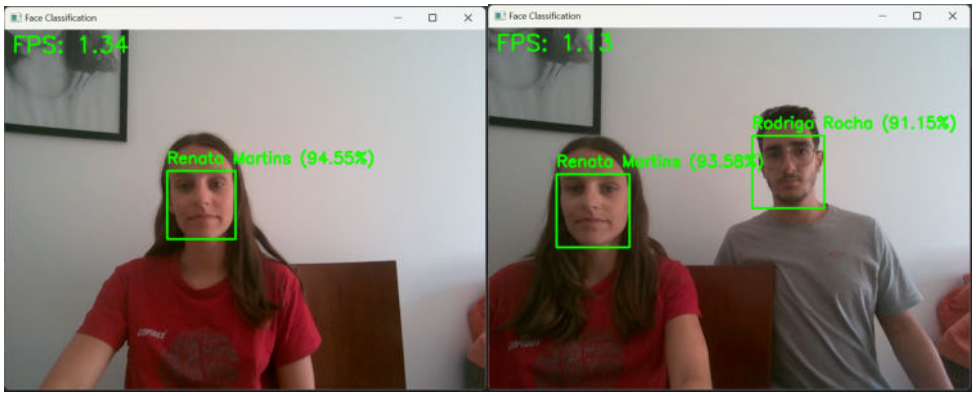
Tabela 8: Tempo médio para treino do classificador SVM

resultados obtidos demonstram um rendimento bom.

4.1.4.2 Reconhecimento Facial com SVM em *stream* de vídeo

Foi também testado o correto funcionamento do classificador SVM em tarefas de reconhecimento facial em *stream* de vídeo. Foram utilizados os modelos *FaceNet* do *Keras* e o modelo 2 da arquitetura *Inception ResNet*. O modelo 2 foi treinado com a arquitetura *Inception ResNet V1*, com um tamanho de *batch* de 64, com o otimizador *Adam*, um *learning rate* de 0.0001 e uma taxa de *dropout* de 0.5.

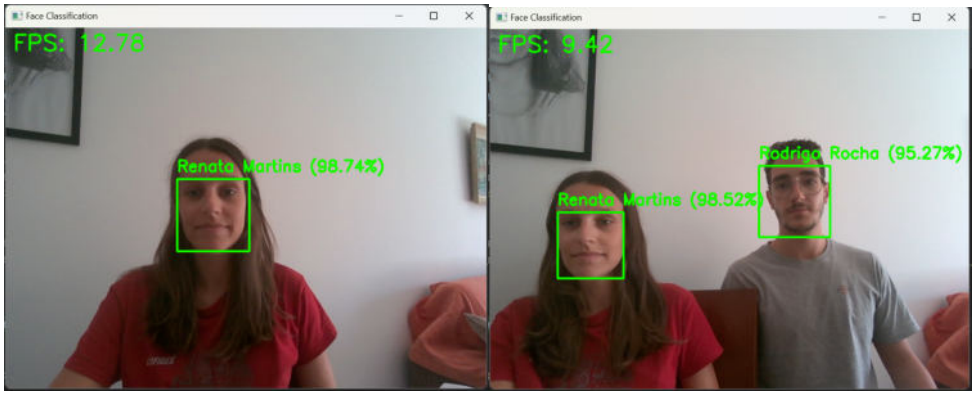
Em *stream* de vídeo são utilizados os 3 módulos mencionados anteriormente, a) deteção facial através de *MediaPipe*, b) a matriz das faces é introduzida no extrator de características *Inception ResNet* ou no modelo pré-treinado *FaceNet* do *Keras* e obtido o vetor *embedding* e c) o vetor resultado do passo b) passa por um classificador SVM treinado, obtendo a identidade e a confiança associada. Se a confiança for superior a um determinado limiar de previsão, neste caso 80%, é apresentada a imagem original com a identidade correspondente associada a uma caixa delimitadora. Nas figuras 60 e 61 estão exemplos de quando está presente na imagem apenas uma pessoa e quando estão dois utilizadores a ser detetados e classificados ao mesmo tempo.



(a) (b)

Figura 60: Resultados com o modelo FaceNet para a) um utilizador e b) dois utilizadores

Ao utilizar o modelo *FaceNet* o número de fps obtidas é bastante mais baixo do que quando utilizado o modelo 2. Isto deve-se ao facto do modelo *FaceNet* ser de uma maior complexidade, o que torna o processamento mais pesado e consecutivamente mais lento.



(a) (b)

Figura 61: Resultados com o modelo 2 da arquitetura *Inception Resnet* V1 para a) um utilizador e b) dois utilizadores

Na figura 62 está representada uma situação onde está presente um utilizador conhecido e um desconhecido. Neste caso e tal como é visível quando se tratam de utilizadores desconhecidos é classificado como "*Unknown*". A percentagem associada ao utilizador desconhecido corresponde ao valor de confiança que o modelo considera que aquela pessoa tem, por outras palavras, o modelo assume com 45.57% de confiança que o utilizador desconhecido é alguém do *dataset*, mas sendo a percentagem tão baixa, é considerado alguém desconhecido. Este valor é meramente informativo para quem analisa os resultados, sendo que a informação relevante é ser alguém *"Unknown"*.

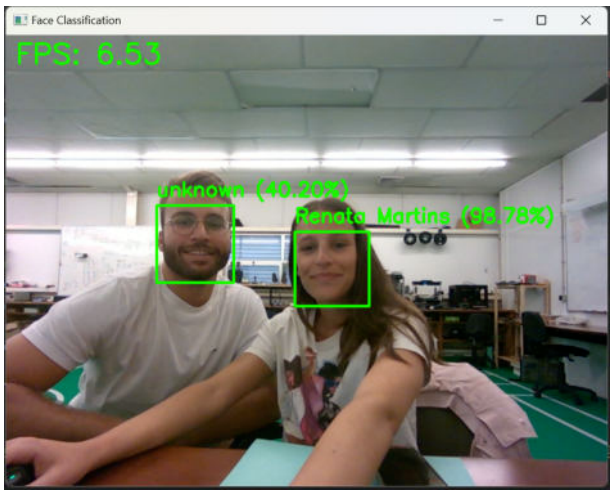


Figura 62: Resultado com o modelo 2 com um conhecido e um desconhecido

Foi também analisado o caso onde estão 3 utilizadores presentes na imagem como na figura 63. Neste caso estão dois utilizadores conhecidos e um utilizador desconhecido.

De forma a sustentar os resultados obtidos, na figura 64 estão apresentadas matrizes de confusão, traçadas após a execução da previsão de cada subconjunto de teste. Estas apresentam uma representação visual das previsões feitas pelo modelo em comparação com as classes reais. Esta ferramenta de

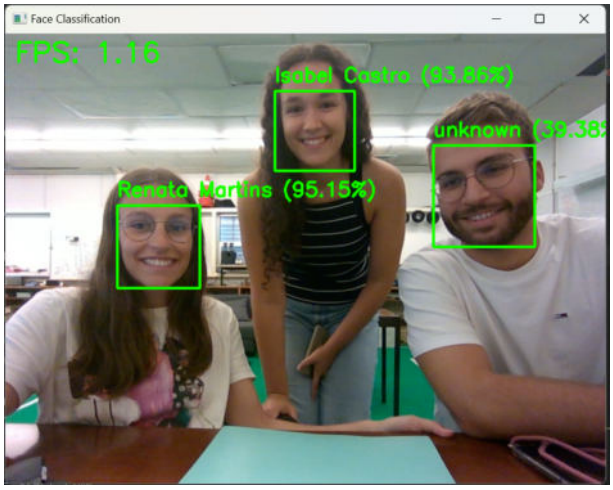
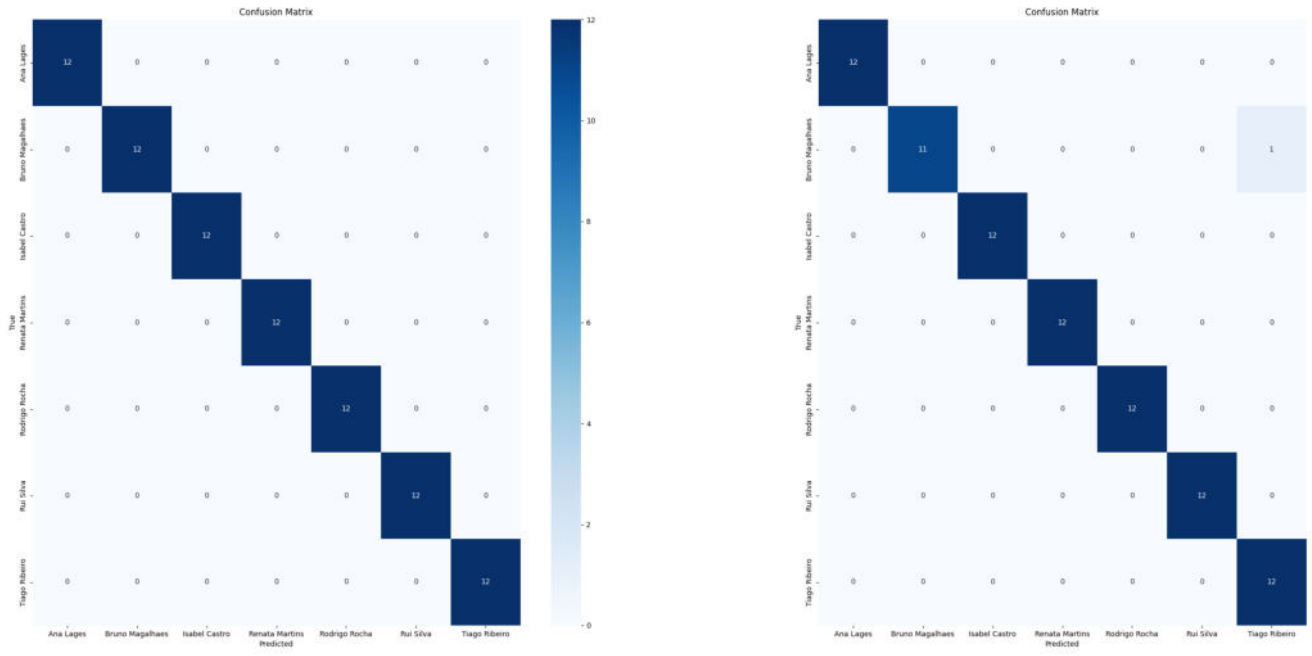


Figura 63: Resultado com o modelo 2 com dois conhecidos e um desconhecido

visualização foi importante para compreender a forma como o SVM se estava a comportar.



(a) (b)

Figura 64: Matrizes de confusão dos melhores modelos utilizados a) *FaceNet* e b) modelo 2 *Inception ResNet* V1

4.1.4.3 Conclusões

Ao testar os módulos anteriormente implementados em *stream* de vídeos os resultados obtidos demonstraram sucesso na tarefa. O sistema de reconhecimento com o modelo proveniente do treino com a arquitetura *Inception ResNet* V1 manteve entre 9 a 14 fps, sendo mais elevado quando estava apenas uma pessoa na imagem e descendo quando existiam mais faces. Relativamente ao modelo *FaceNet*, o número de *fps* obtidas é bastante mais baixo estando a justificação para isto acima retratada. Através dos resultados dos testes efetuados é possível concluir que os módulos implementados anteriormente, nomeadamente deteção facial, extração de características e reconhecimento facial funcionaram em conjunto com sucesso. Ao usar o *dataset* criado utilizando ambos os modelos o sistema foi capaz de identificar corretamente as faces e com confianças elevadas. Algumas possíveis falhas que possam existir justificam-se com um pequeno *dataset* e pouco adaptado a diferentes situações como variações de poses, estar com óculos ou sem óculos entre outras condições.

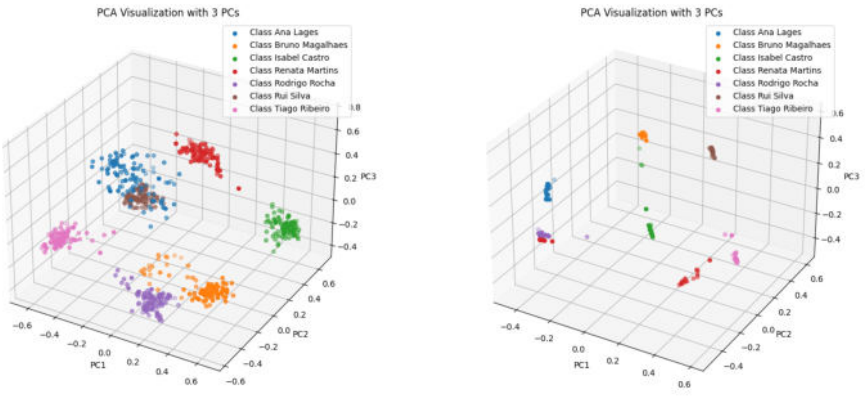
4.1.5 Visualização e análise de *embeddings* de faces utilizando PCA e t-SNE

Nesta secção são apresentados os resultados da análise sobre a visualização de *embeddings* de faces utilizando a análise de componentes principais PCA e a incorporação de vizinhos estocásticos distribuídos t-SNE. Estas técnicas são amplamente utilizadas na aprendizagem automática e na visualização de dados para reduzir dados de alta dimensão em representações de baixa dimensão que podem ser facilmente visualizadas, preservando a estrutura subjacente.

4.1.5.1 Principal Component Analysis

Foi aplicada a técnica PCA ao conjunto de dados de *embeddings* de faces para reduzir a dimensionalidade a três componentes principais. Cada componente principal captura uma combinação linear das características originais, permitindo visualizar os dados num espaço tridimensional.

A visualização da PCA revelou grupos distintos dentro do *dataset*, correspondendo a diferentes classes onde cada ponto no gráfico de dispersão tridimensional representa uma incorporação de rosto individual, e as cores representam as classes. Notavelmente, a visualização destaca a separabilidade de diferentes classes, sugerindo que os *embeddings* de rosto carregam informações essenciais de discriminação. O resultado da PCA num exemplo onde foi utilizado o modelo pré-treinado *FaceNet*, é apresentado na figura 65(a). Na figura 65(b) está o resultado PCA de um modelo com a arquitetura *Inception ResNet* onde é possível perceber que o PCA provou que o extrator de características *Inception ResNet* é capaz de diferenciar classes num espaço euclidiano de dimensão inferior.



(a) (b)

Figura 65: Análise do PCA com o modelo a) *FaceNet* e b) *Inception ResNet* V1

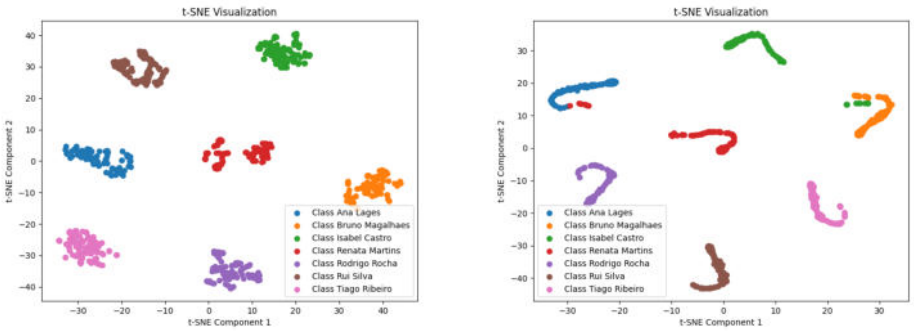
4.1.5.2 t-distributed Stochastic Neighbor Embedding

Tal como mencionado anteriormente, o *t-SNE* é uma técnica de redução de dimensionalidade não linear que enfatiza a preservação das relações locais entre os pontos de dados no espaço reduzido. Inicialmente é calculada a probabilidade de semelhança dos pontos no espaço de alta dimensão e calculada a probabilidade de semelhança dos pontos no espaço de baixa dimensão correspondente.

O t-SNE é utilizado para mapear os *embeddings* faciais num espaço bidimensional, facilitando uma visualização abrangente dos dados.

A visualização t-SNE exibe um arranjo dos pontos dos dados. Os *clusters* correspondentes a diferentes classes são evidentes, e as distâncias relativas entre os pontos sugerem a sua similaridade ou dissimilaridade. A capacidade da visualização de captar padrões complexos nos dados melhora a nossa compreensão das relações entre as diferentes classes.

Depois de calculadas as interações suficientes, as imagens foram agrupadas de acordo com os vizinhos semelhantes. Analisando a figura 66(a) e a figura 66(b), as imagens em todas as localizações espaciais eram da mesma identidade, comprovando uma boa união entre as categorias para ambos os modelos utilizados. Contudo é preciso salientar que quando utilizado o modelo *Inception ResNet* é visível que alguns pontos de diferentes classes estão na mesma região do gráfico. Isto acontece devido a parecenças existentes nas imagens obtidas daquelas classes, tanto a nível do tipo de fotos como parecenças físicas entre duas classes.



(a) (b)

Figura 66: Análise do t-SNE com o modelo a) *FaceNet* e b) *Inception ResNet* V1

4.1.6 Aplicação no robô CHARMIE com recurso a ROS2

Esta secção apresenta os resultados obtidos relativamente à aplicação no robô CHARMIE de todos os módulos apresentados anteriormente. Para isto e tal como mencionado foi utilizado o ROS2 para a integração dos módulos.

O objetivo passa então por demonstrar o correto funcionamento da tarefa de reconhecimento facial do robô e para isto é então preciso verificar se a tarefa de deteção facial e classificação está a funcionar corretamente. É também relevante demonstrar a capacidade do robô CHARMIE de perceber quando o seu utilizador principal está presente na imagem e que neste caso trata-se da "Renata Martins".

Na figura 67, está um exemplo quando está presente na imagem uma pessoa. Aa figura 68 mostra o *output* do módulo *speaker* mencionado em 3.3.2.2. Nestas figuras é possível verificar que para além de a pessoa ser corretamente classificada, é pronunciado através da voz do robô o nome correspondente.

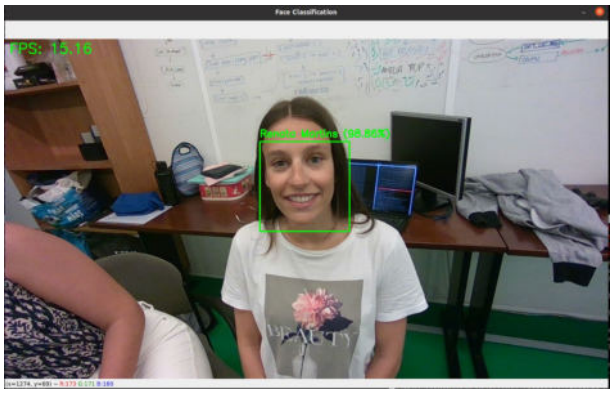


Figura 67: Resultado obtido com o modelo 2 com pessoa presente

```
Received String: ['Renata Martins']
[INFO] [1694008635.379593194] [Speaker]: Received Speech String
```

Figura 68: *Output* do módulo *Speaker*

Os mesmo testes foram feitos para quando estão duas pessoas presentes na imagem, tanto com duas pessoas conhecidas, figuras 69 e 70, mas também quando apenas uma pessoa é conhecida, figuras 71 e 72.

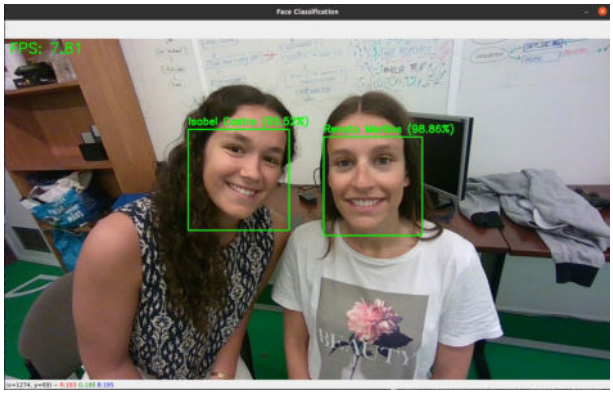


Figura 69: Resultado com o modelo 2 com duas pessoas conhecidas

```
Received String: ['Renata Martins', 'Isabel Castro']
[INFO] [1694078412.181195283] [Speaker]: Received Speech String
```

Figura 70: *Output* do módulo *Speaker*

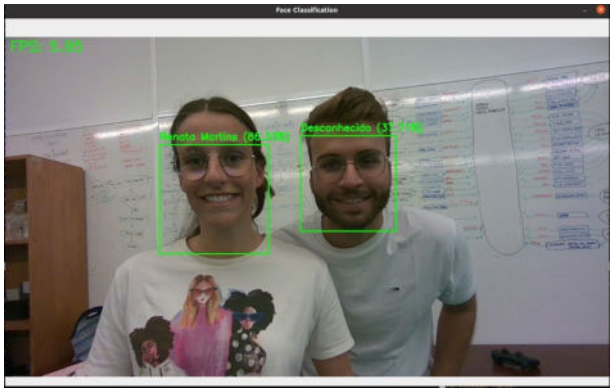


Figura 71: Resultado com o modelo 2 com um conhecido e um desconhecido

```
Received String: ['Desconhecido', 'Renata Martins']
[INFO] [1694010070.284948477] [Speaker]: Received Speech String
```

Figura 72: *Output* do módulo *Speaker*

De forma a validar também o correto funcionamento quando estão presentes mais pessoas na imagem, as figuras 73 e 74 mostram o resultado obtido quando estão presentes 3 pessoas na imagem.

Na figura 75 está visível o *output* quando é detetada a pessoa "Renata Martins"na imagem. Desta forma é possível verificar que independentemente de quando a mesma aparece, é reconhecida e identificada como sendo o utilizador principal do robô.

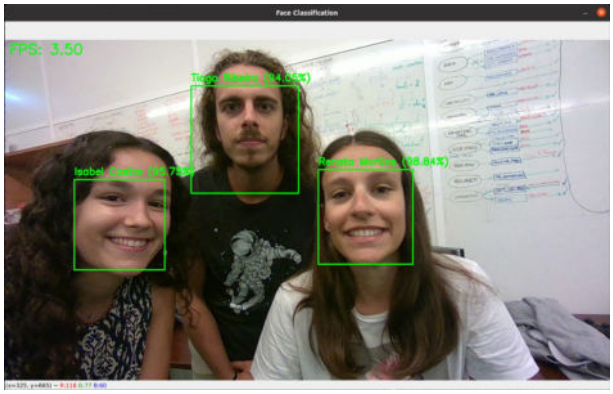


Figura 73: Resultado com o modelo 2 com três pessoas conhecidas

```
Received String: ['Renata Martins', 'Isabel Castro', 'Tiago Ribeiro']
[INFO] [1694078553.578883288] [Speaker]: Received Speech String
```

Figura 74: *Output* do módulo *Speaker*

```
Renata Martins chegou- Principal utilizador
```

Figura 75: *Output* quando a "Renata Martins"é detetada

Foi também analisado e comparado o resultado obtido quando os utilizadores estão diferentes do *dataset*, por exemplo se o penteado é diferente e se estão ou não de óculos comparativamente às imagens retiradas e o resultado obtido está na imagem 76.

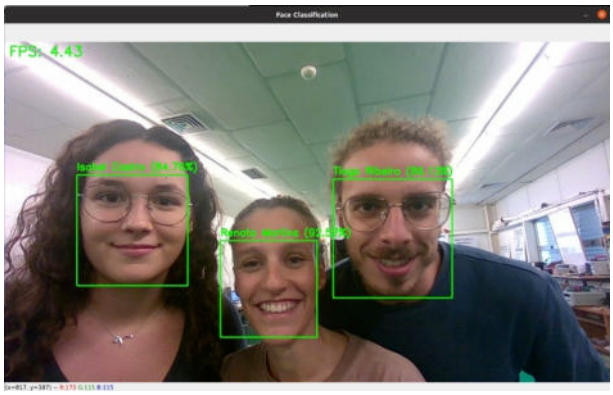


Figura 76: Comparação da *accuracy* quando os utilizadores estão diferentes comparativamente ao dataset

Na tabela 9 é possível ver a diferença entre a *accuracy* obtida para cada uma das pessoas. Como seria de esperar a percentagem associada a cada identidade diminuiu quando as pessoas estão em condições diferentes do que estavam quando o *dataset* foi obtido.

4.1.6.1 Conclusões

Os objetivos principais desta dissertação são demonstrados nesta secção do documento uma vez que com os resultados obtidos é possível afirmar que apesar de algumas limitações associadas aos modelos treinados os resultados são satisfatórios, na medida em que foi comprovado o correto funcionamento

Utilizador	Utilizador semelhante	Utilizador diferente
Isabel Castro	96%	84%
Renata Martins	98%	92%
Tiago Ribeiro	94%	86%

Tabela 9: Comparação da *accuracy* obtida entre o utilizador nas mesmas/diferentes condições do *dataset*

do reconhecimento facial quando aplicados os módulos desenvolvidos em ROS2 no robô. Tal como é possível verificar acima, independentemente do número de pessoas, a classificação está correta e é possível notar que à medida que aumenta o número de faces presentes na imagem, o número de fps baixa, contudo os valores obtidos são aceitáveis uma vez que se considerou 7 fps como o valor mínimo . Apesar dos resultados expostos, foi percetível que há uma necessidade de proximidade da câmara para uma correta identificação das faces e consequentemente correta classificação. Para além de testado o modelo 2 abordado anteriormente foi também analisado o funcionamento do modelo *FaceNet*. Foi percetível que principalmente quando está presente na imagem um utilizador desconhecido, o modelo tem um comportamento melhor e mais estável, isto advém do modelo *FaceNet* ter sido treinado com uma grande quantidade de dados, o que ajuda a extrair características genéricas de rostos humanos. Isto pode permitir que o modelo *FaceNet* generalize melhor para rostos desconhecidos, uma vez que aprendeu representações abrangentes. Algo que não acontece com o modelo 2 que foi treinado com um *dataset* bastante mais pequeno. Relativamente à comparação da *accuracy* obtida quando os utilizadores estão em diferentes condições do que estavam no *dataset*, é notório que a percentagem de *accuracy* associada a cada identidade é menor, contudo os resultados são satisfatórios na medida em que consegue reconhecer corretamente os utilizadores.

4.2 Abordagem usada no RoboCup@Home

O principal teste ao trabalho desenvolvido no âmbito desta dissertação na abordagem competitiva foi na prova Rececionista e *Stickler for the Rules* do RoboCup@Home.

4.2.1 Prova Rececionista e *Stickler for the Rules*

Na prova Rececionista diversas questões foram necessárias validar, nomeadamente:

- a voz do robô e o áudio, que é verificado tendo em conta se o robô conseguiu compreender corretamente o que lhe é dito ou não
- os *LEDs RGB* que funcionaram como ferramenta de *debug* e foram utilizados por exemplo para confirmar a correta captura da fotografia do convidado
- o pescoço, validando se o mesmo olhou para os locais corretos e também se se ajustou tendo em consideração a face da pessoa.

- a imagem obtida, é também analisada no que toca à correta captura a partir da câmara e é também analisada a correta extração das características
- deteção de faces

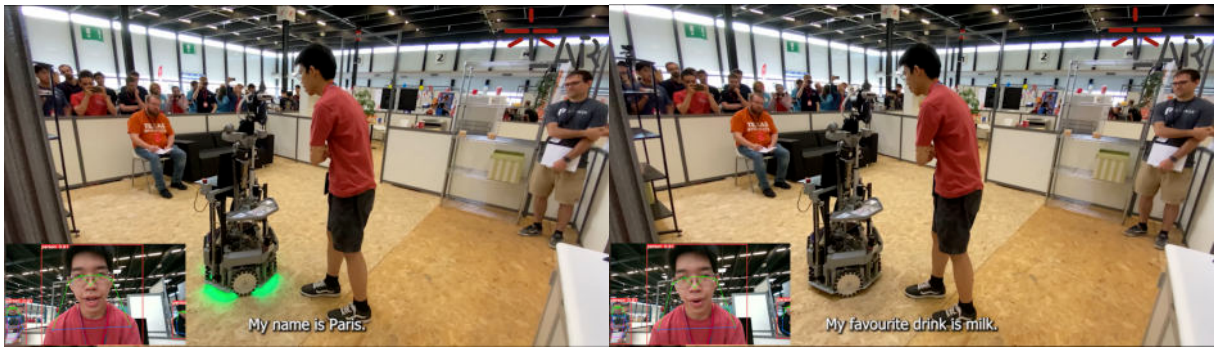
Na prova *Stickler for the rules*, foi necessário validar:

- a voz do robô e os *LEDs RGB* tal como no caso anterior
- deteção de pessoas

Os resultados obtidos que são comuns às duas provas são expostos nas secções seguintes. Os módulos que foram utilizados em apenas uma das provas são explicados posteriormente

4.2.1.1 Áudio

Na prova do Rececionista é solicitado ao convidado que indique o seu nome e bebida favorita de uma lista de possíveis opções. Tanto o convidado 1 como o 2 tiveram de indicar estas informações ao robô, como é possível verificar através das imagens 77 e 78.



(a) (b)

Figura 77: Informação dada pelo convidado 1, nome e bebida favorita

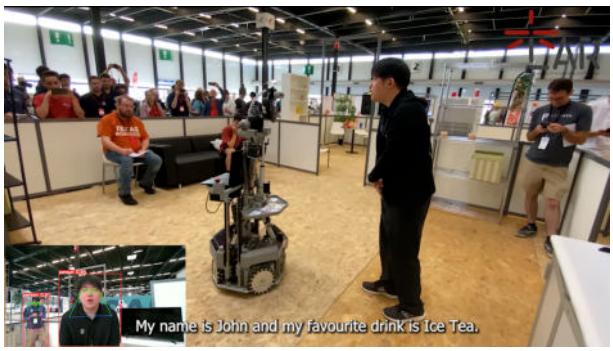
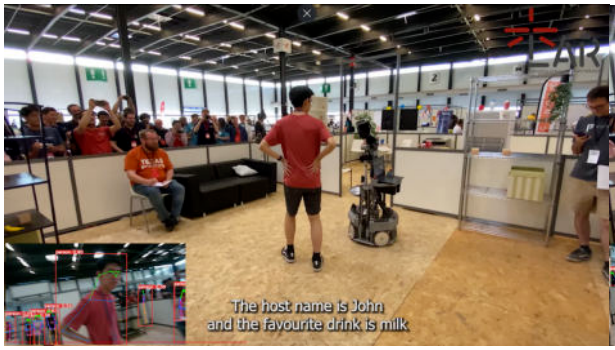
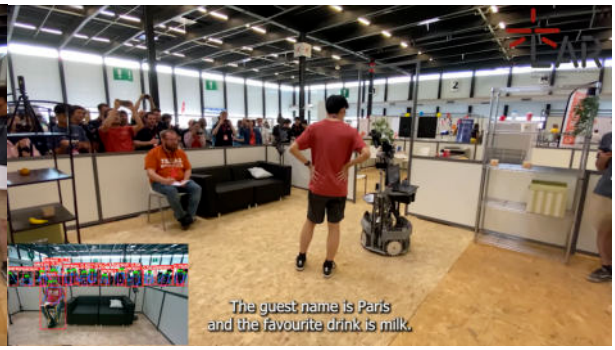


Figura 78: Informação dada pelo convidado 2, nome e bebida favorita



(a) Apresentação do *host*



(b) Apresentação do convidado 1



(c) Apresentação do convidado 2

Figura 79: Informação retida pelo robô e transmitida na apresentação dos elementos presentes na sala

É também possível verificar na figura 79, que para as duas situações o robô compreendeu corretamente aquilo que lhe foi dito.

A partir deste momento o robô tem do seu lado as informações relativas ao anfitrião e dos dois convidados que foram obtidas a partir do áudio.

4.2.1.2 Voz

A voz é um dos recursos mais utilizados e importantes durante a prova uma vez que é proibido algum elemento da equipa comunicar com os convidados e dar algum tipo de ajuda ou informação, obrigando o robô a dar todas as informações necessárias, as quais tanto são relativas à fase da prova em que o robô se encontra como àquilo que precisa que façam ou digam.

Tendo isto em consideração estão abaixo exemplos disponíveis de falas que o robô teve necessárias para o decorrer da prova.

Na figura 80 está um exemplo de uma comunicação que foi feita no início da prova para o júri, que o robô está pronto para começar.

No final da prova há também um exemplo (ver figura 81) da comunicação feita pelo robô para demonstrar o término da tarefa e que o robô finalizou tudo o que tinha a fazer, sendo então uma informação para o árbitro que está a avaliar a prova e a cronometrar o tempo.



Figura 80: Imagem referente ao inicio da prova do Rececionista



Figura 81: Imagem referente ao fim da prova do Rececionista

Na prova *Stickler for the Rules* a voz do CHARMIE mais uma vez foi utilizada para falar com os convidados. Nesta prova o CHARMIE fala com o convidado quando deteta que o mesmo está a desrespeitar uma regra. A ação a ser tomada é indicada pelo robô e fica a repetir essa ação enquanto o convidado não sair efetivamente do espaço. Após a ação realizada o mesmo refere que o convidado respeitou a regra e vai avançar.

4.2.1.3 Navegação

Tal como mencionado no capítulo anterior a parte relativa à navegação representa algo importante nas provas, nomeadamente pela necessidade de se tratar de um problema com visão por computador e no caso de o robô não se mover para uma dada posição ou ficar orientado para um local errado, o que pode implicar o sucesso da prova. Se o robô olhar na direção de navegação são-lhe atribuídos mais pontos. Assim à semelhança dos humanos quando o CHARMIE se move, olha ligeiramente para o chão de forma a indicar que está a olhar para o caminho que está a seguir (ver figura 82).

O robô apesar de ter como destino um local e uma orientação previamente definida, tem a capacidade de se desviar de obstáculos fixos ou móveis durante a navegação pela casa.

Na prova *Stickler for the Rules* o CHARMIE navegou pela casa até um ponto específico do quarto,

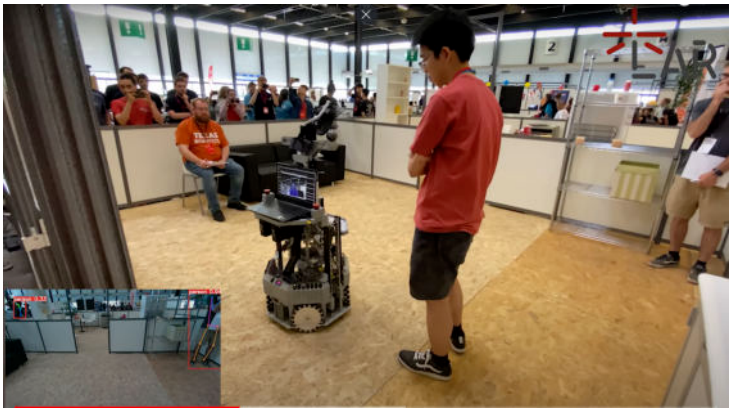


Figura 82: Imagem referente à navegação do robô onde o pescoço orienta-se para baixo

desviando-se dos obstáculos que existem na sala, sendo que nessa divisória para onde se dirige deve ser capaz de fazer uma espécie de varrimento pelo quarto até detetar alguém. Este movimento é feito alterando a orientação do robô e quando detetar alguém o seu movimento deve ser com o objetivo de se orientar sempre para a direção da pessoa de forma a falar fazendo contacto visual e ainda mais importante, perceber se o convidado está a respeitar as regras. Na figura 83(a) está um exemplo do robô a olhar para a pessoa detetada e na figura 83(b) é visível a olhar para a pessoa, embora neste caso estar fora do quarto.



(a) (b)

Figura 83: Deteção de pessoa em espaço proibido: a) dentro ou b) fora

4.2.1.4 Sistema de *Debug*

Relativamente ao sistema de *debug* na prova do Rececionista o *buzzer* não foi utilizado uma vez que o botão de emergência não foi premido, e as baterias estavam na carga máxima quando foram colocadas no robô. Contudo o mesmo foi utilizado em situações de teste e treino da prova, pelo que foi validado que o mesmo funciona. Na prova *Stickler for the Rules*, o botão de Emergência foi premido uma vez que tal como mencionado anteriormente não foi desenvolvido todo o *software* para a prova e era apenas abordada uma das regras. Já o sistema de *debug* visual (LEDs RGB) foi utilizado em diferentes situações.

Na prova do Rececionista, a figura 84 mostra a luz com a cor roxa após ter sido premido o *"start button"* .



Figura 84: Sinal luminoso referente ao inicio da prova associado a premir o *start button*

Na interação entre o CHARMIE e o humano após o robô compreender e validar o nome e bebida mencionados, os *LEDs* acendem com a luz verde como forma de demonstrar sucesso na execução deste processo.

Os *LEDs* também são utilizados para servir como ponto de referência para os humanos. Normalmente os utilizadores do robô, quando não estão habituados a lidar com robôs, não têm noção de quando o robô está a ouvir ou preparado para executar uma ação. Na figura 85 está demonstrada uma situação onde o CHARMIE solicita que o convidado apenas responda às questões após acender a luz verde. Isto permite que o convidado tenha conhecimento sobre quando deve executar uma ação ajudando ao sucesso da prova.



Figura 85: Informação transmitida aos convidados para saberem quando responder

Como é possível verificar através da figura 77(a) o convidado apenas está a responder quando teve a sinalização com os *LEDs* verdes.

Na prova *Stickler for the Rules* os *LEDs* são utilizados cada vez que é detetada uma regra quebrada e quando o convidado respeita a regra. Quando o robô deteta o convidado na sala proibida os *LEDs* acendem

com as cores azuis e vermelhas. Quando o convidado segue a ação a ser tomada, nomeadamente sair da sala proibida, acende uma luz verde como forma de confirmar que o convidado saiu.

4.2.2 Módulos relativos à prova do Rececionista

4.2.2.1 Deteção Facial

Relativamente à deteção facial há dois módulos que devem ser abordados nomeadamente o pescoço e a câmara, uma vez que um módulo sem o outro não permitiria o sucesso nesta componente. Para a deteção facial é necessária a câmara para detetar as pessoas e é necessário o pescoço para fazer o ajuste do centro da face com o centro da imagem.

Na figura 86 é possível analisar a imagem que inicialmente o robô tinha. São detetados os *keypoints* do tronco mas a face não é visível, o que apesar do robô estar em condições para detetar uma pessoa com uma elevada percentagem de confiança, sendo necessário obter imagens da face o resultado não está adequado.

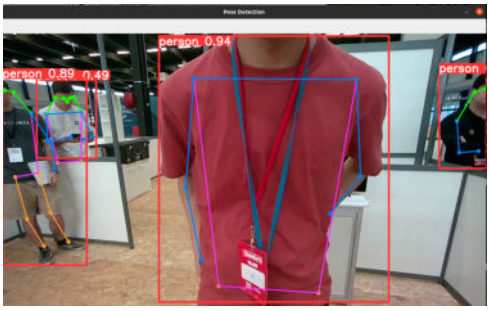


Figura 86: Imagem inicial com face não visível

Foi necessário um ajuste do pescoço para centrar o nariz com o centro da imagem e o resultado deste ajuste é visível na figura 87.

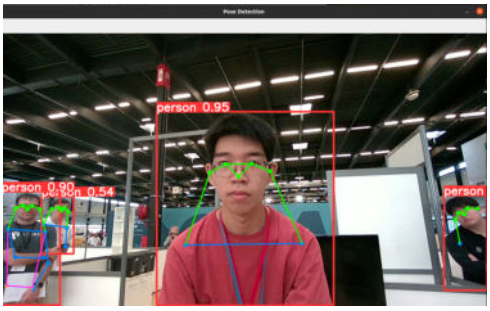


Figura 87: Imagem após o ajuste do pescoço

Na figura 88 é possível verificar que os *keypoints* da face são detetados mas não estão centrados com o centro da imagem.

Tal como na primeira situação do convidado 1 a imagem foi ajustada para o centro da face corresponder ao centro da imagem e neste caso a imagem ajustada está na figura 89.

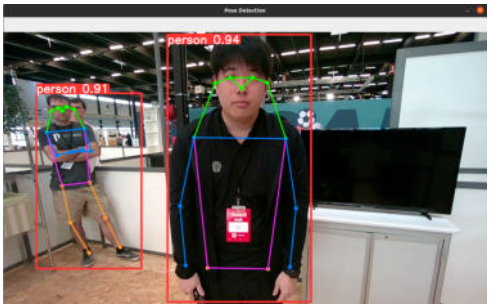


Figura 88: Imagem inicial com face está visível mas não centrada

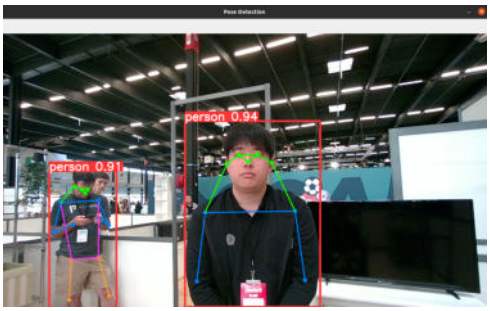


Figura 89: Imagem após o ajuste do pescoço

Após o ajuste do pescoço é possível capturar imagens para extração das características.

4.2.2.2 Recolha de Imagens

A captura de imagens foi apenas executada para o convidado 1 uma vez que em contexto de prova apenas foi necessário extrair as características do convidado 1. Na figura 90 é possível visualizar a imagem que foi guardada e processada.



Figura 90: Imagem capturada do convidado 1

De salientar que a equipa LAR@Home na prova do Rececionista obteve 800 pontos tendo ficado em 1° lugar empatada com outras duas equipas. De referir que a pontuação máxima que era possível obter são os 800 pontos uma vez que este robô não possui braços para abrir a porta. Em [51] está disponível o vídeo relativo à prova onde é possível ver tanto o ponto de vista do robô, como o ponto de vista dos espetadores.

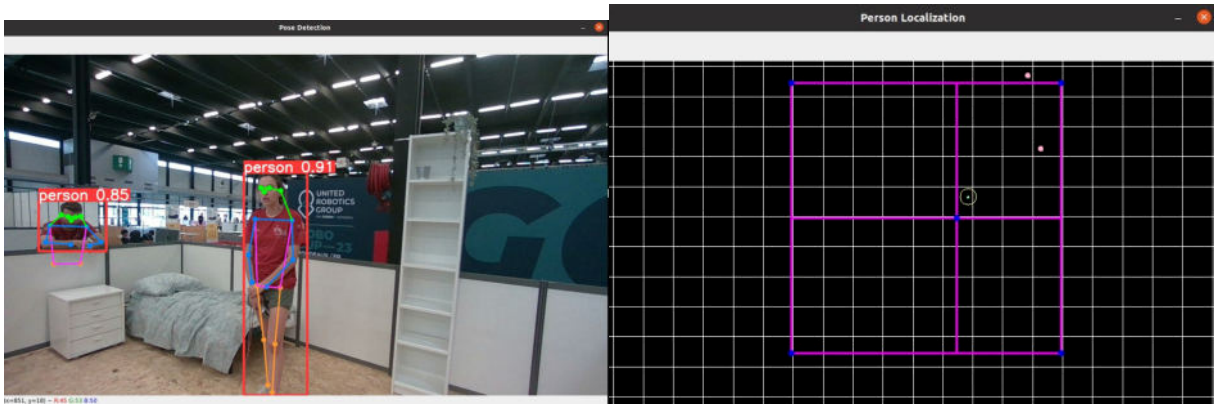
4.2.3 Módulos relativos à prova *Stickler for the Rules*

De seguida serão apresentados os resultados obtidos relativamente à deteção de convidados na sala proibida no âmbito da prova *Stickler for the Rules*.

4.2.3.1 Deteção de pessoas

Para validar o módulo de deteção de pessoas são usados a voz e os *LEDs RGB* para comunicar e transmitir o que está a acontecer.

Para a deteção de pessoas foi necessário utilizar a câmara e a base omnidirecional do robô, pois ao entrar na sala proibida desloca-se para um canto estratégico do espaço e a partir daí faz uma rotação da base até ter um *output*. Se encontrar uma pessoa, como está demonstrado na figura 91(a), é associado à coordenada desta pessoa uma localização da casa, como se pode ver na figura 91. Após passada a informação através da voz e dos *LEDs RGB*, é comparada novamente a localização da pessoa, quando a mesma se encontra noutra divisão que não o quarto, como na figura 92, e nesse caso é feito outro aviso por parte do robô, terminando assim a componente da prova de encontrar alguém na sala proibida.



(a) (b)

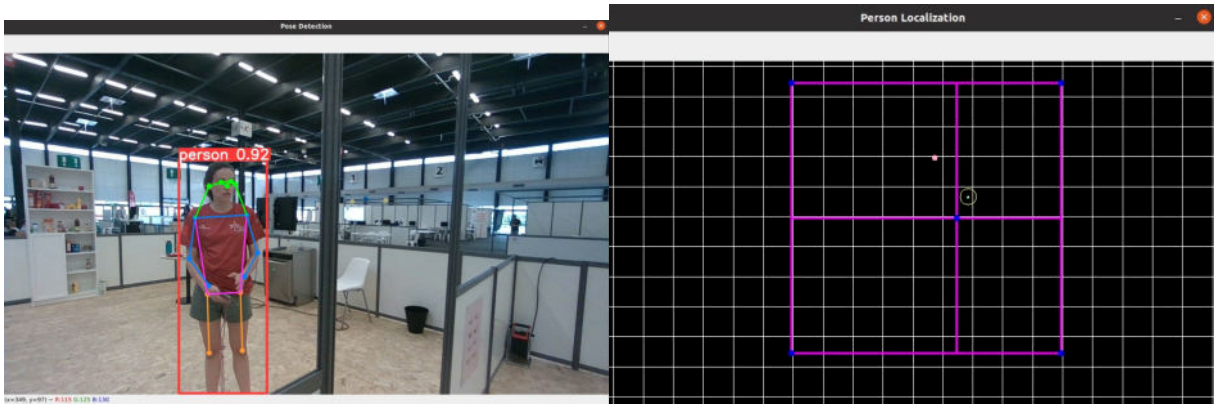
Figura 91: *Output* da posição das pessoas detetadas na casa

Nesta prova foram obtidos 100 pontos relativos à correta identificação de alguém na sala proibida.

4.3 Extração de Caraterísticas

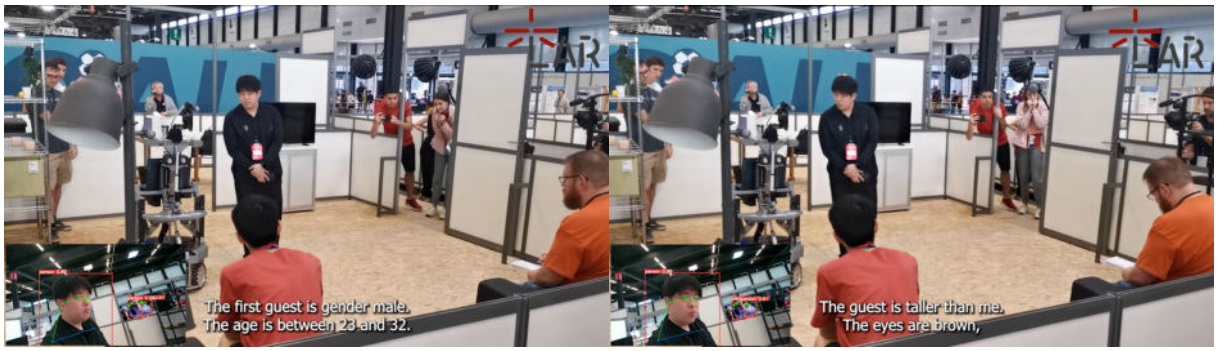
Os resultados obtidos relativamente à extração de características referem-se tanto à abordagem social como à abordagem competitiva, pois são uma parte integrante da dissertação e não exclusiva de nenhuma das abordagens. De seguida serão demonstrados os resultados obtidos neste módulo que foram recolhidos num contexto de prova e os mesmo estão representados na figura 93.

Como é possível observar a etnia dita pelo robô não corresponde à etnia do convidado 1, contudo após a competição e uma análise do trabalho e dos resultados obtidos foi possível confirmar que o CHARMIE



(a) (b)

Figura 92: *Output* da posição das pessoas detetadas, a) fora da casa e b) dentro da casa



(a) (b)



(c)

Figura 93: Informação do género, idade, etnia, altura e cor dos olhos do convidado 1

avaliou a etnia não do convidado mas sim de uma segunda pessoa na imagem, que como se vê na figura 94 foram detetados um árbitro e o convidado 1.

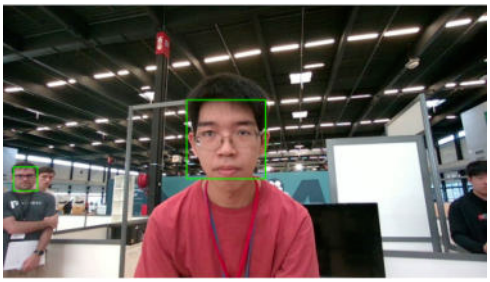


Figura 94: Imagem com as faces detetadas

Foram guardadas duas etnias correspondentes a cada uma das pessoas detetadas e ao invés de ter sido dita a etnia correspondente ao convidado, foi dita a etnia que estava na primeira posição do *array* onde esta informação foi guardada. Na figura 95 é possível ver os resultados obtidos relativamente às duas regiões onde foram detetadas faces. A região 2 seria a correta quando nos referimos ao convidado da prova.

```
Região 1:
  Raça Dominante: white
  Coordenadas da Região: x=30, y=427, w=63, h=63

Região 2:
  Raça Dominante: asian
  Coordenadas da Região: x=487, y=250, w=208, h=208
```

Figura 95: Resultados relativos à etnia

Tendo em consideração este detalhe foi preciso voltar atrás e rever os resultados obtidos. Na figura 96 está representado o resultado obtido relativamente ao género identificado. É possível verificar que ambos os elementos presentes na imagem são do género masculino e foram corretamente identificados.

```
Região 1:
  Gender: {'Woman': 0.0019797425920842215, 'Man': 99.99802112579346}
  Coordenadas da Região: x=30, y=427, w=63, h=63

Região 2:
  Gender: {'Woman': 0.002670889807632193, 'Man': 99.9973971191406}
  Coordenadas da Região: x=487, y=250, w=208, h=208
```

Figura 96: Resultados relativos ao género

Relativamente à idade e tal como mencionado em 3.4.2 foi utilizado outro método. Neste caso foi dito pelo robô a idade relativa ao convidado 1, uma vez que não foi possível detetar a idade relativa à face 1 (árbitro), tal como é possível analisar na figura 97.

```
Estimated Age for Face 1: None
Estimated Age for Face 2: (23 and 32)
(23 and 32)
```

Figura 97: Resultados relativos à idade

Uma forma de resolver esta situação seria ter apenas em consideração a região mais centrada da imagem algo que já foi desenvolvido após a participação no *RoboCup@Home*.

Capítulo 5

Conclusões

5.1 Conclusões

Nesta dissertação foi apresentado um sistema de deteção e reconhecimento facial em imagem e em *stream* de vídeo utilizando métodos de DL, e também um extrator de caraterísticas para ajudar no reconhecimento de utilizadores em tarefas de longa duração. Foram também desenvolvidas soluções de *hardware* e *software* para participar no *RoboCup@Home* em Julho de 2023.

Foi construído um *dataset* personalizado de utilizadores do Laboratório de Automação e Robótica (LAR) da Universidade do Minho e outros alunos da mesma universidade para suportar a execução e teste de todos os algoritmos implementados. O principal objetivo para este projeto foi a implementação do trabalho desenvolvido no robô de serviço e de cuidados de saúde CHARMIE. Para um robô de serviços como é o caso do CHARMIE e sendo tão importante a capacidade de interação do robô com humanos, é fundamental a capacidade de reconhecimento do utilizador, tanto a nível de saber quem é a pessoa, como de saber as características da mesma. Desta forma o CHARMIE pode adaptar a sua abordagem e também a forma de interação dependendo dos utilizadores com quem interage. Estas informações podem ser retiradas a partir do trabalho desenvolvido relativamente à extração de características, como idade, género, etnia e altura. Estas informações ajudam a criar uma experiência mais personalizada para todas as interações do utilizador, uma vez que o CHARMIE pode utilizar esta tecnologia para adaptar diretamente os seus comportamentos de acordo com a pessoa com quem está a interagir.

Inicialmente, foi feita um introdução detalhada de tópicos como Machine Learning, onde foi abordado o classificador SVM e como este funciona. Foram também abordados os temas de Deep Learning e CNN, tanto conceitos básicos como arquiteturas mais avançadas. Foi também abordado a importância das imagens de profundidade em tarefas de reconhecimento facial quando há questões de segurança em causa, e de forma a ser possível integrar e aplicar o trabalho no robô CHARMIE foi também feita uma

breve explicação relativamente a ROS2.

Para a deteção de faces foi utilizada a biblioteca *MediaPipe*, tendo demonstrado a sua eficiência, prevendo as coordenadas da caixa delimitadora. Esta biblioteca foi também utilizada para processar as imagens das faces de forma a ser possível criar o *dataset*. Todo este sistema foi testado utilizando a câmara do computador e também a Intel *RealSense Depth* D455 usada pelo robô, obtendo entre 26 e 30 fps de forma consistente, o que a torna fiável em cenários em tempo real. Detetou facilmente rostos em diferentes poses e para diferentes condições de iluminação exceto quando alguma parte da face não estava visível.

Relativamente à arquitetura *Inception ResNet*, tanto a versão 1 como a 2 foram treinadas com o *dataset* anteriormente criado e foi apresentado o estudo de diferentes hiperparâmetros bem como a aplicação de diferentes valores melhoraram determinados resultados quando associados a outros parâmetros. Devido aos maus resultados obtidos quando foram utilizados os modelos treinados nas tarefas de classificação, foi escolhida outra abordagem nomeadamente, através da utilização de modelos apenas como extrator de características faciais, utilizando depois um SVM para a tarefa de classificação, que provou ser eficiente e bastante prático quando necessário adicionar novas pessoas ao *dataset*. O classificador SVM foi treinado com imagens de alunos da Universidade do Minho e as amostras utilizadas como entrada no classificador foram codificadas com êxito pelo módulo *Inception ResNet* com melhor desempenho. Além disso, foram efetuadas diferentes experiências relativamente ao número de imagens necessárias para obter um classificador com bons resultados.

Foram descritos os resultados obtidos em tempo real relativamente aos diversos módulos e a junção dos mesmos, sendo possível concluir que o sistema conseguiu detetar e classificar corretamente as faces dos alunos com uma elevada *accuracy*.

Relativamente ao trabalho desenvolvido para o *RoboCup@Home* e sendo esta a primeira participação da equipa na prova, o resultado obtido foi considerado extremamente positivo. Este documento, para além de descrever o trabalho do autor, menciona pequenos módulos desenvolvidos por outros elementos da equipa, sendo que a junção de todo este trabalho, representa o resultado final apresentado e avaliado em competição.

Na prova do Rececionista, e tal como é possível observar no vídeo da prova, todos os módulos desenvolvidos e apresentados funcionaram corretamente. Daí ter sido obtida a pontuação máxima na prova o que se traduziu num 1° lugar na prova em questão. Relativamente à parte de deteção facial e extração das características que foi onde o trabalho incidiu principalmente, os resultados obtidos são também positivos. A deteção facial funcionou corretamente, uma vez que quando este foi preciso, nomeadamente para extrair as fotografias, fazer contacto visual com o convidado ou com a pessoa a que o robô se dirigia, este funcionou como esperado.

Relativamente à extração de características de um convidado, bastava indicar quatro características para obter a classificação máxima, contudo foram mencionadas cinco características. Tal como mencionado no capítulo anterior, a etnia não foi dita corretamente. Este erro foi analisado e é possível afirmar que apesar do que aconteceu em prova, foi identificada corretamente esta característica. Este erro motivou

uma análise posterior à competição dos resultados obtidos sendo que com esta análise foi possível perceber que o mesmo aconteceu relativamente ao género. Isto demonstrou uma falha uma vez que deveria ter sido considerada apenas a região mais central da imagem, algo que já foi corrigido após a competição. O resultado final relativo a esta prova foi extremamente positivo o que se retratou na pontuação obtida.

Na prova *Stickler for the Rules*, apesar do reduzido tempo em que a mesma foi desenvolvida, a equipa conseguiu somar 100 pontos, aos 800 anteriormente feitos. Em testes anteriores à prova foi verificado o correto funcionamento tanto da parte de deteção das pessoas dentro e fora do espaço proibido mas também a parte de voz onde o robô tem de ser capaz de se dirigir ao humano com diversas indicações. Esta parte não foi conseguida em prova, contudo o resultado foi positivo apesar das melhorias que são necessárias. A acrescentar a esta pontuação a equipa obteve ainda 32 pontos relativos à apresentação de um poster sobre o robô CHARMIE, fazendo um total de 932 pontos obtidos na competição.

Desta forma, é possível assumir que os objetivos estabelecidos inicialmente para esta dissertação tanto a nível competitivo como social foram cumpridos.

5.2 Trabalho Futuro

No âmbito de melhorar ou ajustar o trabalho até agora desenvolvido para uma solução mais completa e robusta, é necessário aumentar a complexidade e desempenho dos modelos. Um dos tópicos em consideração é a criação de um *dataset* maior, com mais classes e mais diversificado para melhorar a capacidade dos modelos de extraírem características faciais. Isto envolve um tempo superior para treino bem como maior capacidade computacional, contudo é um ponto fulcral para obter melhores resultados. Sobre o módulo da deteção facial, o método escolhido foi o *MediaPipe* pelos resultados analisados, contudo e tendo em conta o trabalho que tem sido desenvolvido com o robô CHARMIE relativamente à deteção de pessoas com recurso a *Yolo v8 Pose*, seria interessante analisar de que forma o sistema se comportava relativamente a deteção de faces com esta solução. Com o objetivo de continuar a investigar relativamente ao tema, seria relevante analisar outras arquiteturas e também diferentes combinações de hiperparâmetros de forma a encontrar junções que resultem em melhores resultados tanto ao nível de tempo de treino como *accuracy* obtida. De forma a validar o correto funcionamento do trabalho desenvolvido, seria ainda importante executar alguns testes com situações mais próximas da realidade nomeadamente testar o robô num ambiente hospitalar ou lar de idosos. Como última sugestão deixa-se aqui o desafio de continuar a desenvolver e a aprimorar as tarefas da competição *RoboCup@Home*.

Bibliografia

[1] D. Kragic, M. Vincze et al. “Vision for robotics”. Em: *Foundations and Trends® in Robotics* 1.1 (2009), pp. 1–78 (ver p. 2).

[2] T. Ribeiro et al. “CHARMIE: A Collaborative Healthcare and Home Service and Assistant Robot for Elderly Care”. Em: (2021). https://www.mdpi.com/2076-3417/11/16/7248/htm (ver p. 3).

[3] *Robocup@Home*. url: https://athome.robocup.org/ (ver p. 3).

[4] T. Ribeiro et al. “LAR@Home 2023 Team Description Paper”. Em: (2022). url: https://lar.dei.uminho.pt/images/downloads%5CTDP%5C%20Lar@Home%5C%202023.pdf (ver pp. 4, 39).

[5] M. van der Burgh et al. “Tech United Eindhoven @Home 2019 Team Description Paper”. Em: (2019) (ver p. 5).

[6] Y. Tanaka et al. “Hibikino-Musashi@Home 2019 Team Description Paper”. Em: (2019) (ver p. 5).

[7] J. Savage et al. “Pumas@Home 2019 Team Description Paper”. Em: (2019) (ver p. 5).

[8] R. Memmesheimer et al. “RoboCup 2019 - homer@UniKoblenz (Germany)”. Em: (2019) (ver p. 5).

[9] A. L. Samuel. “Some studies in machine learning using the game of checkers”. Em: *IBM Journal of Research and Development* 44.1.2 (2000), pp. 206–226. doi: 10.1147/rd.441.0206 (ver p. 9).

[10] B. Mahesh. “Machine Learning Algorithms - A Review”. Em: (2020). url: https://www.researchgate.net/publication/344717762%5C_Machine%5C_Learning%5C_Algorithms%5C_-A%5C_Review (ver p. 9).

[11] A. Bisht. *ML | Classification vs Regression*. 2022. url: https://www.geeksforgeeks.org/ml-classification-vs-regression/ (ver p. 9).

[12] L. Breiman. “Random forests”. Em: *Machine learning* 45 (2001), pp. 5–32 (ver p. 10).

[13] M. Goadrich, L. Oliphant e J. Shavlik. “Gleaner: Creating ensembles of first-order clauses to improve recall-precision curves”. Em: *Machine Learning* 64 (2006), pp. 231–261 (ver p. 12).

[14] J. Cervantes et al. “A comprehensive survey on support vector machine classification: Applications, challenges and trends”. Em: *Neurocomputing* 408 (2020), pp. 189–215. issn: 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.10.118. url: https://www.sciencedirect.com/science/article/pii/S0925231220307153 (ver p. 13).

[15] M. Matsugu, K. Mori e T. Suzuki. “Face Recognition Using SVM Combined with CNN for Face Detection”. Em: *Neural Information Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 356–361. isbn: 978-3-540-30499-9 (ver p. 14).

[16] N. Hayes. *Neuron*. url: https://pixabay.com/vectors/motor-neurone-neuron-neurone-nerve-2040692/ (ver p. 14).

[17] N. Hayes. *Artificial Neural Network - Basic Concepts*. url: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm (ver p. 14).

[18] S. Srimamilla. “Image Classification Using Convolutional Neural Networks”. Em: (2022). doi: https://doi.org/10.22214/ijraset.2022.47085 (ver p. 16).

[19] K. O’Shea e R. Nash. “An Introduction to Convolutional Neural Networks”. Em: (2015). url: http://arxiv.org/abs/1511.08458 (ver pp. 16, 17).

[20] M. Gurucharan. *Basic CNN Architecture*. 2022. url: https://www.upgrad.com/blog/basic-cnn-architecture/ (ver p. 17).

[21] J. Briggs e L. Carnevali. *Embedding Methods for Image Search*. Cap. 4. url: https://www.pinecone.io/learn/image-search/ (ver p. 17).

[22] M. Asthana. *Layers of a Convolutional Neural Network*. 2020. url: https://medium.com/analytics-vidhya/layers-of-a-convolutional-neural-network-168daddd2dd1 (ver p. 18).

[23] Y. LeCun et al. “LeNet-5, Convolutional Neural Networks”. Em: *URL: http://yann. lecun. com/exdb/lenet* 20.5 (2015) (ver p. 20).

[24] A. Krizhevsky, I. Sutskever e G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. Em: *Communications of the ACM* 60.6 (2017) (ver p. 20).

[25] K. Simonyan e A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. Em: *arXiv preprint arXiv:1409.1556* (2014) (ver p. 21).

[26] C. Szegedy et al. “Going deeper with convolutions”. Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015 (ver p. 22).

[27] K. He et al. “Deep residual learning for image recognition”. Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016 (ver p. 24).

[28] C. Szegedy et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. arXiv: 1602.07261 [cs.CV] (ver p. 25).

[29] R. Tellez. *A History of ROS (Robot Operating System)*. 2019. url: https://www.theconstructsim.com/history-ros/ (ver p. 26).

[30] S. Macenski et al. "Robot Operating System 2: Design, architecture, and uses in the wild". Em: *Science Robotics* 7 (mai. de 2022). doi: 10.1126/scirobotics.abm6074 (ver p. 26).

[31] C. Lugaresi et al. "Mediapipe: A framework for building perception pipelines". Em: *arXiv preprint arXiv:1906.08172* (2019) (ver p. 28).

[32] K. Zhang et al. "Joint face detection and alignment using multitask cascaded convolutional networks". Em: *IEEE signal processing letters* 23.10 (2016), pp. 1499–1503 (ver p. 29).

[33] D. Wen, H. Han e A. Jain. "Face Spoof Detection With Image Distortion Analysis". Em: *IEEE Transactions on Information Forensics and Security* 10 (abr. de 2015). doi: 10.1109/TIFS.2015.2400395 (ver p. 33).

[34] M. Dindorkar et al. "Improved Face Spoofing Detection Using Random Forest Classifier with Fusion of Luminance Chroma Features". Em: *International Journal of Computer Information Systems and Industrial Management Applications* 12 (fev. de 2021), pp. 374–386 (ver p. 33).

[35] C.-S. Fahn, C.-P. Lee e M.-L. Wu. "A Cross-Dataset Evaluation of Anti-Face-Spoofing Methods Using Random Forests and Convolutional Neural Networks". Em: *Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference*. AICCC '19. Kobe, Japan: Association for Computing Machinery, 2020, pp. 89–96. isbn: 9781450372633. doi: 10.1145/3375959.3375985. url: https://doi.org/10.1145/3375959.3375985 (ver p. 33).

[36] W. Zhao et al. "Face recognition: A literature survey". Em: *ACM computing surveys (CSUR)* 35.4 (2003), pp. 399–458 (ver p. 34).

[37] K. Hao. "This is how we lost control of our faces". Em: (2021). url: https://www.technologyreview.com/2021/02/05/1017388/ai-deep-learning-facial-recognition-data-history/ (ver p. 34).

[38] M. Wang e W. Deng. "Deep face recognition: A survey". Em: *Neurocomputing* 429 (2021), pp. 215–244 (ver pp. 34, 35).

[39] F. Schroff, D. Kalenichenko e J. Philbin. "FaceNet: A unified embedding for face recognition and clustering". Em: (2015), pp. 815–823. doi: 10.1109/CVPR.2015.7298682 (ver p. 36).

[40] Q. Cao et al. "VGGFace2: A Dataset for Recognising Faces across Pose and Age". Em: (2018), pp. 67–74. doi: 10.1109/FG.2018.00020 (ver p. 36).

[41] D. Yi et al. "Learning Face Representation from Scratch". Em: *ArXiv* abs/1411.7923 (2014) (ver p. 36).

[42] *Intel RealSense Depth Camera D455: Twice the Range, Better Performance*. url: https://www.intelrealsense.com/depth-camera-d455/ (ver p. 40).

[43] J. Terra. *Keras vs Tensorflow vs Pytorch: Key Differences Among Deep Learning*. url: https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article?tag=TensorFlow%5C%20and%5C%20Keras (ver p. 46).

[44] *GitHub- facenet*. url: https://github.com/davidsandberg/facenet (ver p. 54).

[45] *Pre-trained Keras FaceNet model provided by Hiroki Taniai*. url: https://drive.google.com/file/d/1e6PHRlIeayAsvRGpYUwvstklvJy-3H5B/view?usp=drive_link (ver p. 55).

[46] *A Brief History of RoboCup*. url: https://www.robocup.org/a_brief_history_of_robocup (ver p. 58).

[47] *GitHub- RoboCup@Home 2023 Bordeuax*. url: https://github.com/RoboCupAtHome/Bordeaux2023#stickler-for-the-rules (ver pp. 59, 60).

[48] J. Hart et al. *RoboCup@Home 2024: Rules and Regulations*. https://athome.robocup.org/rules/2024_rulebook.pdf. 2024 (ver pp. 60, 61).

[49] *Imagem de um grupo de pessoas*. url: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.eusemfronteiras.com.br%2Fcomo-atrair-pessoas-incriveis-em-nossa-vida%2F&psig=AOvVaw2b2Lwo7swXM05eYYozqabG&ust=1692885910759000&source=images&cd=vfe&opi=89978449&ved=0CBAQjRxqFwoTCIjy5cz58oADFQAAAAAdAAAAABAE (ver p. 72).

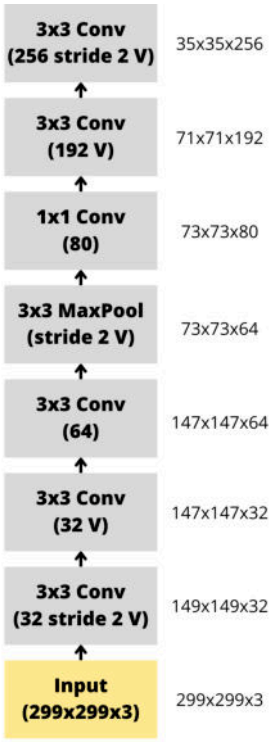
[50] *Imagem de um grupo de pessoas*. url: https://www.google.com/url?sa=i&url=https%3A%2F%2Fdescobre-a-biblia.webnode.pt%2Fencontra-uma-igreja%2F&psig=AOvVaw2qgUhl-tSkAqd4VSv5_ypn&ust=1692885835714000&source=images&cd=vfe&opi=89978449&ved=0CBAQjRxqFwoTCJja9qj58oADFQAAAAAdAAAAABAE (ver pp. 72, 73).

[51] *Receptionist Task CHARMIE LAR@Home RoboCup @Home 2023*. url: https://www.youtube.com/watch?v=rYzFhlLRpxc (ver p. 98).

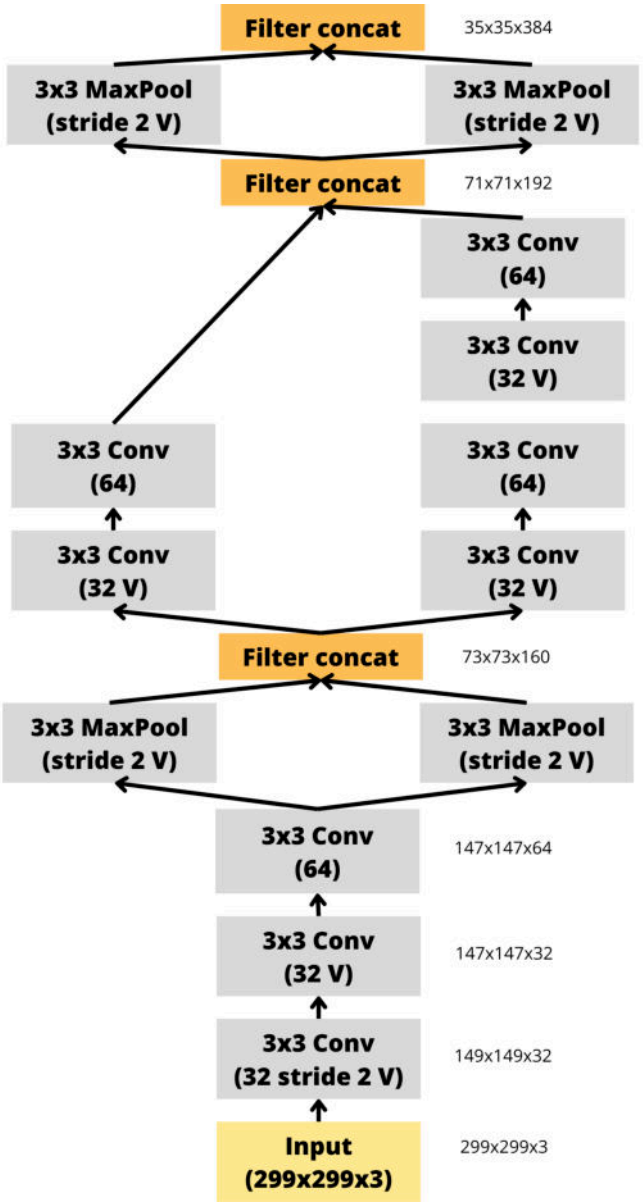
Anexo I

Anexos

I.1 Arquitetura Inception ResNet V1



I.2 Arquitetura Inception ResNet V2

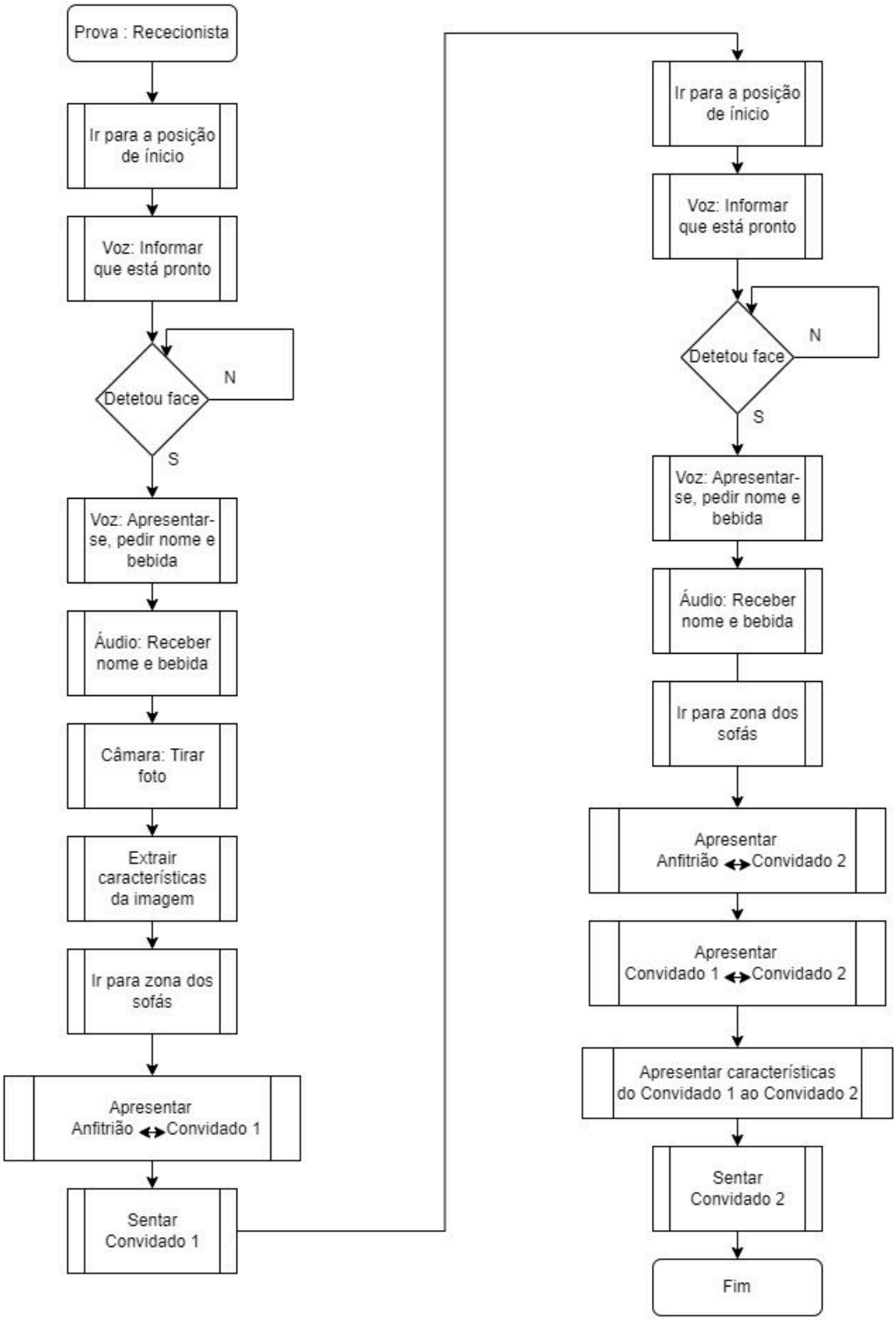


Anexo II

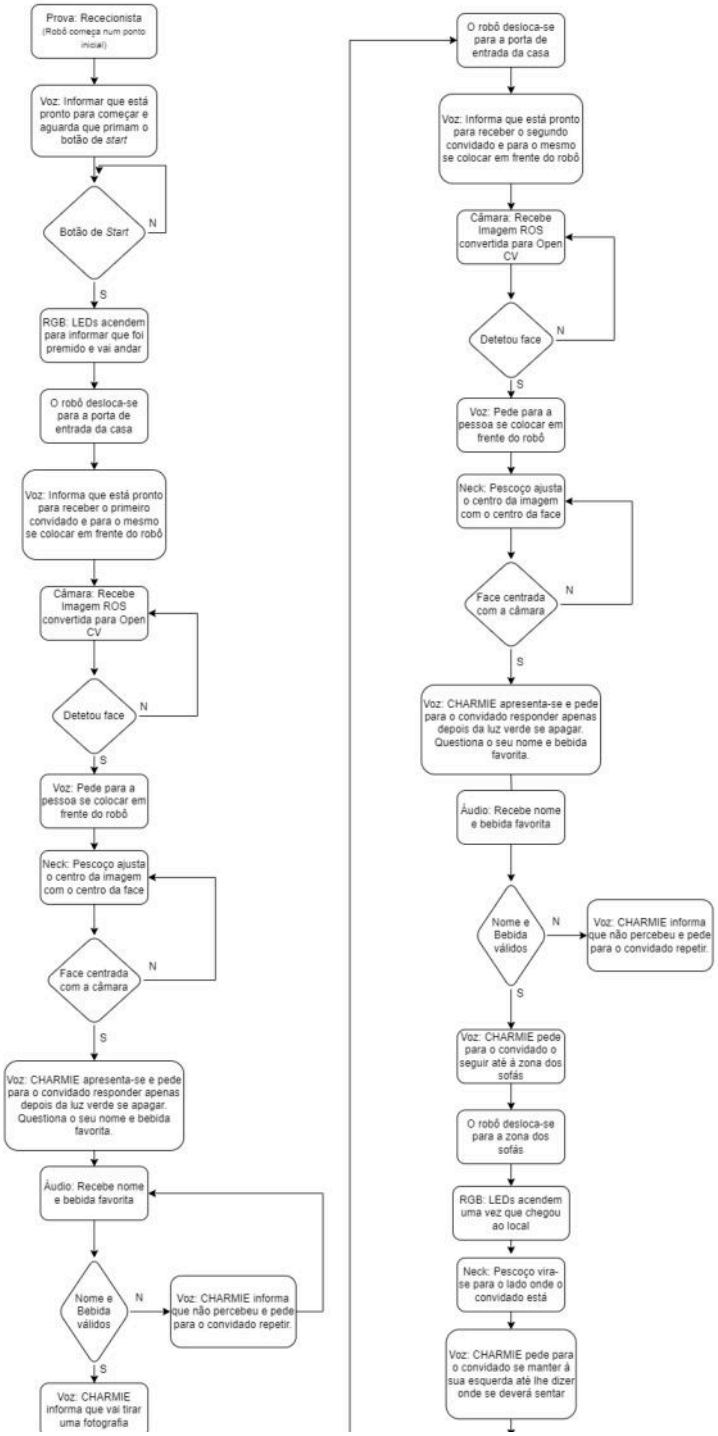
Anexos

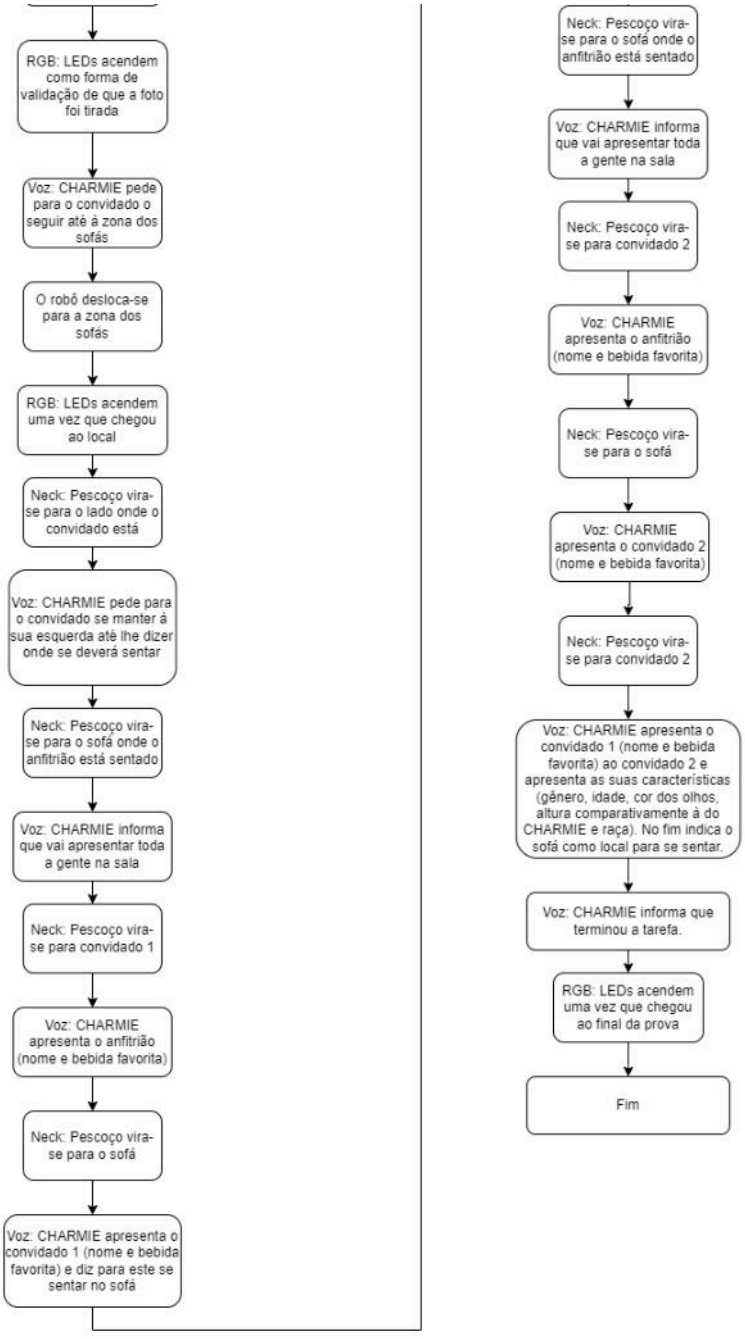
II.1 Fluxogramas da Abordagem Competitiva

II.1.1 Fluxograma da prova Rececionista

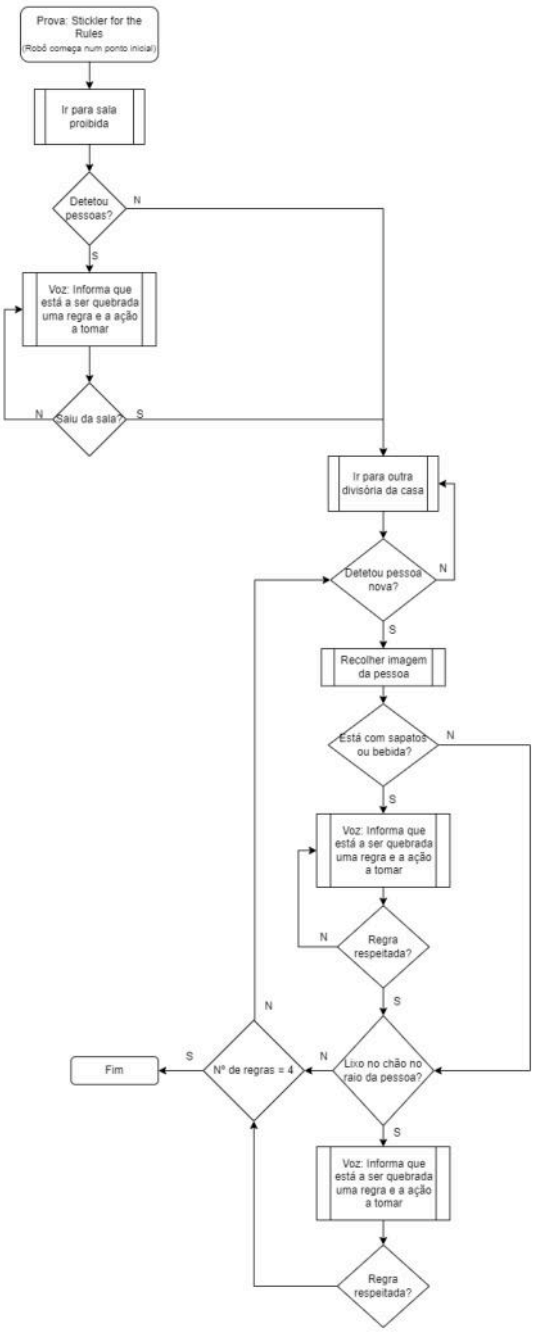


II.1.2 Fluxograma do trabalho desenvolvido na prova Rececionista (parte 1 e 2

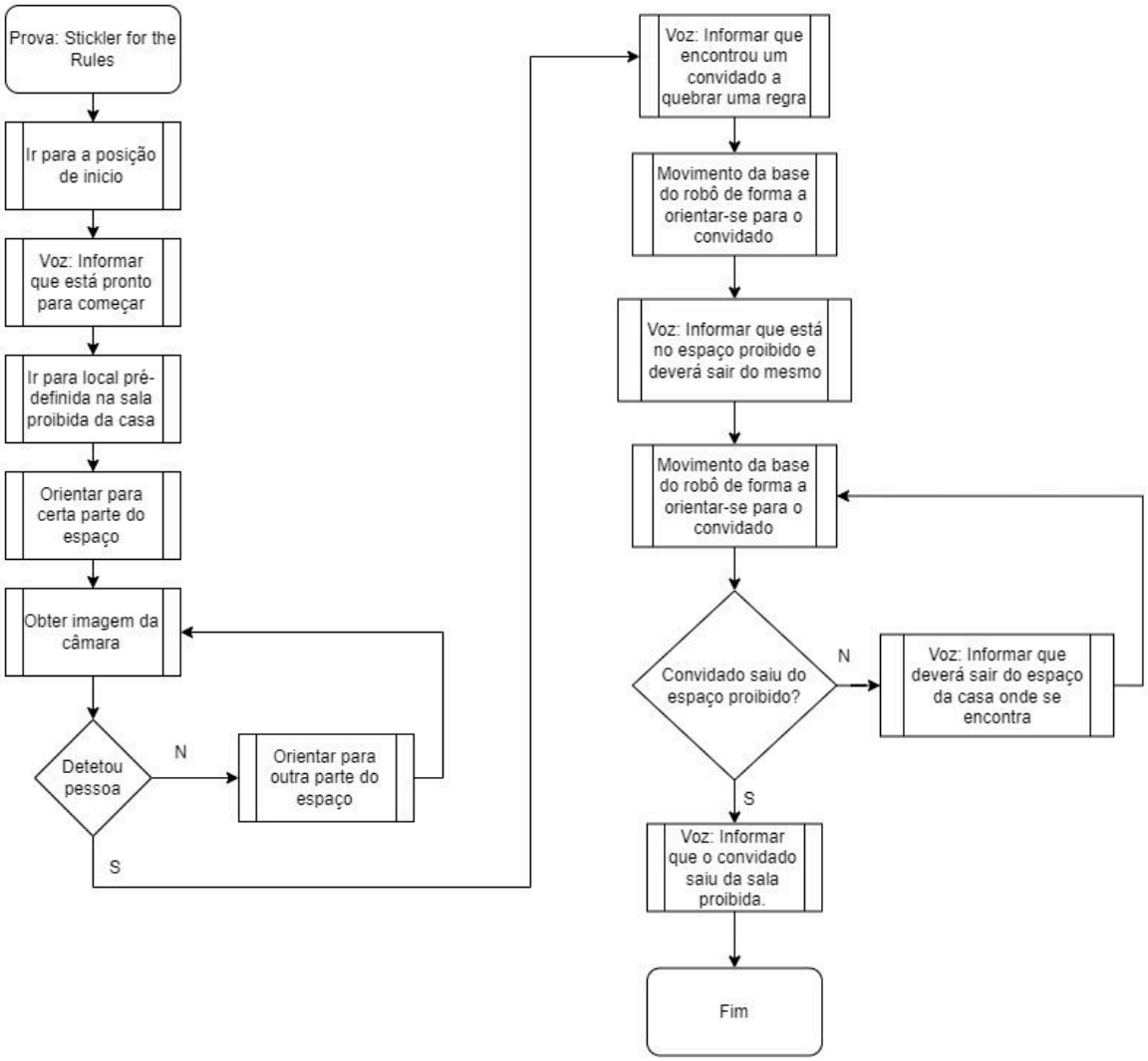



RGB: LEDs acendem como forma de validação de que a foto foi tirada
Voz: CHARMIE pede para o convidado o seguir até à zona dos sofás
O robô desloca-se para a zona dos sofás
RGB: LEDs acendem uma vez que chegou ao local
Neck: Pescoço vira-se para o lado onde o convidado está
Voz: CHARMIE pede para o convidado se manter à sua esquerda até lhe dizer onde se deverá sentar
Neck: Pescoço vira-se para o sofá onde o anfitrião está sentado
Voz: CHARMIE informa que vai apresentar toda a gente na sala
Neck: Pescoço vira-se para convidado 1
Voz: CHARMIE apresenta o anfitrião (nome e bebida favorita)
Neck: Pescoço vira-se para o sofá
Voz: CHARMIE apresenta o convidado 1 (nome e bebida favorita) e diz para este se sentar no sofá
Neck: Pescoço vira-se para o sofá onde o anfitrião está sentado
Voz: CHARMIE informa que vai apresentar toda a gente na sala
Neck: Pescoço vira-se para convidado 2
Voz: CHARMIE apresenta o anfitrião (nome e bebida favorita)
Neck: Pescoço vira-se para o sofá
Voz: CHARMIE apresenta o convidado 2 (nome e bebida favorita)
Neck: Pescoço vira-se para convidado 2
Voz: CHARMIE apresenta o convidado 1 (nome e bebida favorita) ao convidado 2 e apresenta as suas características (género, idade, cor dos olhos, altura comparativamente à do CHARMIE e raça). No fim indica o sofá como local para se sentar.
Voz: CHARMIE informa que terminou a tarefa.
RGB: LEDs acendem uma vez que chegou ao final da prova
Fim

II.1.3 Fluxograma da prova *Stickler for the Rules*



II.1.4 Fluxograma da trabalho desenvolvido na prova *Stickler for the Rules*

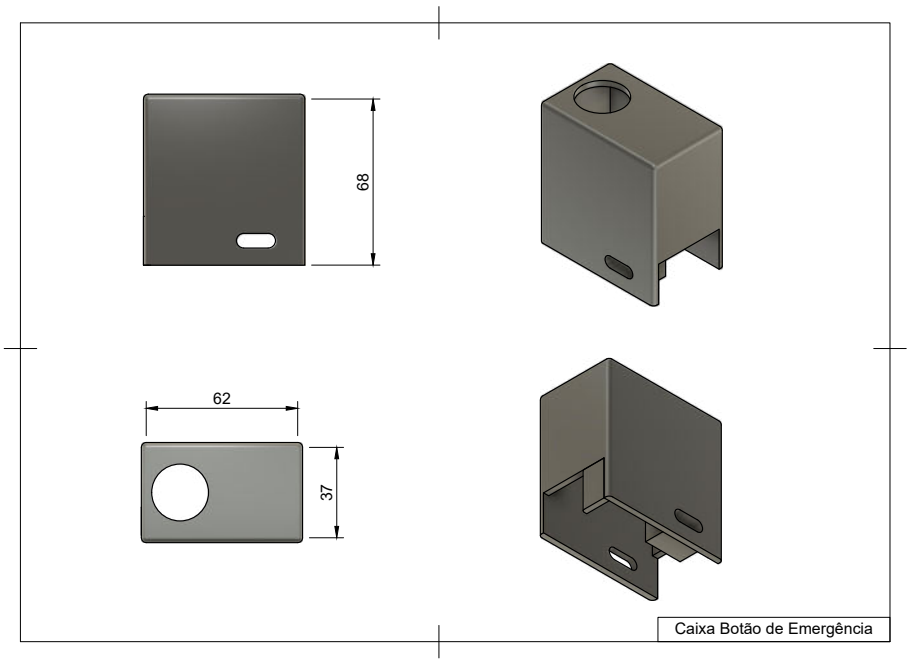


Anexo III

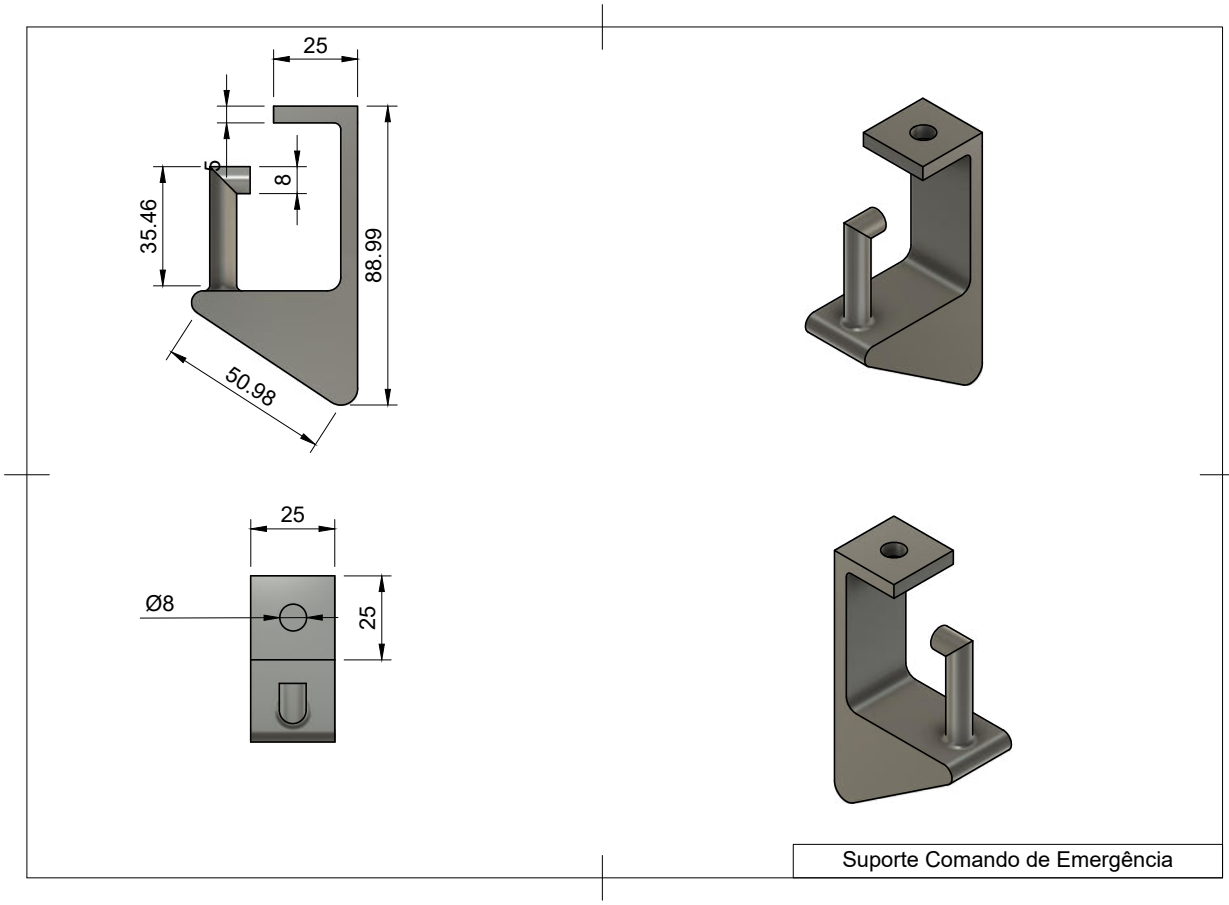
Anexos

III.1 Desenhos peças 3D

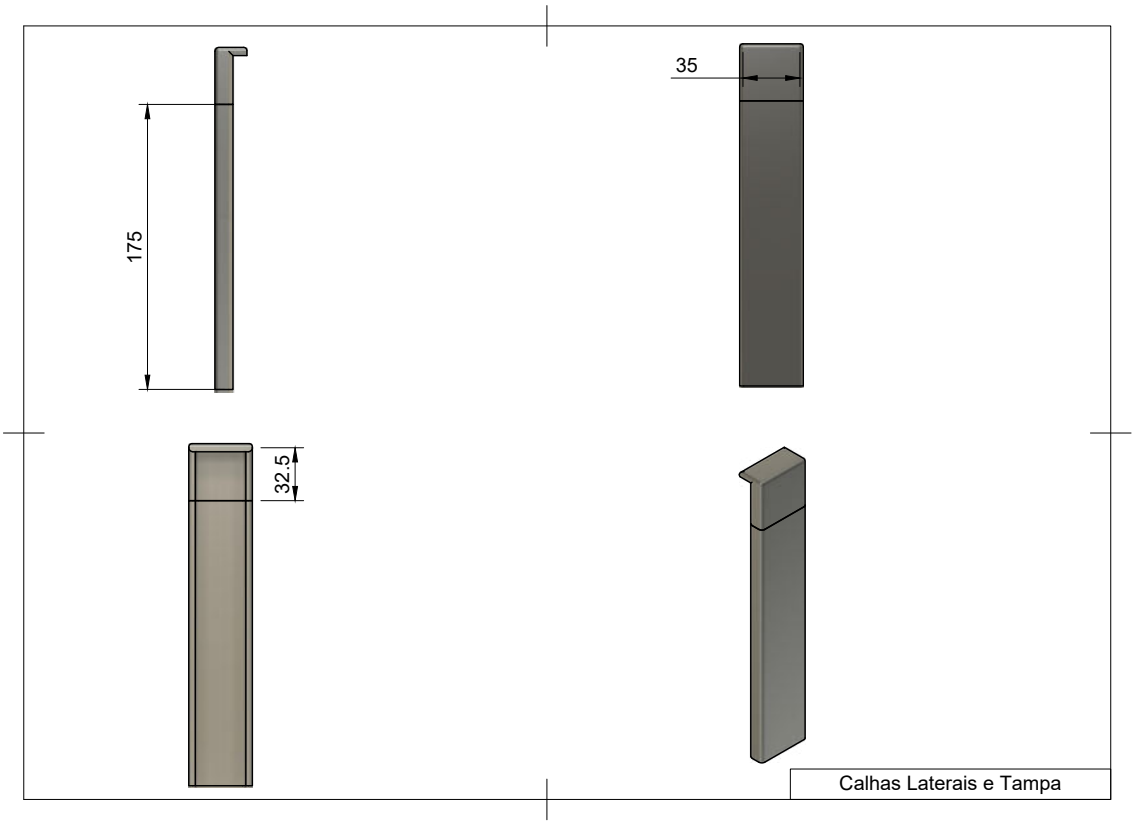
III.1.1 Caixa para Botão de Emergência



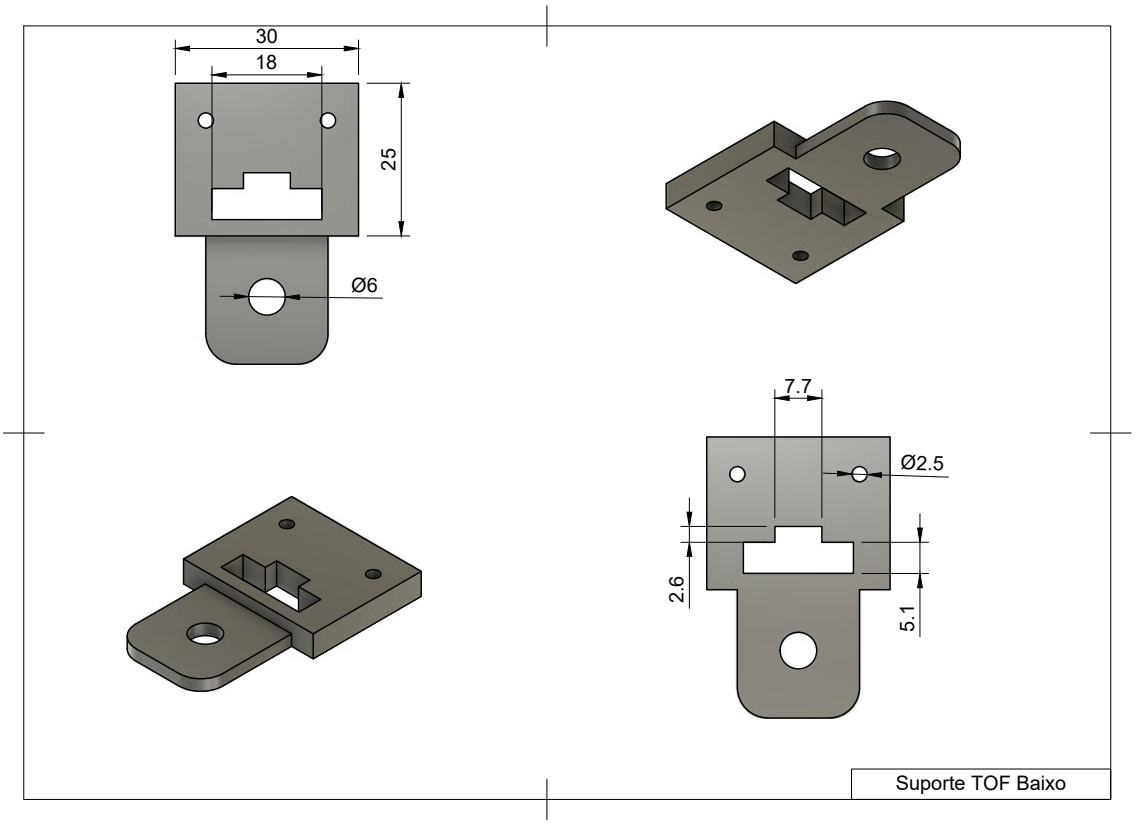
III.1.2 Suporte para o Comando de Emergência



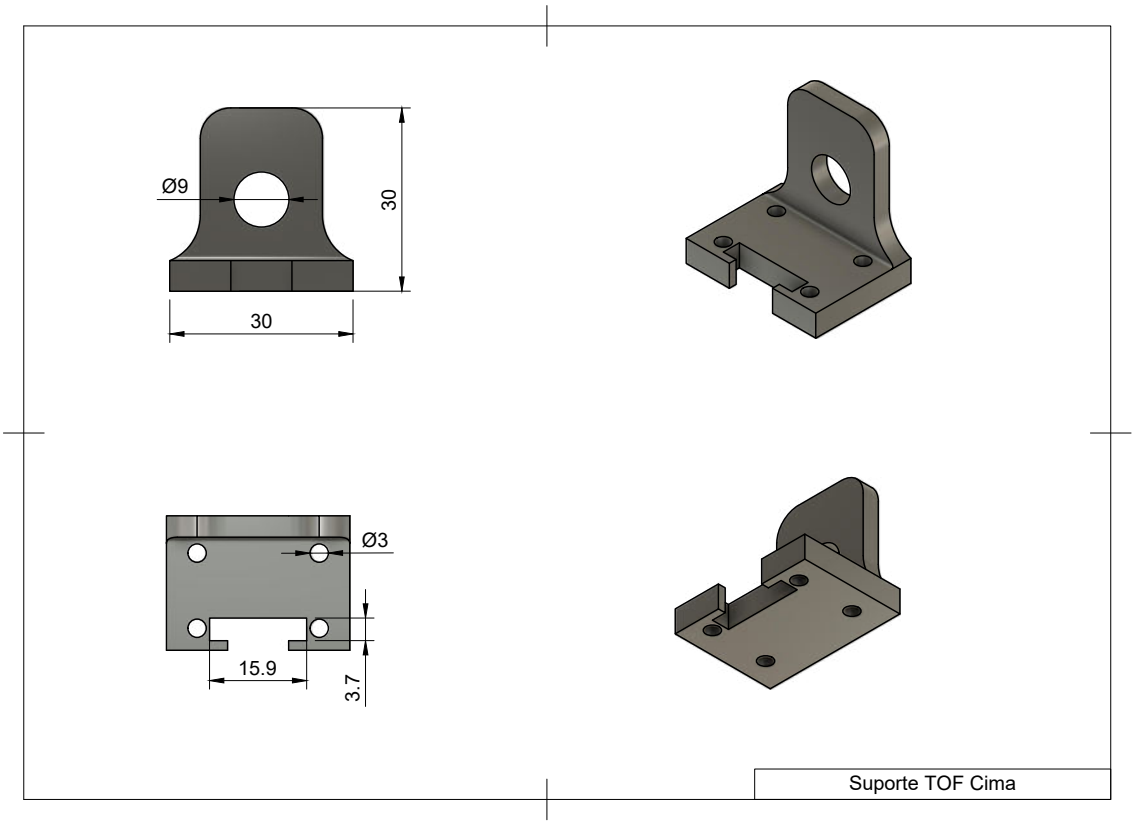
III.1.3 Calhas Laterais e Tampa



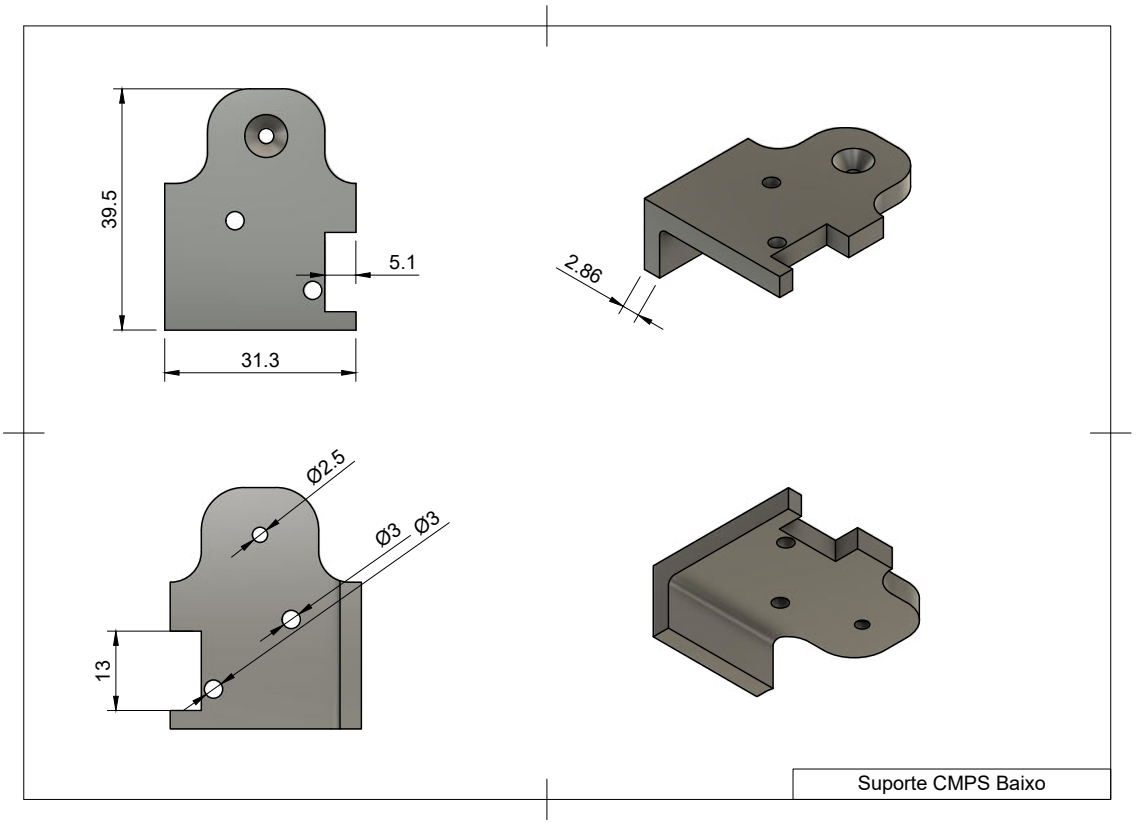
III.1.4 Suporte para o sensor TOF inferior



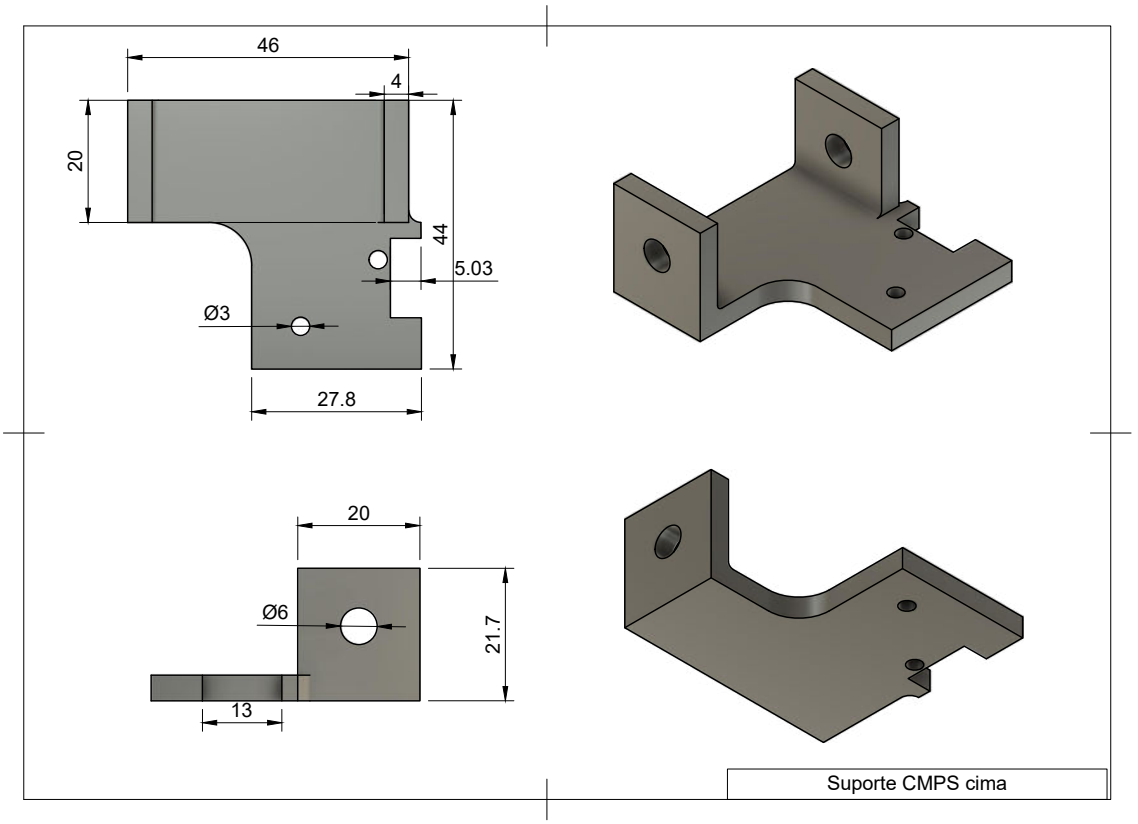
III.1.5 Suporte para o sensor TOF superior



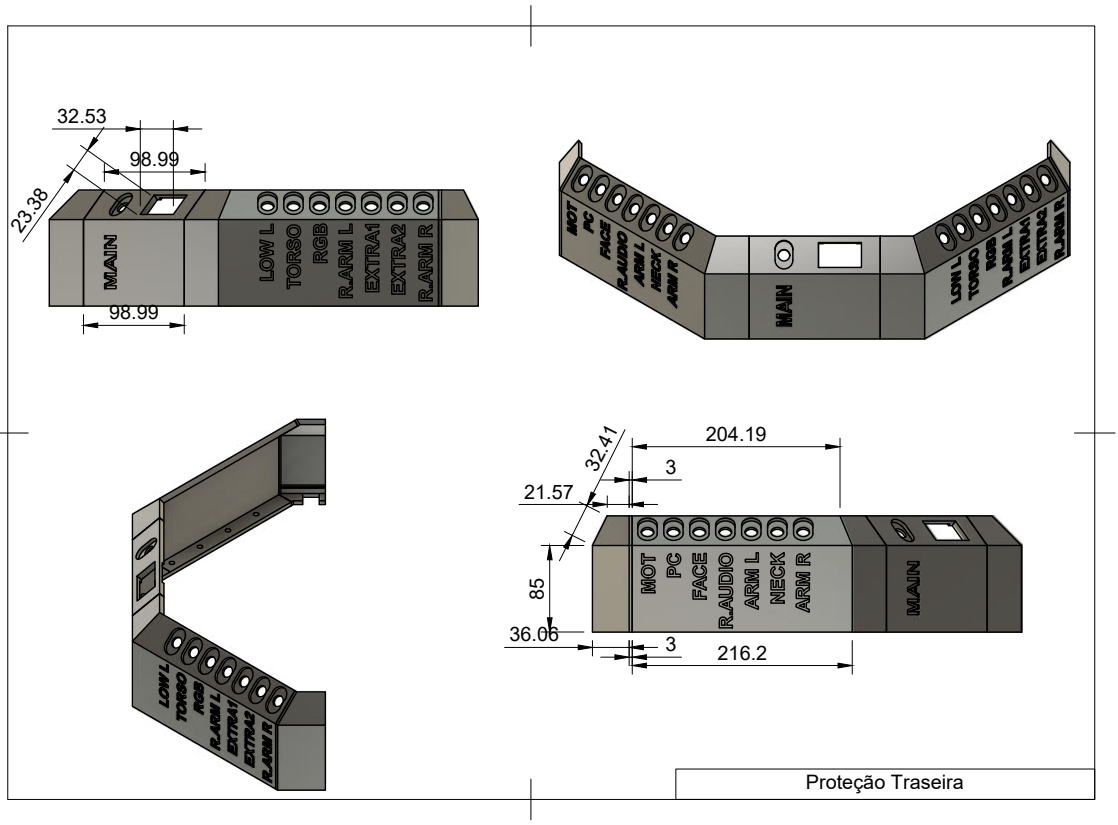
III.1.6 Suporte para a bússola CMPS inferior



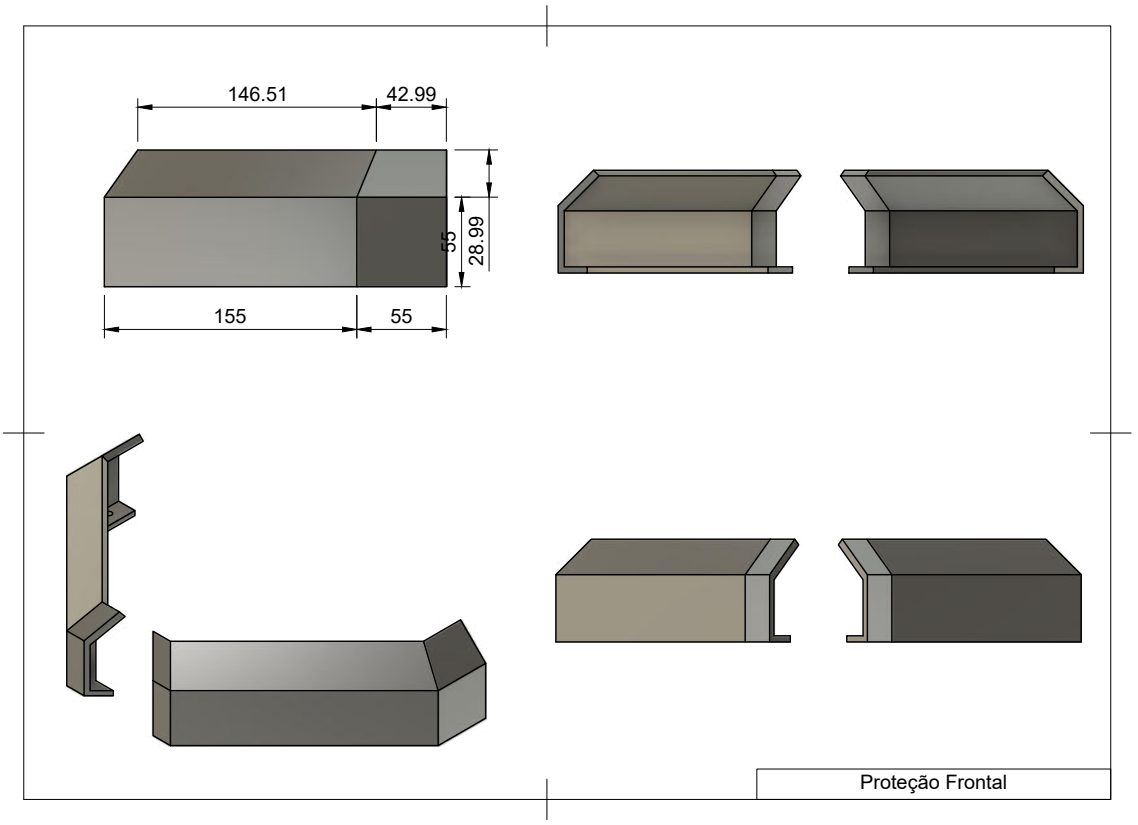
III.1.7 Suporte para a bússola CMPS superior



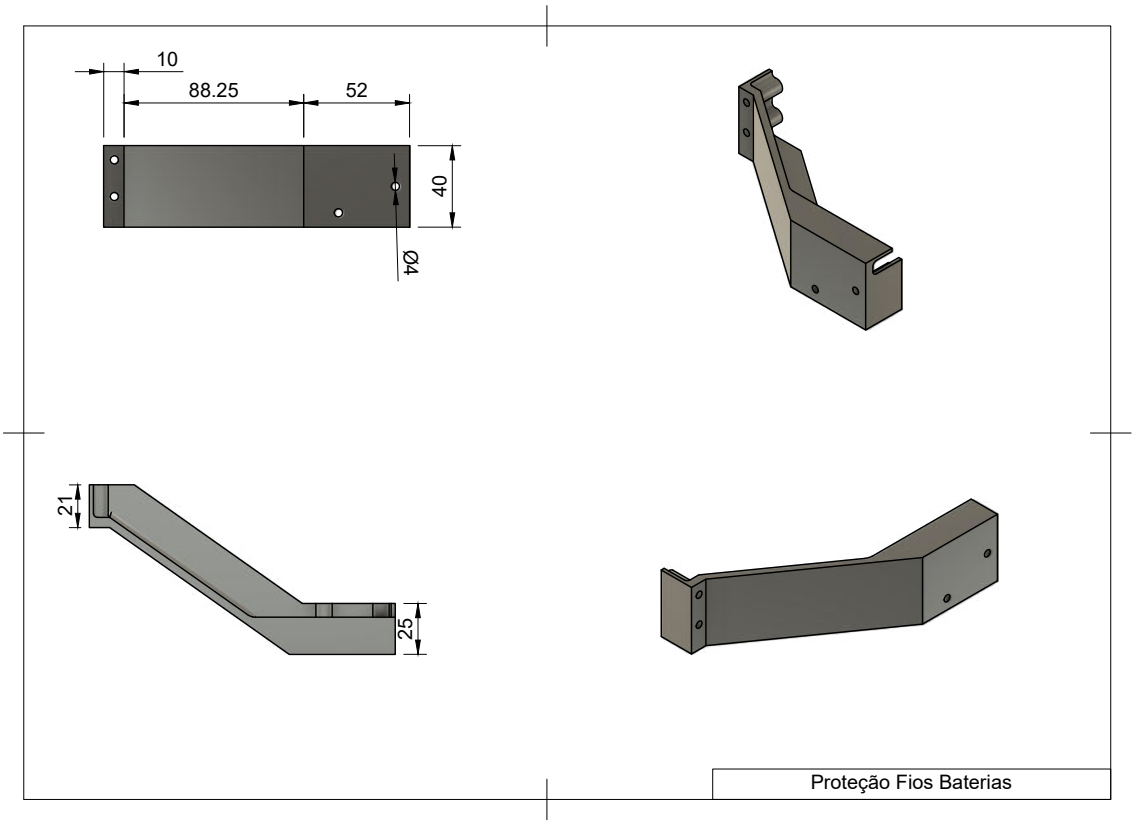
III.1.8 Proteção Traseira



III.1.9 Proteção Frontal



III.1.10 Proteção dos Fios das Baterias



III.1.11 Ombros e Suporte do Microfone

