INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Desenvolvimento de um Sistema de Reconhecimento Facial

SARA RAQUEL DOS SANTOS RODRIGUES
Outubro de 2020

POLITÉCNICO
DO PORTO

DESENVOLVIMENTO DE UM SISTEMA DE RECONHECIMENTO FACIAL

Sara Raquel dos Santos Rodrigues

Departamento de Engenharia Eletrotécnica

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Sistemas e Planeamento Industrial

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de Tese/Dissertação do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Sara Raquel dos Santos Rodrigues, Nº 1161646, 1161646@isep.ipp.pt
Orientação científica: Professor Filipe Azevedo, fta@isep.ipp.pt

Departamento de Engenharia Eletrotécnica

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Sistemas e Planeamento Industrial

2020

Dedico este trabalho aos meus pais, Eduarda e Nelson

Agradecimentos

Os meus agradecimentos começam por se direcionarem aos meus pais, Eduarda e Nelson, que proporcionaram, ao longo destes anos, a minha frequência no ensino superior, apoiando-me tanto financeiramente como emocionalmente, com especial direção para esta fase da minha vida; ao meu namorado Marcos, que passou estes anos do meu lado, abdicando de várias coisas para que eu pudesse dedicar-me exclusivamente ao término deste capítulo da minha vida, dando-me suporte em todos os momentos. Agradeço, do fundo do coração, aos três, que juntamente com a minha família e amigos, sempre me impulsionaram a finalizar esta epata, apesar de toda a dificuldade por que passei em conciliar a vida académica, com a profissional e pessoal.

Não poderia deixar de agradecer também ao meu orientador Professor Filipe Azevedo que, apesar da minha desistência inicial, decidiu continuar a ajudar-me e a apoiar-me no desenvolvimento deste projeto.

Agradeço também ao ISEP, Instituto Superior de Engenharia do Porto, assim como a todos os professores com os quais tive a oportunidade de me cruzar ao longo deste percurso, pelo acompanhamento e possibilidade de me tornar um pouco do que sou hoje.

Resumo

O Reconhecimento Facial tornou-se um dos sistemas mais importantes e utilizados de reconhecimento de indivíduos já que reproduz uma capacidade natural importante do Ser Humano. Isto porque, a primeira coisa que identificamos quando nos deparamos com uma pessoa é o seu rosto, e não as suas impressões digitais ou geometria das mãos. Além disso, é um sistema de fácil implementação e não necessita de interação por parte dos usuários. Este tipo de reconhecimentos é aplicado nas mais diversas áreas, podendo ir desde a utilização mais básica, como por exemplo, no login em dispositivos de uso pessoal, a uma utilização mais abrangente, na segurança em aeroportos e análise de vídeos de vigilância. No entanto, apresenta algumas desvantagens quando transportado para sistemas computacionais, sobretudo ao nível das variações ambientais, como a posição da face, a iluminação do ambiente, o uso de maquiagem e acessórios. Ainda assim, é possível reconhecer um indivíduo, recorrendo a métodos e algoritmos, para análise de características e padrões.

Posto isto, este trabalho visa abranger um pouco mais o conhecimento nesta área de interesse e procura desenvolver um sistema de reconhecimento facial com o auxílio do MATLAB. Para tal, foi criada uma base de dados com imagens de indivíduos que servirão de ponto de conexão para realizar o reconhecimento. O sistema vai recorrer a uma câmara conectada ao computador, para recolha de informação em tempo real. O mesmo será composto por duas vertentes, uma responsável pela deteção de faces e outra responsável pelo reconhecimento das mesmas. No entanto, a vertente de reconhecimento pode realizar o mesmo recorrendo tanto em imagens capturadas e armazenadas no momento em se trabalha com o sistema, como em imagens armazenadas anteriormente, provenientes de qualquer outro dispositivo.

Palavras-Chave

Reconhecimento Facial, Deteção Facial, Biometria, MATLAB, Algoritmos, *Viola-Jones*, Análise de Componentes Principais, *Eigenfaces*.

Abstract

Facial Recognition has become one of the most important and used systems of individual recognition since it reproduces an important natural capacity of the Human Being. This is because, the first thing we identify when we come across a person is his face, not his fingerprints or hand geometry. In addition, it is an easy-to-implement system and does not require user interaction. This type of recognition is applied in the most diverse areas, ranging from the most basic use, for example, when logging in to personal devices, to a more comprehensive use, in airport security and analysis of surveillance videos. However, it has some disadvantages when transported to computer systems, especially in terms of environmental variations, such as the face position, the ambient lighting, the use of makeup and accessories. Even so, it is possible to recognize an individual, using methods and algorithms, to analyze characteristics and patterns.

That said, this work aims to cover a little more knowledge in this area of interest and seeks to develop a facial recognition system with the help of MATLAB. To this end, a database was created with images of individuals that will serve as a connection point to perform the recognition. The system will use a camera connected to the computer to collect information in real time. It will consist of two strands, one responsible for detecting faces and the other responsible for recognizing them. However, the recognition aspect can do the same using images captured and stored at the time when working with the system, as well as images previously stored, coming from any other device.

Keywords

Facial Recognition, Facial Detection, Biometrics, MATLAB, Algorithms, Principal Component Analysis, Eigenfaces.

Résumé

La reconnaissance faciale est devenue l'un des systèmes de reconnaissance individuelle les plus importants et les plus utilisés car elle reproduit une capacité naturelle importante de l'être humain. En effet, la première chose que nous identifions lorsque nous rencontrons une personne est son visage, pas ses empreintes digitales ou la géométrie de sa main. De plus, c'est un système facile à mettre en œuvre et ne nécessite pas d'interaction de l'utilisateur. Ce type de reconnaissance est appliqué dans les domaines les plus divers, allant de l'utilisation la plus basique, comme par exemple la connexion aux appareils à usage personnel, à une utilisation plus complète, dans la sécurité aéroportuaire et l'analyse des vidéos de surveillance. Cependant, il présente certains inconvénients lorsqu'il est transporté vers des systèmes informatiques, notamment en termes de variations environnementales, telles que la position du visage, l'éclairage de l'environnement, l'utilisation de maquillage et d'accessoires. Même ainsi, il est possible de reconnaître un individu, à l'aide de méthodes et d'algorithmes, pour analyser les caractéristiques et les modèles.

Cela dit, ce travail vise à couvrir un peu plus de connaissances dans ce domaine d'intérêt et cherche à développer un système de reconnaissance faciale avec l'aide de MATLAB. À cette fin, une base de données a été créée avec des images d'individus qui serviront de point de connexion pour effectuer la reconnaissance. Le système utilisera une caméra connectée à l'ordinateur pour collecter des informations en temps réel. Il sera composé de deux volets, l'un chargé de détecter les visages et l'autre chargé de les reconnaître. Cependant, le volet de reconnaissance peut faire de même en utilisant des images capturées et stockées au moment où vous travaillez avec le système, ainsi que des images précédemment stockées, provenant de tout autre appareil.

Mots-clés

Reconnaissance Faciale, Détection Faciale, Biométrie, MATLAB, Algorithmes, Viola-Jones, Analyse en Composantes Principales, Eigenfaces.

Índice

Índice de Figuras

Índice de Tabelas

Siglas e Abreviaturas

ISEP – Instituto Superior de Engenharia do Porto

MATLAB – Matriz Laboratory

FFTW – Fastest Fourier Transform in the West

HTML – HyperText Markup Language

PDF – Portable Document Format

LaTeX – Lamport TeX

GPU – Graphics Processing Unit

MEF – Método dos Elementos Finitos

VHDL – VHSIC Hardware Description Language

VHSIC – Very High Speed Integrated Circuits

FPGA – Field Programmable Gate Array

ASIC – Application Specific Integrated Circuits

HDL – Hardware Description Language

CUDA – Compute Unified Device Architecture

AUTOSAR – Automotive Open System Architecture

PLC – Power Line Communication

PAC – Programmable Automation Controller

FMU – Funtional Mockup Units

TI – Tecnologias de Informação

LTE – Long Term Evolution

WLAN – Wireles Local Área Network

RF – Radiofrequência

LIDAR – Light Detection And Ranging

CNN – Convolution Neural Network

API – Application Programming Interfarce

FDA – Food and Drug Administration

ISO – International Organization for Standardization

FIR – Finite Impulse Response

IIR – Infinite Impulse Response

IP – Internet Protocol

RGB – Red-Green-Blue

GUI – Graphical User Interface

EM – Espectro Eletromagnético

TP – True Positive

TN – True Negative

FP – False Positive

FN – False Negative

IV – Infravermelhos

PCA – Análise de Componentes Principais

ICA – Análise de Componentes Independentes

LDA – Análise Discriminante Linear

LBPH – Local Binary Patterns Histograms

LBP – Local Binary Patterns

SURF – Speeded Up Robust Features

SIFT – Scale Invariant Feature Transform

EP – Evolutionary Pursuit

SVM – Support Vector Machine

AAM – Active Appearance Model

EBGM – Elastic Bunch Graph Matching

1. INTRODUÇÃO

Neste capítulo é apresentada uma introdução ao tema do Reconhecimento Facial, abordando tópicos como contextualização do projeto, motivação que levou à realização do mesmo, os objetivos pretendidos e a estruturação do documento.

1.1. CONTEXTUALIZAÇÃO

É impossível falar em Reconhecimento Facial sem abordar temas como Biometria e Processamento de Imagens. A Biometria é responsável pela análise de características físicas do Ser Humano, sendo mais comummente realizadas análises ao ADN, impressões digitais, retina e íris dos olhos, face e padrões da face (localização dos olhos, nariz e boca), voz, geometria das mãos, entre outros. Estas análises possibilitam identificar um indivíduo, já que, à partida, cada indivíduo possui características físicas únicas. No

entanto, embora muitas destas caraterísticas sejam inalteráveis ao longo do tempo, muitas delas podem variar com a idade, estados de saúde, cirurgias, entre outras causas, podendo ocorrer também roubo de identidade. Cada vez mais, a Biometria é utilizada em diversas áreas e aplicações, das quais se destacam a identificação de pessoas pelos agentes de autoridade, segurança em bancos e sistemas de pagamento, controlo de fronteiras no reconhecimento de viajantes, acessos físicos a determinadas empresas e locais públicos, cadeias, entre muitos mais.

O Reconhecimento facial é a capacidade de reconhecer um indivíduo através das suas características físicas faciais, possibilitando a sua identificação e verificação da mesma. Para tal, é possível recorrer-se a diversos algoritmos que auxiliam tanto na deteção de uma face como no posterior reconhecimento, com auxilio de uma base de dados previamente criada e que permite a comparação de dois rostos, para possível verificação da identidade de um indivíduo. Destes algoritmos destacam-se sobretudo os algoritmo de *Viola-Jones*, Análise de Componentes Principais (PCA), Análise de Componentes Independentes (ICA), Análise Discriminante Linear (LDA), Redes Neurais, *Local Binary Patterns Histograms* (LBPH), *Local Binary Patterns* (LBP), *Speeded Up Robust Features* (SURF), *Scale Invariant Feature Transform* (SIFT) e *Brute-Force Matcher*. A Biometria Facial tornou-se no sistema de reconhecimento mais utilizado, já que além de ser de fácil implementação, é o sistema que permite de forma mais rápida o reconhecimento, desde que todas as condições sejam favoráveis, além de não ser necessária a interação dos indivíduos.

Uma vez que a partir de um Sistema de Reconhecimento Facial é possível obter uma imagem de um individuo, nem sempre com as condições mais favoráveis, recorre-se ao processamento de imagem que, por meio de operações matemáticas, possibilita o melhoramento de uma imagem inicial. O processo consiste em obter de uma imagem inicial, uma imagem baseada nesta mas com características de interesse realçadas. Para tal, passa por etapas que permitem a separação da imagem em regiões de interesse, por um processo de segmentação da imagem que visa a diminuição e correção de defeitos que possam existir, e extração e categorização de objetos de interesse. O processamento de imagens, além de operar no espaço visível do espectro eletromagnético (EM), opera

em imagens geradas a partir de ultrassons, microscopia eletrónica e outras imagens geradas por computador.

O MATLAB é um *software* interativo de alta performance, no qual é possível aplicar os algoritmos mencionados anteriormente e desenvolver um sistema que apesar de simples é capaz de detetar faces e reconhecer indivíduos. Assim, com este projeto, pretende-se, essencialmente, desenvolver um sistema em MATLAB que permita o reconhecimento facial recorrendo a uma simples câmara de computador. A motivação para a sua realização passa por alargar o conhecimento em áreas como Biometria, Reconhecimento de Faces e Processamento de Imagens, além de entender o funcionamento de um sistema de Reconhecimento Facial, praticamente utilizado em todas as coisas do dia-a-dia, através do desenvolvimento de um. O mesmo poderá ser uma ferramenta prática, possível a qualquer indivíduo que a queira utilizar.

1.2. OBJETIVOS

O principal objetivo desta dissertação passa pelo desenvolvimento de um Sistema de Reconhecimento Facial, recorrendo a uma câmara de computador. No entanto, para chegar a essa fase, é necessário alagar o conhecimento em diversas áreas. Por isso, tem-se também como objetivos:

- Conhecimento aprofundado da área de Processamento de Imagem;
- Conhecimento aprofundado da área de Biometria;
- Conhecimento aprofundado da área de Reconhecimento Facial;
- Conhecimento aprofundado dos Algoritmos existentes na Deteção e Reconhecimento Facial;
- Conhecimento aprofundado do MATLAB e das suas *Toolboxes,* para posterior desenvolvimento do sistema de Reconhecimento Facial.

1.3. ORGANIZAÇÃO DA DISSERTAÇÃO

A presente dissertação está estruturada em sete capítulos. No primeiro capítulo, designado de Introdução, é feita uma breve introdução sobre o tema da dissertação, revelados os motivos e as metas a alcançar com a mesma, bem como a apresentação da sua estruturação. Segue-se o segundo capítulo, MATLAB, onde é apresentado o *software* e a sua constituição, a história que o acompanhou até ao dia de hoje, as *Toolboxes* que fazem dele um dos melhores *softwares* do mercado, as suas utilizações nas mais diversas áreas e, por fim, as vantagens e desvantagens da sua utilização. No terceiro capítulo, Processamento de Imagem, é brevemente descrito o que é o processamento de imagem, seguindo-se o processamento digital de imagem. É também detalhado o processo em si, composto pelas etapas de Aquisição de Imagens, Pré-Processamento, Segmentação, Pós-Processamento, Extração de Atributos e Classificação e Reconhecimento. No quarto capítulo, Algoritmos de Reconhecimento Facial, são abordados temas como Biometria e Reconhecimento Facial. São ainda enumeradas aplicações para este tipo de reconhecimento e enunciados alguns algoritmos mais utilizados nesta área. No quinto capítulo, Desenvolvimento do Sistema de Reconhecimento Facial, é mostrado de forma detalhada o sistema de reconhecimento desenvolvido com o auxílio do MATLAB. No sexto capítulo, são abordados resultados obtidos com o desenvolvimento do sistema e o sucesso do mesmo. No sétimo e último capítulo, são mostradas as conclusões às quais se chegou com a realização deste projeto, recorrendo à análise de alguns resultados tidos como mais importantes.

2. MATLAB

Este capítulo contém uma breve descrição do MATLAB, a sua constituição e a sua origem e história ao longo do tempo. Conta ainda com a enumeração das *Toolboxes* existentes, da aplicação do MATLAB em situações e projetos reais e das vantagens e desvantagens que acarreta.

2.1. DESCRIÇÃO DO PRODUTO

O MATLAB é visto como um *software* interativo de elevada *performance*, baseado no cálculo numérico. Permite ter acesso a problemas e respetivas soluções de forma rápida e escritos de igual forma caso fossem escritos matematicamente, já que integra análises

numéricas, cálculos com matrizes, processamento de sinais e construção de gráficos. A linguagem utilizada é a linguagem MATLAB, muitas vezes denominada de M-Código mas, no entanto, trata-se de uma combinação entre as linguagens mais utilizadas, como é o caso da linguagem C, Java e Basic.

O logotipo do próprio *software* e da companhia responsável pelo lançamento das suas versões é a solução para a equação de onda, uma base da física matemática onde é representada a vibração de uma membrana esticada sobre uma região em forma de L (Figura 1). A MathWorks é a única empresa a nível mundial cujo logotipo é a solução para uma equação de diferencial potencial [1].

Figura 1: Logotipo MATLAB

O sistema MATLAB é constituído pelas componentes [2]:

a) **Linguagem:** possibilidade de criação e manipulação de matrizes, com vasto leque de funções utilizadas na resolução rápida e eficaz de problemas complexos, além de permitir a construção de expressões matemáticas sem declaração do formato numérico ou dimensão;
b) **Ambiente de trabalho:** possibilidade de gestão, visualização, leitura e gravação de variáveis, criação de programas em linguagem MATLAB e automatização de cálculos;
c) **Gráficos:** possibilidade de criação, visualização, manipulação e interpretação de gráficos;
d) ***Toolbox:*** possibilidade de utilização de caixas de ferramentas de funções nas diversas áreas de cálculo. Estas serão abordadas com mais detalhe, mais à frente nesta dissertação.

2.2. HISTÓRICO

O MATLAB foi criado em 1970 por Cleve Moler, presidente do departamento de ciência da computação da Universidade do Novo México. Até então, o *software* não passava de uma simples calculadora matricial interativa, com apenas 71 palavras e funções armazenadas (Figura 2). Para adicionar uma nova função, era necessário obter um código fonte, escrever uma sub-rotina *Fortran*, adicionar o nome da função à tabela e recompilar o *software* [3].

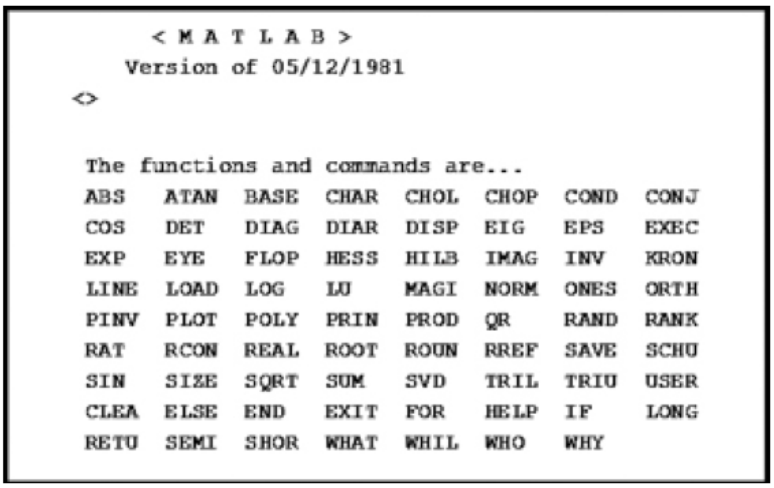

Figura 2: Tela de início com as palavras e funções reservadas

Em 1983, Jack Little verificou que o *software* tinha potencial e, juntamente com Moler e Steve Bangert, adicionaram-lhe novos recursos: um analisador/interprete e bibliotecas de matemática, incluindo escrita em Linguagem C. Não obstante, surgiu a possibilidade de adicionar facilmente novas funções organizadas em caixas de ferramentas (*Toolboxes*), que permitiam que o sistema estivesse disponível para uma elevada gama de equipamentos, e utilizar gráficos disponíveis. No entanto, era preocupante o tamanho de código nas versões iniciais do MATLAB.

Em 1984, Moler, Jack e Steve, fundaram a MathWorks, que lançou o MATLAB 1.0, implementado em C para computadores MS-DOS. O MATLAB estreou-se na IEEE *Conference on Design and Control,* em Las Vegas.

Desde então, foi sempre a crescer: em 1985, a Mathworks vendeu o primeiro pedido de 10 cópias do MATLAB, para o MIT; em 1986 foi lançava a versão MATLAB 2, para estações

de trabalho UNIX e em 1987 a versão 3; e em 1990 foi lançado o Simulink 1.0. O MATLAB 4, com gráficos 2D e 3D a cores e matrizes esparsas, foi lançado em 1992, juntamente com o MATLAB *Student Edition*. As matrizes esparsas foram uma forma eficiente de armazenar memória para representar matrizes, já que apenas os elementos diferentes de zero eram armazenados.

Em 1993, o *site* mathworks.com foi lançado como um dos 75 primeiros *sites* comerciais registados e foi desenvolvido o MATLAB para Microsoft Windows.

Em 1995, foi lançado o MATLAB para Linux e o Compilador MATLAB, que faz a conversão de programas escritos em MATLAB para linguagem C.

Com o crescimento da empresa e do avanço do próprio *software*, realizaram-se várias conferências de usuários do MATLAB em diversos países. Foi possível, então, em 1996 o lançamento da versão 5, que incluía tipos de dados, visualização avançada, um depurador, um criador de perfil e um construtor de GUI, além da introdução de matrizes de células, estruturas e "noção de ponto". As matrizes de células são criadas por {} e não passam de coleções indexadas de objetos de MATLAB.

Em 2000, iniciou-se o suporte à aritmética de precisão única e foi lançada a versão MATLAB 6, com o novo *desktop* MATLAB, LAPACK (bibliotecas LINPACK e EISPACK), e Transformada de *Fourier* mais rápida do Oeste (FFTW). Surge também a área de trabalho do MATLAB (Figura 3).

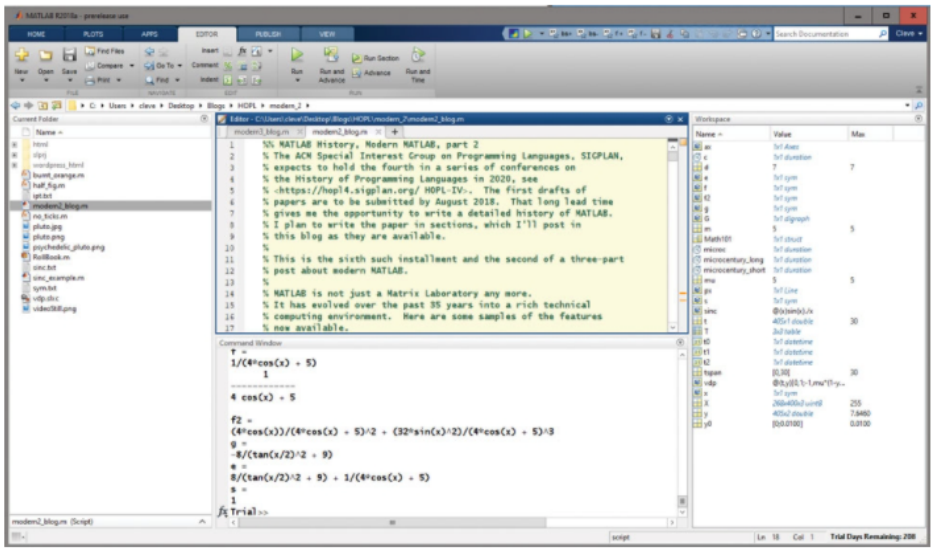

Figura 3: Versão atualizada da tela

Na versão de 2018 da área de trabalho, é já possível visualizar quatro painéis: o visualizador de pastas (à esquerda), o visualizador da área de trabalho (à direita), o editor/depurador (centro superior) e a janela de comando (centro inferior). Pode também alterar-se o *layout* e incluir-se um visualizador de arquivos e uma janela do histórico de comandos, ou apenas fechar e tornar independente qualquer painel [3].

Em 2002 ocorreu o lançamento do MATLAB 6.5, com aceleração *just-in-time* e suporte renovado ao Macintosh.

Em 2004 surge o MATLAB 7, com o qual é possível obter uma redução para metade da necessidade de memória direcionada para matrizes de elevada dimensão e matemática inteira, funções alinhadas e anónimas, lote de ferramentas e desenvolvimento de um algoritmo interativo. Neste mesmo ano, introduziu-se na conferência *SuperComputing*, o *Parallel Computing Toolbox*, uma caixa de ferramentas que suporta paralelismo de memória distribuída de granulação grossa, executando funções do MATLAB em várias máquinas ou núcleos de uma máquina única.

Bill Gates usou o MATLAB, em 2005, para apresentar a entrada da Microsoft na computação de alto desempenho.

No ano de 2006, o número de produtos que complementam o MATLAB o e Simulink excederam os 300, e estavam já disponíveis mais de 800 livros baseados no *software* e nos respetivos complementos.

Em 2018, realizaram-se os principais aprimoramentos dos recursos de programação orientada a objetos do MATLAB, nomeadamente, a criação de classes.

Em 2016 foi introduzido o *Live Editor*, com possibilidade de combinar num único documento interativo e exportado para HTML, PDF ou LaTex, texto descritivo e gráficos do MATLAB.

A versão 14, com *Service Pack* 3, veio fornecer atualizações para MATLAB, Simulink e outros produtos, recorrendo a novos recursos para análise de dados, modelagem em larga escala e geração de código.

Aquando do lançamento da versão 2018a, havia cerca de 63 *Toolboxes* distribuídas pelas áreas de Implantação de Aplicativo, Geração de Código, Biologia Computacional, Finanças Computacionais, Sistemas de Controlo, Acesso a bancos de dados e relatórios, Processamento de imagem e visualização computacional, Matemática, Estatística e Otimização, Computação Paralela, Processamento de sinais e comunicações sem fio, Teste e medição.

Atualmente, o MATLAB continua a evoluir de forma a atender às necessidades de engenheiros, cientistas e todos aqueles que o utilizam, tendo em conta a facilidade de uso, potência e velocidade.

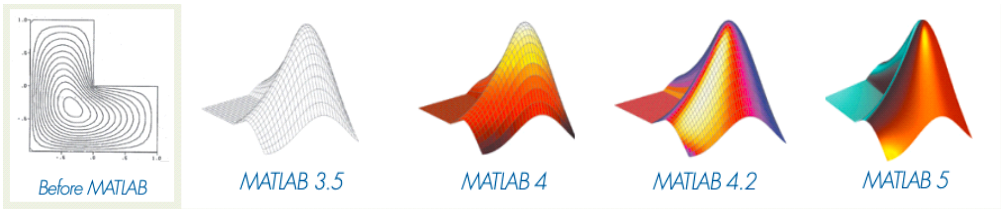

Figura 4: Evolução gráfica da membrana em forma de L

A Figura 4 mostra como a membrana em forma de L e a computação gráfica evoluíram ao longo dos tempos: antes do MATLAB era possível a criação de um gráfico de contorno bidimensional; com a versão 3.5, tornou-se possível fazer o gráfico tridimensional em

preto e branco, usando algoritmos de linha oculta; o MATLAB 4 e 4.2 introduziram a cor e o sombreado nos gráficos; a versão 5, trouxe o sombreado e a iluminação de forma mais trabalhada; e, posteriormente, foi possível alterar o ponto de vista, o sombreado e a iluminação, observando a vibração da membrana em tempo real.

2.3. TOOLBOXES

O MATLAB dispõe de diversas extensões, denominadas *Toolboxes*, que oferecem funcionalidades profissionalmente desenvolvidas, testadas e documentadas, com o intuito de serem aplicadas nas mais diversas áreas. Das *Toolboxes* do MATLAB destacam-se as seguintes [4]:

a) ***Parallel Computing*:** possibilidade de acelerar tarefas longas e com elevada memória, aproveitando recursos de computação de alto desempenho, sem alterar os fluxos de trabalho. Assim, torna-se possível reduzir o esforço de programação, executar um aplicativo em recursos de computação de alto desempenho e executar código paralelo. Fazem parte deste grupo:

   - *Parallel Computing Toolbox*;
   - *MATLAB Parallel Server*.

b) ***Math and Otimization:*** possibilidade de, através de regressão, interpolação e suavização, ajustar curvas e superfícies; resolver problemas de otimização linear, não linear, quadrática, de máximos e mínimos; executar cálculos matemáticos simbólicos; analisar e visualizar informações geográficas; e, utilizando o Método dos Elementos Finitos (MEF), resolver equações diferenciais parciais. Para tal, conta-se com:

   - *Curve Fitting Toolbox;*
   - *Otimization Toolbox;*
   - *Global Optimization Toolbox;*
   - *Symbolic Math Toolbox;*
   - *Mapping Toolbox;*

- *Partial Differential Equation Toolbox.*

c) ***AI, Data Science and Statistics***: responsável por, através de estatísticas e aprendizado de máquina, analisar e modelar dados; projetar, treinar e analisar redes de aprendizado profundo; projetar políticas de treino por meio de aprendizado reforçado; analisar e modelar dados de texto e monitorizar condições e algoritmos de manutenção preditiva. Deste grupo fazem parte:

- *Statistics and Machine Learning Toolbox;*
- *Deep Learning Toolbox;*
- *Reinforcement Learning Toolbox;*
- *Text Analytics Toolbox;*
- *Predictive Maintenance Toolbox.*

d) ***Code Generation***: possibilidade de gerar código C e C++ a partir de um código MATLAB e para sistemas embarcados; gerar código VHDL e *Verilog* para projetos FPGA e ASIC; testar e verificar *Verilog* e VHDL usando simuladores HDL e placas FPGA; gerar código HDL para filtros de ponto fixo; modelar e otimizar algoritmos de ponto fixo e ponto flutuante e gerar código CUDA para GPUs NVIDIA. Destacam-se:

- *MATLAB Coder;*
- *Embedded Coder ;*
- *HDL Coder ;*
- *HDL Verifier;*
- *Filter Design HDL Coder;*
- *Fixed-Point Designer;*
- *GPU Coder.*

e) ***Application Deployment:*** permite a criação de aplicativos da *Web* independentes e a integração de algoritmos MATLAB nos mesmos e em bancos de dados, bem como partilha de aplicativos MATLAB e simulações Simulink como aplicativos *Web* baseados num navegador; e construção de componentes de *software*. Fazem parte do grupo:

- *MATLAB Compiler;*
- *MATLAB Compiler SDK ;*
- *MATLAB Production Server ;*
- *MATLAB Web App Server.*

f) ***Database Access and Reporting*****:** responsável por trocar dados com bancos de dados relacionais e não relacionais e por projetar e gerar automaticamente relatórios. Este grupo conta com:

- *Database Toolbox;*
- *MATLAB Report Generator.*

Além de todas as extensões abordadas anteriormente, o MATLAB conta com o Simulink, um ambiente de simulação baseado em diagramas e bibliotecas de blocos. É utilizado sobretudo em teoria de controlo e processamento digital de sinais.

a) ***Event-Based Modeling:*** possibilita modelar e simular lógica de decisão, sistemas de comunicação de mensagens e eventos discretos. O grupo conta com:

- *Stateflow;*
- *SimEvents.*

b) ***Physical Modeling*****:** permite modelar e simular sistemas físicos de vários domínios, sistemas mecânicos de rotação e translação, sistemas de energia eletrónica, mecatrónica e elétrica e sistemas de fluidos. Para tal, existem:

- *Simscape;*
- *Simscape Driveline;*
- *Simscape Electrical;*
- *Simscape Fluids;*
- *Simscape Multibody.*

c) ***Real-Time Simulation and Testing:*** permite construir, executar e testar aplicativos e modelos Simulink em tempo real. Deste grupo fazem parte:

- *Simulink Real-Time;*

- *Simulink Desktop Real-Time.*

d) ***Code Generation**:* possibilita gerar Código C e C++ e código otimizado para sistemas embarcados; projetar e simular AUTOSAR; modelar e otimizar algoritmos de ponto fixo e ponto flutuante; gerar diagramas estruturados de texto e escala IEC 61131-3 para PLCs e PACs; automatizar as revisões de código-fonte para os padrões de segurança; quantificar as ferramentas de verificação Simulink e Poliscape para DO-178, DO-278 e DO-254, assim como as ferramentas de geração e verificação de código para a certificação ISO 26262 e IEC 61508; gerar código VHDL e Verilog para projetos FPGA e ASIC e testar e verificar Verilog e VHDL usando simuladores HDL e placas FPGA. Destacam-se:

- *Simulink Coder;*
- *Embedded Coder;*
- *AUTOSAR Blockset;*
- *Fixed-Point Designer;*
- *Simulink PLC Coder;*
- *Simulink Code Inspector;*
- *DO Qualification Kit (for DO-178);*
- *IEC Certification Kit (for ISO 26262 and IEC 61508);*
- *HDL Coder;*
- *HDL Verifier*.

e) ***Application Deployment**:* responsável pela partilha de simulações como executáveis autónomos, aplicativos da *Web* e FMU, e do qual faz parte:

- Simulink Compiler.

f) ***Verification, Validation and Test**:* permite criar, gerir e rastear requisitos e medir a cobertura de modelos, código gerado e casos de teste; verificar a conformidade com as diretrizes de estilo e os padrões de modelagem; identificar erros de projeto e comprovar a conformidade, assim como erros de *software* e prova da sua inexistência; desenvolver, gerir e executar testes baseados em simulações. Fazem parte deste grupo:

- *Simulink Requirements;*
- *Simulink Check;*
- *Simulink Coverage;*
- *Simulink Design Verifier;*
- *Simulink Test;*
- *Polyspace Bug Finder;*
- *Polyspace Code Prover.*

g) ***Simulation Graphics and Reporting**:* possibilidade de visualizar o comportamento dinâmico do sistema num ambiente de realidade virtual, assim como projetar e gerar automaticamente relatórios a partir de modelos e simulações do Simulink. Este grupo conta com:

- *Simulink 3D Animation;*
- *Simulink Report Generator.*

2.4. APLICAÇÕES

O MATLAB é utilizado por uma vasta comunidade que, com os seus *inputs*, contribui para a existência das *Toolboxes*. No entanto, o MATLAB destaca-se pela sua utilização em [5]:

i. **Análise de Dados:** utilizado em praticamente todas as tarefas que englobam a análise e tratamento de dados. Especificamente, permite criar modelos preditivos e de aprendizado de máquina; implementar modelos em sistemas TI com base em ferramentas de acesso e pré-processamento de dados; ter acesso a dados já armazenados em bancos de dados de qualquer tipo; limpar e gerir dados, documentar análises previamente realizadas, recorrendo para tal a gráficos do MATLAB e do *Live Editor*; aplicar técnicas de engenharia de recursos ligadas ao domínio para sensor, texto, imagem e vídeo; utilizar aplicativos de aprendizado de máquina e profundo para o ajuste de modelos e sistemas de TI de produção e converter automaticamente modelos para código C e C++ independente. Pode ser aplicado em situações reais como:

a. **Industria Automóvel, pela BMW**, no reconhecimento de excessos de direção, em automóveis [6]. Esta é uma condição de extrema insegurança, na qual ocorre perde de aderência por parte dos pneus traseiros e deve-se, sobretudo a pneus degastados, condições de terreno, velocidade, fretamento brusco ou qualquer combinação entre as causas abordadas. As soluções existentes atualmente procuram executar ações corretivas aquando da deteção de uma viragem acentuada por parte do automóvel. No entanto, estas soluções são de difícil implementação, o que levou ao desenvolvimento de um modelo de aprendizado de máquina, capaz de detetar excessos de direções com mais de 98% de precisão.

b. **Serviços públicos e energia, pela Baker Hughes**, na manutenção de equipamentos de extração de gás e óleo [7]. Para exploração de apenas um reservatório de petróleo e gás natural são necessários cerca de 20 camiões com bombas de deslocamento positivo, para injeção simultânea de uma mistura de água e areia a altas pressões. Estas bombas e os seus componentes representam um custo elevado para a empresa, o que levou à procura de algoritmos de aprendizado de máquina para monitorização das bombas, dando informação de desgaste e previsão de falhas antes da sua ocorrência.

c. **Neurociência, pela Battelle,** na restauração do movimento do braço e da mão de um individuo paralisado (Figura 5) [8]. Uma paralisia permanente pode resultar de uma doença ou lesão, com interrupção das vias neurais que conectam o córtex motor do cérebro aos músculos. De forma a auxiliar o indivíduo no aumento da sua independência, foi desenvolvido um sistema que visa restaurar o controlo muscular, utilizando sinais neurais gravados intercorticalmente em humanos, recorrendo para isso a microeléctrodos implantados no cérebro do paciente, permitindo-lhe recuperar o controlo do antebraço, mão e dedos.

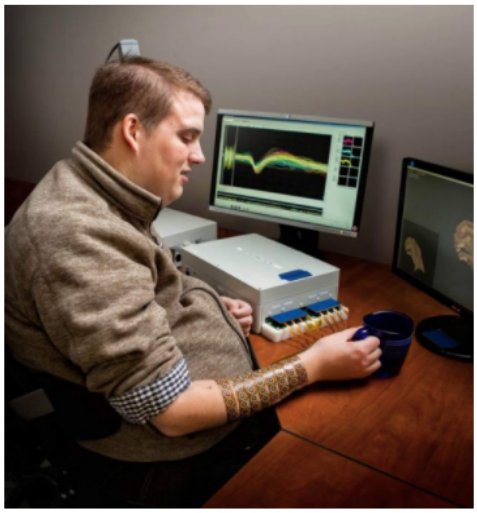

Figura 5: Paciente usando o sistema Battelle NeuroLife (sistema desenvolvido com auxilio do MATLAB)

ii. **Comunicação sem fio:** utilizado para redução do tempo necessário para desenvolver e otimizar testes e verificações. Para tal, permite testar algoritmos e projetos de sistemas através de simulações e sinais aéreos; personalizar formas de onda para verificação da conformidade com os padrões 5G, LTE e WLAN; criar modelos com base em elementos digitais, RF e antena com a finalidade de otimizar o comportamento do sistema, e modelos para verificação iterativa de projetos, protótipos e implementações sem fios; gerar código C e HDL para prototipagem e implementação; e por fim, automatizar a análise de dados de teste de campo em larga escala e visualizar os resultados da simulação.

iii. ***Deep Learning*:** utilizado na criação e análise de arquiteturas de aprendizado profundo por meio de aplicativos e ferramentas de visualização, no processamento e automatização de dados de imagem, vídeo e áudio; acelerar algoritmos em recursos de GPUs NVIDIA, nuvem e *datacenter* sem programação especializada; usar *TensorFlow*, *PyTorch* e *MxNet*; simular o comportamento dinâmico do sistema e gerar dados e testes, tendo por base a simulação de sistemas fixos (usando MATLAB e Simulink). Aplica-se em áreas como:

    a. **Geologia, pela Shell**, no reconhecimento de *tags* em imagens industriais e reconhecimento de terreno em imagens de satélite hiperespectrais, para exploração e identificação de recursos deológicos subterrâneos [9]. Recorre-se a dados sísmicos para criação de imagens de geologia

subterrânea e identificação de possíveis acumulações de depósitos de hidrocarbonetos. No entanto, muitas das vezes, essas imagens não possuem qualidade suficiente, o que leva à perfuração de poços não produtivos. Isto engloba perda de recursos, tempo e, consequentemente, aumento de custos. Para tal, foram desenvolvidos algoritmos que usam medições de recursos geológicos conhecidos, para prever outros recursos na mesma região, fornecendo prognósticos estatísticos de presença e carácter de reservatórios (Figura 6).

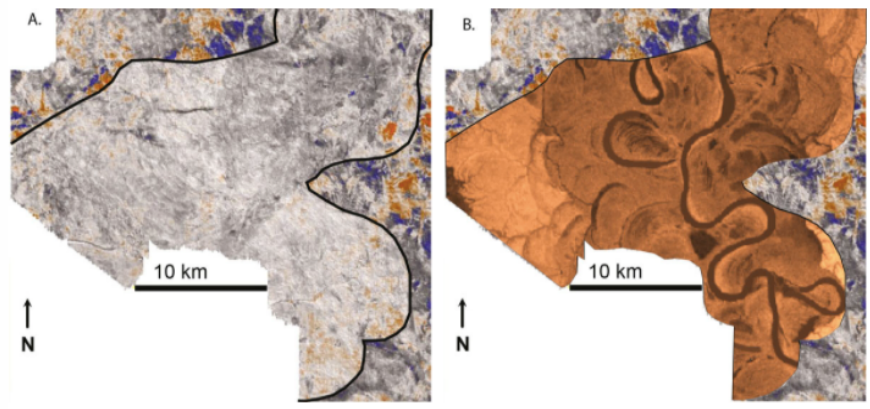

Figura 6: (A.) Imagem obtida através de dados sísmicos 3D, de um potencial reservatório a ser explorado. (B.) Previsão de uma possível configuração para a geologia interna do reservatório

b. **Industria automóvel, pela Autoliv,** na anotação automatizada da nuvem de pontos LIDAR para verificação de sensores [10]. Neste tipo de ramo, é fundamental uma boa precisão na deteção e funcionalidade correta dos sensores. Assim, foi desenvolvida uma ferramenta em MATLAB, que auxilia na verificação de sensores de radar, automatizando o processo de verificação (deteta e rastreia objetos; deteta e verifica pontos cegos e métricas de distâncias).

c. **Saúde, pela Universidade de Ritsumeikan,** na redução do risco de exposição à radiação para obtenção de imagens de tomografia computadorizada [11]. Estudos revelam que, embora imagens 3D apresentarem um melhor diagnóstico do que um simples raio-X, acarretam riscos para quem se expõe à radiação. Esta questão levou ao desenvolvimento de um sistema baseado na regressão da CNN que utiliza

tomografia computadorizada de baixa de radiação, mas que gera imagens de igual qualidade à tomografia normal. Com este sistema, é possível fornecer um elevado nível de qualidade de diagnóstico e reduzir em 95% a exposição à radiação do paciente.

iv. **Processamento de imagem e visão computacional**: Juntamente com o Simulink, permite obter informações relativamente a dados de imagem e vídeo, e desenvolver algoritmos. É possível criar soluções de visão através de algoritmos de referência em processamento de imagens, visão por computador e aprendizado profundo; usar OpenCV, Python, C e C++ pelo meio de APIs interoperáveis e ferramentas de integração; usar aplicativos de fluxo de trabalho para automatizar tarefas comuns e acelerar a exploração de algoritmos, nomeadamente algoritmos em recursos GPUs, nuvem e *datacenter* da NVIDIA, sem programação especializada ou conhecimento TI; e implementar algoritmos em dispositivos incorporados, incluindo GPUs NVIDIA, processadores Intel e FPGAs e processadores embarcados baseados em ARM. Tem aplicação em áreas como:

   a. **Dispositivos médicos, pela Medviso,** na análise de imagens para ressonância magnética e aprovação regulatória segura para uso clínico [12]. Antes de proceder ao tratamento de um paciente com determinada condição cardíaca, o médico necessita de saber o funcionamento do órgão. São as ressonâncias magnéticas e tomografias que fornecem imagens da estrutura do coração, embora não forneçam informações a cerca da quantidade de sangue bombeado em cada contração, o volume diastólico e sistólico do ventrículo, a massa do miocárdio, entre outros. Desta forma, a empresa desenvolveu um *software* de análise de imagens, em MATLAB, que auxilia na quantificação automática das medidas de avaliação cardíaca. Com este feito, a empresa recebeu ainda aprovação do FDA 510 (k) nos EUA e a marcação CE na Europa, certificado de acordo com a norma ISO 13485.

v. **Processamento de sinais:** juntamente com o Simulink, é possível, por meio de aplicativos internos, visualizar e pré-processar sinais nos domínios de tempo,

frequência e frequência de tempo, de forma a detetar padrões e tendências sem precisar de escrever manualmente um código. Permite também projetar e analisar filtros digitais, desde passagens simples de frequência até formas avançadas de FIR e IIR, visualizando a magnitude, fase, atraso de grupo e resposta a impulsos, e avaliar o desempenho do filtro, tal como estabilidade e linearidade. Pode ocorrer apenas visualização e análise, mas também simulação.

vi. **Finanças quantitativas e gestão de riscos:** utilizado para criar protótipos e validar modelos de finanças computacionais; acelerar modelos através de processamento paralelo, colocando-os diretamente em produção; determinar taxas de juros; gerir contas e negociar instrumentos complexos. Toda a documentação é gerida automaticamente e os modelos podem ser implementados, de forma protegida por IP, diretamente em aplicativos da *Web* e de *desktop,* como Excel, Java, C++ e *Pytyhon*.

vii. **Robótica e Sistemas Autónomos:** permite conectar e controlar robots (sistema operacional Robot – ROS e ROS2) através de algoritmos de *hardware*. A ligação pode realizar-se através de sensores e atuadores, de forma a enviar sinais de controlo ou análise de dados. É possível também eliminar a codificação manual, gerando código automaticamente para destinos incorporados e conectar-se a um *hardware* de baixo custo, como o Arduino e Raspberry Pi.

    a. **Desenvolvimento de Robot Humanóide Autónomo com *Design* Baseado em Modelo, pelo Centro Aerospacial Alemão (DLR),** capaz de interagir com o ambiente [13]. Trata-se de um robot humanoide móvel, de dois braços, com 53 graus de liberdade, que deteta o ambiente através de câmaras estéreo e sensores RGB-D, sensores de torque e sensores táteis. Para a sua utilização foram desenvolvidos, em MATLAB, algoritmos avançados de controle, calibração e planeamento do caminho.

2.5. VANTAGENS E DESVANTAGENS DA SUA UTILIZAÇÃO

Das grandes vantagens da utilização deste *software*, destacam-se [14]:

- A linguagem aplicada que segue de forma fiel a maioria dos aspetos das linguagens que estiveram na sua origem. Desta forma, qualquer que seja o utilizador, não terá qualquer dificuldade na adaptação a MATLAB, já que os códigos-fonte são bastante parecidos e a construção de objetos GUI bastante fácil de ser conseguida;

- Os comandos são muito próximos da forma como são escritas as expressões algébricas, o que facilita a sua utilização;

- Podem incorporar-se rotinas predefinidas e pacotes para cálculos específicos;

- As *Toolboxes* oferecem funcionalidades altamente profissionais, rigorosamente testadas e com documentação completa, desenvolvidas em especialidades como engenharia, economia, bioinformática, finanças, entre outras;

- Capacidade de gerar códigos rapidamente, com as operações matemáticas distribuídas ao longo do processo, bibliotecas otimizadas e todos os códigos compilados *just-in-time;*

- Possibilidade de cálculo matemático, manipulação de matrizes e visualização de gráficos;

- Elevado grau de Confiabilidade, já que, a MathWorks possui uma equipa de engenheiros dedicados à verificação da qualidade do *software*, realizando milhões de testes nas bases de códigos MATLAB todos os dias;

- É independente da plataforma utilizada, uma vez que o código desenvolvido corre em diferentes SO, como por exemplo, Windows, Linux, Unix, Macintosh.

No entanto, conta também com algumas desvantagens, tais como:

- Apesar de o código poder ser compilado, é mais demorado na sua execução que Fortran ou C (C++);

- É uma linguagem interpretada, logo a sua execução torna-se mais lenta que linguagens compiladas;

- A licença para o seu uso é bastante cara, cerca de 800 euros para uma licença *Standard* anual. No entanto, este valor pode ser compensado pela redução de tempo de execução e portanto, torna-se um produto atrativo quando relacionando o custo-benefício.

3. PROCESSAMENTO DE IMAGENS

Este capitulo apresenta, sobretudo, definições para Processamento de Imagem e Processamento Digital de Imagem. Apresenta ainda as ferramentas disponíveis para o processo e as aplicações do método.

3.1. PROCESSAMENTO DE IMAGEM

Uma definição simples de imagem é, para qualquer individuo, uma representação visual de uma pessoa, animal ou objeto, ou seja, é aquilo que se conhece de algo que se consegue observar. No entanto, matematicamente falando, uma função bidimensional, onde as variáveis x e y são coordenadas espaciais e a amplitude f, em qualquer par de

coordenadas (x,y), é a intensidade de imagem nesse ponto, pode também ser uma definição de Imagem. A imagem torna-se uma imagem digital, quando os valores de x, y e f são quantidades finitas e discretas, cada uma com uma localização e valores específicos, e o seu processamento é feito por meios digitais, nomeadamente um computador [15]. O *pixel*, abreviatura de *picture elements*, é o menor elemento que compõe uma imagem digital, pelo que, um conjunto de *pixels* forma uma imagem completa. A forma mais comum de apresentação do *pixel* é na forma quadrada (Figura 7).

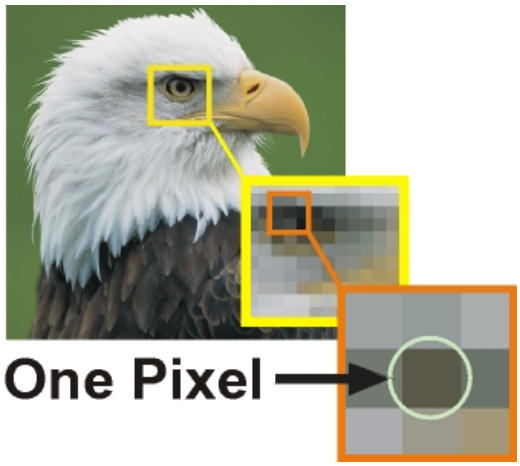

Figura 7: Imagem com destaque para uma região de 3x3 *pixels*

3.2. PROCESSAMENTO DIGITAL DE IMAGEM

O corpo humano acarreta com ele cinco sentidos de extrema importância, que o auxiliam no seu dia-a-dia: a visão, por exemplo, apresenta um papel fundamental na perceção humana, mas é limitado à banda visível do espectro eletromagnético (EM). Contrariando esta limitação do ser humano, surgem máquinas de imagem que cobrem todo o EM, permitindo operar imagens cujo humano não consegue processar, nomeadamente imagens geradas a partir de ultrassons, microscopia eletrónica e outras imagens geradas por computador.

Apesar de não haver concordância relativamente à definição de processamento digital de imagens, pode considerar-se como um processo com entrada e saída de imagens, onde ocorre, por meio de operações matemáticas, transformações nos valores de pixéis e

melhoramento de imagem, permitindo a extração de atributos e o seu reconhecimento individualmente.

Assume-se a existência de três tipos de processamento de imagens [16]:

a) De nível baixo: caracterizado pelo facto de as entradas e saídas serem uma imagem e por envolver ações como pré-processamento de imagem, tendo como finalidade reduzir o ruído e aprimorar o contraste e a nitidez da imagem.

b) De nível médio: caracterizado pelo facto de as entradas serem imagens e as saídas serem atributos extraídos dessas imagens e por envolver tarefas de segmentação, divisão da imagem em objetos, e descrição dos objetos, com a finalidade de reduzi-los para um formato adequado e classifica-los de forma individualmente.

c) De nível alto: caracterizado por interpretar objetos reconhecidos, desempenhando funções cognitivas normalmente associadas à visão.

3.3. ETAPAS DO PROCESSAMENTO DIGITAL DE IMAGEM

Como visto anteriormente, o processamento digital de imagens é uma ferramenta que recorre a operações matemáticas para transformar a própria imagem, ao alterar os valores dos *pixels* e melhorar as suas características e apresentação. Usualmente, o processamento digital de imagem segue a ordem ilustrada pela Figura 8, em etapas que vão desde a Aquisição da Imagem até à Classificação e Reconhecimento de objetos de uma imagem.

O processo tem então início com a Aquisição da imagem, quando ocorre a formação ou digitalização de uma imagem. Nesta fase do processo, a imagem chama-se normalmente, imagem original. A partir daqui, o processo torna-se digital e processável.

A segunda etapa do processo é designada de Pré-Processamento da imagem, que visa proporcionar o melhoramento da mesma. É a etapa que antecede a Segmentação,

responsável pela separação da imagem em regiões de interesse. Nestas duas últimas etapas opera-se ao nível do *pixel*, tendo com resultado regiões de *pixels*.

De forma a corrigir potenciais defeitos, torna-se necessária uma etapa de Pós-Processamento, onde a imagem é segmentada. Aqui é possível extrair objetos sem interesse, juntando-os ou separando-os [17].

Na etapa de Extração de Atributos é possível a extração de determinadas características para posterior categorização dos objetos.

Por fim, a última etapa, designada de Classificação e Reconhecimento, possibilita dar um nome e significado a um objeto extraído. Diferente das etapas de Pré-Processamento e Segmentação, nestas duas últimas etapas trabalha-se já ao nível de dados mais complexos [17].

Todo o processo vai ser abordado com mais detalhe nos pontos que se seguem.

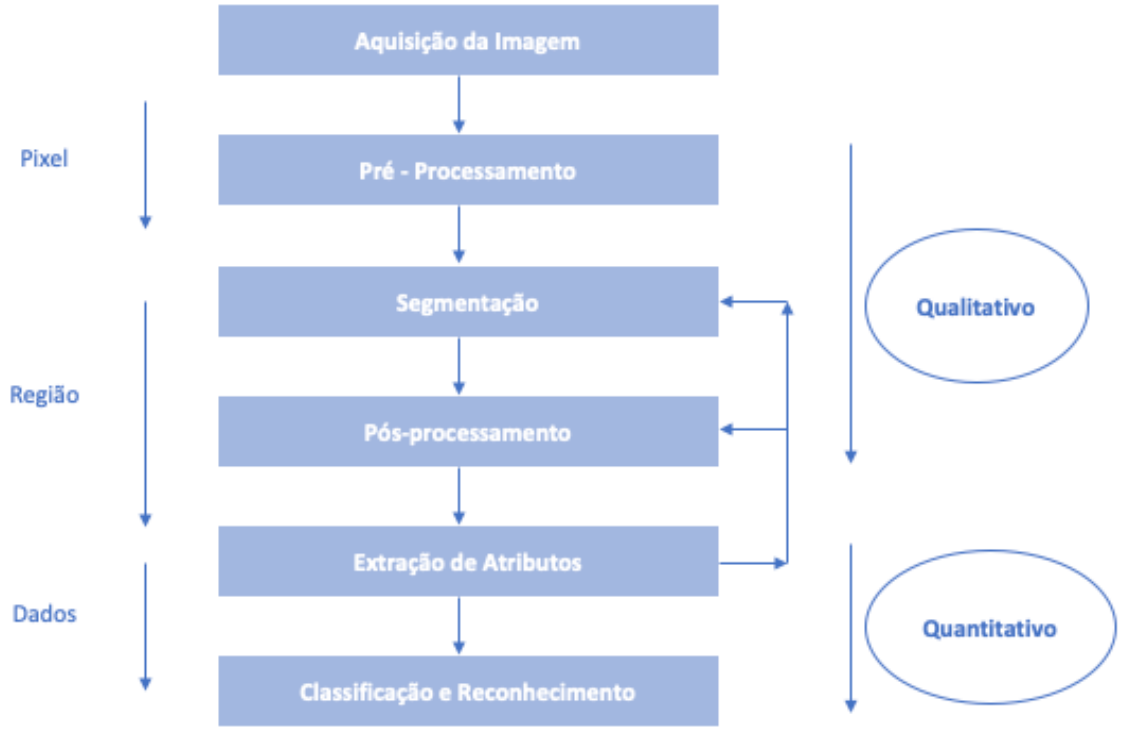

Figura 8: Etapas de um sistema de Processamento Digital de Imagem

3.3.1. AQUISIÇÃO DE IMAGENS

Para iniciar esta etapa é necessário um instrumento capaz de formar uma imagem, um dispositivo físico sensível ao EM, para produzir um sinal elétrico com nível de energia semelhante ao que recebeu e também um digitalizador que permita converter o sinal elétrico analógico produzido pelo dispositivo físico, num sinal digital [15].

Tal como vimos anteriormente, uma imagem digital pode ser vista como uma função bidimensional, onde as coordenadas x e y identificam um ponto na imagem, com valor correspondente ao nível de cor, comummente o nível cinza, designado de *pixel*.

A resolução de imagem consiste na quantidade de pontos presentes numa imagem, por polegada, facilmente representada por uma matriz (Figura 9). Por sua vez, a profundidade ou quantização remete ao número máximo de intensidades que uma imagem pode apresentar, sendo que, quanto maior o intervalo de tonalidades, mais definidas estarão as intensidades, menos progressiva será a alteração de tons e mais intensidades estarão representadas na escala (Figura 10). Uma imagem digital torna-se binária quando apenas apresenta duas intensidades. A profundidade mais comum e satisfatoriamente alcançada é de 8 *bits/byte* = 2^8 =256 tons ou intensidades [17].

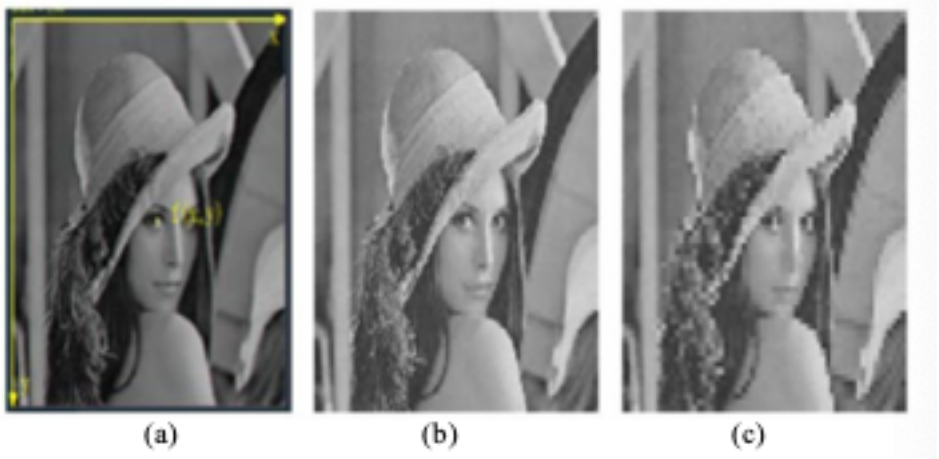

Figura 9: Imagem com diferente resolução: (a) resolução de 256x256 *pixels*; (b) resolução de 128x128 *pixels*; (c) resolução de 64x64 *pixels*

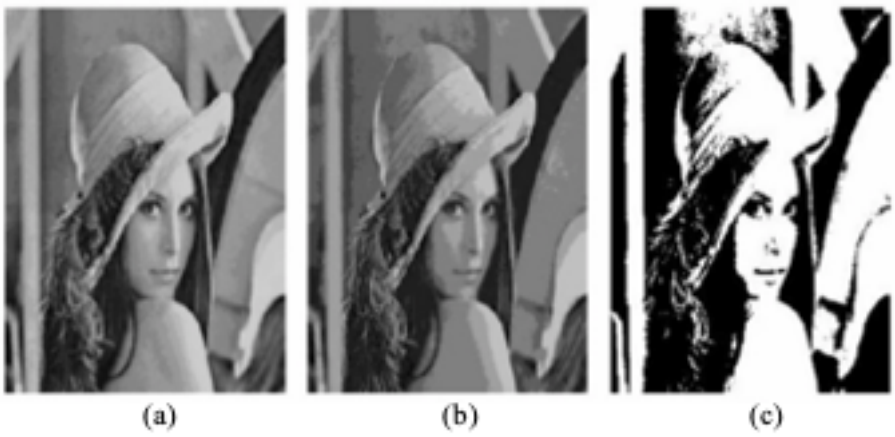

Figura 10: Imagem com diferente profundidade: (a) 16 níveis de cinza; (b) 8 níveis de cinza; (c) 2 níveis de cinza

Embora a resolução e a profundidade sejam, muitas das vezes, pouco percetíveis ao olho humano, o mesmo não acontece com o brilho e contraste de uma imagem. Uma imagem percebe-se clara e de alto brilho quando apresenta até 255 (branco) tons de cinza, e escura e sem brilho com 0 (preto) valores de cinza. Por sua vez, uma imagem com grande variação de tons de cinza é uma imagem com alto contraste, contrariamente a uma imagem com limitada variação de tons de cinza, que apresenta baixo contraste. Em termos matemáticos, o brilho pode ser definido como a média e o contraste como o desvio padrão dos tons de cinza de todos os *pixels* da imagem [18]. As Figuras 11 e 12 mostram as alterações de brilho e de contraste de duas imagens de objetos.

Figura 11: A partir de uma imagem original (a) é possível aumentar (b) ou diminuir (c) o brilho da mesma

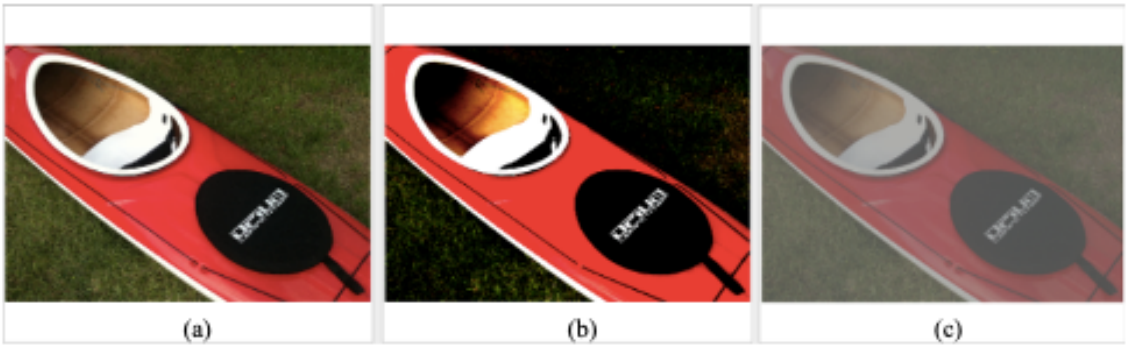

Figura 12: A partir de uma imagem original (a) é possível aumentar (b) ou diminuir (c) o contraste da mesma

Muitas das vezes, é usado um histograma para a mostrar a distribuição de intensidade dos *pixels*. Este histograma representa a intensidade dos *pixels* para 256 tons e a probabilidade de ocorrência de tons cinza na imagem [15]. Pode ser ainda representado pela equação (1):

$$p(k) = n_k / n \; . \qquad (1)$$

onde *k* é o nível de cinza, variando entre valores 0 e 255; n_k é o número de *pixels* na imagem com nível de cinza *k*; *n* é o número total de *pixels* na imagem e *p(k)* é a estimativa da probabilidade de ocorrência do nível cinza *k.*

Apesar de ser impossível, por meio de um histograma, obter uma descrição exata do conteúdo da imagem, é possível obter estatisticamente a distribuições de *pixels*, do brilho e do contraste. Obtendo esta informação, melhoram-se as condições de captura já que são cruciais no processamento digital de imagens [15].

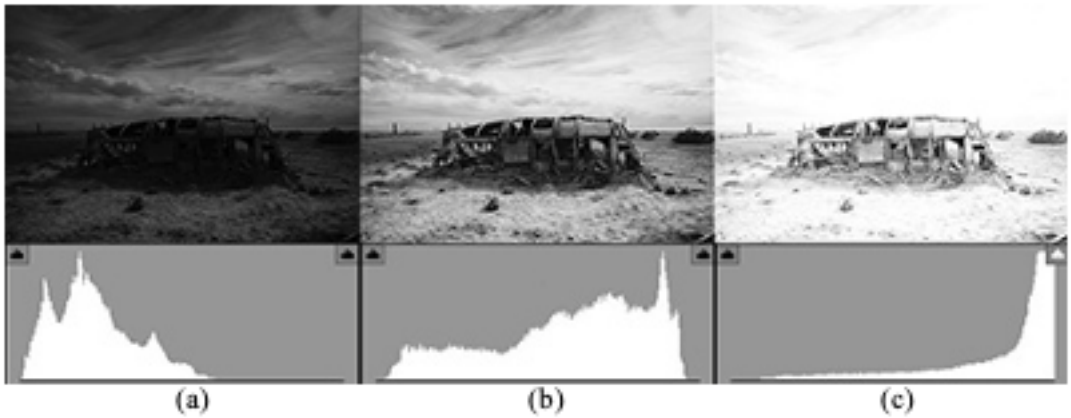

Figura 13: Histograma de diferentes exposições de uma imagem

A Figura 13 mostra como através de um histograma se pode obter diversas informações referentes a imagens: a imagem (a) apresenta um histograma onde os níveis de cinza estão mais concentrados na região esquerda, considerada a mais escura, e cujos valores estão menos distribuídos que nos restantes. Por isso, pode afirmar-se que se trata de uma imagem escura e com pouco contraste. Por sua vez, a imagem (b) apresenta os níveis de cinza distribuídos uniformemente e um histograma mais largo, o que sugere uma imagem mais clara e com mais contraste que a imagem anterior. Por fim, a imagem (c) prevê-se mais clara e com bastante contraste, já que os níveis de cinza estão concentrados na região da direita, considerada a mais clara, e o histograma também é mais largo. Em conclusão, pode afirmar-se que a imagem (a) é a mais escura e com menos contraste, e as imagens (b) e (c), apesar de terem contrastes semelhantes, têm brilhos diferentes, já que a imagem (c) é mais clara que a (b).

O histograma assume ainda um papel importante nas etapas de Pré-Processamento (quando necessário) e Segmentação, em técnicas onde se utiliza a similaridade de *pixels*.

3.3.2. PRÉ-PROCESSAMENTO

A etapa de Pré-Processamento é de extrema importância, já que visa melhorar a imagem proveniente da etapa anterior, corrigindo potenciais defeitos ou realçando detalhes que possam vir a ser importantes na sua análise. São várias as soluções para a correção destes

defeitos destacando-se as melhorias no brilho e no contraste, redução de ruídos, correção de iluminação indesejada, realce de bordas, entre outros [19].

Nem todas as imagens necessitam do mesmo tipo de correção, pelo que, o procedimento adequado para determinada imagem, pode não ser o mais correto a se utilizar em outra.

Os procedimentos da etapa de Pré-Processamento podem realizar-se em dois domínios [16], [17] e [19]:

- Domínio da frequência: as técnicas neste domínio baseiam-se, essencialmente, em filtros que exercem a sua função sobre o espectro da imagem. Neste contexto, os procedimentos são realizados por meio de modificações na imagem original através da Transformada de *Fourier*. Posto isto, obtém-se um mapa de altas frequências e, após aplicação dos procedimentos, diretamente sobre as mesmas, modifica-se novamente a imagem original recorrendo à Transformada Inversa de *Fourier*. Neste tipo de domínio, para uma imagem com 256 tonalidades de cinza, cada *pixel* ocupa 8 *bytes*.

- Domínio do espaço real: as técnicas neste domínio, têm por base filtros que manipulam o plano de imagem. Os procedimentos são, diferentemente do domínio anterior, realizados por meio de operações matemáticas que agem diretamente sobre os *pixels* das imagens, e que se dividem em [20]:

    - **Operações Pontuais:** quando a intensidade de cada *pixel* na imagem de saída depende unicamente da intensidade do seu correspondente na imagem de entrada e da função que modifica as intensidades de entrada (Figura 14).

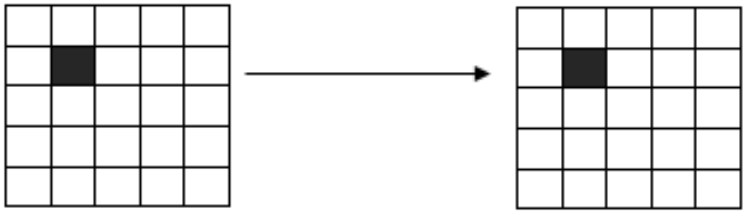

Figura 14: Operação Pontual

Estas operações definem-se, normalmente, de acordo com a equação (2):

$$g(x,y) = M[f(x,y)] . \quad (2)$$

onde, *f(x,y)* é a imagem de entrada, *g(x,y)* a imagem de saída, *(x,y)* as coordenadas dos *pixels* nas imagens e *M* a função transformação, que designa a operação e que mapeia um tom de cinza *s* na imagem de saída, para cada tom cinza *r* da imagem de entrada e, por esta razão, é também chamada função de mapeamento [15]. Deste modo, as operações pontuais podem também ser expressas pela equação (3):

$$s = M(r) . \quad (3)$$

A função M pode ser linear, quando modifica o brilho e o contraste de uma imagem sem alterar a forma do histograma. Neste caso, as operações podem ser do tipo expansão ou redução de contraste e brilho; operações de negativos com alteração de tons de cinza claro da imagem de entrada em tons de cinza escuros na imagem de saída, ou vice-versa, finalizando a operação com uma imagem de saída que é um negativo da imagem de entrada; operações de aplicação de cor com alteração de cada tom de cinza na imagem de entrada para uma determinada cor na imagem de saída, obtendo assim uma imagem colorida como imagem de saída e, por fim, operações de normalização com alteração da faixa de tons de cinza da imagem de entrada para uma faixa desejada na imagem de saída. Por outro lado, a função M pode também ser uma função não linear, quando modifica uma dada região do histograma ou o altera para uma forma pré-definida. Aqui, as operações podem ser do tipo logarítmicas, com aumento de contraste das regiões escuras, redução de contraste nas regiões claras e consequente aumento do brilho de uma imagem de saída, quando comparada a uma imagem de entrada; do tipo exponencial, com efeito contrário à logarítmica, isto é, aumento de contraste nas regiões claras, redução de contraste nas regiões escuras e consequente diminuição do

brilho da imagem [18]; ou aplicadas diretamente no histograma, numa operação conhecido como Equalização do Histograma ou Linearização do Histograma, responsável por gerar um histograma distribuído de forma mais uniforme, de modo que o número de pels, em qualquer escala de cinza, seja o mais semelhante possível (Figura 15). Esta operação baseia-se na utilização da função de distribuição acumulada do histograma, calculando, para cada nível de cinza na imagem de entrada, um determinado tom de cinza na imagem de saída [15], tal como mostra a equação 4:

$$s = M(r) = \sum_{i=0}^{r} n_i / n = \sum_{i=0}^{r} p(i) \quad . \qquad (4)$$

onde, *r* e *s* representam os tons de cinza na imagem de entrada e de saída, respetivamente; n_i o número de *pixels* com o tom de cinza *i* na imagem de entrada; *n* o número total de *pixels* da imagem de entrada; e *p(i)* a função distribuição de probabilidades de tons cinza *i* que define o histograma da imagem de entrada.

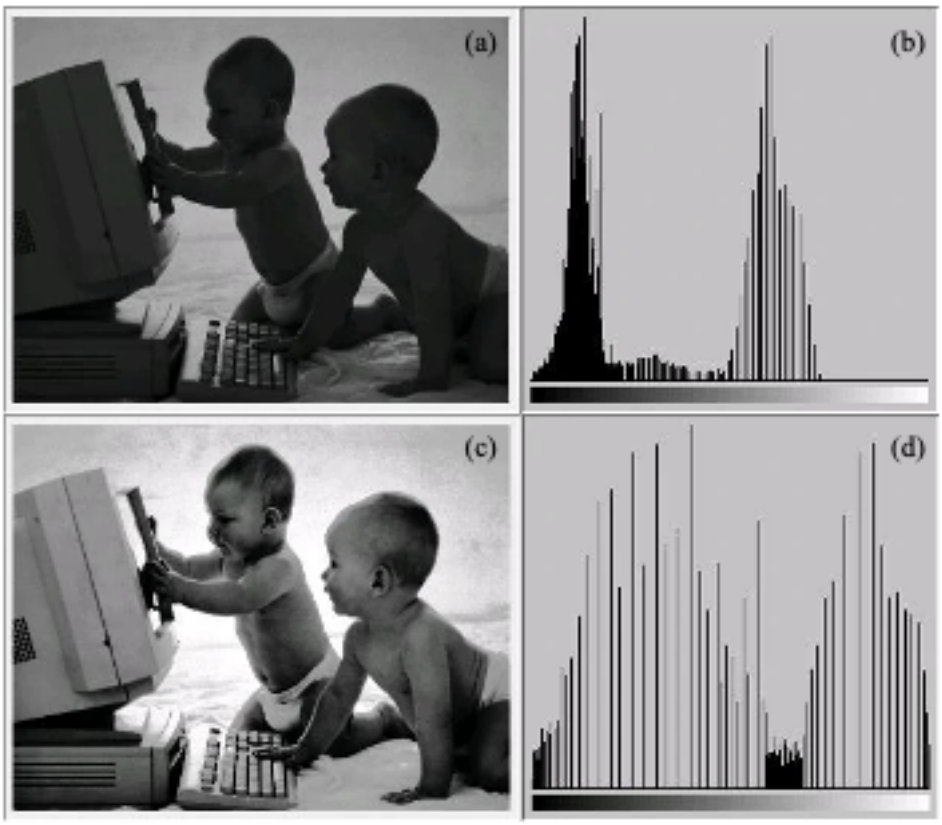

Figura 15: (a) Imagem original; (b) Histograma da imagem original; (c) Imagem com equalização de histograma e (d) Histograma Equalizado

- **Operações Locais:** quando o resultado de um *pixel* na imagem de saída depende dos valores dos *pixels* vizinhos na imagem de entrada (Figura 16). Por esta razão, são também chamadas de operações de vizinhança, usualmente representada por uma matriz e que engloba os *pixels* que cercam o *pixel* de interesse.

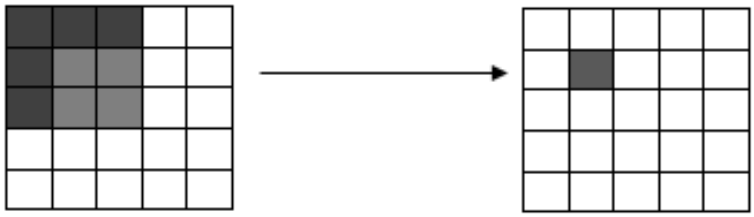

Figura 16: Operação Local

Para que a operação ocorra, realiza-se uma operação designada de *Convolução*, na qual, com incentivo de um filtro espacial, um delimitador da vizinhança percorre a imagem de entrada, analisando cada *pixel*

individualmente, e atribuindo o valor médio a cada *pixel* com a posição correspondente à do *pixel* central na vizinhança analisada. Filtros como este podem ser aplicados neste tipo de operações, em procedimentos como:

- **Redução de ruído,** com aplicação de filtros Passa-Baixa (beneficiam a passagem de componentes de baixa frequência espacial de forma a amenizar os componentes de alta frequência espacial, como o caso de ruídos), que se subdividem em filtros de Média (aplicam um valor médio dos valores de cinza dos *pixels* da vizinhança a um *pixel* da imagem de saída) e filtros Gaussianos (funcionam de igual forma como a média mas destaca-se por apresentar um peso maior no centro e menor nas extremidades); ou filtros Mediana (analisam estaticamente a vizinhança de forma a identificar a intensidade mediana da mesma, sendo os filtros que mais reduzem o ruído de uma imagem) [15];

- **Correção de iluminação irregular,** onde podem ser aplicados filtros Passa-Baixa de Média, de forma a obter-se uma imagem totalmente borrada do fundo da imagem de entrada;

- **Deteção de bordas,** onde são aplicados filtros Passa-Alto (efeito contrário aos filtros Passa-Baixo), que se subdividem em filtros Básicos (formados por pesos centrais positivos e pesos na extremidade negativos, que se anulam e retornam valores de zero quando atravessam regiões de baixa frequência espacial, gerando uma imagem de saída com bordas grossas sobre um fundo escuro), em filtros Direcionais (que também realçam bordas mas de acordo com zonas de interesse), em filtros *Laplacianos* (recorrem a simulações de derivadas) e em Diferenciais Parciais (estimulam o módulo de gradiente, que

pode ser calculado através da soma dos módulos das derivadas parciais em *x* e *y* numa imagem *f(x,y));*

- **Realce de bordas e detalhes,** onde podem ser aplicados filtros Passa-Alta, com destaque para filtros de Alto Reforço (beneficiam a passagem de componentes de alta frequência espacial, possibilitando a passagem parcial de componentes de baixa frequência espacial).

- **Operações Globais:** quando cada ponto da imagem de saída depende dos valores de todos os restantes pontos de imagem. São exemplos deste tipo de operações, as transformadas bidimensionais na compressão de uma imagem por transformadas, determinação do histograma de uma imagem e transformações geométricas de imagem, quando definidas por meio de fixação de uma certo número de pares de *pixels* correspondentes [20].

3.3.3. SEGMENTAÇÃO

O termo segmentar uma imagem significa separar a imagem em regiões que se diferenciam entre si, salientando pontos ou áreas de interesse para o processamento e análise posterior. Estas áreas passam a ser reconhecidas como objetos. Esta etapa é considerada a mais importante em todo o fluxo do processamento de imagem, uma vez que qualquer erro ocorrido nesta fase, irá refletir-se nas etapas seguintes, não produzindo os resultados desejados no final do processo.

A segmentação é um processo que visa adaptar-se às características e necessidades particulares de cada tipo de imagem e metas a alcançar na mesma. Este método segue duas abordagens: a similaridade e a descontinuidade entre *pixels*. A técnica mais utilizada na abordagem da similaridade é a binarização ou *thresholding*, onde ocorre uma diferenciação dos objetos de interesse do fundo da imagem, que resulta numa imagem binária, onde o fundo e os objetos estão representados por *pixels* de cor preta e de cor

branca, respetivamente, ou vice-versa (Figura 17) [19]. Por outro lado, uma abordagem de descontinuidade de imagens visa detetar *pixels* que delimitam contornos ou bordas dos objetos, separando a imagem de acordo com as alterações bruscas dos níveis de cinza. A técnica de segmentação mais utilizada baseada nesta abordagem é, por isso, a deteção de bordas (Figura 18).

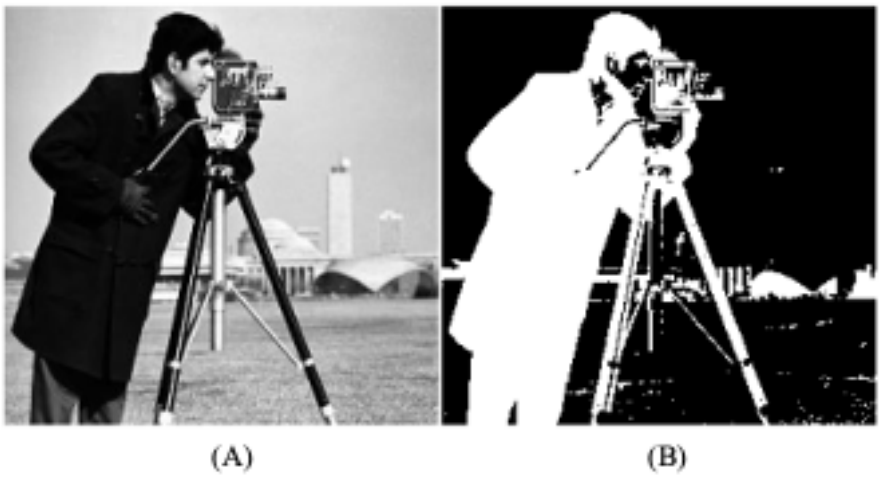

Figura 17: (A) Imagem original e (B) Imagem binária após processo de segmentação

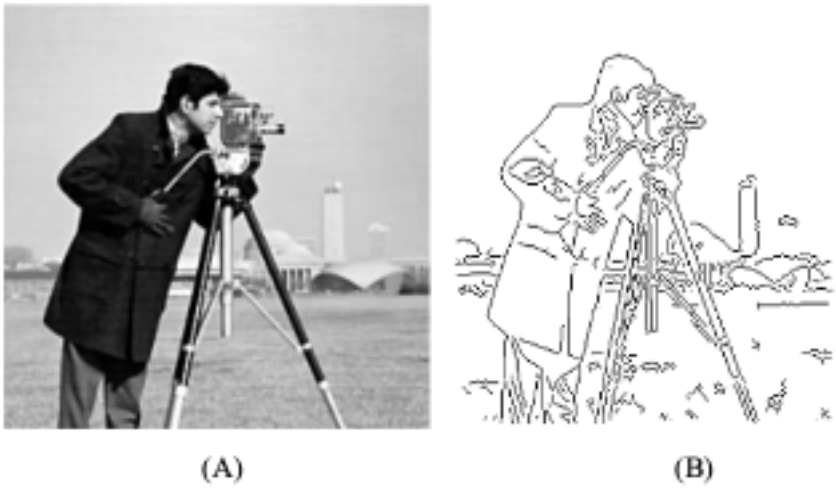

Figura 18: (A) Imagem original e (B) Imagem segmentada por deteção de bordas

Dado que a segmentação é uma das tarefas mais importantes do processamento de imagem é necessário torna-la o mais automática possível. Para tal, a escolha do limiar passa pela análise de histogramas de imagens. Como decisão para o tom de corte entre as fases, utiliza-se a técnica dos mínimos do histograma, onde os limiares correspondem às tonalidades intermediárias entre dois picos do histograma [21].

Estas etapas podem aplicar-se tanto em imagens monocromáticas como em imagens de cor, para as quais se recorre a três histogramas correspondentes a cada componente R, G e B de uma imagem de cor. RGB é a abreviatura de um sistema de cores aditivas, em que o Vermelho (*Red*), o Verde (*Green*) e o Azul (*Blue*) são combinados de várias formas de modo a reproduzir um largo espectro cromático. Neste caso, a escolha dos limiares é realizada nos três histogramas, de forma que a faixa de cores selecionada pertença à região segmentada [21].

3.3.4. PÓS-PROCESSAMENTO

A etapa de Pós-Processamento tem como principal objetivo corrigir defeitos residuais nas imagens provenientes da etapa de Segmentação. Recorre à separação de objetos, à posterior eliminação de objetos dos quais não se deseja extrair qualquer informação e, por fim, ao agrupamento de objetos de interesse. Para tal, realizam-se operações [19] e [21]:

i. **Lógicas** – tomadas como operações pontuais que possibilitam, por meio de operadores lógicos, a formação de uma imagem de saída contendo *pixels* preservados ou invertidos, a partir de uma imagem de entrada. As operações lógicas mais comuns, a partir das quais são geradas outras funções lógicas são [15] e [19], tal como mostra a Figura 19:

    a. União (OR) – une das duas imagens de entrada, uma imagem de saída onde apresentam a cor branca os *pixels* que são brancos em pelo menos uma das imagens;

    b. Interseção (AND) – intersecta duas imagens de entrada, gerando uma imagem de saída onde apresentam cor branca os *pixels* que são brancos em ambas as imagens;

    c. Complemento (NOT) – inverte todos os *pixels* da imagem de entrada, originando o seu negativo como imagem de saída;

d. Ou Exclusivo (XOR) – gera uma imagem de saída onde apresentam a cor branca os *pixels* que são brancos em apenas uma das imagens de entrada.

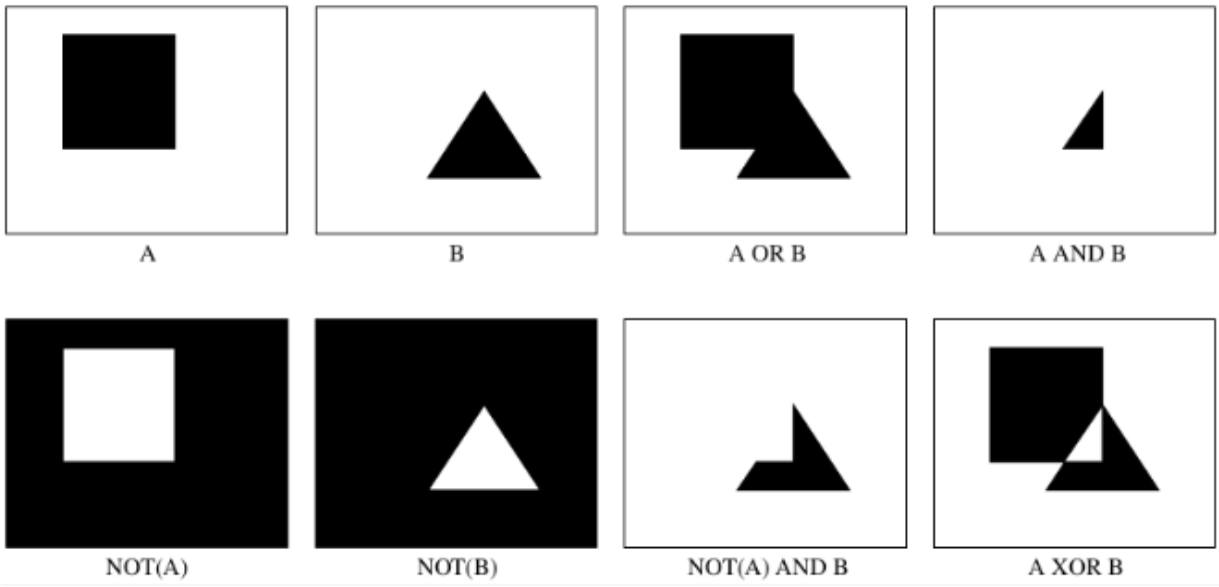

Figura 19: Imagens de entrada A e B, e respetivas imagens de saída, de acordo com operações lógicas possíveis

ii. **Morfológicas** – tomadas como operações locais, onde um *pixel* da imagem de saída é formado, a partir, do valor de *pixels* de uma vizinhança da imagem de entrada. A vizinhança é definida por uma imagem binária, comummente chamada de elemento estruturante, que varre a imagem de entrada, preservando ou invertendo o *pixel* central da vizinhança de acordo com a função aplicada. Estas funções podem ser:

   a. **Erosão –** tem como objetivo fazer o objeto encolher. Cada *pixel* branco, na imagem de entrada, que não tenha na vizinhança onde se localiza o elemento estruturante mais nenhum *pixel* dessa cor, é invertido na imagem de saída correspondente. Obtém-se uma imagem com objetos brancos diminuídos ou eliminados, fundo preto e com aumento de buracos nos objetos (Fig. 20 - B);

   b. **Dilatação –** tem por objetivo fazer o objeto aumentar em área realizando o processo de forma contraria à erosão, ou seja, cada *pixel* preto numa imagem de entrada é invertido na imagem de saída correspondente, caso

haja pelo menos um *pixel* branco na vizinhança onde se localiza o elemento estruturante. Obtém-se uma imagem com aumento de objetos brancos e diminuição da visualização do fundo e de buracos nos objetos (Fig. 20 - C);

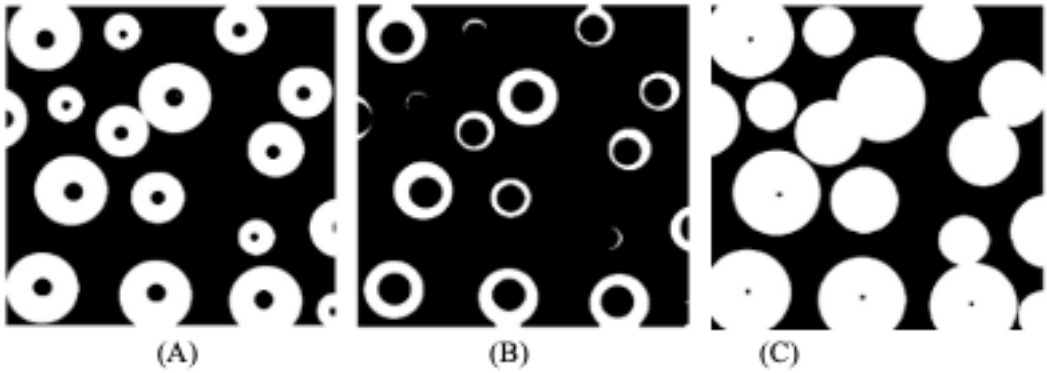

Figura 20: Operações Morfológicas: (A) Imagem original, (B) Resultado de Erosão e (C) Resultado de Dilatação

Estas operações podem combina-se e originar outras operações, tais como:

c. **De abertura (OPEN)** – resultado de uma erosão seguida de uma dilatação (Fig.21 – B);

d. **De Fechamento (CLOSE)** – resultado de uma dilatação seguida de uma erosão (Fig. 21 – C).

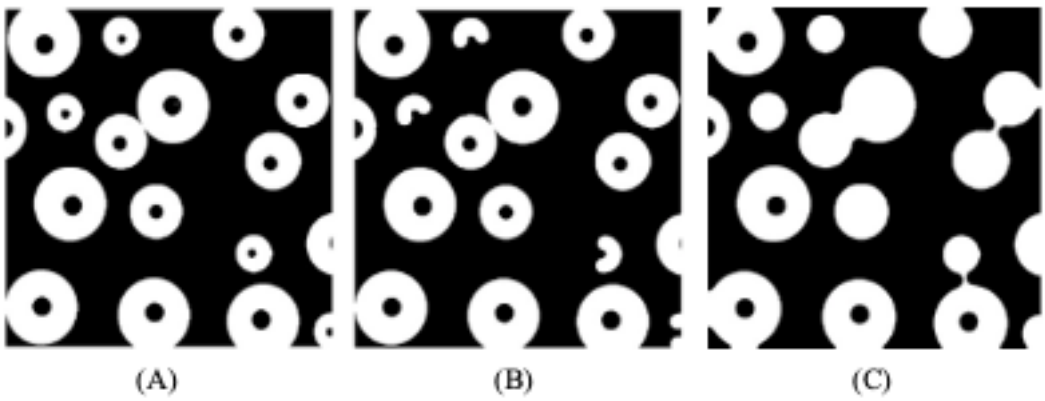

Figura 21: Operações Morfológicas: (A) Imagem original, (B) Resultado da Abertura e (C) Resultado do Fechamento

Também é possível gerar operações a partir da junção de operações lógicas e operações morfológicas, nomeadamente:

- o preenchimento de buracos *(hole filling);*
- a eliminação de objetos que tocam na borda da imagem *(border object killing);*
- eliminação de objetos por área *(scrapping)*, onde são eliminados objetos que tenham um número de *pixels* dentro de um dado intervalo de valores.

3.3.5. PÓS-PROCESSAMENTO

Extrair características ou atributos com maior importância numa imagem permite realçar diferenças e semelhanças que possam existir entre os objetos. A etapa de Extração de Atributos é responsável, por meio da descrição de atributos com determinadas características, criar dados qualitativos a partir de medidas na imagem proveniente de etapas anteriores, isto é, caracterizar os atributos para que possam ser utilizados na etapa seguinte.

A Extração de Atributos divide-se, sobretudo, em duas tipologias de medida [17], [19] e [21]:

I. **Medidas de Campo** – relacionadas com o campo como um todo, como por exemplo, área total dos objetos, número de objetos, fração de área (percentagem de pixéis brancos numa imagem binária), entre outros. A contabilização do número de objetos obtém-se pela contabilização de *pixels* contíguos, onde cada um corresponde a um objeto. Esta contiguidade, também denominada de conetividade, pode resultar em *pixels* pertencentes a dois objetos separados (Conetividade 4) ou *pixels* pertencentes ao mesmo objeto (Conetividade 8);

II. **Medidas de Região** – relacionadas com os objetos de forma individual, sendo extraído uma característica de cada objeto e que pode estar ligada à área, tamanho, forma, entre outros. A área que um objeto ocupa numa imagem é a

quantidade de *pixels* presentes no mesmo. Esta área pode dizer-se convexa, quando a mesma é equivalente a um elástico em torno do objeto e que, por isso, pode albergar tanto o objeto como alguma área vizinha; ou preenchida, quando são incluídos buracos internos. Por sua vez, o perímetro é considerado o tamanho da fronteira dos objetos, ou seja, o contorno do objeto. Também o perímetro pode ser convexo ou preenchido. A forma está relacionada com parâmetros sem grandeza dimensional, que variam entre 0 (formas irregulares) e 1 (formas padrão), obtidos pela combinação de parâmetros de tamanho.

3.3.6. CLASSIFICAÇÃO E RECONHECIMENTO

A etapa de Classificação e Reconhecimento visa analisar os dados quantitativos, provenientes da fase de Extração de Atributos, e agrupa-los em classes [15]. O reconhecimento de atributos pode ser usado na classificação somente dos objetos ou de toda a região de uma imagem. Partindo dos atributos dos objetos, com determinadas características, é construído um espaço de atributos, onde cada objeto é representado por um vetor. O posicionamento desse vetor vai ditar a identificação de pontos com características semelhantes [19]. Assim, a escolha de um grande número de características, leva a um espaço de elevada dimensão e consequentemente, uma difícil etapa de aprendizado; por sua vez, um espaço de pequena dimensão pode levar a uma baixa caracterização e erros no processo de reconhecimento.

O reconhecimento pode ser realizado por dois tipos de métodos (Fig. 22):

A. **Supervisionado,** onde um classificador ou analista recebe informações para correta classificação e identificação das classes, em função da proximidade destas aos grupos de características já conhecidas;

B. **Não-Supervisionado**, onde não há qualquer informação referente às classes e a sua definição é realizada de acordo com um grupo de pontos próximos entre si, designados *Clusters* [19].

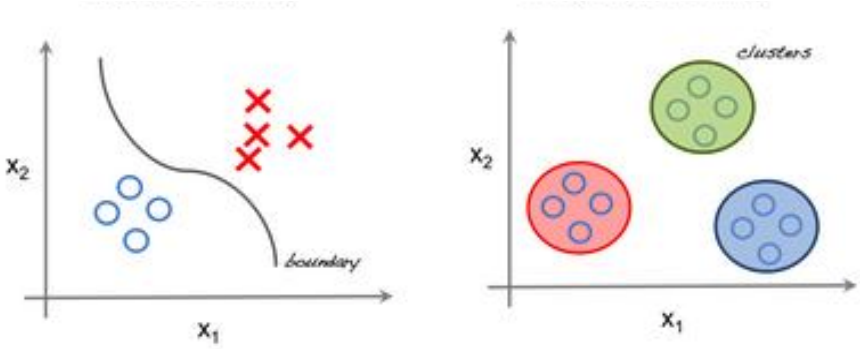

Figura 22: Métodos de Reconhecimento, Supervisionado e Não-Supervisionado, respetivamente

Existem outros métodos de reconhecimento de formas, tais como, métodos baseados em propriedades estatísticas de objetos (classificador de *Bayes*), métodos baseados em técnicas de distância entre objetos e as respetivas formas padrão e métodos baseados na descrição da forma através de uma linguagem básica.

4. ALGORITMOS DE RECONHECIMENTO FACIAL

Este capitulo aborda os principais tópicos do Reconhecimento Facial, nomeadamente os algoritmos utilizados para tal. São também abordados conceitos de interesse em Biometreia.

4.1. BIOMETRIA

A palavra Biometria deriva das palavras “bio”, que significa vida, e “metria”, que significa medida. Deste modo, pode afirmar-se que a Biometria está relacionada com técnicas que

possibilitam a identificação de pessoas através da medição de características fisiológicas e comportamentais, com auxílio de um sistema automático, baseado em inteligência artificial e reconhecimento de padrões. Para tal, é necessária a existência de propriedades como a unicidade, uma vez que cada pessoa apresenta características e padrões que se distinguem dos demais; coletabilidade, permitindo a possibilidade de coleta e medida; e permanência, já que deve permanecer invariável durante um longo período de tempo [22].

Os sistemas biométricos podem, então, dividir-se em (Figura 23) [22], [23] e [24]:

I. **Sistemas biométricos fisiológicos**, que analisam características de:

   a. **Face –** é a característica mais utilizada para identificação de pessoas que tem por base a localização e análise dos olhos, nariz e boca, ou até mesmo uma análise global de todas as características.

   b. **Mãos** – analisa a forma da mão, o comprimento e largura da palma e dos dedos. Apesar de ser uma técnica simples e de baixo custo, trata-se de uma característica inalterável do individuo, já que a geometria da mão pode ser alterada devido ao crescimento, uso de joias e doenças.

   c. **Olhos** – pode analisar duas característica dos olhos, a iris e a retina. Na análise da iris é tomada em conta a sua textura, uma vez que esta estabiliza-se nos primeiros dois anos de vida de um indivíduo e se diferencia dos demais, inclusivamente de gémeos idênticos. Por outro lado, é de fácil identificação quando artificial (uso de lentes de contacto) e difícil manipulação cirúrgica. Também a retina permite identificar pessoas através do padrão criado pelos vasos sanguíneos presentes, que é único de cada individuo.

   d. **Impressões digitais** – é a mais antiga e usada técnica, devido à sua taxa de exatidão, uma vez que cada indivíduo apresenta uma impressão digital única e de difícil alteração ao longo dos tempos (pode alterar-se devido a fatores genéticos, ambientais ou profissionais).

II. **Sistemas biométricos comportamentais**, que analisam:

   a. **Voz** – recorrendo à forma e tamanho do aparelho fonador, composto pela boca, fossas nasais e lábios, mas também ao comportamento do discurso que varia com a idade, estado emocional, condições de saúde ou ambientais;

   b. **Caligrafia** – analisando semelhanças na caligrafia, legibilidade na hora de escrever, direção e velocidade da escrita, espaçamento entre os caracteres e personalização da assinatura.

No entanto, apesar de não tanto comuns, existem outras formas para identificação, tais como [23] e [24]:

i. **Termogramas –** analisa o calor irradiado pelo corpo humano, com auxílio a raios infravermelhos que permitem a criação de um mapa de valores com forma da pessoa. Apesar de ser vantajoso pelo facto de não ter contacto com o individuo que se quer identificar, pode ser alterado devido à presença de aparelhos de calefação, estado emocional e de saúde.

ii. **Forma de caminhar –** utilização de sequências do movimento de caminhar de uma pessoa para medição dos movimentos de cada articulação. No entanto, estes movimentos podem variar com a idade, presença de lesões, alterações de peso, alterações do estado de sobriedade, posse de determinada roupa, entre outros.

iii. **Forma de digitar no teclado do computador –** reconhece um padrão na forma de digitalizar de cada pessoa, medindo características de rapidez, habilidade, efetividade e dificuldade. No entanto, perde pelo facto de haver ainda um número considerável de pessoas que não sabe utilizar um computador.

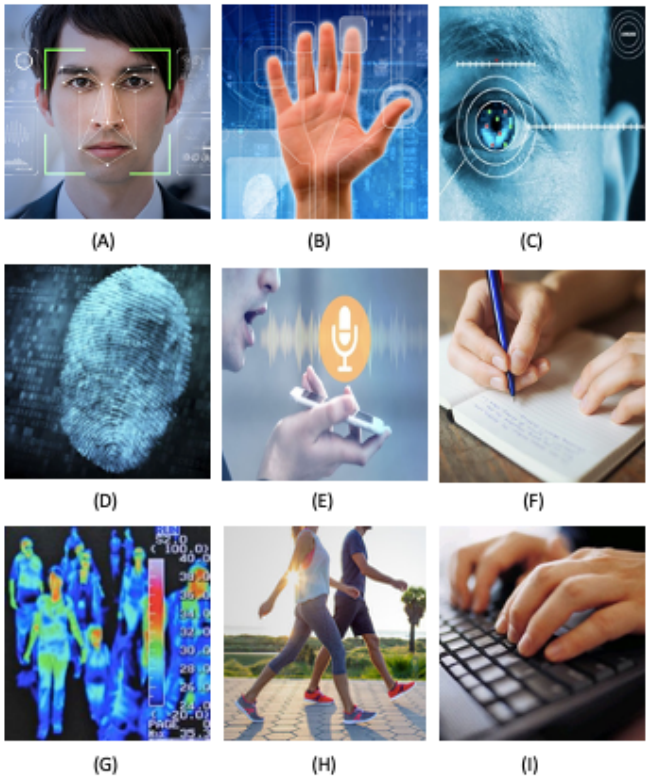

Figura 23: Exemplos de sistemas biométricos: (A) Face, (B) Mãos, (C) Olhos, (D) Impressão Digital, (E) Voz, (F) Caligrafia, (G) Termogramas, (H) Forma de Caminhar e (I) Forma de Digitar no Computador

Para identificação por qual sistema biométrico se deve optar, devem ser tomadas em consideração propriedades como a performance, isto é, a facilidade e rapidez de processamento; a precisão obtida com a aplicação do sistema; a aceitabilidade por parte das pessoas que vão estar sujeitas a análise e a dificuldade de imitação da impostura já que, quanto maior o grau de impostura, maior a dificuldade para imitar determinada ação.

A Tabela 1 compara os diferentes sistemas biométricos de acordo com as propriedades referidas acima:

Tabela 1: Comparação entre sistemas biométricos

	Unicidade	Coletabilidade	Permanência	Performance	Aceitabilidade	Imitação Impostura
Face	Alta	Alta	Média	Alta	Alta	Baixa
Mãos	Média	Baixa	Média	Média	Média	Alta
Olhos	Alta	Baixa	Alta	Alta	Baixa	Baixa
Impressão digital	Alta	Alta	Alta	Alta	Média	Baixa
Voz	Baixa	Alta	Baixa	Baixa	Alta	Média
Caligrafia	Baixa	Alta	Baixa	Baixa	Alta	Alta
Termogramas	Alta	Média	Baixa	Média	Média	Alta
Caminhar	Baixa	Alta	Baixa	Baixa	Alta	Alta
Digitar	Baixa	Alta	Baixa	Baixa	Média	Alta

Tendo em conta a Tabela 1, pode afirmar-se que não existe um sistema biométrico que satisfaça todas as propriedades mencionadas, e por isso, a escolha de um sistema biométrico depende para que efeitos se vai aplicar. No entanto, este trabalho vai ter maior enfase em sistemas biométricos da face.

4.2. RECONHECIMENTO FACIAL

O reconhecimento através da face tornou-se num dos principais sistemas biométricos utlizados, já que, diariamente, se recorre a este tipo de reconhecimento para identificação de pessoas com quem se cruza. Assim, pode afirmar-se que o reconhecimento facial é a capacidade de reconhecer pessoas através das suas características faciais, sob as mais diversas condições de observação, quer em imagens distorcidas, com pouca luminosidade ou até mesmo com a face parcialmente visível. O

reconhecimento facial é, sobretudo, usado para identificação e verificação de pessoas, detetando facilmente informações como idade, sexo, raça, humor e emoção [25] e [26].

Um sistema de reconhecimento facial é constituído pelas etapas presentes na Figura 24, e que são as seguintes [22] e [26]:

- **Deteção da face** – a partir de uma imagem de entrada, é possível verificar a presença de uma face e, caso exista, determinar a localização da mesma. A deteção da face pode sofrer influência da utilização de adereços, maquilhagem, barba/bigode e mal formações, pela orientação da face e pela presença de ruídos e oclusões. As técnicas utilizadas nesta epata podem basear-se em características, quando se pretende localizar características como os olhos, boca e nariz; em modelos, que possibilitam agir com variações de pose e expressão; e por fim, em aparência, havendo origem de regiões retangulares na imagem com o objetivo de procurar prováveis candidatos a faces [28].

- **Extração das características –** uma face, numa imagem, encontra-se rodeada por um ambiente, comummente chamado de fundo, sendo necessário realizar o isolamento da mesma. Para tal, recorta-se a imagem nos limites onde a face foi localizada e extrai-se as características com maior interesse.

- **Reconhecimento da face** – a parir de uma imagem de entrada de uma face, efetua-se a sua comparação com imagens de faces já conhecidas, normalmente armazenadas em bases de dados anteriormente criadas, e verificam-se as semelhanças encontradas. Este reconhecimento pode ser utilizado para operar em duas situações distintas:

    - Para verificação, onde existe a necessidade de determinar se o individuo é quem diz ser, utilizando para tal uma imagem de uma pessoa desconhecida, assim como uma reivindicação de identidade;

    - Para identificação, onde a partir de uma imagem da face de um desconhecido, recorre-se a uma base de dados com imagens de rostos conhecidos, a fim de determinar a identidade da pessoa.

É possível aceitar o indivíduo como "verdadeiro" ou rejeitar o mesmo como "falso" (chamado impostor), com auxílio de um "limiar de casamento" (*matching threshold*), que define o grau de confiança a ser utilizado na distinção entre escolhas "verdadeiras" e escolhas "falsas". Assim, podem ocorrer quatro opções [22]:

1. Verdadeiro positivo (*True Positive* – TP) – o padrão é verdadeiro, é classificado como tal e ocorre confirmação da sua identidade ou acesso a determinado recurso;

2. Verdadeiro negativo (*True Negative* – TN) – o padrão é falso, é classificado como tal, e a sua identidade ou acesso a determinado recurso são negados;

3. Falso Positivo (*False Positive* – FP) – o padrão é falso, é classificado como verdadeiro e ocorre confirmação da identidade ou acesso a determinado recurso, neste caso por alguém impostor;

4. Falso Negativo (*False Negative* – FN) – o padrão é verdadeiro, é classificado como falso e a sua identidade ou acesso a determinado recurso são negados, neste caso seria alguém "verdadeiro" com identidade recusada.

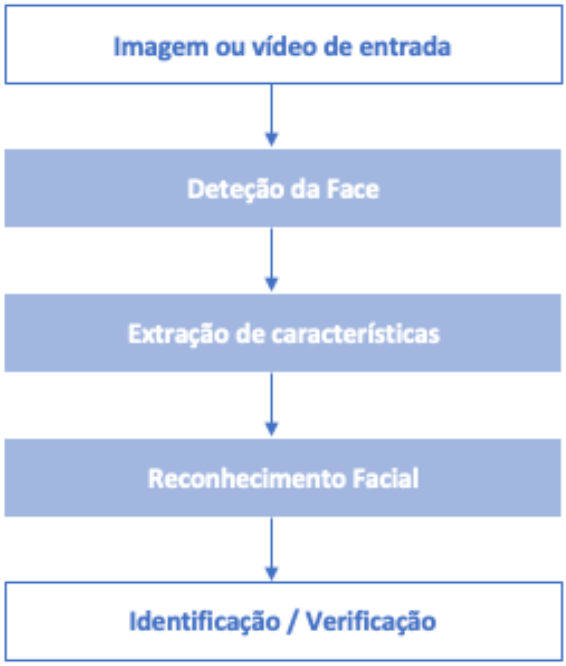

Figura 24: Esquema de um sistema de Reconhecimento Facial

De acordo com o método de aquisição da imagem, as técnicas de reconhecimento de rosto podem também dividir-se em três categorias [27]:

I. **Técnicas baseadas em imagens de intensidade (estáticas**), podendo ser técnicas holísticas de correspondência, onde, com auxilio de transformadores, é utilizado todo o rosto para reconhecimento; ou técnicas de correspondência baseados em recursos, onde são apenas consideradas algumas áreas da imagem (informações faciais como olhos, boca e nariz), com o intuito de reconhecer padrões e combinar rostos;

II. **Técnicas baseadas em sequências de vídeo**, onde a imagem é extraída a partir de uma sequência de imagens, o que se torna uma vantagem pela possibilidade de rastreio do rosto durante o tempo necessário;

III. **Técnicas baseadas em dados sensoriais**, onde as imagens consideradas são do tipo 3D e IV, o que proporciona a obtenção de informações acerca da geometria da face e uma maior precisão das características faciais.

4.3. APLICAÇÕES

São várias as áreas de aplicação deste tipo de reconhecimento, destacando-se [24], [26], [27] e [29]:

- Segurança, nos aeroportos, caixas automáticas, fronteiras e até mesmo no inicio de sessão de computadores e telemóveis, aplicações, Internet, registos médicos;
- Vigilância e cumprimento da lei, no controlo CCTV, controlo de portais, acompanhamento de suspeitos, identificação e rastreamento de criminosos;
    - Emissão de documentos de identidade;
    - Verificações de borda para comparação do retrato de um passaporte;
    - Exames policiais;

    - Procura em bancos de dados;
    - Drones com câmaras aéreas para reconhecimento facial durante eventos em massa;
- Sistemas de justiça criminal, nomeadamente no apoio a investigações e áreas forenses;
- Investigações de banco de dados e envelhecimento computadorizado:
    - Identificar crianças desaparecidas;
    - Identificar adultos desorientados;
    - Identificar crianças exploradas;
- Verificação geral de identidade, por meio de cartas de condução, passaporte, registo eleitoral, bilhete de identidade;
- Autenticação de usuários, acesso a instalações e acesso a veículos;
- Validação de transações bancárias e com cartão de crédito;
- Desenvolvimento de interfaces homem-máquina, onde os sistemas tentam reconhecer e interpretar a face do usuário de um terminal de computador;
- Monitorização de multidões em estações, centros comerciais, estádios;
    - Possibilidade de ver comportamento de clientes e usuários;
- Na saúde, no acompanhamento do uso de medicamentos com maior precisão, na deteção de doenças genéticas e no apoio a procedimentos de controlo de dor.

4.4. ALGORITMOS

As técnicas de reconhecimento facial eram, inicialmente, utilizadas apenas para reconhecimento de uma única face num ambiente que podia ser simples. No entanto, e devido ao facto de cada vez mais haver uma necessidade de avanço nesta área, desenvolveram-se novas técnicas e métodos, que possibilitaram o reconhecimento facial em ambientes com características mais adversas. Seguidamente serão abordados os algoritmos mais utilizados mas, no entanto, será dado maior enfâse aos algoritmos com os quais se irá trabalhar no desenvolvimento da sistema, neste caso os algoritmos *Viola-Jones*, PCA e *Eigenfaces*.

4.4.1. VIOLA-JONES

O algoritmo de *Viola-Jones* permite a deteção de objetos ou faces em imagens, de forma rápida e com reduzida taxa de falsos positivos. Destaca-se das outras abordagens pelo facto de possuir quatro estágios, nos quais a imagem de entrada é representada na forma de imagem integral, possibilitando que as características sejam processadas numa só passagem pela imagem; permite a construção de um classificador capaz de selecionar um pequeno número de características dentro de um conjunto possível elevado e também permite descartar imagens de fundo rápida e eficazmente, combinando classificadores em cascata [22].

O primeiro estágio inicia-se com o uso de características denominadas de *Haar,* para identificação de padrões, que representam as semelhanças que os rostos humanos partilham, nomeadamente a localização dos olhos, nariz e boca. Estas características resultam numa diferença de intensidades luminosas entre áreas da imagem, podendo agrupar-se em três classes [28] e [30]:

- Características de dois retângulos – diferença entre a soma da intensidade dos *pixels* dentro de duas regiões retangulares e adjacentes do mesmo tamanho;

- Características de três retângulos – soma dos valores de *pixels* de um retângulo central menos a soma dos valores de *pixels* de dois retângulos externos;

- Característica de quatro retângulos – diferença entre valores dos pares diagonais de retângulos.

A utilização de características *Haar* tomam um papel importante na deteção facial, já que, de forma eficaz e rápida, consegue perceber-se qual a região que se está a tratar: a região dos olhos é, frequentemente, mais escura que a parte superior das bochechas e a região da ponto da nariz é mais brilhante que a parte dos olhos (Figura 25).

Figura 25: Exemplo de características *Haar* em reconhecimento facial

Segue-se a criação de uma Imagem Integral, com o cálculo das características de retângulos utilizando uma representação intermediária da imagem. Numa Imagem Integral, cada *pixel* é igual à soma dos valores dos *pixels* acima e à esquerda do mesmo, incluindo o próprio *pixel*, tal como se representa na Figura 26. Assim, facilmente se calcula o valor da soma numa determinada região retangular da imagem [29].

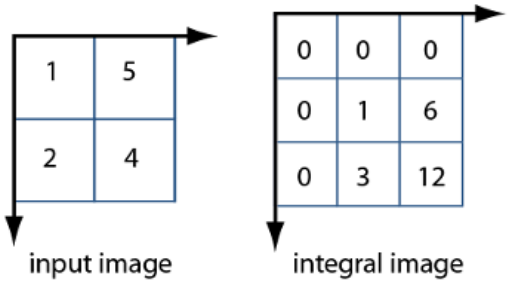

Figura 26: Exemplo de uma Imagem Integral

Para obter um processo mais eficaz e rápido é, normalmente, realizada uma combinação de características *Haar*, retirando as características com mais interesse. Para tal, é utilizado o método de *Adaboost*, que promove a construção de um classificador forte através da combinação de classificadores mais fracos e dependentes de uma única característica. O método *Adabooost* pode ser usado para escolha de características adequadas mas também para treinar os classificadores com as características escolhidas. A combinação dos classificadores mais fracos intitula-se de combinação em Cascata onde, em cada estágio do processo, aplica-se um classificador mais específico que o anterior, por forma a serem rejeitadas as regiões que não contêm as características pretendidas e evitando que sejam realizados passos desnecessários. Ou seja, utilizando uma imagem como entrada para o primeiro classificador e se ela for considerada verdadeira (neste caso, uma imagem facial), é submetida ao próximo classificador e assim sucessivamente até chegar ao último classificador, que a considera uma imagem facial; se a imagem for considerada como falsa em qualquer um dos classificadores, deixa de ser considerada no restante fluxo (Figura 27) [28].

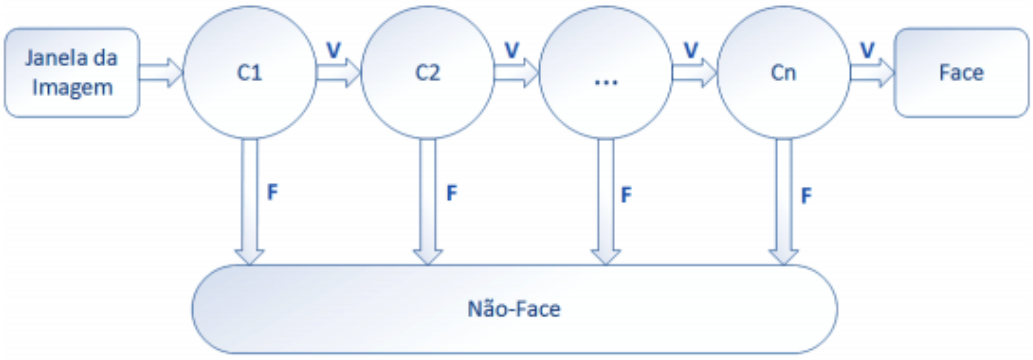

Figura 27: Esquema de classificadores em cascata

De salientar que, apesar do algoritmo apresentar uma boa precisão e um reduzido número de falsos positivos, nem sempre é capaz de detetar faces de perfil, em situações em que as os olhos estão cobertos ou faces inseridas num ambiente com uma iluminação desfavorável, ou até mesmo quando existe na imagem um objeto com padrões marcantes que possa "confundir" o algoritmo.

4.4.2. ANÁLISE DE COMPONENTES PRINCIPAIS (PCA)

A Análise de Componentes Principais promove a transformação de um conjunto de variáveis correlacionadas entre si, num conjunto de variáveis de menor dimensão que reúne a informação do conjunto original, sem perdas de informação, e que não se correlacionam entre si. Fundamentalmente, o objetivo deste método passa por explicar a estrutura da variância e covariância de um vetor aleatório, composto por *p*-variáveis, por meio de combinações lineares das variáveis originais [31]. Estas combinações lineares ou variáveis de reduzida dimensão são chamadas de componentes principais e são determinadas por ordem decrescente de importância, num determinado espaço dimensional. Ou seja, a primeira explica o máximo possível da variância dos dados originais, a seguinte explica o máximo possível da variância ainda não explicada e assim sucessivamente, até chegar à última componente que será a que fornece menor contribuição para a explicação da variância total dos dados originais [32].

Este algoritmos passa pelas seguintes etapas [33]:

1. Padronização, onde o objetivo passa por padronizar as variáveis iniciais contínuas, de forma que cada uma delas tenha um contributo igual para a análise. Este passo torna-se importante no sentido que, caso existam diferenças significativas entre os intervalos das variáveis iniciais, um dado intervalo pode ser dominante em relação a outro, levando a resultados tendenciosos. A padronização pode ser realizada subtraindo a média e dividindo pelo desvio padrão, para cada valor da variável, tal como mostra a equação (5):

$$z = (\text{valor} - \text{média}) / \text{desvio padrão}. \quad (5)$$

   Realizada a padronização, as variáveis devem ser transformadas na mesma escala.

2. Cálculo da Matriz de Covariância, que tem por objetivo analisar a relação entre as variáveis. Trata-se de uma matriz *NxN*, com *N* como número de dimensões e cujas entradas são as covariâncias associadas aos possíveis pares de variáveis iniciais. Para uma matriz A_{3x3}, a matriz de covariância tem os valores: a_{11} = Cov (a,a); a_{12} = Cov (a,b); a_{13} = Cov (a,c); a_{21} = Cov (b,a); a_{22} = Cov (b,b); a_{23} = Cov (b,c); a_{31} = Cov (c,a); a_{32} = Cov (c,b) e a_{33} = Cov (c,c). Dado que a covariância de uma variável

consigo mesma é a sua variância (Cov(a,a)=Var(a)), na diagonal principal encontram-se as variações de cada variável inicial. Uma vez que a variância é comutativa (Cov (a,b) = Cov (b,a)), as entradas da matriz de covariância são simétricas em relação à diagonal principal.

3. Cálculo dos vetores próprios e valores próprios da matriz de covariância para identificação dos componentes principais, onde cada autovetor está sempre acompanhado de um autovalor, sendo que o seu valor é sempre igual ao número de dimensões dos dados (ex. para um conjunto de dados tridimensionais, existem 3 variáveis, e consequentemente, 3 autovetores e 3 autovalores correspondentes. São os autovetores e autovalores que estão na base da transformação dos vários conjuntos de componentes principais, já que os autovetores da matriz de covariância correspondem às direções dos eixos onde existe mais variância e os autovalores são os coeficientes anexados aos autovetores, fornecendo a quantidade de variância em cada componente principal. Quando já se tem os componentes principais, calcula-se a percentagem de variância contabilizada por cada componente e divide-se o valor próprio de cada componente pela soma dos valores próprios totais.

4. Cálculo do vetor de recurso, onde se escolhe entre manter todos os componentes ou descartar aqueles que apresentam menos significância (autovalores mais baixos) e se forma uma matriz de vetores chamada vetor de característica, na qual as colunas são os autovetores dos componentes que se decidem manter.

5. Reformulação dos dados ao longo dos eixos dos componentes principais, com o objetivo de utilizar o vetor de características formado com base nos autovetores da matriz covariância, de forma a reorientar os dados dos eixos originais para os representados pelos componentes principais. Para tal, multiplica-se a transposição do conjunto de dados original pela transposição do vetor de características.

No que toca a reconhecimento facial, cada face numa imagem, pode então ser representada por um conjunto de pontos no espaço, que correspondem a amostras de probabilidade. Os vetores das imagens faciais criam autovetores que têm como função

representar as características que descrevem as variações entre as imagens. Estes vetores são denominados *Eigenfaces*, que será abordado seguidamente trabalho.

4.4.2.1 EIGENFACES

Conhecido como um dos métodos mais aplicados em reconhecimento facial, o método de *Eigenfaces* (autofaces) tem por base o método de Análise de Componentes Principais, para calcular o melhor sistema vetorial para compressão de imagens e, seguidamente, aplicar os *Eigenfaces* para tratar do problema de reconhecimento, extraindo características relevantes de uma imagem de face. O reconhecimento é obtido pela projeção do rosto no espaço formado pelos *eigenfaces*, não trabalhando com o rosto como um todo [28] e [34]. Matematicamente falando, o algoritmo visa encontrar as componentes principais da distribuição das faces ou os autovetores da matriz de covariância do conjunto de imagens, tratando a imagem como um vetor [28]. É realizada uma comparação com base na distância euclidiana dos autovetores das autofaces e da autoface da imagem utilizada: se a distância for pequena, a pessoa é identificada; caso contrário, a imagem é considerada como pertencente a um indivíduo para o qual se deve treinar o sistema [34].

O fluxograma do algoritmo pode ser visto na Figura 28.

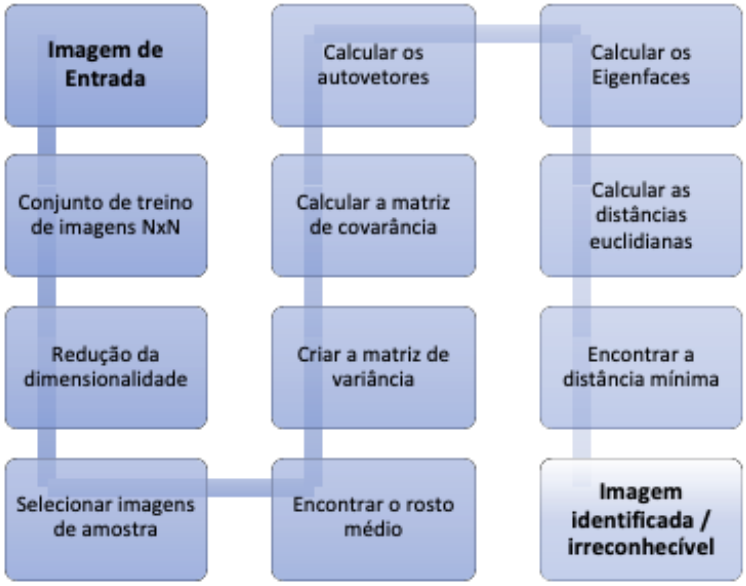

Figura 28: Fluxograma Algoritmo Eigenfaces

A utilização deste método prevê que uma imagem facial possa ser facilmente representada por um vetor com dimensão NxN. Assim, com auxílio a técnica PCA, pode diminui-se a sua dimensionalidade e encontrar os vetores que melhor representam a distribuição das imagens faciais dentro de todo o espaço da imagem. Calcula-se a imagem de face média com a média de todas as faces de amostra consideradas (Figura 29), representada por μ e que visa eliminar informação excessiva da face, como apresentado na equação (6):

$$\mu = (1 / M) \sum^{M}_{i=1} \Gamma_i \, . \qquad (6)$$

onde, M é o número de imagens faciais para treinamento e Γ o vetor linha que representa cada imagem facial.

Figura 29: Face média (direita) gerada com base em imagens de treinamento (esquerda)

Os *eigenfaces* correspondentes aos autovalores mais altos são retirados de forma a definirem o espaço facial. O autoespaço é criado projetando-se a imagem para o espaço facial formado pelos *eigenfaces* [34]. Cada vetor da imagem é transformado num novo vetor Φ, com o qual se faz a subtração do vetor de face média ao vetor de cada imagem facial, tal como mostra a equação (7):

$$\Phi = \Gamma_i - \mu \, . \tag{7}$$

Utilizando os resultados desta subtração, é gerada uma matriz A que multiplicada pela sua transposta gera a matriz de covariância C, apresentada na equação (8):

$$C = AA^T \, . \tag{8}$$

Calculando os autovetores e autovalores da matriz C, é possível encontrar as *eigenfaces*. No entanto, é ainda necessária a sua redução de C e, por isso, multiplica-se a matriz A pelo conjunto selecionado de autovetores e cria-se, dessa forma, um subespaço de eigenfaces reduzido [34].

Os autovalores permitem classificar autovetores de acordo com a utilidade na caracterização da variação entre imagens [28]. As imagens *eigenfaces* apresentam um conjunto de base, possibilitando a descrição das imagens faciais e portanto, realizar o reconhecimento de faces. O reconhecimento facial é o reconhecimento de padrões e os autovetores significativos são escolhidos como aqueles com os maiores autovalores associados. Forma-se então a *eigenface* da imagem em questão. Para determinar a melhor classe que descreve uma imagem facial de entrada, deve encontrar-se a classe *k* que melhor minimiza a distância Euclidiana entre vetores, assim como mostra a equação (9):

$$\epsilon_k = \| \Omega - \Omega_k \|^2 . \quad (9)$$

onde Ω_k é o vetor que descreve a classe. Estas são calculadas através da média dos resultados da representação *eigenfaces* sobre um pequeno número de imagens faciais de cada indivíduo. A face é classificada como fazendo parte da classe k quando ϵ_k mínimo é inferior a um limiar definido. Em caso contrário, a face é classificada como "irreconhecível" [28].

Em suma, este método possibilita o reconhecimento de características relevantes na diferenciação de uma face para outra, analisando a variação dos valores dos *pixels*, dentro de um conjunto de imagens faciais [35]. Ele é bastante utilizado devido à independência da geometria facial, simplicidade de realização, possibilidade de a realização ser em tempo real; facilidade e rapidez no reconhecimento assim como elevada taxa de sucesso [34].

4.4.3. ANÁLISE DE COMPONENTES INDEPENDENTES (ICA)

A Análise de Componentes Independentes é tida como uma particularização da PCA, evidenciando a sua capacidade de extração de componentes independentes e não gaussianos [24]. Destaca-se pelo facto de maximizar a independência entre os componentes das imagens, separando fontes linearmente misturadas, e pela redução da dimensionalidade do espaço original da imagem.

Este método está diretamente relacionado com a Separação Cega da Fonte, cujo objetivo é decompor um sinal numa combinação linear de sinais independentes desconhecidos.

O método inicia-se prevendo que as imagens de entrada não passam de combinações de fontes independentes: se esta combinação for linear, constrói-se uma matriz cujos coeficientes definem a combinação linear, onde as imagens de faces correspondem a variáveis e os valores dos *pixels* fornecem informações acerca das mesmas. Se contrariamente, os *pixels* corresponderem a variáveis e as imagens fornecerem informações sobre as mesmas, passa a trabalhar-se com a matriz inversa da matriz de

combinação. É de salientar que na primeira forma, obtém-se como resultado propriedades locais da face, enquanto que pela segunda forma obtém-se propriedades globais da face [30].

4.4.4. ANÁLISE DISCRIMINANTE LINEAR (LDA)

A Análise Discriminante Linear consiste em separar classes de objetos com o objetivo de também reduzir a dimensão do espaço. Para tal, recorre a combinações lineares. O método inicia-se com a utilização de amostras de todas as classes. Posteriormente, calcula-se a matriz de dispersão entre as classes, que representa a diferença na aparência relacionada com a diferença de identidade, e a matriz de dispersão dentro da mesma classe, que representa as características intrapessoais e variações relacionadas com a aparência da mesma pessoa, com a finalidade de maximizar a relação entre os seus determinantes [30]. Os elementos que maximizam esta relação são denominados de *Fisherfaces*.

4.4.4.1 FISHERFACES

O método *Fisherfaces*, como visto no ponto anterior, é baseado na Análise de Discriminante Linear (LDA) e desenvolvido de forma a que a sua função não se altere com variações de condições de imagem e expressões faciais, ao mesmo tempo que reduz a dimensionalidade no espaço de características [28].

A existência de classes que representam a identidade de cada indivíduo de acordo com imagens já conhecidas e que formam as imagens de treinamento, proporciona a determinação da classe que melhor representa as imagens fornecidas, maximizando a variância entre classes e minimizando a variância dentro das classes.

A primeira etapa no método é a projeção das imagens num espaço de baixa dimensionalidade, recorrendo à técnica PCA, e posteriormente à aplicação da técnica LDA, para procurar a melhor característica discriminante linear no subespaço de PCA. Esta

projeção ocorre nos autovetores, comparando as faces projetadas no subespaço. Neste algoritmo recorre-se ao calculo da matriz de tamanho M com os valores de intensidade dos *pixels* correspondentes à imagem, à média total das imagens faciais, à face média de cada classe, à matriz de dispersão interclasses (dispersão das faces médias de cada classe em relação à média total), à matriz de dispersão intraclasses (dispersão em relação à face média da classe). Após a projeção das imagens de faces, imagens pertencentes à mesma classe (neste caso, à mesma pessoa) são distribuídas próximas umas das outras, afastando-se de imagens não pertencentes a essa pessoa. Assim, é possível determinar a pertença da imagem facial a determinado indivíduo, comparando a similaridade das novas imagens de teste com as imagens faciais do conjunto de treinamento.

4.4.5. REDES NEURAIS

A técnica é das mais utilizadas para reconhecimento facial. Para reconhecer uma imagem facial, recorre a uma preparação prévia das redes, recorrendo a diferentes imagens que podem ou não conter faces. As redes trabalham diretamente sobre as imagens de entrada com níveis de cinza, e percorrem a imagem detetando se existe uma imagem facial ou não.

São vários os modelos com diferentes complexidades, bem como as abordagens locais de redes neurais. O objetivo comum passa por encontrar estruturas locais e, com base nas mesmas, reconhecer o objeto pretendido [24].

4.4.6. LOCAL BINARY PATTERNS HISTOGRAMS (LBPH)

Local Binary Patterns (LBP) é sobretudo um método utilizado para extração de características durante processos de reconhecimento e de classificação facial. O principal objetivo prende-se com a classificação dos *pixels* numa imagem por meio de uma matriz, na qual o *pixel* central é usado como limiar para definição dos *pixels* vizinhos. Estes *pixels* definem-se binariamente, sendo que, para *pixels* com valores iguais ou superiores ao

limiar é dado o valor 1 e para os restantes com valor inferior ao limiar é dado o valor 0 [28]. Após esta epata, ocorre correspondência dos valores binários, com origem num novo valor binário que é atribuído à posição central da matriz. O resultado é uma imagem com diferentes tipos de texturas (Figura 30).

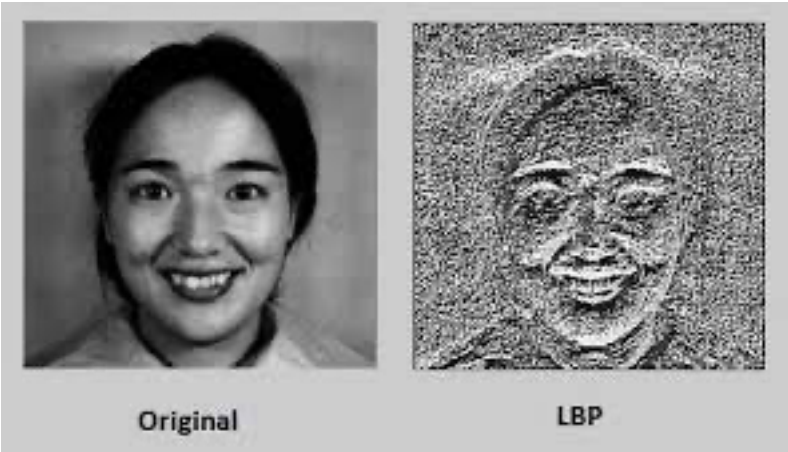

Figura 30: Operação Binária do LBP

Esta imagem é dividida em regiões com a possibilidade de extração de padrões, que conciliados formam um padrão geral da face. As regiões sofrem redução de dimensão dos dados, para que seja possível a classificação da imagem facial, de acordo com a sua similaridade e são convertidas em histogramas. Esta redução de dimensionalidade é designada de *Local Binary Patterns Histograms* (LBPH). Os *pixels* vizinhos podem também ter as suas dimensões alteradas, recorrendo a vizinhanças circulares e interpolação bilinear. É de notar que quanto maior o número de *pixels* vizinhos, maior o número de padrões obtidos.

4.4.7. SCALE INVARIANT FEATURE TRANSFORM (SIFT)

O método *Scale Invariante Feature Transforme* (SIFT) destaca-se pelo facto de originar uma elevada quantidade de características que cobrem toda a imagem. O algoritmo tem início com a extração de características de determinado conjunto de imagens predefinidas e com a procura pela sua correspondência. A facilidade com que encontra a sua correspondência prende-se com facto de o padrão de cada ponto chave ser único,

apesar de haver inúmeras possibilidades de correspondência. No entanto, muitas das características não encontram tão facilmente a sua correspondência, devido ao facto de a área de plano de fundo possuir uma área consideravelmente elevada. Neste caso, as características são filtradas possibilitando a identificação de subconjuntos de pontos chave que concordam com o objeto e com a localização, escala e orientação que está a ser criar [28].

4.4.8. SPEEDED UP ROBUST FEATURES (SURF)

O algoritmo *Speeded Up Robust Features* foi desenvolvido baseado no algoritmo anterior mas com a finalidade de ser mais rápido e eficaz. Tem início com a seleção, através de detetores de boa repetibilidade, de pontos de interesse em diferentes locais da imagem. A vizinhança de cada ponto de interesse é representada recorrendo a um vetor de características de grande eficiência na deteção de ruídos, erros de deteção e deformações. Os vetores são combinados entre as imagens, realizando-se a sua correspondência com base na distância entre eles [28].

4.4.9. BRUTE-FORCE MATCHER

O *Brute-Force Matcher* é um método utilizado por SIFT e SURF para correspondência entre os descritores da imagem facial que se pretende reconhecer e os descritores das imagens faciais contidas na base de dados preparada [28]. É realizada a comparação de cada ponto chave numa imagem original e calcula-se a distância entre os mesmo: aquele que apresentar menor distância é considerado o ponto chave correspondente.

4.5. VANTAGENS E DESVANTAGENS

Os sistemas biométricos apresentam diversas vantagens , destacando-se pelo facto de:

- Os sistemas biométricos são únicos de cada pessoa, não podendo ser passados para outros ou esquecidos, isto é, a identificação pessoal deixa de ser baseada em algo que o individuo tem ou sabe, para passar a considerar o próprio como modo de identificação;

- Sendo intransmissíveis, é incomum ou até mesmo impossível, haver duas pessoas com as mesmas características biométricas;

- Fácil identificação desde que haja uma base de dados previamente preparada;

- De difícil imitação, sendo por isso, pouco suscetíveis de ataques semelhantes ao que acontecem com senhas e cartões de passe.

O uso destes sistemas acarretam também desvantagens, como, por exemplo [29]:

- Impossível substituir determinada característica biométrica;

- Com o passar dos anos e avanço de idade, é possível que parte das características biométricas se vão degradando, diminuindo a precisão de identificação e autenticação das pessoas;

- Doenças ou acidentes podem influenciar a autenticação de indivíduos;

- Necessidade constante de atualização de base de dados;

- O reconhecimento facial realizado por computador é afetado pela complexidade dos rostos e expressão facial, pelas variações de luz e ângulo de captura;

- É possível ocultar características faciais com a óculos, bigode, cabelo e maquilhagem;

- Desafio ético e social da proteção de dados;

- Em alguns países existe proibição para reconhecimento facial;

- Como qualquer sistema informático, é possível ser roubado e alterado (*hackeado*).

5. DESENVOLVIMENTO DO SISTEMA DE RECONHECIMENTO FACIAL

Este capítulo é responsável por mostrar, de forma detalhada, o desenvolvimento do sistema de reconhecimento facial, bem como os métodos e ferramentas escolhidas para a criação da mesma.

5.1. ESCOLHA DO SOFTWARE E ALGORITMOS

O MATLAB foi o *software* escolhido para desenvolvimento da aplicação, por apresentar diversas vantagens, abordadas anteriormente. Destas, destacam-se sobretudo, o facto de

ser uma linguagem simples, de fácil entendimento e que se aproxima bastante das formas matemáticas; de apresentar um vasto leque de *Toolboxes*, incluindo para o tema abordado nesta dissertação; de permitir desenvolver programas de forma rápida e correr em diferentes SO, neste caso concreto, em Macintosh.

No que toca a implementação de algoritmos, optou-se por utilização do algoritmo de *Viola-Jones* para deteção de faces e o algoritmo de Análise de Componentes Principais para reconhecimento das mesmas.

5.2. CRIAÇÃO E TREINAMENTO DA BASE DE DADOS

Como visto anteriormente, para obter um bom sistema de reconhecimento facial, deve-se primeiramente avaliar e testar o mesmo através de uma base de dados com imagens de faces de pessoas. O presente trabalho vai recorrer a uma base de dados criada especificamente para este trabalho (Figura 31), uma vez que a deteção e reconhecimento das faces vai passar pela construção de imagens em tempo real através da câmara do computador. Foram utilizadas 10 imagens de faces de cada indivíduo, sendo que cada imagem tem uma resolução de 180 x 200 pixéis (tamanho recomendável para fotografias de perfil, tipo avatar), havendo indivíduos de ambos os sexos para uma maior variedade. As imagens estão no formato .jpg.

O primeiro passo passa por preparar a base de dados para que ocorra um reconhecimento viável. Deve ter-se em consideração que as imagens deverão ter todas a mesma localização e deve converter-se, inicialmente, todas as imagens (2D) para vetores (1D) com tamanho, neste caso, de 36 000. No entanto, este número é bastante grande o que poderá provocar um problema de memória ou erros no *software*, assim, é necessária a sua redução de pelo menos para metade (90x100). Neste caso, são utilizados também 50 autovalores uma vez que é o valor que resulta em melhores resultados.

Figura 31: Imagens da Base de Dados

O código MATLAB para preparação/treino da base de dados é apresentado no Anexo A. Nas duas primeiras linhas do código é necessária a colocação do número de imagens que se pretende treinar, bem como do número de autovalores que se quer considerar. Isto vai permitir que, mesmo que se altere o número de imagens a serem preparadas, não é necessária a alteração do código.

Utilizando este algoritmo é também possível obter a imagem média (Figura 32), bastando apenas fazer correr as linhas **MI=reshape(m,[100,90]); imshow(uint8(imresize(MI ,[200,180])),[]):**

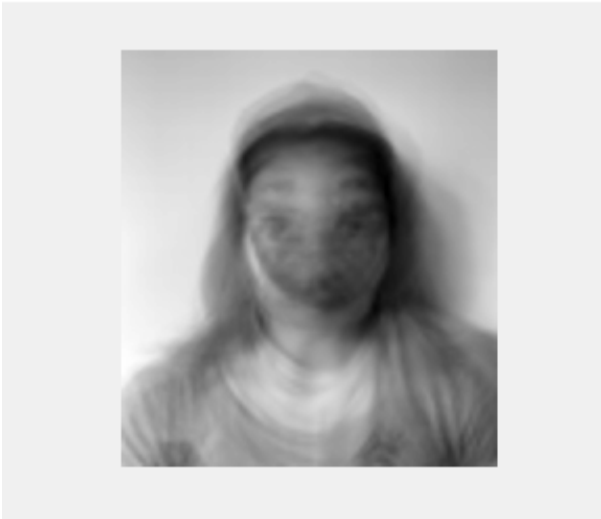

Figura 32 Imagem Média Resultante

Uma vez que a preparação da base de dados é um processo um pouco longo, não há necessidade de o repetir a cada reconhecimento que se realize. Assim, apenas se precisa de armazenar varáveis que serão importantes numa fase posterior, tais como número de imagens para treinar (n), imagem média (m), dimensões das imagens (M e N), conjunto de dados transformados no espaço PCA (T) e redução matriz de Transformação (Ppca).

5.3. DESENVOLVIMENTO DO SISTEMA DE RECONHECIMENTO FACIAL

O sistema de Reconhecimento Facial desenvolvido divide-se em duas partes: Deteção da Face e Reconhecimento da Face.

5.3.1. DETEÇÃO DAS FACES

Para que a deteção seja possível em tempo real, é necessário aceder à *webcam* do computador. Para tal, adiciona-se uma extensão do MATLAB para *webcams* USB, através de pacotes disponíveis do *software*. Para verificar se já possui a extensão, basta digitar o código ***webcam*** na janela de comando. Caso não se possua a extensão, automaticamente, é gerada uma mensagem com a possibilidade de redireccionamento para *Support Package Installer*. Neste, procura-se a extensão desejada, neste caso "MATLAB Support Package for USB Webcams", desenvolvida pela *MathWorks Image Acquisition Toolbox Team* e procede-se à sua instalação.

Neste trabalho vai ser utilizada uma *webcam* externa da Microsoft com as seguintes propriedades (Figura 33).

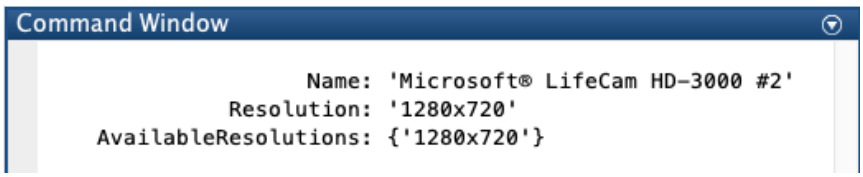

Figura 33: Propriedades da Webcam utilizada

O principal objetivo desta etapa é detetar a presença de faces quando se recorre a uma *webcam*. Para tal, é sobretudo utilizado o detetor de objetos em cascata, que utiliza o algoritmo *Viola-Jones* para detetar rostos. Este algoritmo também é capaz de detetar apenas olhos, nariz, boca ou qualquer outra parte superior do corpo de um individuo. No entanto, neste trabalho, apenas será detetado o rosto na sua generalidade. É utilizado um método – *Kanade-Lucas-Tomasi* (KLT), para rastreamento de recursos. Este método é também utilizado para estabilizar um vídeo, estimar movimentos da câmara e rastrear objetos. A sua função passa por, em cada ponto no quadro anterior, encontrar o ponto correspondente no quadro atual, para posteriormente, se estimar a translação, rotação e escala entre os pontos antigos e os pontos novos. A transformação é aplicada ao retângulo que delimita a face.

O processo inicia-se quando é exibido um quadro de vídeo, neste caso proveniente da *webcam*. Quando uma face é detetada, um retângulo de cor amarela, preenchido com pontos de cor branca, envolve o rosto detetado e segue-o quando este se movimenta (Figura 34 e Figura 35). Nesta etapa é importante escolher um recurso que seja único e permaneça invariável às movimentações do objeto.

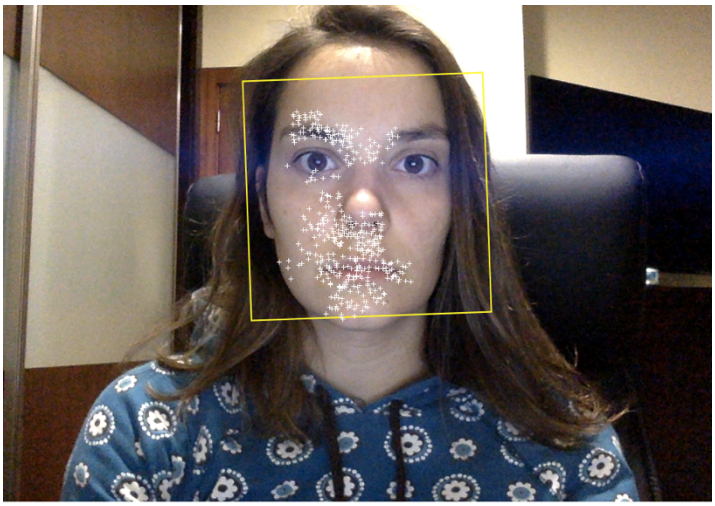

Figura 34: Deteção da Face (posição 1)

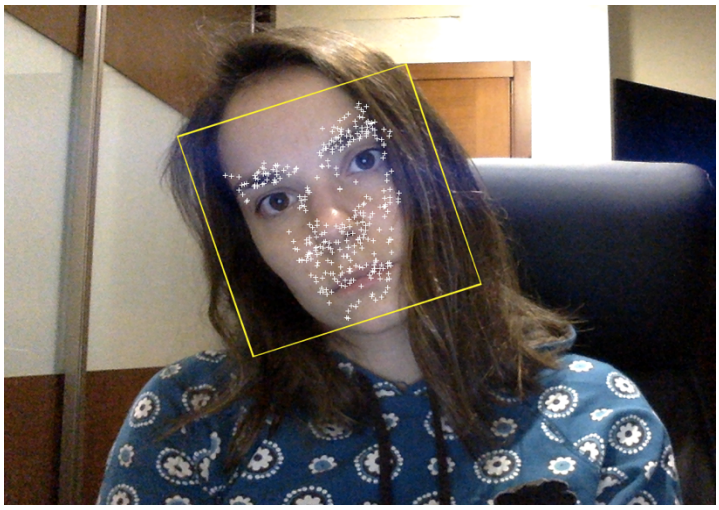

Figura 35: Deteção da Face (posição 2)

O código MATLAB para Deteção de Faces é apresentado no Anexo B.

5.3.2. RECONHECIMENTO DAS FACES

Uma vez criada a base de dados, resta apenas cruzar as faces detetadas em tempo real com a base de dados criada, de forma a encontrar a melhor correspondência. Para tal, recorre-se inicialmente às variáveis armazenadas na preparação das imagens da base de dados. O processo em si começa quando é pedido para selecionar uma imagem de uma localização no computador, imagem essa que será a imagem a ser reconhecida e que não pode estar inserida na mesma localização que a base de dados, uma vez que teria uma combinação de 100% . Posteriormente, é feita uma cópia para fins de exibição, é convertida para uma imagem de escala cinza e redimensionada para uma escala 100 x 90. Após redimensionar o vetor, a imagem é projetada para o espaço PCA, onde ocorre a subtração da imagem média e a multiplicação da mesma pela matriz de transformação, para utilização de dimensões mais reduzidas. Dá-se início à matriz de distâncias, para colocação de todas as distâncias, com a finalidade de comparar a imagem de entrada com todas as imagens armazenadas na base de dados. Após encontrar todas as distâncias, calcula-se o mínimo, que significa a melhor combinação da imagem. Posteriormente, o resultado é mostrado com a comparação visual entre as duas imagens, tal como mostram as Figuras 36 e 37.

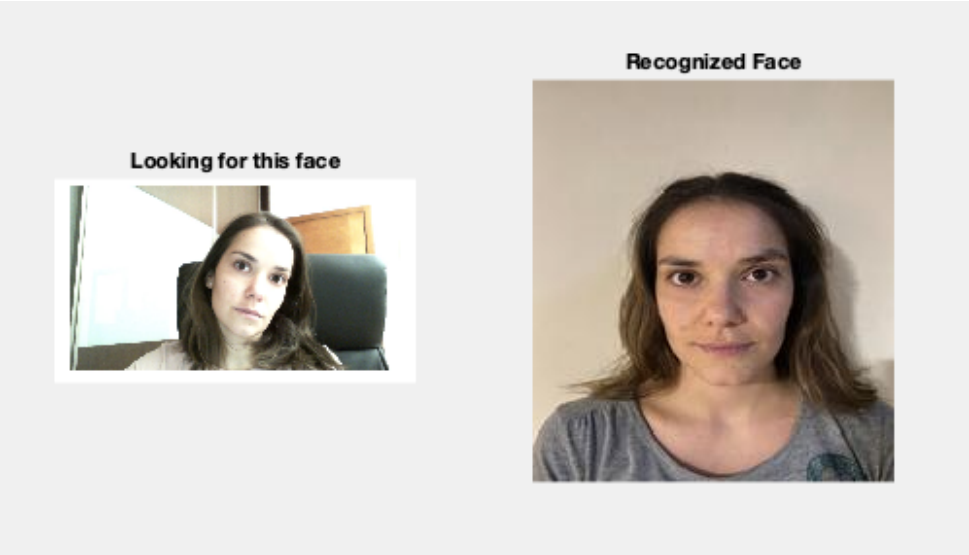

Figura 36: Reconhecimento Facial Indivíduo 1

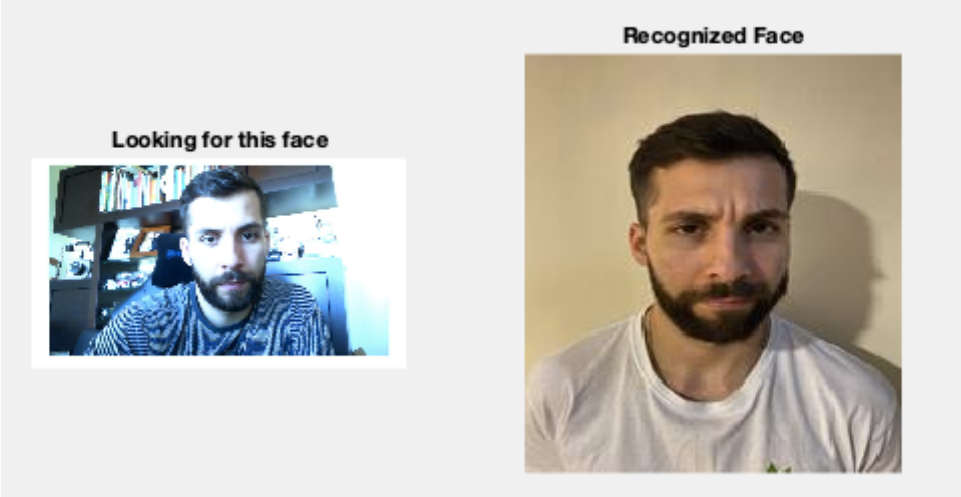

Figura 37: Reconhecimento Facial Indivíduo 2

Nos dois casos apresentados, ambos os indivíduos foram identificados corretamente.

O código MATLAB para reconhecimento de faces é apresentado no Anexo C.

5.3.3. SISTEMA DE RECONHECIMENTO DAS FACES

Como mencionado anteriormente, o sistema desenvolvido está dividido em duas partes, tal como mostra a Figura 38, que corresponde à tela inicial do sistema. Recorreu-se a interfaces gráficas do usuário (GUI) no MATLAB, para apresentação do mesmo.

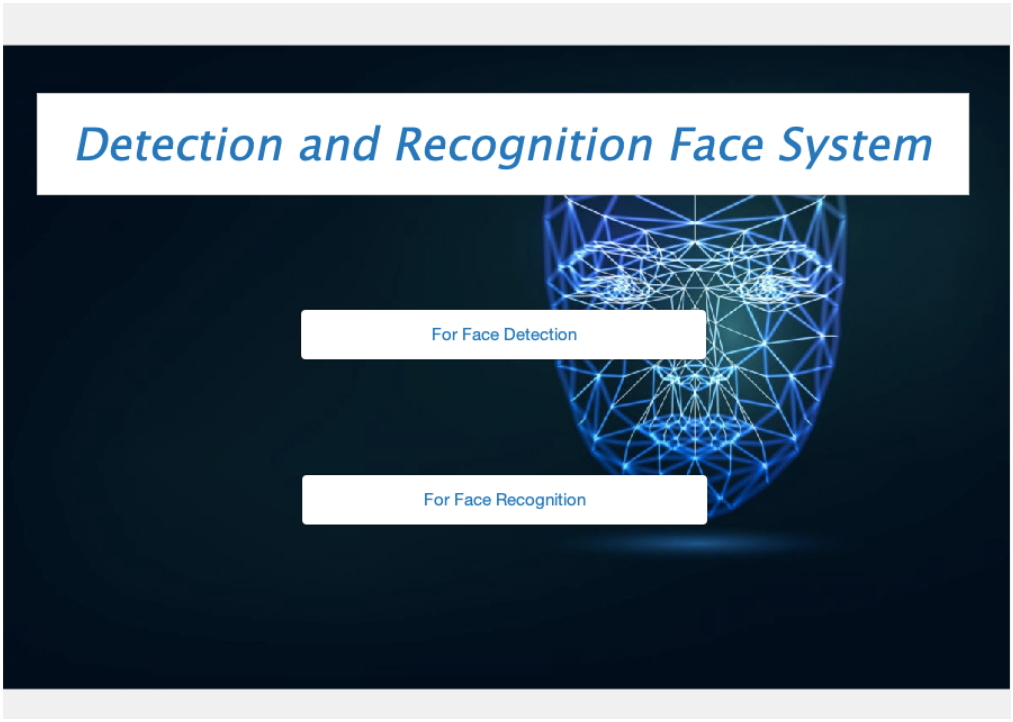

Figura 38: Visão geral do sistema desenvolvido

Tal como é apresentado, existem duas opções para iniciar o sistema. Escolhendo a primeira ("For Face Detection"), abre uma outra GUI que redireciona para a deteção de faces. Esta deteção é feita em tempo real, através da *webcam* existente e com os recursos a acompanhar a movimentação do rosto, sempre que ocorre deteção de um. Estes recursos são, como já mencionado anteriormente, um retângulo de cor amarela, preenchido com pontos de cor branca. A Figura 39 mostra um exemplo de deteção com as características abordadas, aquando da escolha da opção "For Face Detection". Esta imagem assemelha-se às conseguidas anteriormente (Figuras 34 e 35).

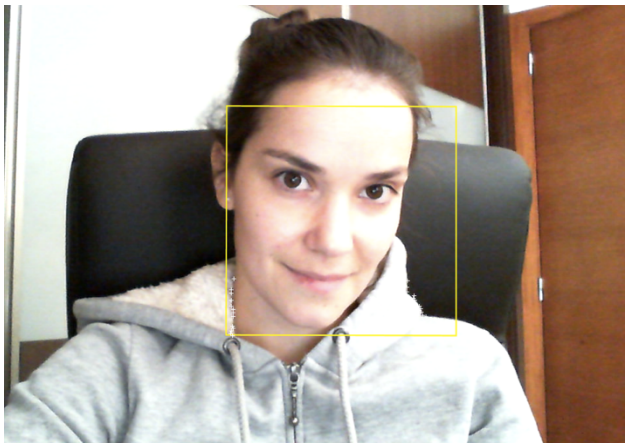

Figura 39: Deteção de face no sistema desenvolvido

Se se optar por escolher a segunda opção (“For Face Recognition”), também é gerada uma nova GUI (Figura 40) que faz a captura da imagem que aparece na *webcam*, logo após a escolha da opção. Esta imagem vai ser a escolhida para proceder ao posterior reconhecimento de faces. Desta forma, caso a imagem não esteja de acordo com o desejado, é possível, através da opção “Capture New Image”, capturar nova imagem. Posteriormente, é essencial armazenar essa imagem numa pasta vazia (ou com as imagens destinadas para reconhecimento) á escolha e fora da base de dados. É importante que o nome das imagens contenha números, isto é, que as imagens estejam numeradas, de forma a facilitar a procura da imagem selecionada. Esta etapa é possível escolhendo a opção “Save Image”. Os códigos para captura e armazenamento das imagens é apresentado no Anexo D.

Figura 40: Vista do sistema quando se escolhe a opção para Reconhecimento facial

Após armazenamento da imagem deve, então, proceder-se ao reconhecimento da mesma. A opção "Go to Recognition" torna isso possível. Escolhendo a mesma, é gerada uma nova GUI (Figura 41) onde apenas é possível voltar para trás no processo ou escolher a imagem para reconhecimento, através do *pushbotton* "Select Image". Selecionando essa opção, vai abrir as pastas existentes no computador, por defeito, onde o sistema se encontra, e deve proceder-se à escolha da pasta onde se localiza a imagem armazenada. Após seleção da imagem a considerar para reconhecimento, irá surgir o resultado final (Figura 42) que consiste na apresentação da imagem seleciona (quadro "Looking for this Face") e da correspondente imagem com a face reconhecida ("Recognized Face"). Realizado o reconhecimento é possível retornar e repetir o processo ("Return") ou fechar o sistema (X).

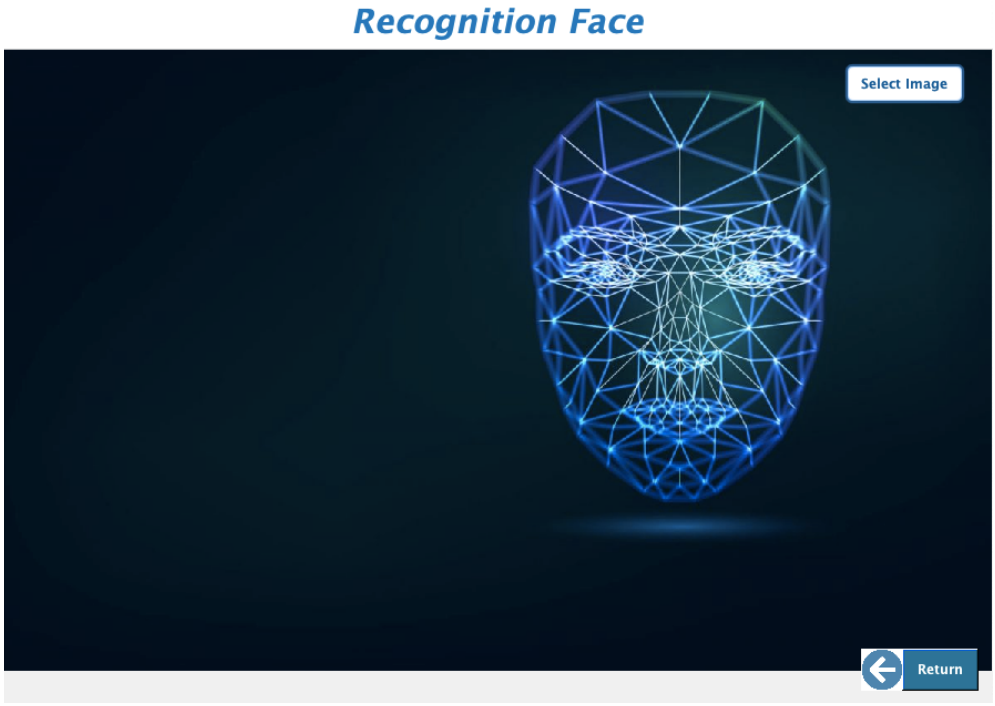

Figura 41: Visão geral da última etapa para Reconhecimento

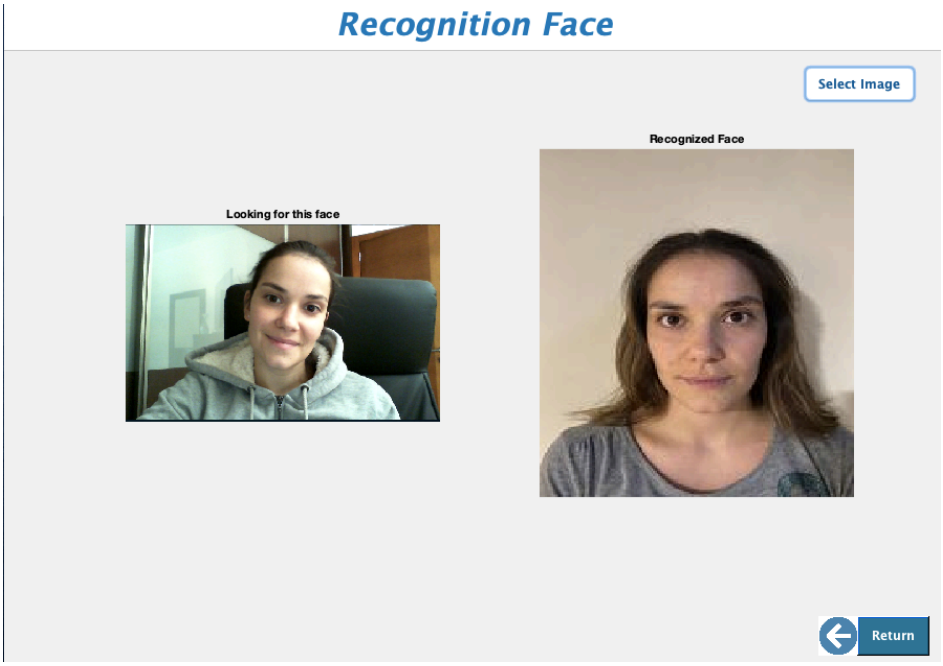

Figura 42: Reconhecimento da imagem gerada

O código MATLAB do sistema de reconhecimento desenvolvido é apresentado no Anexo E.

6. RESULTADOS FINAIS

Este capítulo é responsável por apresentar os resultados finais bem como abordar o sucesso do sistema de reconhecimento desenvolvido.

O principal objetivo deste trabalho vai de encontro ao desenvolvimento de um sistema de reconhecimento facial. Após o desenvolvimento do mesmo é possível, com sucesso, efetuar a deteção e o reconhecimento das faces e, por isso, pode dizer-se que os resultados foram bem sucedidos. No entanto é importante analisar, individualmente, cada ação e os seus resultados, já que, por vezes, ocorreram algumas falhas, destacando-se principalmente o facto de o sistema detetar erradamente objetos que não eram faces e reconhecer, também, faces sem correspondência.

No que concerne à deteção de faces, os resultados obtidos foram maioritariamente satisfatórios, uma vez que sempre que uma face era apresentada, o sistema detetava a mesma e acompanhava-a a cada movimento da mesma, tal como mostrado anteriormente nas Figuras 34, 35 e 39. No entanto, em determinadas situações, o sistema reconhecia outros objetos, mesmo não se tratando de faces. Esta situação estava, grande parte das vezes, ligada a variações de ambiente e padrões no vestuário, tal como mostra a Figura 43. Este obstáculo foi ultrapassado com a movimentação do indivíduo e/ou disfarce do objeto sem face, nomeadamente o encobrimento do mesmo.

Figura 43: Exemplo 1 de Deteção incorreta de face

Um outro fator importante a ter em consideração na deteção de faces é o tempo que demora a que a mesma ocorra. Em todos os testes realizados, verificou-se que o sistema já iniciava com a deteção de rosto feita, ou seja, a deteção de faces era instantânea.

Situações semelhantes às abordadas anteriormente acontecem quando se passa para a etapa de reconhecimento facial. Nem sempre o sistema respondeu de forma mais eficiente, já que para uma determinada face considerada, a face reconhecida não correspondia à face correta. Para um melhor entendimento do bom funcionamento do sistema, além se de realizarem testes com imagens provenientes da *webcam*, analisaram-se imagens de faces em diferentes condições ambientais, de iluminação e nitidez. De um total de 30 imagens testadas, 21 tiveram sucesso na correspondência e 7 não tiveram sucesso na correspondência. As Figuras 44 e 45 são exemplos desta última situação.

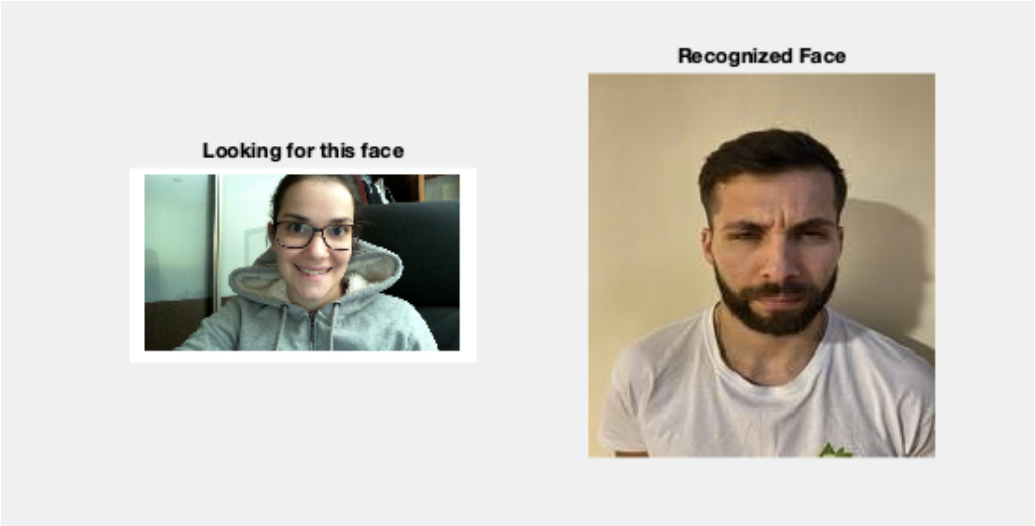

Figura 44: Exemplo de um incorreto reconhecimento facial

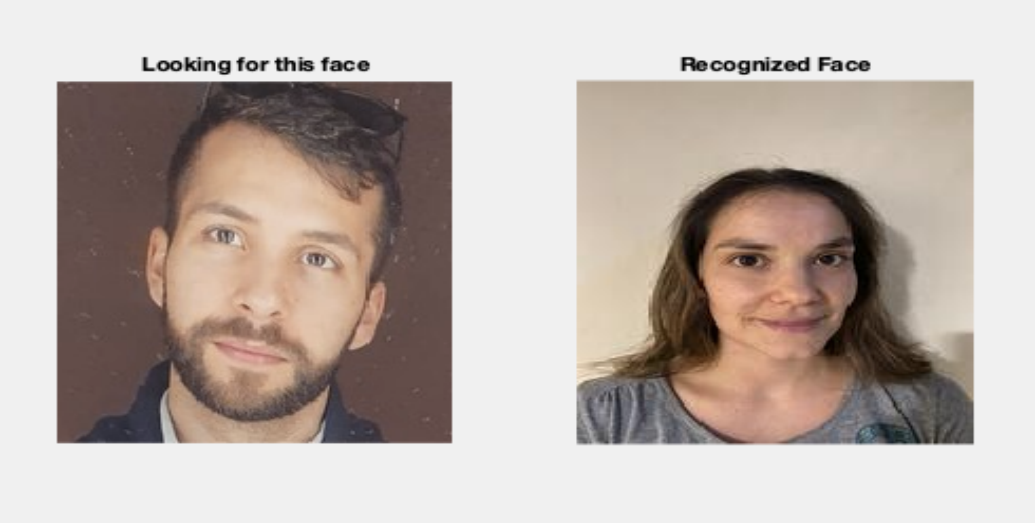

Figura 45: Exemplo de um incorreto reconhecimento facial

Situações como estas podem ser ultrapassadas com o aperfeiçoamento da base de dados, bem como o aumento de faces e indivíduos treinados. Além disso, uma imagem média mais "perfeita" poderá auxiliar neste processo, que pode ser conseguida com o melhor ajuste de posição de cada indivíduo, em cada imagem da base de dados. Outra forma de ultrapassar é a melhor preparação do sistema, isto é, quantas mais vezes o sistema for testado, mais sucesso irá ter.

7. CONCLUSÕES

Este trabalho permitiu conhecer de forma mais aprofundada em que consiste o processamento de imagens e o reconhecimento facial também em imagens, estendendo-se desde a parte teórica até ao desenvolvimento prático de um sistema, permitindo assim um maior conhecimento do tema.

Para um sistema de reconhecimento facial é necessário recorrer a determinados algoritmos e expressões algébricas, capazes de lidar com variações no ambiente das imagens, desde existência de adereços até às condições de iluminação, pose, entre outros.

Relativamente a este trabalho em concreto pode afirmar-se que os objetivos principais foram alcançados com sucesso já que foi possível o desenvolvimento de um sistema que

além de realizar o reconhecimento facial de imagens provenientes de uma *webcam* é também capaz de identificar qualquer tipo de imagem, desde que criada devidamente uma base de dados, e de detetar a presença de faces em imagens em tempo real.

A realização deste projeto não foi, de todo, uma tarefa fácil na medida que nem sempre se tornou possível conciliar a sua execução com a vida profissional e pessoal; o código muitas das vezes apresentava erros difíceis de decifrar, uma vez que não é um tema com o qual houvesse muito contacto, pelo que, muitas das funções utilizadas eram desconhecidas. No entanto, é de salientar a quantidade e qualidade de material de auxílio fornecido pela MathWorks, desde códigos já definidos e construídos pela comunidade, explicações sobre todas as funções existentes, assim como exemplos práticos, e feedbacks e questões/resposta entre a comunidade e a equipa da MathWorks que, muitas das vezes, ajudaram na resolução de erros no código. Uma outra dificuldade encontrada está ligada com a *webcam*, na medida que a câmara incorporada no computador não era capaz de capturar imagens nítidas e com claridade suficiente para obter imagens de qualidade. Essa foi a principal razão para se trabalhar com uma câmara externa da Microsoft que, embora com qualidade melhor, também capturava imagens com pouca qualidade, sendo que grande parte das vezes eram claras demais, enquanto que em outras era impossível distinguir objetos, devido ao facto de a imagem ser demasiado escura. No final, o mais importante é que foi possível ultrapassar estes obstáculos, conseguindo desenvolver o sistema proposto. O sistema em si, como falado no capítulo anterior, não está construído da forma perfeita, uma vez que, durante os testes, ocorreram alguns erros, mas na maioria das vezes, o sistema teve sucesso.

Futuramente, pretende-se aprofundar ainda mais o conhecimento desta matéria de forma a tentar reduzir o número de imagens com faces detetadas e reconhecidas erradamente. Isto é, agora com o sistema desenvolvido, o objetivo passa por melhora-lo para que seja segura a sua utilização.

Referências Documentais

[1] Moler, C. (2016). Introducing Cleve's Laboratory. *MathWorks.*

[2] J. Veira, *Matlab num Instante (versão 1.4)*. Departamento de Eletrónica e Telecomunicações da Universidade de Aveiro, 2004.

[3] Moler, C. (2018). A Brief History of MATLAB. *MathWorks.*

[4] MathWorks, "Products and Services", https://www.mathworks.com/products.html?s_tid=gn_ps.

[5] MathWorks, "MATLAB", https://www.mathworks.com/products/matlab.html.

[6] MathWorks, "Videos and Webinars – Car Drift Recognition Using Machine Learning", https://www.mathworks.com/videos/car-drift-recognition-using-machine-learning-1525327873602.html.

[7] MathWorks, "User Stories – Baker Hughes Develops Predictive Maintenance Software for Gas and Oil Extraction Equipment Using Data Analytics and Machine Learning", https://www.mathworks.com/company/user_stories/baker-hughes-develops-predictive-maintenance-software-for-gas-and-oil-extraction-equipment-using-data-analytics-and-machine-learning.html.

[8] MathWorks, "User Stories – Battelle Neural Bypass Technology Restores Movement to a Paralyzed Man's Arm and Hand", https://www.mathworks.com/company/user_stories/battelle-neural-bypass-technology-restores-movement-to-a-paralyzed-mans-arm-and-hand.html.

[9] MathWorks, "Videos and Webinars – How MATLAB Distributed Computing Server and Machine Vision Tools Are Transforming Shell", https://www.mathworks.com/videos/how-matlab-distributed-computing-server-and-machine-vision-tools-are-transforming-shell-1541001149396.html.

[10] MathWorks, "Videos and Webinars – Automated LiDAR Point-Cloud Annotation for Sensor Verification", https://www.mathworks.com/videos/automated-lidar-point-cloud-annotation-for-sensor-verification-1527491006097.html.

[11] Nakayama, R. (2018). Using Deep Learning to Reduce Radiation Exposure Risk in CT Imaging. *Ritsumeikan University.*

[12] MathWorks, "User Stories – Medviso Receives FDA 510(k) Approval and CE Marking for Cardiovascular Analysis Software", https://www.mathworks.com/company/user_stories/medviso-receives-fda-510k-approval-and-ce-marking-for-cardiovascular-analysis-software.html.

[13] MathWorks, "User Stories - German Aerospace Center (DLR) Robotics and Mechatronics Center Develops Autonomous Humanoid Robot with Model-Based Design", https://www.mathworks.com/company/user_stories/german-aerospace-

center-dlr-robotics-and-mechatronics-center-develops-autonomous-humanoid-robot-with-model-based-design.html.

[14] OPENCADD, “AS 7 VERDADES DO MATLAB”, https://opencadd.com.br/7-verdades-do-matlab/.

[15] Gonzales R.C., Woods R.E. *Digital Image Processing*. 3. Ed. Pearson Education, 2011

[16] Faria, Diogo, *Análise e Processamento de Imagem*. 2010. 44f. Trabalhos Práticos em Mestrado Integrado em Engenharia Biomédica – Faculdade de Engenharia da Universidade do Porto, Porto.

[17] Oliveira, Cristiane de Queiroz, *Metodologia de Processamento Digital Aplicada à Exploração de Imagens Radiológicas.* 2004. 128f. Dissertação de Mestrado em Engenharia Nuclear do Instituto Militar de Engenharia, Rio de Janeiro.

[18] Weeks, A.R.J. Fundamentals of Electronic Image Processing. SPIE OPTICAL ENGINEERING PRESS, 1996.

[19] Gomes, O. Processamento e Análise de Imagens Aplicados à Caracterização Automática de Materiais. 2001. 151f. Dissertação de Mestrado Ciências da Engenharia Metalúrgica da Universidade Católica do Rio de Janeiro. Rio de Janeiro.

[20] Padilha, A.J, “Processamento e Análise de Imagem”, https://web.fe.up.pt/~padilha/PAI/ficheiros/Cap3-ac.pdf.

[21] Maxwell, “Processamento e Análise Digital de Imagens”, https://www.maxwell.vrac.puc-rio.br/21365/21365_6.PDF.

[22] Moraes, Jairo Lucas, *Controle de Acesso Baseado em Biometria Facial*. 2010. 102f. Dissertação de Pós-Graduação em Informática do Centro Tecnológico da Universidade Federal do Espírito Santo. Vitória.

[23] Lourenço, Gonçalo Filipe da Fonseca. *Reforço da Segurança das Biométricas utilizando Codificação de Fonte Distribuída*. 2009. 105f. Dissertação de Mestrado em Engenharia Eletrotécnica e de Computadores do Instituto Superior Técnico da Universidade de Lisboa. Lisboa.

[24] Nunes, Fernanda Todesco. *Técnicas de Biometria Baseadas em Padrões Faciais e sua utilização na Segurança Pública*. 2015. 65f. Dissertação de Pós-Graduação em Tecnologias da Informação e Comunicação aplicadas à Segurança Pública e Direitos Humanos na Universidade Federal de Santa Catarina. Araranguá.

[25] Neto, Ernesto Luiz Andrade. *Sistemas de Identificação Pessoal Utilizando Técnicas de Reconhecimento e Verificação Facial Automáticas*. 1997. 137f. Tese de Mestrado em Engenharia Elétrica da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas. Campinas.

[26] Nascimento, Andreia Vieira do. *Detecção de Faces Humanas em Imagens Digitais: Um algoritmo Baseado em Lógica Nebulosa*. 2005. 139f. Dissertação de Mestrado

em Engenharia Elétrica da Escola de Engenharia de São Carlos da Universidade de São Paulo. São Carlos.

[27] Henriques, Marco António Silva. *Reconhecimento facial com base em compressão de imagem*. 2016. 92f. Dissertação de Mestrado em Engenharia Eletrónica e Telecomunicações da Universidade de Aveiro. Aveiro.

[28] Prado, Kelvin Salton do. *Comparação de Técnicas de Reconhecimento Facial para Identificação de Presença em um Ambiente Real e Semicontrolado*. 2018. 172f. Dissertação em Pós Graduação em Sistemas de Informação da Escola de Artes, Ciências e Humanidades da Universidade de São Paulo. São Paulo.

[29] THALES, "Facial recognition: top 7 trends (tech, vendors, markets, use cases and latest news)", https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition.

[30] Braga, Luiz Filipe Zenicola. *Sistemas de Reconhecimento Facial*. 2013. 84f. Trabalho de Conclusão de Curso de Engenharia Elétrica em ênfase em Eletrónica da Escola de Engenharia de São Carlos da Universidade de São Paulo. São Carlos.

[31] Hongyu, K; Sandanielo, V. L. M.; Oliveira Junior, G. J.. (2016). Principal Component Analysis: theory, interpretations and applications. *Engineering and Science.*

[32] ESTGV. "Análise de componentes Principais e Análise Factorial", http://www.estgv.ipv.pt/PaginasPessoais/lucas/material/Acetatos%20ACP%20aluno.pdf.

[33] bulitin. "A STEP BY STEP EXPLANATION OF PRINCIPAL COMPONENT ANALYSIS", https://builtin.com/data-science/step-step-explanation-principal-component-analysis.

[34] Çarikçi, M.; Ozen, F. (2012). A Face Recognition System Based on Eigenfaces Method. *SciVerse ScienceDirect.*

[35] Guimarães, Rafael Miranda. *Desenvolvimento de um protótipo de software de reconhecimento facial de tempo real para registro eletrônico de ponto em ambientes indoor com utilização do dispositivo kinect*. (2015). 197f. Dissertação de Mestrado em Sistemas de Informação e Gestão do Conhecimento na Faculdade de Ciências Empresariais da Universidade FUMEC. Belo Horizonte.

[36] MathWorks, "MATLAB GUI", https://www.mathworks.com/discovery/matlab-gui.html.

[37] MathWorks, "Help Center", https://www.mathworks.com/help/index.html.

[38] MathWorks, "MATLAB Answers", https://www.mathworks.com/matlabcentral/answers/index

Anexo A. Código MATLAB para Preparação/Treino da Base de Dados.

Neste anexo são apresentadas as linhas de código utilizadas para preparar/treinar a Base de Dados.

```
n=input('Enter No. of Images for training:');
L=input('Enter No. of Dominant Eigen Values
to keep:');

% Dimensões da Imagem
M=100; N=90;
% Matriz
X=zeros(n,(M*N));
% Conjunto de dados transformados no espaço
PCA
T=zeros(n,L);

for count=1:n
    % Ler as imagens
    I=imread (sprintf('%d.jpg',count));
    I=rgb2gray(I);
    I=imresize(I,[M,N]);
    % Transformar as imagens em vetores
    X(count,:)=reshape(I,[1,M*N]);
end

% Copiar a base de dados
Xb=X;
% Calcular a Media de todas as imagens
m=mean(X);
for i=1:n
% Subtrair a media de cada imagem da base de
dados
    X(i,:)=X(i,:)-m;
end
```

```
% Calcular a Matriz de Covariancia
Q=(X'*X)/(n-1);
% Obter dos autovalores e autovetores da
matriz de Covariancia
[Evecm,Evalm]=eig(Q);
% Extrair os autovalores
Eval=diag(Evalm);
% Ordenar os autovalores
[Evalsorted,Index]=sort(Eval,'descend');
Evecsorted=Evecm(:,Index);
% Reduzir a matriz de Transformacao
Ppca=Evecsorted(:,1:L);
for i=1:n
    T(i,:)=(Xb(i,:)-m)*Ppca;
end
```

Anexo B. Código MATLAB para Deteção de Faces

Neste anexo são apresentadas as linhas de código para a Deteção de Faces.

```
clear all

% Iniciar webcam
cam=webcam('Microsoft');
% Identificar a resolucao da webcam
cam.Resolution='1280x720';
% Leitura de quadros
video_Frame=snapshot(cam);

% Inicio do objeto de video
video_Player=vision.VideoPlayer('Position', [100 100
1280 720]);

% Definir Detetor de Faces
face_Detector=vision.CascadeObjectDetector();
% Rastrear as Faces
point_Tracker=vision.PointTracker('MaxBidirectionalErr
or',2);

% Variaveis para Loop
run_loop=true;  % Funcionamento do Loop
number_of_Points=0; % Numero de pontos
frame_Count=0; % Contagem de quadros

while run_loop && frame_Count<1200 % Accao do loop
para uma contagem de quadros menor que 1200

    % Armazenar os quadros do objeto
    video_Frame=snapshot(cam);
    % Escala de cinza
    gray_Frame=rgb2gray(video_Frame);
    % Aumento do contador de quadros em cada iteracao
    frame_Count=frame_Count+1;

    if number_of_Points < 10
        % Localizacao de um rectangulo que envolve o
rosto (funcao degrau detetor de faces)
        face_Rectangle=face_Detector.step(gray_Frame);
```

```
        % Para rectangulo preenchido, encontrar os
pontos do rectangulo
        if ~isempty(face_Rectangle)
            % Especificar a regiao de interesse
            points =
detectMinEigenFeatures(gray_Frame, 'ROI',
face_Rectangle(1, :));

            % Converter os pontos para valores XY
            % Iniciar o rastreador de pontos e
obtencao dos valores de X
            xy_Points=points.Location;
            number_of_Points=size(xy_Points,1);
           release(point_Tracker);
       initialize(point_Tracker,xy_Points,gray_Frame);

           % Atribuicao dos pontos reais a variavel
anterior
           previous_Points=x_Points;
           % Converter o retangulo ao redor do rosto
detetado em pontos
           rectangle=bbox2points(face_Rectangle(1,:));
           % Transformaçao do rectangulo na
movimentacao da face
           face_Polygon=reshape(rectangle',1,[]);
           % Caracteristicas do rectangulo e pontos

video_Frame=insertShape(video_Frame,'Polygon',face_Pol
ygon,'LineWidth',3);

video_Frame=insertMarker(video_Frame,xy_Points,'+','Co
lor','white');
        end
    else
        % Presença e armazenamento de pontos
   [xy_Points,isFound]=step(point_Tracker,gray_Frame);
        new_Points=xy_Points(isFound,:);
        old_Points=previous_Points(isFound,:);

        number_of_Points=size(new_Points,1);

        % Para alteracao significativa da face
        if number_of_Points >= 10
            % Matriz - transformacao da geometria
estimada
```

```
            [xform,old_Points,new_Points] =
estimateGeometricTransform(...

old_Points,new_Points,'similarity','MaxDistance',4);

            % Transformacao de pontos para a funcao
    rectangle=transformPointsForward(xform,rectangle);
            % Redimensionar o rectangulo
            face_Polygon=reshape(rectangle',1,[]);

            % Caracteristicas do restangulo e dos
pontos

video_Frame=insertShape(video_Frame,'Polygon',face_Pol
ygon,'LineWidth',3);

video_Frame=insertMarker(video_Frame,new_Points,'+','C
olor','white');

            % Definicao de novos pontos para a
variavel de pontos anteriores
            previous_Points=new_Points;
            % Definicao do rastreador de ponto para os
pontos anteriores atualizados
            setPoints(point_Tracker,previous_Points);
        end
    end

    % Definicao do quadro de video para reproducao
    step(video_Player,video_Frame);
    run_loop=isOpen(video_Player);
end

% Limpar a webcam e libertar os objetos
clear cam;
release(video_Player);
release(point_Tracker);
release(face_Detector);
```

Anexo C. Código MATLAB para Reconhecimento de Faces

Neste anexo são apresentadas as linhas de código para a Deteção de Faces.

```
clc;
% Carregar as variaveis necessarias no espaço de
trabalho
% Ja armazenadas anteriormente
load pcadb;
% Selecionar uma imagem para reconhecimento
[filename,pathname]=uigetfile('*.*','Select the Input
Image');
% Localizacao da imagem, copia para exibicao,
conversao de cores e redimensionamento
filewithpath=strcat(pathname,filename);
img=imread(filewithpath);
imgo=img;
img=rgb2gray(img);
img=imresize(img,[M,N]);
% Remodelacao para um vetor
img=double(reshape(img,[1,M*N]));

% Projecao da imagem para o espaco PCA
imgpca=(img-m)*Ppca; % Subtracao da media e
multiplicacao pela matriz de transformacao

% Iniciar a matriz de distancia de tamanho
distarray=zeros(n,1);
for i=1:n

    % calcular a distancia L1
    distarray(i)=sum(abs(T(i,:)-imgpca));
end

% Obter a melhor combinacao
[result,indx]=min(distarray);
% Obter a melhor imagem
resultimg=imread(sprintf('%d.jpg',indx));

% Mostrar o resultado final para comparação das
imagens
```

```
subplot(121)
imshow(imgo);
title('Looking for this face');
subplot(122)
imshow(resultimg);
title('Recognized Face');
```

Anexo D. Código MATLAB para Captura e Armazenamento de imagem.

Neste anexo são apresentadas as linhas de código para a Captura de imagem e para o seu Armazenamento.

```
% Captura de imagem
cam=webcam('Microsoft');
imshow(snapshot(cam));
clear('web')
```

```
% Armazenar imagem
cam=webcam('Microsoft');
global image
counter=1;
image=snapshot(cam);
imshow(image)
[Save,savename] = uiputfile('*.jpg','save this file')
fname=fullfile(savename,Save);
imwrite(image,fname);
clear ('web')
```

Anexo E. Código MATLAB para o Desenvolvimento do Sistema de Reconhecimento Facial

Neste anexo são apresentadas as linhas de código para o Desenvolvimento do Sistema de Reconhecimento Facial. Está dividido em três GUI's.

GUI Principal:

```
function varargout = ExemploGui(varargin)
% EXEMPLOGUI MATLAB code for ExemploGui.fig
%      EXEMPLOGUI, by itself, creates a new EXEMPLOGUI
or raises the existing
%      singleton*.
%
%      H = EXEMPLOGUI returns the handle to a new
EXEMPLOGUI or the handle to
%      the existing singleton*.
%
%
EXEMPLOGUI('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in EXEMPLOGUI.M with
the given input arguments.
%
%      EXEMPLOGUI('Property','Value',...) creates a
new EXEMPLOGUI or raises the
%      existing singleton*.  Starting from the left,
property value pairs are
%      applied to the GUI before ExemploGui_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value
makes property application
%      stop.  All inputs are passed to
ExemploGui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose
"GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
ExemploGui
```

```
% Last Modified by GUIDE v2.5 09-Oct-2020 14:33:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton,
...
                   'gui_OpeningFcn',
@ExemploGui_OpeningFcn, ...
                   'gui_OutputFcn',
@ExemploGui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before ExemploGui is made visible.
function ExemploGui_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to ExemploGui (see
VARARGIN)

% Choose default command line output for ExemploGui
handles.output = hObject;
axes(handles.axes1);
imshow('imagem1.jpg');
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ExemploGui wait for user response (see
UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the
command line.
function varargout = ExemploGui_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles
structure
varargout{1} = handles.output;




function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit1 as text
%        str2double(get(hObject,'String')) returns
contents of edit1 as a double



% --- Executes during object creation, after setting
all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
```

```
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

clear all

cam=webcam('Microsoft');
cam.Resolution='1280x720';
video_Frame=snapshot(cam);

video_Player=vision.VideoPlayer('Position', [100 100
1280 720]);

face_Detector=vision.CascadeObjectDetector();
point_Tracker=vision.PointTracker('MaxBidirectionalErr
or',2);

run_loop=true;
number_of_Points=0;
frame_Count=0;

while run_loop && frame_Count<1200
    video_Frame=snapshot(cam);
    gray_Frame=rgb2gray(video_Frame);
    frame_Count=frame_Count+1;

    if number_of_Points < 10
```

```
        face_Rectangle=face_Detector.step(gray_Frame);

        if ~isempty(face_Rectangle)
            points =
detectMinEigenFeatures(gray_Frame, 'ROI',
face_Rectangle(1, :));

            xy_Points=points.Location;
            number_of_Points=size(xy_Points,1);
           release(point_Tracker);

initialize(point_Tracker,xy_Points,gray_Frame);

           previous_Points=xy_Points;

           rectangle=bbox2points(face_Rectangle(1,:));
           face_Polygon=reshape(rectangle',1,[]);


video_Frame=insertShape(video_Frame,'Polygon',face_Pol
ygon,'LineWidth',3);

video_Frame=insertMarker(video_Frame,xy_Points,'+','Co
lor','white');
        end
    else

[xy_Points,isFound]=step(point_Tracker,gray_Frame);
        new_Points=xy_Points(isFound,:);
        old_Points=previous_Points(isFound,:);

        number_of_Points=size(new_Points,1);

        if number_of_Points >= 10
            [xform,old_Points,new_Points] =
estimateGeometricTransform(...

old_Points,new_Points,'similarity','MaxDistance',4);


rectangle=transformPointsForward(xform,rectangle);

            face_Polygon=reshape(rectangle',1,[]);
```

```
video_Frame=insertShape(video_Frame,'Polygon',face_Pol
ygon,'LineWidth',3);

video_Frame=insertMarker(video_Frame,new_Points,'+','C
olor','white');

            previous_Points=new_Points;
            setPoints(point_Tracker,previous_Points);
        end
    end
    step(video_Player,video_Frame);
    run_loop=isOpen(video_Player);
end

clear cam;
release(video_Player);
release(point_Tracker);
release(face_Detector);



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

GuiRec;
close ExemploGui;
```

GUI Captura e Armazenamento de Imagem:

```
function varargout = GuiRec(varargin)
% GUIREC MATLAB code for GuiRec.fig
%      GUIREC, by itself, creates a new GUIREC or
raises the existing
%      singleton*.
%
```

```
%      H = GUIREC returns the handle to a new GUIREC
or the handle to
%      the existing singleton*.
%
%
GUIREC('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in GUIREC.M with the
given input arguments.
%
%      GUIREC('Property','Value',...) creates a new
GUIREC or raises the
%      existing singleton*.  Starting from the left,
property value pairs are
%      applied to the GUI before GuiRec_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value
makes property application
%      stop.  All inputs are passed to
GuiRec_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose
"GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
GuiRec

% Last Modified by GUIDE v2.5 10-Oct-2020 12:37:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton,
...
                   'gui_OpeningFcn',
@GuiRec_OpeningFcn, ...
                   'gui_OutputFcn',
@GuiRec_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GuiRec is made visible.
function GuiRec_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to GuiRec (see
VARARGIN)

% Choose default command line output for GuiRec
handles.output = hObject;
axes(handles.axes1);
imshow('imagem1.jpg');


axes(handles.axes4);
imshow('seta2.png');

axes(handles.axes2);
Captura;


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GuiRec wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the
command line.
function varargout = GuiRec_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles
structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit1 as text
%        str2double(get(hObject,'String')) returns
contents of edit1 as a double



% --- Executes during object creation, after setting
all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit2 as text
%        str2double(get(hObject,'String')) returns
contents of edit2 as a double



% --- Executes during object creation, after setting
all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

```
Captura;



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

GuiPca;
close GuiRec;



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

salvarimagem;



% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)



% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
```

```
% handles    structure with handles and user data (see
GUIDATA)
ExemploGui;
close GuiRec;
```

GUI Reconhecimento Facial:

```
function varargout = GuiPca(varargin)
% GUIPCA MATLAB code for GuiPca.fig
%      GUIPCA, by itself, creates a new GUIPCA or
raises the existing
%      singleton*.
%
%      H = GUIPCA returns the handle to a new GUIPCA
or the handle to
%      the existing singleton*.
%
%
GUIPCA('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in GUIPCA.M with the
given input arguments.
%
%      GUIPCA('Property','Value',...) creates a new
GUIPCA or raises the
%      existing singleton*.  Starting from the left,
property value pairs are
%      applied to the GUI before GuiPca_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value
makes property application
%      stop.  All inputs are passed to
GuiPca_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose
"GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
GuiPca

% Last Modified by GUIDE v2.5 10-Oct-2020 20:17:30
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton,
...
                   'gui_OpeningFcn',
@GuiPca_OpeningFcn, ...
                   'gui_OutputFcn',
@GuiPca_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GuiPca is made visible.
function GuiPca_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to GuiPca (see
VARARGIN)

% Choose default command line output for GuiPca
handles.output = hObject;
axes(handles.axes2);
imshow('imagem1.jpg');
axes(handles.axes3);
imshow('seta2.png');
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GuiPca wait for user response (see
UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the
command line.
function varargout = GuiPca_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles
structure
varargout{1} = handles.output;




function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit1 as text
%        str2double(get(hObject,'String')) returns
contents of edit1 as a double



% --- Executes during object creation, after setting
all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
```

```
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit2 as text
%        str2double(get(hObject,'String')) returns
contents of edit2 as a double


% --- Executes during object creation, after setting
all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
```

```
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
testpca1;



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
GuiRec;
close GuiPca;
```

Histórico

- 12 de Outubro de 2020, Versão 1.0, fta@isep.ipp.pt
- 26 de Outubro de 2020, Versão 1.0a, fta@isep.ipp.pt
- 31 de Outubro de 2020, Versão 1.0.b