Universidade Federal da Paraíba
Centro de Energias Alternativas e Renováveis
Departamento de Engenharia Elétrica

Alef Kaian Feitosa Barbosa

Sistema de Autenticação por Biometria utilizando Técnicas de Machine Learning e Processamento Digital de Sinais

João Pessoa, 2019

Alef Kaian Feitosa Barbosa

Sistema de Autenticação por Biometria utilizando Técnicas de Machine Learning e Processamento Digital de Sinais

Monografia apresentada ao curso de Engenharia Elétrica do Centro de Energias Alternativas e Renováveis, da Universidade Federal da Paraíba, como requisito para a obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Alexsandro José Virgínio dos Santos

João Pessoa, 2019

Catalogação na publicação
Seção de Catalogação e Classificação

B238s Barbosa, Alef Kaian Feitosa.
Sistema de Autenticação por Biometria utilizando Técnicas de Machine Learning e Processamento Digital de Sinais / Alef Kaian Feitosa Barbosa. - João Pessoa, 2019.
120 f. : il.

Orientação: Alexsandro José Virgínio dos Santos.
Monografia (Graduação) - UFPB/CEAR.

1. Sistema de autenticação. 2. Biometria. 3. Reconhecimento facial. 4. Processamento digital de sinais. 5. Machine Learning. I. Santos, Alexsandro José Virgínio dos. II. Título.

UFPB/BC

Trabalho de Conclusão de Curso de Engenharia Elétrica intitulado ***Sistema de Autenticação por Biometria utilizando Técnicas de Machine Learning e Processamento Digital de Sinais*** de autoria de Alef Kaian Feitosa Barbosa, aprovado pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Alexsandro José Virgínio dos Santos
DEE/UFPB

Prof. Dr. Fabrício Braga Soares de Carvalho
DEE/UFPB

Prof. Dr. José Maurício Ramos de Souza Neto
DEE/UFPB

Coordenador(a) do Departamento de Engenharia Elétrica
Prof. Dr. Alexsandro José Virgínio dos Santos
DEE/UFPB

João Pessoa, 2019

Centro de Energias Alternativas e Renováveis, Universidade Federal da Paraíba
Lot. Cidade Universitaria, Castelo Branco, João Pessoa, Paraíba, Brasil CEP: 58051-900
Fone: +55 (83) 3216-7200

“Quando algo é importante o suficiente, você realiza mesmo que as chances não estejam a seu favor.”

(Elon Musk)

AGRADECIMENTOS

A Deus, por ter criado um universo tão perfeito que até hoje se faz ciência para tentar entendê-lo.

Aos meus pais, Teresinha e Aziz, por desde sempre me mostrarem o caminho da educação e proporcionarem o melhor ambiente possível, independente do que fosse necessário.

As minhas irmãs, Wlly e Anne, por terem me levado a ser uma pessoa melhor.

Aos meu familiares, em especial, Paula, Mosir e Fernando, por desde pequeno me conduzirem direta e indiretamente até aqui.

A Aurélio, Jean, Alexandre, Thiago e Wanderly, por terem me dado tantos momentos de descontração, mesmo nas horas mais difíceis.

A Rodrigo, por ter me ajudado a continuar seguindo meu caminho diversas vezes ao longo da graduação.

A Matheus, por todas as caronas e toda força que me deu durante o curso.

A Thommas, que sempre me ajudou quando eu precisei, e sempre esteve junto até quando eu não precisei.

A Thalyne, Pedro, Felipe e Marote, porque mesmo estando longe me trouxeram muitos momentos de alegria e descontração como se estivessem do meu lado.

A Lucas, Viviane e Bia, porque os três são como um só para mim, e esse é um exemplo de família que eu quero seguir.

A Amanda, por ter me mostrado que superheróis existem no mundo real.

A Bia, Cyn, Gabriel, Eliézio, Eugênio, Marcos, Bianca e Giovanna, que apesar de não estarem por perto agora, estavam no começo disso tudo.

A todos os professores do curso, em especial, Alexsandro, Antônio, Carlos Alberto, Fabrício, Lucas e Maurício, por fazer eu manter minha paixão por Engenharia Elétrica por todos esses anos.

RESUMO

O objetivo deste trabalho foi desenvolver um sistema de autenticação utilizando biometria, que são características de cada indivíduo que os tornam único em relação aos demais. Essa ideia foi proposta para substituir sistemas de autenticação clássicos, como senhas e cartões, que estão sujeitos a erros humanos, como perda e furto. O tipo de biometria escolhido foi reconhecimento facial e o sistema foi desenvolvido utilizando linguagem de programação Python, algoritmos de processamento digital de sinais, como o histograma dos gradientes orientados e transformações afins em imagens, e técnicas de *machine learning*, como máquinas de vetores de suporte e redes neurais convolucionais. Foram feitas duas abordagens para a medição do desempenho e aperfeiçoamento do sistema: uma quantitativa e uma qualitativa. A quantitativa consistiu em medir as taxas de falsa rejeição e falsa aceitação para o conjunto de dados disponível e encontrar os parâmetros que minimizassem essas taxas. A qualitativa consistiu em selecionar alguns dados que, a critério do autor, teriam mais chance de gerar saídas erradas, por serem imagens de pessoas com a aparência próxima, chamadas de decisões difíceis, e ajustar os parâmetros do sistema para minimizar o erro para esse conjunto de amostras. Foi concluído que o método qualitativo gerou resultados mais confiáveis e foi sugerido que a melhor abordagem seria uma combinação dos dois métodos. O tempo médio medido para o sistema fazer uma autenticação utilizando uma webcam foi de 2,7 s. Finalmente, foi verificado que o detector facial possuía um tempo de detecção de, em média, 430 ms contra 200 ms dos detectores no estado da arte e foram sugeridas soluções para a melhoria desse tempo. Além disso, foi detectada uma falha de segurança, em que o sistema pode ser fraudado por uma pessoa utilizando uma imagem estática de outra pessoa, e foram feitas sugestões de correção para essa falha ao se utilizar ou um sensor de profundidade, ou os pontos de referência da face ou uma imagem do histórico de movimento.

Palavras-chave: Sistema de autenticação, Biometria, Reconhecimento facial, Processamento digital de sinais, *Machine learning*.

ABSTRACT

The objective of this work was to develop an authentication system using biometrics, which are characteristics of each individual that make them unique in relation to the others. This idea was proposed to replace classic authentication systems, such as passwords and cards, which are prone to human error, such as loss and theft. The type of biometrics chosen was facial recognition and the system was developed using Python programming language, digital signal processing algorithms, like the histogram of oriented gradients and affine transformations in images, and machine learning techniques, like support vector machines and convolutional neural networks. Two approaches were taken to measure performance and improve the system: a quantitative and a qualitative one. The quantitative approach consisted in measure the false rejection and false acceptance rates for the available dataset and to adjust the parameters so this rates were minimized. The qualitative one consisted in selecting some data that, at the author's criteria, would be more likely to generate wrong outputs, because they were images of people with near appearance, called hard decisions, and adjusting the system parameters so the error for this set of data was minimized. The measured average time for an authentication be done by the system using a webcam was 2.7 s. Finally, it was verified that the face detector had a detection time of 430 ms, on average, against 200 ms from state of the art detectors and solutions were suggested to improve this time. In addition, a security flaw was detected, where the system may be tricked by a person using an static image of another person, and suggestions were made to correct this flaw by using either a depth sensor, the facial landmarks or a motion history image.

Keywords: Authentication system, Biometrics, Facial recognition, Digital signal processing, Machine learning.

LISTA DE FIGURAS

1 Exemplos de biometria 22
2 Exemplo de impressão digital 24
3 Funcionamento de um sensor ótico para aquisição de uma impressão digital 25
4 Funcionamento de um sensor capacitivo para aquisição de uma impressão digital 26
5 Exemplo de íris 27
6 Exemplo de sensor de reconhecimento de íris de um smartphone moderno 27
7 Exemplo de padrão de veias capturado por um sensor de imagem vascular 28
8 Funcionamento de um sensor vascular 29
9 Sensor vascular comercial 29
10 Estrutura de um microfone dinâmico 31
11 Estrutura de um microfone condensador 31
12 Estrutura de um microfone de eletreto 32
13 Representação do processo de reconhecimento facial 33
14 Exemplo de câmera que pode ser utilizada para reconhecimento facial 34
15 Diagrama de blocos das etapas de PDS 35
16 Circuito de sample and hold 35
17 Conversor flash 37
18 Conversor por aproximações sucessivas 38
19 Conversor do tipo contador 39
20 Conversor do tipo R2R 40
21 Imagem original e em tons de cinza 41
22 Imagem original e resultado de algumas transformações afins 42
23 Exemplo de árvore de decisão e sua representação no espaço 43
24 Exemplo de hiperplano ótimo e de vetores de suporte 46
25 Estrutura de uma rede neural artificial 52
26 Gráfico da função sigmoide 54
27 Estrutura de uma rede neural convolucional 58
28 Representação dos gradientes de uma imagem 59

29 Exemplos de pontos de referência da face de alguns indivíduos 62
30 Comparativo 1 entre o detector de Viola-Jones (à esquerda) e o detector por HOG (à direita) 66
31 Comparativo 2 entre o detector de Viola-Jones (à esquerda) e o detector por HOG (à direita) 67
32 Algumas imagens do dataset FaceScrub 68
33 Algumas imagens do dataset LabelMe 68
34 Cálculo do HOG utilizando a classe do OpenCV 69
35 Criação e treinamento de uma SVM utilizando o pacote scikit-learn.svm 70
36 Método utilizado para o teste de desempenho da SVM 71
37 Primeiro problema de detecção de face 72
38 Segundo problema de detecção de face 73
39 Terceiro problema de detecção de face 73
40 Imagem original com a face desalinhada 75
41 Imagem com a posição da face corrigida 75
42 68 pontos de referência das faces 76
43 Sintaxe da função utilizada para detectar os 68 pontos de referência da face 76
44 Primeira parte do código do algoritmo de alinhamento da face 77
45 Segunda parte do código do algoritmo de alinhamento da face 78
46 Terceira parte do código do algoritmo de alinhamento da face 78
47 Quarta parte do código do algoritmo de alinhamento da face 78
48 Quinta parte do código do algoritmo de alinhamento da face 79
49 Etapa de treinamento da RNCP baseado em triplets 81
50 Sintaxe da função que extrai 128 medidas da face de uma pessoa . . . 81
51 Sintaxe da função que compara dois conjuntos de medidas da face . . 82
52 Fluxograma do algoritmo para reconhecimento de face 83
53 Taxa de acerto da primeira SVM treinada para cada pacote de amostras 85
54 Taxa de acerto da SVM retreinada com hard mining para cada pacote de amostras 86
55 Algumas imagens dos pacotes 22, 23 e 24 que foram classificadas de forma errada 87

56 Taxa de acerto da terceira SVM treinada para cada pacote de amostras 88
57 Imagens usadas como exemplo de mudança de parâmetros do detector de face 89
58 Regiões detectadas para um limiar de decisão de 0,5 90
59 Regiões detectadas para um limiar de decisão de 1,5 90
60 Regiões detectadas para 2 redimensionamentos da janela de detecção 91
61 Regiões detectadas para 8 redimensionamentos da janela de detecção 91
62 Regiões detectadas para uma taxa de redimensionamento da janela de 0,3 92
63 Regiões detectadas para uma taxa de redimensionamento da janela de 1 93
64 Regiões detectadas para um stride de 1/2 do tamanho da janela 94
65 Regiões detectadas para um stride de 1/16 do tamanho da janela . . . 94
66 Regiões detectadas paras as imagens de exemplo utilizando os parâmetros ótimos 95
67 Tempo de execução médio em função do número de imagens cadastradas do indivíduo 96
68 Taxa de falsa aceitação (FAR) em função do número de imagens cadastradas do indivíduo 97
69 Taxa de falsa rejeição (FRR) em função do número de imagens cadastradas do indivíduo 97
70 Par de indivíduos que caracterizam uma decisão difícil 99
71 Exemplo 1 de resultados de uma decisão difícil após o ajuste do limiar . 100
72 Exemplo 2 de resultados de uma decisão difícil após o ajuste do limiar . 100
73 Exemplo 3 de resultados de uma decisão difícil após o ajuste do limiar . 100
74 Tela do programa onde o usuário declara quem ele é 101
75 Exemplo 1 de funcionamento do sistema com imagens de pessoas famosas 101
76 Exemplo 2 de funcionamento do sistema com imagens pessoas famosas 102
77 Exemplo 3 de funcionamento do sistema com imagens de pessoas famosas 102
78 Exemplo 1 de funcionamento do sistema com a webcam 103
79 Exemplo 2 de funcionamento do sistema com a webcam 103

80 Exemplo 3 de funcionamento do sistema com a webcam 103
81 Problema de regiões detectadas erroneamente como face pelo detector 104
82 Detecção de imagem estática 105
83 Funcionamento de um sensor de profundidade 106
84 Pontos de referência da face de uma pessoa real 107
85 Pontos de referência da face de uma imagem estática 107
86 Exemplo de imagem do histórico de movimento 108

LISTA DE ABREVIATURAS

CART - Árvores de classificação e regressão (*Classification and regression trees*)

CCD - Dispositivo de carga acoplada (*Charged coupled device*)

CMOS - Semicondutor de óxido metálico complementar (*Complementary metal oxide semiconductor*)

FAR - Taxa de falsa aceitação (*False acceptance rate*)

FRR - Taxa de falsa rejeição (*False rejection rate*)

HOG - Histograma dos gradientes orientados (*Histogram of oriented gradients*)

IBGE – Instituto Brasileiro de Geografia e Estatística

IBOPE – Instituto Brasileiro de Opinião Pública e Estatística

MHI - Imagem do histórico de movimento (*Motion history image*)

ML - Aprendizado de máquina (*Machine learning*)

PDS - Processamento digital de sinais

ReLU - Unidade linear retificada (*Rectified linear unit*)

RNA - Rede neural artificial

RNC - Rede neural convolucional

RNCP - Rede neural convolucional profunda

ROI - Região de interesse (*Region of interest*)

SVM - Máquina de vetores de suporte (*Support vector machine*)

SUMÁRIO

1 INTRODUÇÃO **15**

1.1 DEFINIÇÃO DO PROBLEMA 18

1.1.1 Objetivo geral 20

1.1.2 Objetivos específicos 20

1.2 ESTRUTURA DA MONOGRAFIA 21

2 CONCEITOS GERAIS E REVISÃO DA LITERATURA **22**

2.1 BIOMETRIA 22

2.1.1 Reconhecimento de impressão digital 23

2.1.2 Reconhecimento de íris 26

2.1.3 Reconhecimento vascular 28

2.1.4 Reconhecimento de voz 30

2.1.5 Reconhecimento facial 32

2.2 PROCESSAMENTO DIGITAL DE SINAIS 34

2.2.1 Imagens digitais 40

2.3 ÁRVORES DE DECISÃO E DE REGRESSÃO 42

2.3.1 Definição 42

2.3.2 Classificação e treinamento 43

2.4 MÁQUINAS DE VETORES DE SUPORTE 45

2.4.1 Definição 45

2.4.2 Classificação e treinamento 47

2.5 REDES NEURAIS ARTIFICIAIS 52

2.5.1 Definição 52

2.5.2 Modelo matemático 53

2.5.3 Treinamento 55

2.6 REDES NEURAIS CONVOLUCIONAIS 56

2.7 HISTOGRAMA DOS GRADIENTES ORIENTADOS 58

2.7.1 Definição 58

2.7.2 Algoritmo 59

2.8 ESTIMATIVA DOS PONTOS DE REFERÊNCIA DA FACE 62

2.8.1 Definição 62
2.8.2 Algoritmo 62

3 METODOLOGIA 65
3.1 ETAPAS DO ALGORITMO 65
3.1.1 Identificação de faces em uma imagem 66
3.1.2 Correção da posição das faces 75
3.1.3 Extração de *features* do rosto 79
3.1.4 Verificação dos indivíduos 81
3.1.5 Algoritmo final 82
3.2 OBTENÇÃO DOS RESULTADOS 83

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS 85
4.1 RESULTADOS DOS TREINAMENTOS DA SVM 85
4.2 RESULTADOS DA MUDANÇA DE PARÂMETROS DO DETECTOR DE FACES 88
4.2.1 Mudança do limiar de distância do hiperplano 89
4.2.2 Mudança do número de redimensionamentos da janela 91
4.2.3 Mudança da taxa de redução do tamanho da janela 92
4.2.4 Mudança do passo de varredura da janela 93
4.2.5 Parâmetros ótimos 95
4.3 TESTES DE DESEMPENHO DO SISTEMA 95
4.4 REFINAMENTO DO SISTEMA POR DECISÕES DIFÍCEIS 99
4.5 FUNCIONAMENTO DO SISTEMA 101
4.6 PROBLEMAS DO SISTEMA E SUGESTÕES DE CORREÇÃO 104
4.6.1 Detector de face 104
4.6.2 Imagem estática 105

5 CONCLUSÕES E TRABALHOS FUTUROS 109

6 REFERÊNCIAS 110

APÊNDICE 1 – PARES DE PESSOAS ESCOLHIDAS PARA O REFINAMENTO POR DECISÕES DIFÍCEIS 120

1 INTRODUÇÃO

Autenticação ”é o ato de confirmar que algo ou alguém é autêntico, ou seja, uma garantia de que qualquer alegação de ou sobre um objeto é verdadeira” (AMOROSO, 2009).

Historicamente, identificação e autenticação foram a base de acesso de pessoas a determinados lugares. As senhas sempre foram as guardiãs do espaço. O propósito da existência delas foi de incluir ou excluir pessoas de um certo espaço. A utilização de senhas começou na Roma Antiga, em busca de restrição de acesso. Por ser uma sociedade militar, foi desenvolvido um sistema de senhas em forma de lemas, que determinava se um indivíduo possuía acesso ou não a um determinado local. Para os romanos as senhas eram basicamente o que diferenciava os aliados dos inimigos. Nesse caso, as senhas eram um meio de exclusão. Na era medieval as senhas serviam como uma ”autenticação de dois fatores” para as guildas medievais. Para pertencer a esses grupos normalmente se exigia um treinamento e a adoção de um conjunto de bons modos, e as senhas serviam como confirmação da confiança no indivíduo. Nesse caso, elas serviam para inclusão (DEVELLE, 2016).

Nos dias atuais, autenticação e identificação são presentes no cotidiano das pessoas, até de forma imperceptível. Seja para embarcar em um avião, para buscar um filho na escola, acessar uma rede social ou fazer compras com um cartão, é necessário que a identidade do indivíduo seja confirmada (BOLLE et al., 2013). Para casos em que a quantidade de pessoas seja relativamente pequena, como acesso a um condomínio ou a uma festa privada, fazer a identificação de forma é suficiente, porém, quando o número de pessoas começa a aumentar, utilizar um sistema que faça isso de forma automática é mais eficiente.

Tradicionalmente, há três tipos de autenticação: posses, conhecimento e biometria (BOLLE et al., 2013). Posses são objetos que precisam ser mostrados ou utilizados para ser feita a autenticação, como chaves ou cartões. Conhecimento são informações que supostamente somente o indivíduo deve possuir, que são utilizadas na hora da autenticação, como senhas ou frases secretas. Biometria são características únicas de uma pessoa, sejam físicas ou comportamentais, que as dinstinguem das outras. Ela começou a ser utilizada em sistemas de autenticação nos últimos anos, e

há vários tipos, como reconhecimento de impressão digital, reconhecimento facial e reconhecimento de voz.

Na China, desde 2017, há um circuito fechado de mais de 170 milhões de câmeras com tecnologia de reconhecimento facial que podem encontrar um indivíduo no meio da multidão em até sete minutos (BBC, 2017), e a polícia da cidade de Zhengzhou, do mesmo país, já utiliza óculos escuros com câmeras integradas com a mesma tecnologia (VINCENT, 2018). Esse tipo de aplicação de biometria também já é feita pelo Reino Unido (ZUIN, 2019) e durante o mês de março de 2019, um programa experimental da Secretaria de Estado da Polícia Militar do Rio de Janeiro monitorou o bairro de Copacabana com 28 câmeras de reconhecimento facial e identificação de placas de veículos com o objetivo de identificar suspeitos foragidos durante o Carnaval (BOM DIA RIO, 2019). Em sistemas de autenticação, alguns modelos de *smartphone* utilizam reconhecimento facial ou de íris para fazer o desbloqueio de tela, e alguns caixas eletrônicos e mesmo aplicativos de banco já utilizam reconhecimento de impressão digital para garantir acesso da conta ao usuário. Ainda no Brasil, desde 2008 foi adotado o sistema de biometria para autenticação dos eleitores nas eleições. Em 2019 já existem cidades no Brasil em que os transportes públicos possuem sistema de biometria, seja por reconhecimento de impressão digital ou reconhecimento facial.

Algumas das vantagens da utilização de biometria em sistemas de autenticação são a não rejeição, ou seja, o usuário não consegue ”errar” a autenticação, se utilizar do jeito correto; a intransferibilidade, que é o caso de uma característica biométrica não poder ser utilizada por outra pessoa; a impossibilidade de perda, salvo alguns casos raros; e um grande nível de proteção contra fraudes (THAKKAR, 2017).

Para que seja criado um sistema de autenticação automático baseado em características físicas ou comportamentais é necessário que se transforme essas características em dados que possam ser lidos por um computador. Para isso existe a área da Engenharia chamada de processamento digital de sinais (PDS). Sinais são um conjunto de informações sequenciadas sobre uma manifestação física ou abstrata, que variam com o tempo ou espaço. Exemplos de sinais são dados moleculares, químicos, biológicos, sequências de atributos ou quantidades de sensores, imagens, sons, etc (MOURA, 2009). Ainda, segundo Moura, processamento se refere a operações de representação, filtragem, codificação, transmissão, detecção, reconhecimento, síntese,

gravação ou reprodução de sinais. Os sinais podem ser representados matematicamente por uma função de uma ou mais variáveis, por vetores ou por matrizes. E o termo digital se refere ao domínio do sinal no processamento. Neste caso, ele é discreto em tempo e em amplitude, em oposição a sinais analógicos que possuem uma faixa de amplitude infinita, pois são contínuos. Os computadores e processadores modernos só trabalham com os sinais no domínio digital, portanto a área de PDS também se preocupa em como representar um sinal do domínio analógico no domínio digital. Em resumo, ”PDS é a matemática, os algoritmos e as técnicas utilizadas para manipular esses sinais depois que eles são convertidos para uma forma digital” (SMITH, 1999, tradução própria).

No entanto, dependendo do que se deseja como resultado, às vezes o sinal é muito complexo para se encontrar um algoritmo ou um sistema que faça o processamento por si só. Por isso, normalmente são utilizadas técnicas de PDS para extrair características importantes do sinal em relação ao que se quer obter. Por exemplo, em um sistema de reconhecimento de locutor (voz) uma característica importante que pode ser extraída da voz são as componentes de frequência que a compõem. Essas características são referidas como *features*.

Com a complexidade dos dados, passou a se utilizar técnicas de *machine learning* (ML - do inglês, aprendizado de máquina) para processá-los. A ideia geral de ML é que o computador aprenda a realizar uma tarefa ao ser treinado com um conjunto de exemplos e consiga realizá-la posteriormente com dados não vistos antes (LOURIDAS e EBERT, 2016). Há dois tipos de treinamento para modelos de ML: o supervisionado e o não supervisionado. No treinamento supervisionado é fornecido um conjunto de dados de treinamento com suas respectivas respostas esperadas. Assim, quando um novo dado for fornecido se espera uma saída próxima da saída dos dados semelhantes a ele que foram usados no treinamento. Geralmente esse tipo de treinamento é utilizado quando a tarefa desejada é de classificação. No treinamento não supervisionado são fornecidos dados de entrada sem suas saídas, logo o computador deve encontrar soluções por si só. O objetivo de um algoritmo que utiliza esse treinamento é descobrir padrões ou características significativas nos dados de entrada para separá-los em grupos (HAYKIN, 2001). Portanto, trabalha-se com PDS em conjunto com ML de modo que sejam extraídas características importantes dos sinais por PDS e elas

sejam utilizadas como dados de entrada em métodos de ML.

1.1 DEFINIÇÃO DO PROBLEMA

As senhas, por muito tempo, foram o único meio de autenticação automático utilizado pelos serviços. De fato, é um sistema fácil de ser implementado e utilizado, pois cada usuário só precisa se preocupar em lembrar a própria senha e, para o sistema fazer a autenticação, só é necessária a comparação de duas cadeias de caracteres. As senhas de banco, por exemplo, possuem geralmente seis caracteres numéricos e um limite de três tentativas de acesso. Isso daria uma brecha para alguém que quisesse fazer um acesso não autorizado de três chances em um milhão. Como as chances são bem pequenas, criou-se um pressuposto de que apenas o conhecimento pode confirmar a identidade (DEVELLE, 2016). Em outras palavras, há um senso comum de que apenas quem possui um conhecimento oculto (a senha) é a pessoa cuja identidade precisa ser confirmada, então a população adotou as senhas como um método seguro de autenticação.

Quando se considera uma única pessoa utilizando um único serviço, assumir que o sistema de senhas é seguro é fácil. Porém, no mundo digital é necessário se criar uma conta para desde utilizar redes sociais até fazer compras em lojas eletrônicas. Segundo o Instituto Brasileiro de Geografia e Estatística (IBGE), em 2017 o Brasil possuía 126,4 milhões de usuários de internet (SILVEIRA, 2018) e segundo a pesquisa trimestral CONECTAí Express, que é uma plataforma de pesquisas online do Instituto Brasileiro de Opinião Pública e Estatística (IBOPE), 79% dos internautas utilizam pelo menos um aplicativo de banco online (CANTARINO BRASILEIRO, 2017). Em relação às principais redes sociais, em fevereiro de 2019 o Facebook divulgou que a quantidade de usuários diários chegou a 2,3 bilhões (G1, 2019), enquanto o Instagram em junho de 2018 possuía 1 bilhão de usuários ativos (WAKKA, 2018) segundo dados divulgados pela própria empresa, e o Twitter, no último semestre de 2018, possuía 128 milhões de usuários monetizáveis, que de acordo com a empresa são os usuários que utilizam a plataforma ou abrem *tweets* [1] que estão sujeitos a ter propagandas

[1] Nome dado às publicações dos usuários na plataforma Twitter

(RIBEIRO, 2019).

Por isso, os consumidores são forçados a criar e administrar diversas senhas para diferentes serviços. O ponto chave é que eles não fazem isso muito bem. Eles criam senhas fáceis de se descobrir, como a data do aniversário ou o próprio nome, criam listas de senhas fáceis de se achar, e, principalmente, utilizam a mesma senha para diversos serviços diferentes. Ainda assim, mesmo a senha mais complexa e única que se pode criar pode ser descoberta, através de *phishing* [2] e *malwares* [3] (HORN, 2019). Além disso tudo, ainda pode acontecer de o usuário se esquecer da senha, que, a depender do serviço, pode levar alguns minutos ou dias para ser recuperada. Horn afirma que a indústria estima que cerca de 40% das ligações a *call centers* são para redefinição de senha, que levam cerca de 15 minutos e custam algo como U$ 10 por contato.

Sobre sistemas de autenticação por posse, como cartões, existe a possibilidade de perda ou furto, o que leva a um problema análogo ao das senhas. Geralmente nesses sistemas é utilizado o que se chama de autenticação por dois fatores, que seria uma camada extra de proteção. Por exemplo, em uma compra com cartão de crédito, além do número do cartão é necessário a senha ou o dígito verificador. Mas isso não quer dizer que o sistema é totalmente infalível, como são vistas diariamente muitas clonagens de cartões de crédito.

Então, o método de autenticação emergente é a biometria. As características utilizadas em um sistema de biometria são difíceis de serem compartilhadas, perdidas, roubadas, forjadas, e não podem ser alteradas (BOLLE et al., 2013), salvo em raros casos. Se for utilizado um sistema de biometria de dois fatores, as chances de fraude são praticamente nulas. Os maiores problemas da utilização de biometria são a precisão do sistema, a capacidade computacional necessária para processar as informações, o custo-benefício e a aceitação pelo usuário. A precisão não pode ser medida, porém pode ser estimada pela taxa de falsa aceitação e pela taxa de falsa rejeição (THAKKAR, 2017). Um dos desafios dos estudos sobre biometria é o quanto a qualidade dos dados de entrada afetam o sistema como um todo. Em reconhecimento facial, por

---

[2] Prática em que se obtém uma informação de alguém de modo que o próprio indivíduo a entrega, mesmo sem perceber.

[3] *Softwares* mal intencionados

exemplo, fatores como iluminação, pose, expressão facial e ruído afetam diretamente o desempenho do sistema. A capacidade computacional está ligada ao custo-benefício, porém deve-se notar que nos dias atuais muitas pessoas utilizam *smartphones*. De acordo com pesquisa feita pela Fundação Getúlio Vargas de São Paulo (FGV-SP), o Brasil em 2018 possuía cerca de 220 milhões de smartphones ativos (LIMA, 2018), o que seria em teoria mais de um por habitante. Esses aparelhos já possuem processadores e sensores que possibilitam a autenticação biométrica, portanto esses sistemas não estão longe de serem opções viáveis em questão de custo. A aceitação remete a quão invasivo esse tipo de sistema é em relação ao usuário final, mas isso pode ser contornado com um sistema de autenticação alternativo manual, para os consumidores que preferem não utilizar esse tipo de sistema, possuírem algum problema que os impossibilite de utilizá-lo ou estarem em alguma condição temporária em que a autenticação esteja sujeita a falhar (BOLLE et al., 2013).

Dito isso, as pesquisas sobre os temas abordados já estão avançadas o suficiente para se ter sistemas robustos e de alta confiabilidade, de modo que a utilização de biometria em sistemas de autenticação já seja uma realidade.

1.1.1 Objetivo geral

Com tudo o que foi discutido, esse trabalho tem como objetivo desenvolver um algoritmo para um sistema de autenticação com chance mínima de falhas humanas ou fraudes e fácil utilização pelos usuários, utilizando verificação biométrica e técnicas de PDS e ML.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Definir os tipos de biometria mais utilizados atualmente;
- Escolher um tipo e definir os principais algoritmos utilizados no seu processamento;
- Definir um algoritmo para o desenvolvimento do sistema;

- Implementar o sistema em linguagem de programação;
- Mostrar o funcionamento e medir o desempenho do sistema;
- Detectar e propor métodos para corrigir as falhas do sistema.

1.2 ESTRUTURA DA MONOGRAFIA

Este trabalho é dividido essencialmente em três partes. A primeira é uma revisão da literatura e a base teórica para o desenvolvimento do trabalho. A segunda é a metodologia utilizada, que é a descrição dos procedimentos feitos para realizar a concepção do sistema proposto e a explicação das ferramentas e técnicas utilizadas para se obter os resultados. E a terceira é a análise dos resultados obtidos e a apresentação de técnicas de correção e aperfeiçoamento para o que foi desenvolvido.

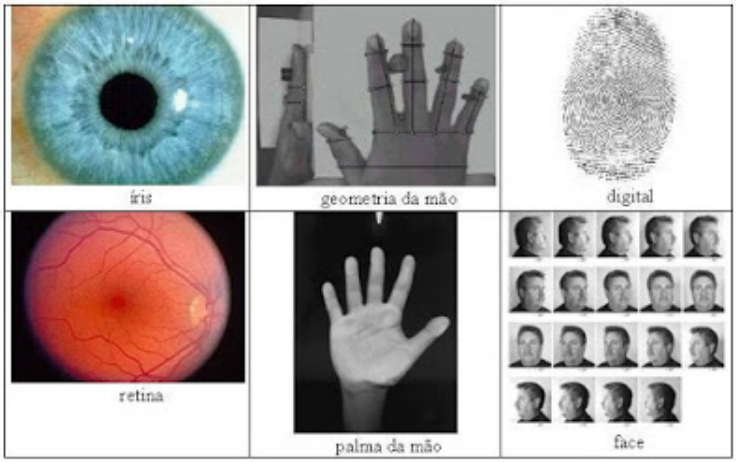
2 CONCEITOS GERAIS E REVISÃO DA LITERATURA

Neste capítulo são explicados os conceitos que foram utilizados no desenvolvimento do trabalho, assim como a base teórica necessária para o entendimento de cada etapa.

2.1 BIOMETRIA

Biometria (ou identificação biométrica) é a identificação de uma pessoa com base em suas características únicas, sejam físicas ou comportamentais (BOLLE et al., 2013). Reconhecimento facial, impressão digital e reconhecimento de íris são alguns exemplos de biometria física, do mesmo modo que assinatura e modo de andar são exemplos de biometria comportamental. A Figura 1 apresenta alguns exemplos de biometria.

Figura 1 – Exemplos de biometria



Fonte: ARAÚJO et al. (2015).

Essas características são capturadas dos usuários com um sistema de aquisição e, após o processamento, são comparadas com os modelos cadastrados até ser encontrada ou não uma combinação. Dificilmente haverá 100% de igualdade entre a leitura e o modelo, por isso nesse tipo de autenticação geralmente se escolhe um limiar de similaridade aceitável para separar as decisões positivas das negativas. É

importante citar a diferença entre identificação e verificação. Em um sistema de identificação, uma amostra é comparada com todas as amostras de um banco de modelos pré-cadastrados e em um sistema de verificação uma amostra é comparada com um modelo específico (RAMOS, 2012).

Existem diversos tipos de biometria, cada um extrai uma característica diferente do indivíduo, cada um possui suas vantagens e desvantagens. De acordo com (THAKKAR, 2017), algumas características que devem ser levadas em consideração na escolha do tipo de biometria para determinada aplicação são:

- **Precisão:** um dos fatores mais importantes. Pode ser estimada pela taxa de falsa aceitação e falsa rejeição e está diretamente ligada ao nível de segurança do sistema.
- **Resistência a fraude:** significa o quão acessível a entrada do sistema é para um possível malfeitor.
- **Custo-benefício:** É a relação entre o custo do sistema com o retorno que ele trará. Thakkar afirma que o investimento inicial de um sistema de biometria pode ser recuperado em um curto período de tempo.
- **Aceitação dos usuários:** é uma característica que depende de vários fatores, pois diferentes usuários podem não querer utilizar esse tipo de sistema por diversos motivos diferentes, como dificuldade, burocracia ou perda de privacidade.
- **Higiene:** outro fator importante. Depende da necessidade de haver contato físico com o *hardware* ou não.

A seguir são apresentados os principais tipos de reconhecimento biométrico e suas características.

2.1.1 Reconhecimento de impressão digital

É o tipo de biometria mais utilizado atualmente, pelo potencial de identificar uma pessoa de forma confiável utilizando uma característica física quase universal (THAKKAR, 2017).

A impressão digital é o desenho formado pelas ranhuras presentes nas pontas dos dedos quando impressos em uma superfície lisa, como no exemplo mostrado na Figura 2. A unicidade dessa característica física é muito alta, porque as impressões digitais de uma pessoa são tão únicas que são diferentes até em gêmeos idênticos (SRIHARI et al., 2008) e não sofrem alterações espontâneas ao longo da vida. Ainda se pode aumentar a segurança de um sistema que utiliza esse tipo de característica ao se utilizar mais de uma impressão digital no processo de autenticação.

Figura 2 – Exemplo de impressão digital

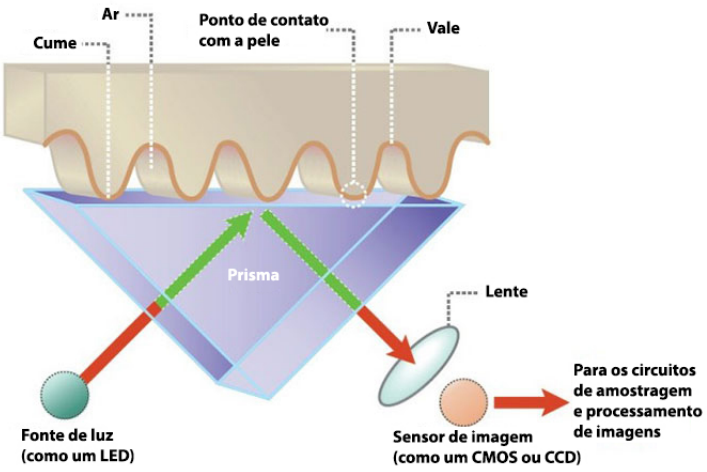
Fonte: Adaptado de SRIHARI et al. (2008).

O reconhecimento por impressão digital possui uma alta aceitação dos usuários por ser um meio de autenticação que já é utilizado há bastante tempo, além do uso ser intuitivo e não invasivo. Muitos fatores contribuíram para a popularidade dessa tecnologia, como o baixo custo dos sensores e a sua miniaturização, e por isso ela está presente até em *smartphones* mais modernos.

Para a aquisição de impressões digitais utiliza-se um scanner específico para isso. Há diversos tipos de sensores projetados para isso, mas os dois tipos mais utilizados são os óticos e os capacitivos (THAKKAR, 2018). Os sensores óticos utilizam um sistema de iluminação interno, geralmente LEDs, para iluminar o dedo e a captura é bem parecida com o de uma câmera digital, pois utilizam sensores óticos do tipo CMOS ou CCD. A imagem capturada recebe um pós-processamento analógico para ser melhorada para ser utilizada nos algoritmos de reconhecimento. O funcionamento da aquisição utilizando sensores óticos é mostrado na Figura 3. A resolução influencia diretamente no nível de segurança, pois quanto maior, mais detalhes da impressão po-

dem ser capturados. Uma das desvantagens desse tipo de sensor é que por capturar apenas a luz ele é mais fácil de ser fraudado, pois não há como diferenciar a impressão digital de uma pessoa presente de uma impressão digital impressa em um papel. Por esse motivo, scanners que utilizam esse sensor geralmente possuem outro sensor embutido para detectar se uma pessoa está presente ou não (THAKKAR, 2018).

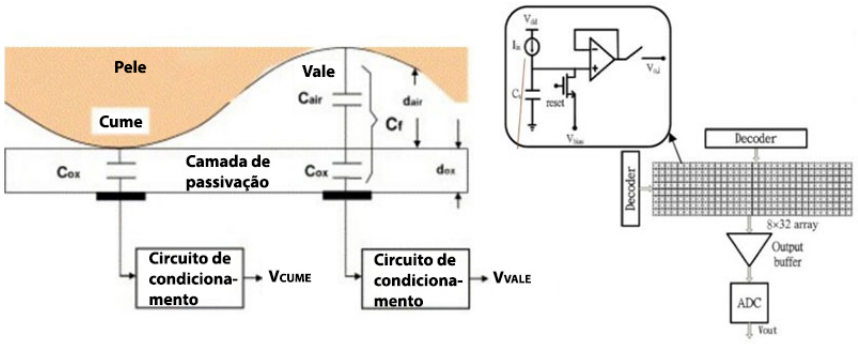
Figura 3 – Funcionamento de um sensor ótico para aquisição de uma impressão digital



Fonte: Adaptado de THAKKAR (2018).

Já os sensores capacitivos utilizam um sistema com princípio capacitivo para fazer a aquisição. São colocadas placas condutoras próximas da região de contato com o dedo e a pele age como o dielétrico entre elas. Como há cumes e vales originados das ranhuras, a capacitância formada por um cume é diferente da formada por um vale, pois a distância da pele para as placas é diferente. Desse modo, esses capacitores ao serem carregados terão tensões diferentes e isso diferencia os cumes dos vales eletricamente; e, ao se fazer um pós-processamento, é formada a imagem da impressão digital. Esse tipo de aquisição é apresentado na Figura 4. Sensores capacitivos podem ser mais compactos que sensores óticos, por isso geralmente a resolução da imagem formada por esses sensores é maior. Além disso, é mais difícil um sistema com esse tipo de sensor ser fraudado (THAKKAR, 2018), pois a pele possui uma constante dielétrica diferente da de outros materiais, e por isso métodos fradulentos como imagens impressas ou próteses não funcionam.

Figura 4 – Funcionamento de um sensor capacitivo para aquisição de uma impressão digital



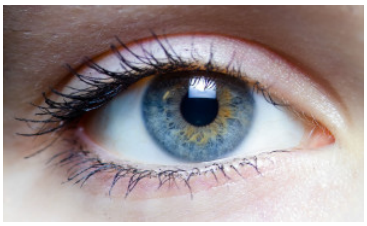
Fonte: Adaptado de THAKKAR (2018).

2.1.2 Reconhecimento de íris

A íris é uma região do olho desenvolvida nos estágios iniciais da vida e, quando totalmente formada, sua textura permanece constante para o resto da vida (THAKKAR, 2017). É a região do olho responsável pela cor dos olhos, e foi descoberto que o padrão da sua textura é único para cada indivíduo, mesmo em gêmeos idênticos, portanto pode ser utilizado para identificar um indivíduo (SHARKAS, 2016).

A textura da íris é um padrão visível que é o conjunto de diferentes formas, como ligamentos em arco, criptas, cumes e bordas em zigue-zague, como mostrado na Figura 5. Esse padrão de formas é diferente em cada pessoa, e foi descoberto que não é relacionado ao código genético dos indivíduos, mas sim a fatores aleatórios durante a formação do feto, por isso os padrões das duas íris do mesmo indivíduo são diferentes, assim como os de gêmeos idênticos (DAUGMAN e DOWNING, 2001). Desse modo, pode-se afirmar que o reconhecimento de íris é um método de biometria seguro.

Figura 5 – Exemplo de íris

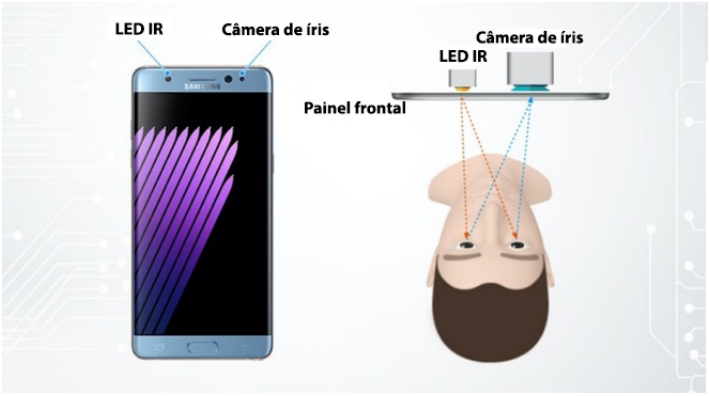


Fonte: TRADER (2015).

Apesar de ser um dos melhores métodos em relação à unicidade e segurança, em questão de usabilidade não é uma tecnologia muito boa, porque é difícil fazer um reconhecimento de íris de uma distância muito grande. Além disso, o usuário tem que ficar parado e olhando fixamente para a câmera com o ângulo correto, o que não é tão desejável em um sistema de autenticação, que requer simplicididade. Para um pequeno número de usuários isso não é tão problemático, mas para um grande número de pessoas não é a melhor opção.

Para fazer a aquisição da imagem utiliza-se uma câmera digital em conjunto com uma iluminação infravermelha para reduzir a reflexão da luz da córnea e conseguir mais detalhes da íris que não são percebidos com a luz natural. A Figura 6 mostra um exemplo de sensor de íris de um *smartphone* moderno.

Figura 6 – Exemplo de sensor de reconhecimento de íris de um smartphone moderno



Fonte: Adaptado de BUNTON (2016).

2.1.3 Reconhecimento vascular

É um dos métodos mais seguros, se não for o mais seguro. Consiste em capturar uma imagem do padrão do conjunto de veias da mão do indivíduo e codificar isso em uma informação para o sistema de autenticação. Um exemplo desse padrão pode ser visualizado na Figura 7. Ao se iluminar a palma da mão do indivíduo com uma luz quase infravermelha, o sangue com hemoglobinas desoxigenadas reflete uma sombra clara que é usada para demarcar os vasos sanguíneos (THAKKAR, 2017). De acordo com (GUO et al., 2010), a faixa de comprimentos de luz ideais para iluminar a mão e extrair essa característica é de 700 a 1000 nm.

Figura 7 – Exemplo de padrão de veias capturado por um sensor de imagem vascular

Fonte: Adaptado de RAUT et al. (2017).

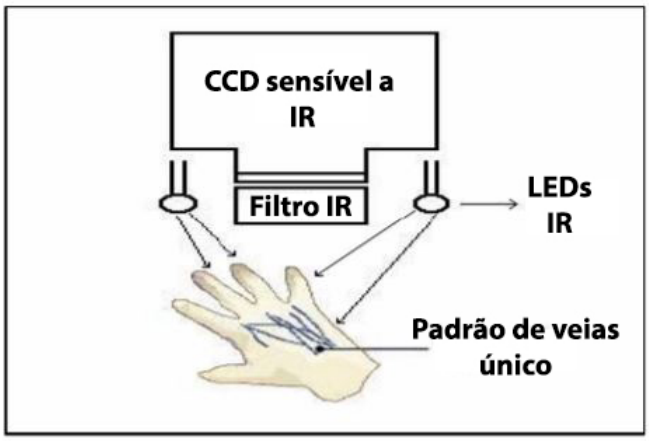
A segurança desse método é muito alta porque a estrutura dos vasos sanguíneos é formada durante a gestação e permanece constante para o resto da vida (RAUT et al., 2017). Ainda, Jiang et al. (2009) observaram que o padrão da estrutura de vasos sanguíneos é único em cada indivíduo, mesmo em gêmeos idênticos. Em comparação com os outros tipos de biometria, essa característica está escondida embaixo da pele, portanto não pode ser capturada nem replicada facilmente, e por ser baseado em uma característica do sistema circulatório, é necessário uma pessoa viva para usar o sistema e, por esses motivos, ela é uma tecnologia muito difícil de ser fraudada.

Por ser uma tecnologia relativamente nova, o custo é maior que o dos outros sistemas de biometria. Pode-se estimar que a aceitação dos usuários deve ser alta

porque a utilização é semelhante à de um leitor de impressões digitais, com a diferença de que não é necessário contato com o *hardware* de leitura.

A forma de aquisição dessa característica é bem parecida com a do reconhecimento de íris. É utilizado um sistema de iluminação (geralmente LEDS) com comprimento de onda quase infravermelho para iluminar a palma da mão e um sensor de imagem do tipo CCD para capturar a imagem. A Figura 8 mostra o funcionamento básico dessa aquisição e a Figura 9 apresenta um sensor vascular comercial.

Figura 8 – Funcionamento de um sensor vascular



Fonte: Adaptado de SARKAR et al. (2010).

Figura 9 – Sensor vascular comercial

Fonte: IDENTYTECH SOLUTIONS (2016).

2.1.4 Reconhecimento de voz

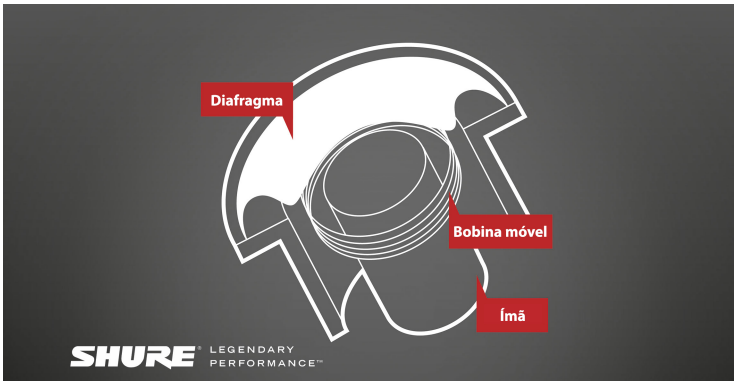
Reconhecimento de voz ou reconhecimento do locutor é a tecnologia que verifica a identidade de um indivíduo baseado em características extraídas da sua voz (THAKKAR, 2017). Essas características são baseadas no processo de formação da voz e não no som produzido ou nas palavras ditas. É muito usado em aplicações forenses, em que o reconhecimento de voz é utilizado para criar uma lista inicial de suspeitos.

Essa tecnologia é capaz de identificar precisamente um indivíduo por características de sua voz com menos de 1% de taxa de erro. Além disso, essa taxa ainda é menor quando uma frase pré-determinada é utilizada na autenticação (THAKKAR, 2017). Muitos fatores podem afetar o desempenho do sistema, como ruído do ambiente, o tempo de voz gravado, a qualidade do sensor utilizado e o uso de uma frase pré-determinada ou não. Outro problema importante é que a voz sofre alterações com o tempo e também pode sofrer alterações temporárias, como problemas de saúde ou o estado emocional do indivíduo.

A taxa de aceitação dos usuários é alta porque é um sistema simples de usar, não invasivo e sem necessidade de contato com o *hardware*. O custo é extremamente baixo em relação aos outros sistemas, porque só é necessário um microfone para se fazer a aquisição.

Existem diversos tipos de sensores utilizados para capturar o som em microfones. Dentre os tipos de microfone mais utilizados estão o dinâmico, o condensador e o de eletreto. O dinâmico possui uma bobina acoplada ao diafragma que envolve um ímã permanente. Desse modo, quando o diafragma é movimentado pelo som, a bobina se move pelo campo magnético e produz um sinal proporcional ao som captado. A Figura 10 apresenta a estrutura interna de um microfone dinâmico.

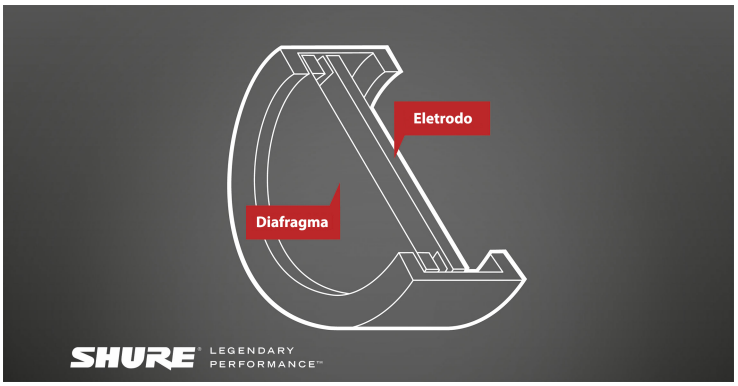
Figura 10 – Estrutura de um microfone dinâmico



Fonte: Adaptado de SHURE (2016).

O microfone condensador consiste em um diafragma carregado eletricamente separado de uma placa fixa também carregada que formam um capacitor variável. Pode-se então carregar esse capacitor em determinados instantes para se gerar uma tensão proporcional à intensidade do som naqueles instantes. A Figura 11 apresenta a estrutura interna de um microfone condensador. Esse microfone, diferentemente do dinâmico, precisa ser alimentado externamente para funcionar.

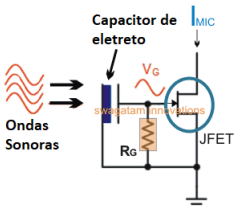
Figura 11 – Estrutura de um microfone condensador



Fonte: Adaptado de SHURE (2016).

O microfone de eletreto funciona baseado em materiais eletretos. Um material eletreto é aquele que possui momentos de dipolo elétrico permanentes, como se fosse um capacitor permanentemente carregado. Assim, ao se acoplar uma placa desse material a um diafragma, a tensão entre as suas faces irá variar conforme o som produzido. Como essa tensão é muito baixa, geralmente se utiliza um transistor de efeito de campo (FET) para amplificar o sinal, logo esse tipo de microfone também precisa ser alimentado. A Figura 12 apresenta a estrutura interna de um microfone de eletreto.

Figura 12 – Estrutura de um microfone de eletreto



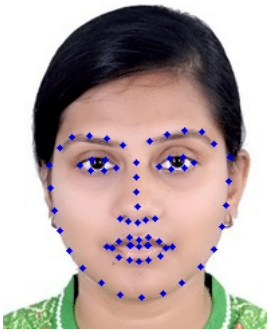
Fonte: Adaptado de DAS (2019).

2.1.5 Reconhecimento facial

É provavelmente o método mais intuitivo de se utilizar, de modo que esse tipo de reconhecimento pode ser feito mesmo sem que o sujeito que está sendo reconhecido perceba. É feito ao se analisar características comuns ao rosto de todas as pessoas (THAKKAR, 2017), mas que são diferentes em cada uma delas. Alguns exemplos dessas características são a distância entre os olhos, tamanho do nariz, distância entre o nariz e a boca e grossura das sobrancelhas, e mesmo que algumas delas sejam semelhantes em mais de uma pessoa, quando analisadas em conjunto tornam o rosto de uma pessoa único. Uma das partes do processo de reconhecimento facial é representado na Figura 13.

Esse método possui um alto grau de aceitação dos usuários, pois não é necessário contato com o *hardware*, é simples de ser utilizado e pode até ser feito de forma imperceptível.

Figura 13 – Representação do processo de reconhecimento facial



Fonte: JAIN (2016).

Existem diversos fatores que podem afetar a leitura que afetam o desempenho do sistema, como a iluminação, a resolução das imagens, objetos que cubram parte do rosto do usuário, como óculos escuros, e as mudanças no rosto devido ao envelhecimento (THAKKAR, 2017). Por esse motivo, há um grande número de pesquisas sobre esse tipo de biometria para reduzir a influência desses fatores. Mesmo assim, um teste feito em 2017 pelo US National Institute of Standards and Technology (NIST) com o melhor sistema de reconhecimento facial na época, da Panasonic, obteve uma taxa de falsa rejeição (FRR) de 1,1% (PANASONIC, 2018). Isso quer dizer que sistemas de reconhecimento facial possuem desempenho maior que seguranças fazendo a comparação manualmente (THAKKAR, 2017).

Para se adquirir as imagens se utiliza uma câmera digital, como mostrada na Figura 14, desse modo os tipos de sensores utilizados nesse sistema são os mesmos das câmeras. Esse sensores são óticos e podem ser do tipo CCD ou CMOS. O funcionamento desses sensores será explicado nas seções seguintes.

Por ser um tipo de biometria com muitos estudos sendo feitos e muitos já feitos e por ser possível implementar um sistema com essa tecnologia somente com um computador pessoal com uma webcam, o reconhecimento facial foi o tipo escolhido para ser utilizado neste trabalho.

Figura 14 – Exemplo de câmera que pode ser utilizada para reconhecimento facial

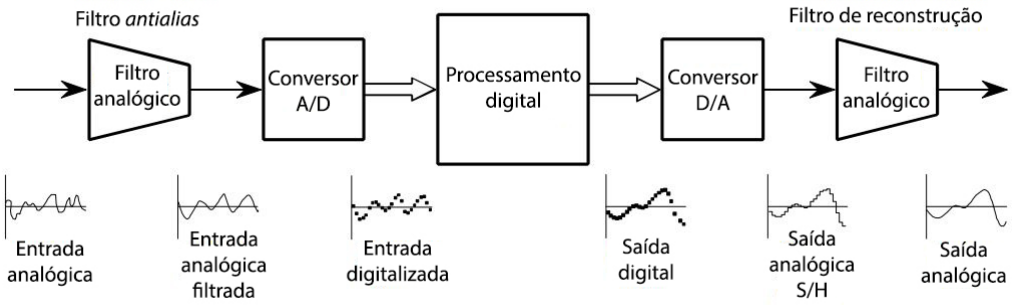
Fonte: MIRANDA (2017).

2.2 PROCESSAMENTO DIGITAL DE SINAIS

Processamento digital de sinais (PDS) é uma área da Engenharia que se preocupa em transformar sinais em informação no domínio digital e utilizar essa informação para realizar um processo, como realizar o controle de uma planta, por exemplo. Uma das grandes vantagens de se fazer o processamento de sinais no domínio digital é que os circuitos digitais podem ser programados, portanto, um mesmo *hardware* pode realizar diferentes tipos de processamento enquanto os circuitos analógicos são muito específicos para um único tipo de processamento. Outra grande vantagem é que os sinais no domínio digital possuem imunidade a ruído, desse modo se pode ter um grau de repetitividade maior dos resultados.

Um processamento de um sinal no domínio digital pode ser divido essencialmente em quatro etapas: aquisição do sinal, conversão analógico-digital, processamento digital e conversão digital-analógica. Essas etapas são apresentadas no diagrama de blocos da Figura 15.

Figura 15 – Diagrama de blocos das etapas de PDS

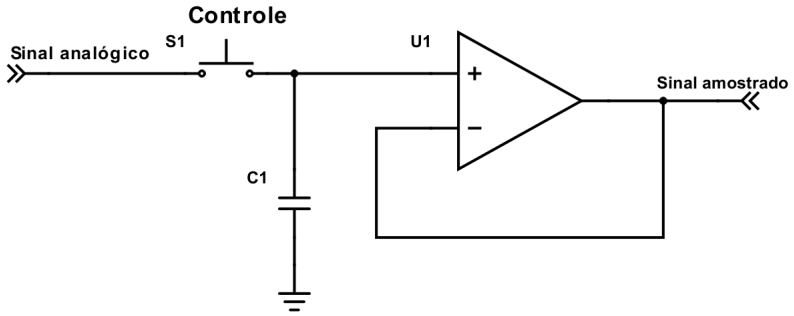


Fonte: Adaptado de SMITH (1999).

A etapa de aquisição do sinal consiste em utilizar um sensor em conjunto com um circuito de condicionamento para se adquirir o sinal no domínio analógico. A função do circuito de condicionamento é adequar o sinal às limitações do circuito da próxima etapa, que é o conversor analógico-digital (A/D).

A etapa de conversão A/D consiste em converter o sinal analógico em um sinal digital. Essa etapa pode ser dividida em três subetapas: amostragem, quantização e codificação. A amostragem consiste em capturar o valor do sinal em determinados intervalos de tempo igualmente separados. Isso é feito ao se utilizar um circuito chamado *sample and hold*, que é mostrado na Figura 16.

Figura 16 – Circuito de sample and hold



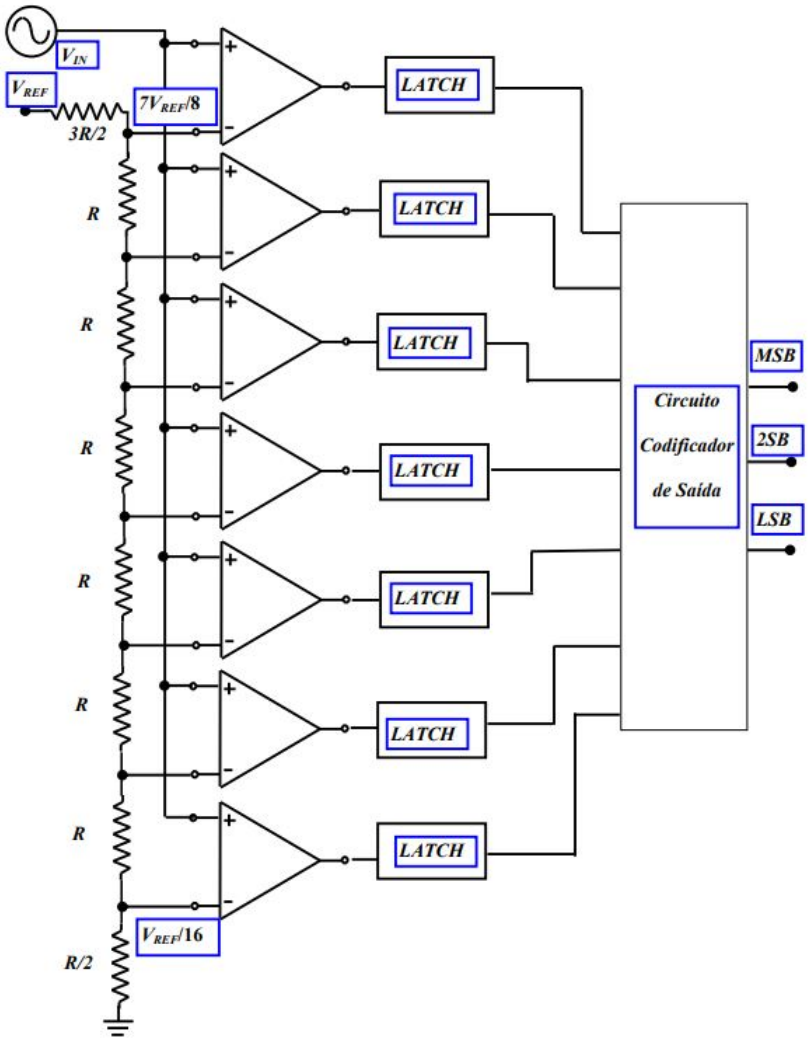
Fonte: Autoria própria.

O sinal de controle define os instantes de tempo em que o sinal analógico é amostrado. O intervalo entre esse instantes é chamado de taxa de amostragem e ela

é muito importante, pois define a frequência máxima de um sinal que se pode amostrar sem a perda de informações. A frequência de amostragem é o inverso da taxa de amostragem e, pelo Teorema de Nyquist, para que o sinal possa ser reconstruído sem perda de informações, a frequência mínima de amostragem tem que ser o dobro da frequência máxima do sinal que se quer amostrar. Na prática, para se ter uma boa representação do sinal é necessário que a frequência de amostragem seja pelo menos 10 vezes a frequência máxima do sinal. Como a frequência máxima do sinal é limitada pela frequência de amostragem, então geralmente na etapa de condicionamento é utilizado um filtro analógico passa-baixas para limitar a banda do sinal.

O próximo passo é a etapa de quantização, em que o sinal amostrado é convertido para um sinal com amplitudes discretas. Essa etapa é feita utilizando conversores A/D propriamente ditos. Alguns desses conversores mais conhecidos são o conversor flash, o conversor por aproximações sucessivas e o conversor contador. O conversor flash consiste em vários circuitos comparadores (um para cada bit de resolução) em que cada amostra do sinal de entrada é comparada com os vários níveis de quantização. A saída desses comparadores passa por um circuito lógico que gera a amostra quantizada. O conversor flash pode ser visto na Figura 17.

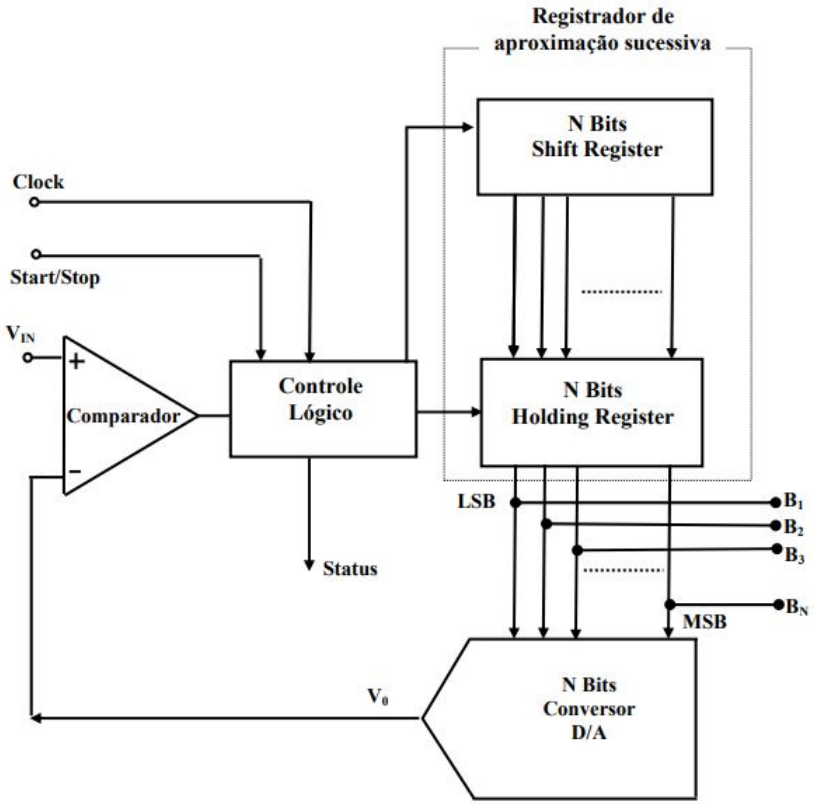
Figura 17 – Conversor flash



Fonte: FERREIRA (2018).

O conversor por aproximações sucessivas utiliza um conversor D/A para fazer a conversão. Inicialmente o bit mais significativo é setado e a palavra é convertida para um valor analógico. Então, um circuito comparador compara esse valor com o valor da amostra. Se a amostra for maior que esse valor, o bit se mantém setado, se não o bit é zerado. Depois, o próximo bit mais significativo é setado e esse procedimento é feito novamente, até que todos os bits tenham sido testados. A palavra final é o valor da amostra no domínio digital. A Figura 18 apresenta o diagrama de um conversor A/D desse tipo.

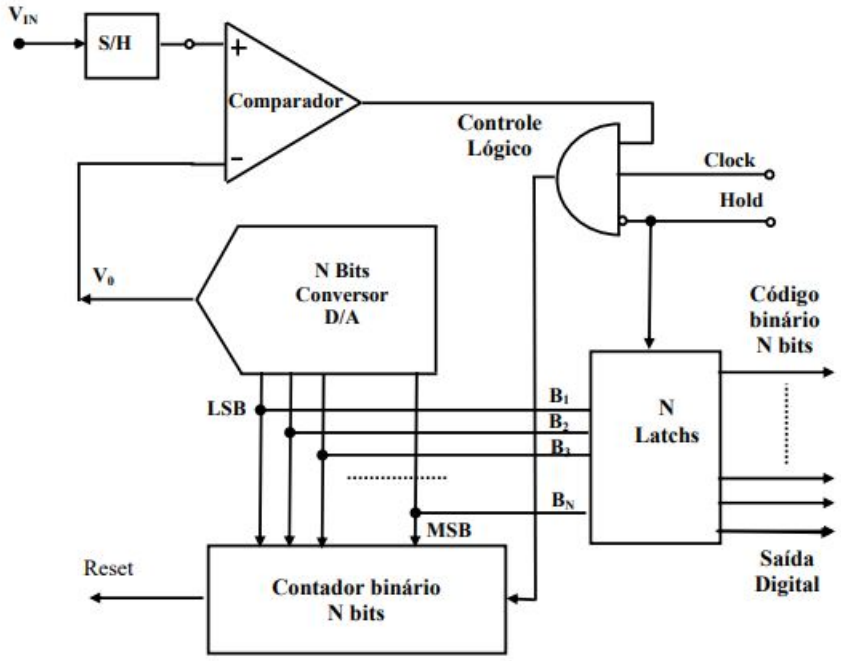
Figura 18 – Conversor por aproximações sucessivas



Fonte: FERREIRA (2018).

O último tipo de conversor é o conversor contador. Ele utiliza um circuito lógico contador que conta de 0 até o valor máximo que se pode obter com a quantidade de bits do conversor. O valor atual do contador é convertido para analógico e comparado com o valor da amostra. Caso o valor dela seja menor, ele é incrementado. Quando o valor da amostra for maior que o valor do contador, esse valor é o valor convertido da amostra. A Figura 19 apresenta a estrutura de um conversor contador.

Figura 19 – Conversor do tipo contador



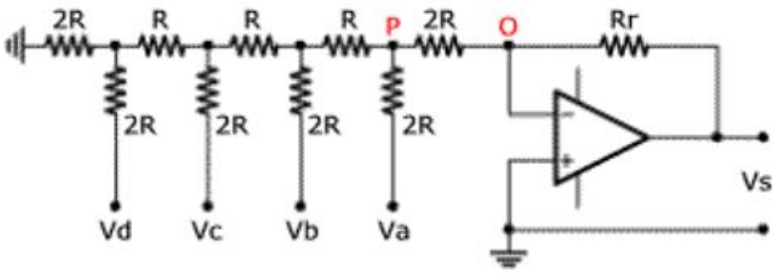
Fonte: FERREIRA (2018).

A última etapa da conversão é a codificação. Essa etapa simplesmente consiste em converter o sinal amostrado e quantizado que foi obtido na etapa anterior em um número binário.

Com o sinal convertido para o domínio digital a próxima etapa é o processamento digital propriamente dito. Ele é feito utilizando um processador digital, que pode ser um microprocessador ou um microcontrolador. Algumas técnicas de processamento digital serão apresentadas posteriomente.

A última etapa de um PDS é converter o sinal resultante do processamento para o domínio analógico. Isso é feito utilizando um circuito chamado de conversor digital-analógico (D/A). Há vários tipos de conversores D/A e um bem simples é o conversor R2R. Ele possui esse nome porque ele só possui 2 valores de resistência em seu circuito, e um é o dobro do outro. Ele é mostrado na Figura 20.

Figura 20 – Conversor do tipo R2R



Fonte: SHIN (2012).

Deve-se notar que, por ser originado de um sinal digital, que só possui valores binários, um sinal analógico convertido por um conversor D/A possuirá componentes de frequência indesejados em sua banda, originados de descontinuidades. Por isso, geralmente se utiliza um filtro passa-baixas no sinal analógico convertido para eliminar essas frequências.

2.2.1 Imagens digitais

A aquisição de uma imagem é feita utilizando uma câmera digital, que é composta por uma lente, um obturador, um diafragma e o sensor. O elemento mais importante é o sensor, que captura a luz e transforma a informação da intensidade de luz em um sinal elétrico. Os dois tipos existentes de sensores são o CMOS (*complementary metal oxide semiconductor* - do inglês, semicondutor de óxido metálico complementar) e o CCD (*charged coupled device* - do inglês, dispositivo de carga acoplada).

Uma imagem digital é formada por um conjunto de unidades fundamentais chamadas de pixels. O sensor possui um conjunto de transdutores e cada um é responsável pela captura de um pixel.

No caso do CCD, os pixels são adquiridos por capacitores MOS. Quando a luz incide sobre esses capacitores eles se carregam de acordo com a intensidade de luz naquela região. Então, as cargas são deslocadas de forma ordenada de um capacitor para o próximo até elas cheguem, uma por uma, a um canto da imagem onde é feita a aquisição dessas cargas.

No caso do CMOS, cada pixel é capturado por um fotodiodo, que é um dispo-

sitivo que gera uma corrente pela absorção de fótons. O sinal de cada fotodiodo é amplificado por um transistor CMOS e cada valor amplificado é lido de forma independente.

Após a conversão dos sinais para a forma digital, uma imagem digital é formada por uma matriz de pixels, da forma $M_{h \times w \times c}$ em que h é o número de pixels no eixo vertical, w é o número de pixels no eixo horizontal e c é o número de canais de cores. Os valores de h e w definem a resolução da imagem. O valor de cada pixel determina a intensidade daquele pixel, no respectivo canal, para o sistema de cores RGB. Nesse sistema, o valor de cada pixel é representado por um byte sem sinal, ou seja, seu valor pode variar de 0 a 255, sendo 0 correspondente à tonalidade mínima (preto) e 255 à tonalidade máxima (branco) do respectivo canal. O canal 1 corresponde a contribuição da cor vermelha para o pixel, o canal 2 corresponde a contribuição da cor verde e o canal 3 corresponde a contribuição da cor azul.

Como imagens digitais são representadas na forma matricial, se pode utilizar operações matemáticas e matriciais para fazer transformações nela. Uma operação muito comum em processamento de imagens é a transformação da imagem para tons de cinza, em que os 3 canais de cores são substituídos por um único, e a intensidade de cada pixel é definida pela média aritmética da intensidade daquele pixel em cada canal. A Figura 21 apresenta uma imagem e sua representação em tons de cinza.

Figura 21 – Imagem original e em tons de cinza

Fonte: Autoria própria.

Outro tipo de operação comum são as transformações afins, que são transformações que preservam as linhas paralelas nas imagens, como rotação, redimensionamento e translação. Elas podem ser feitas ao se multiplicar a matriz da imagem por uma matriz de transformações afins. A Figura 22 apresenta uma imagem e o resultado

de algumas transformações afins.

Figura 22 – Imagem original e resultado de algumas transformações afins

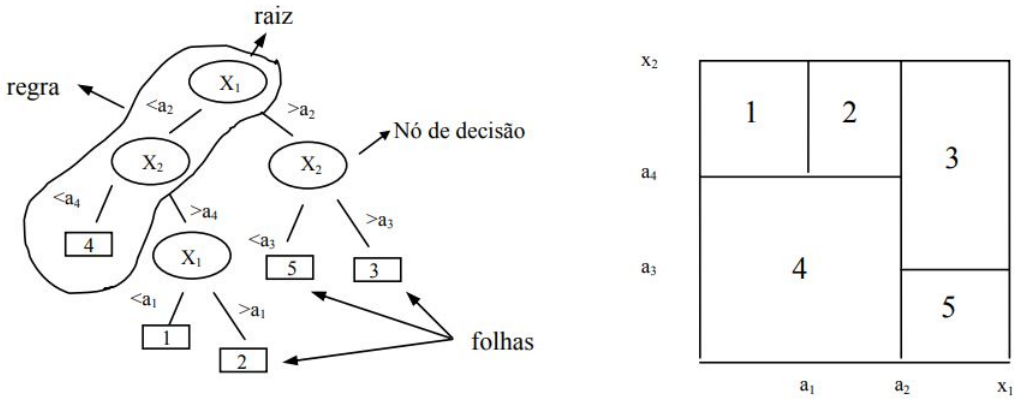
Fonte: Autoria própria.

2.3 ÁRVORES DE DECISÃO E DE REGRESSÃO

2.3.1 Definição

Árvores de decisão são modelos estatísticos usados para classificação e previsão de dados. Elas utilizam um treinamento supervisionado, ou seja, sua construção é feita utilizando um conjunto prévio de entradas com suas respectivas saídas, sendo as saídas chamadas de classes. Cada entrada, chamada de objeto, possui um conjunto de atributos e esses atributos devem estar presentes em todos os objetos (ZUBEN e ATTUX, 2010). Em cada nó de decisão, um atributo é testado de acordo com uma regra e o resultado desse teste pode levar a uma folha, que é uma saída da árvore, ou a um novo nó de decisão. As árvores de decisão separam as classes por uma divisão do espaço definido pelos atributos em espaços menores e a cada um desses espaços é atribuída uma classe (SILVA, 2005). Um exemplo de árvore de decisão e sua representação no espaço é mostrado na Figura 23.

Figura 23 – Exemplo de árvore de decisão e sua representação no espaço



Fonte: GAMA (2002).

O critério utilizado para as divisões é a influência do atributo para a classificação, ou seja, os atributos que mais separam as classes são testados primeiro, e a cada nova divisão a influência do próximo atributo a ser testado é menor que a do anterior.

2.3.2 Classificação e treinamento

Seja um conjunto de dados de treinamento $(x_i,\ y_i) \in \mathbb{R}^p \times \{1, \cdots, c\}$, onde i = 1, $\cdots$, n, *n* sendo o número de dados de treinamento, *p* o número de atributos dos dados e *c* o número de classes. A árvore de classificação *T* define as regiões retangulares $R_1, \cdots, R_m$ que particionam o espaço das preditoras (atributos) $\mathbb{R}^p$. Seja $n_j = \sum_{i=1}^{n} I_{R_j}(x_i)$ o número de preditoras dos dados de treinamento pertencentes à região R_j, para j = 1, $\cdots$, m. A fração de exemplos da classe *k* na região R_j é igual a:

$$\hat{p}_k(R_j) = \frac{1}{n_j} \sum_{\{i:x_i \in R_j\}} I_{\{k\}}(y_i),\ k = 1, \cdots, c$$

A classe predita para a região R_j é $c_j = arg\ max_k\ \hat{p}_k(R_j)$, que é a proporção de exemplos na região R_j que são da classe predominante. O classificador então pode ser escrito como:

$$\hat{\varphi}(x) = \sum_{j=1}^{m} c_j I_{R_j}(x)$$

Existem vários algoritmos para o treinamento da ávore de decisão, entre eles o *Classification And Regression Trees* (CART - do inglês, árvores de classificação e regressão), proposto por Breiman et al. em 1984. Nesse algoritmo é feita uma divisão inicial na raiz em dois nós no próximo nível da árvore. Então, é escolhido o atributo para essa divisão que produz a maior queda no erro de classificação. Esse procedimento continua sendo feito para os nós criados até que se chegue à condição de parada, que pode ser um certo número de iterações ou quando só houver membros de uma classe em cada região formada (MARQUES FILHO e LOPES, 2017).

Matematicamente, o algoritmo CART é iniciado na raíz da árvore, e cria duas regiões R_1 e R_2, tais que:

$$R_1 = \{X \in \mathbb{R}^p : X_j \leq t\} \text{ e } R_2 = \{X \in \mathbb{R}^p : X_j > t\}$$

Com os dados de treinamento, se faz a divisão escolhendo $\hat{j}$ e $\hat{t}$, tais que:

$$(\hat{j}, \hat{t}) = \underset{(j,t)}{arg\ min}\ ((1 - \hat{p}_{c_1}(R_1)) + (1 - \hat{p}_{c_2}(R_2)))$$

$$c_1 = \underset{k=1,\cdots,c}{arg\ max}\ \hat{p}_k(R_1)$$

$$c_2 = \underset{k=1,\cdots,c}{arg\ max}\ \hat{p}_k(R_2)$$

Em que c_1 é a classe dominante do retângulo R_1 e c_2 é a classe dominante do retângulo R_2. Esse processo é repetido para os novos nós criados até que seja satisfeito o critério de parada. Após o fim desse processo, é feito um processo de podagem, em que são eliminadas folhas que não influenciam tanto na decisão final para simplificar a árvore e evitar um problema chamado *overfitting*, que é o que acontece quando o modelo treinado possui uma taxa de erro baixa nos dados de treinamento, mas quando dados nunca vistos antes são testados o erro é maior. Para uma árvore de classificação *T*, sendo $\mid T \mid$ o número de suas folhas e para $\alpha \geq 0$, define-se:

$$C_\alpha(T) = \sum_{j=1}^{|T|}(1 - \hat{p}_{c_j}(R_j)) + \alpha \mid T \mid)$$

O processo de podagem escolhe a árvore *T* que minimiza $C_\alpha(T)$, escolhendo α por validação cruzada, que é quando o conjunto de dados de treinamento é dividido em

um ou mais subconjuntos para treinamento e para validação e o valor de α é variado até se achar um valor ótimo.

No caso de um problema de regressão, são utilizadas árvores de regressão e a variável de resposta é um valor numérico, pois a classificação não é discreta. As folhas das árvores de regressão não classificam as entradas, mas possuem funções de regressão que utilizam as entradas como variáveis e geram uma saída numérica. A construção da árvore de regressão é feita de maneira análoga à da árvore de classificação (MARQUES FILHO e LOPES, 2017), porém a maior diferença é que uma função de perda quadrática é utilizada para definir o atributo do nó e o limiar, de modo que:

$$(\hat{j}, \hat{t}) = \underset{(j,t)}{arg\ min}(\sum_{i:x_i \in R_1} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2} (y_i - \hat{y}_{R_2})^2)$$

Em que $\hat{y}_{R_1}$ e $\hat{y}_{R_2}$ são as médias das saídas dos dados de treinamento que pertencem às regiões R_1 e R_2, respectivamente. Então, para cada região R_j correspondente a uma folha, o CART vincula uma constante c_j que é a média das saídas dos dados de treinamento que pertencem à região R_j.

$$c_j = \frac{1}{n_j} \sum_{\{i:x_i \in R_j\}}^{m} y_i$$

A estimativa para a função de regressão, é então:

$$\hat{\psi}(x) = \sum_{j=1}^{m} c_j I_{R_j}(x)$$

2.4 MÁQUINAS DE VETORES DE SUPORTE

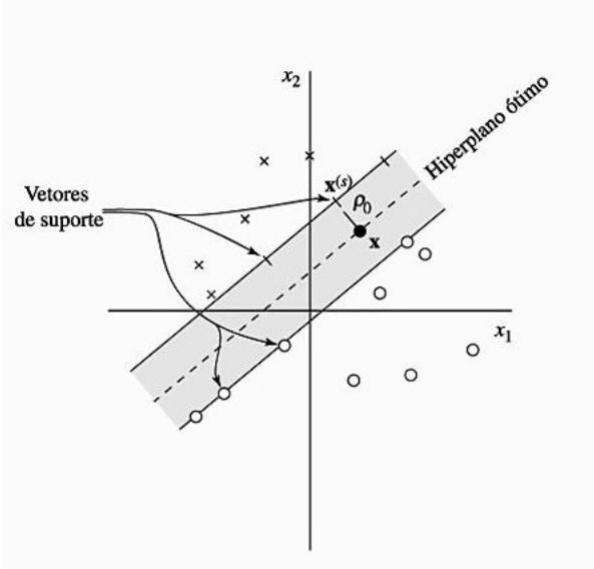
2.4.1 Definição

Uma máquina de vetores de suporte (SVM - do inglês, support vector machine) é um processador linear, que tenta encontrar um hiperplano que seja uma superfície de decisão ótima para a separação entre um conjunto de dados de duas classes diferentes

(HAYKIN, 2001). Por esse motivo, ela pode fornecer um bom desempenho em um problema de classificação de padrões e regressão linear.

A função de decisão é totalmente especificada por um subconjunto dos dados de treinamento, chamados de vetores de suporte. Vetores de suporte são os pontos que estão mais próximos da superfície de decisão e, por isso, são os dados mais difíceis de se classificar (BERWICK, 2008). A Figura 24 mostra um exemplo de dados linearmente separáveis com o hiperplano ótimo entre eles e os vetores de suporte associados.

Figura 24 – Exemplo de hiperplano ótimo e de vetores de suporte



Fonte: HAYKIN (2001).

Um dado de entrada é um conjunto de n-ésimas *features* $x_1, x_2, \cdots, x_n$ e um dado de saída é um valor único *y*. O objetivo do treinamento de uma SVM é encontrar um conjunto de pesos $w_1, w_2, \cdots, w_n$ que ao se fazer uma combinação linear com as *features* façam a predição da saída. A principal característica da otimização de uma máquina de vetores de suporte é que ela faz a maioria dos pesos terem valor nulo, de modo que apenas as *features* importantes na decisão, que são os vetores de suporte, possuam peso correspondente com valor não-nulo (BERWICK, 2008).

2.4.2 Classificação e treinamento

Considera-se uma amostra de treinamento $\{(x_i, d_i)\}_{i=1}^{N}$, onde x_i é o padrão de entrada pra o i-ésimo exemplo e d_i é a resposta desejada respectiva. Ao se assumir que o padrão representado pelo subconjunto d_i = +1 e o padrão representado pelo subconjunto $d_i = -1$ são linearmente separáveis, então a equação de um hiperplano que separa esses subconjuntos é:

$$w^T x + b = 0$$

Onde x é um vetor de entrada, w é um vetor de pesos e b é um bias. Assim, se pode escrever:

$$w^T x_i + b > 0 \text{ para } d_i = +1$$

$$w^T x_i + b < 0 \text{ para } d_i = -1$$

Para um vetor de peso w e um *bias* b, a distância entre o hiperplano e o ponto mais próximo é dada por ρ. O objetivo é encontrar um hiperplano em que o ρ é máximo. A função abaixo fornece uma medida algébrica da distância entre o hiperplano e um vetor de entrada x.

$$g(x) = w_o^T x + b_o$$

Pode-se escrever então que os parâmetros ótimos seguem as seguintes restrições:

$$w_o^T x_i + b_o \geq 1 \text{ para } d_i = +1$$

$$w_o^T x_i + b_o \leq -1 \text{ para } d_i = -1$$

Considerando um vetor de suporte $x^{(s)}$ para o qual $d^{(s)} = +1$, então por definição:

$$g(x^{(s)}) = w_o^T x^{(s)} \mp b_o \mp 1 \text{ para } d^{(s)} = \mp 1$$

Então a distância algébrica do vetor de suporte $x^{(s)}$ até o hiperplano ótimo é:

$$r = \frac{g(x^{(s)})}{\|w_o\|} = \begin{cases} \frac{1}{\|w_o\|}, \ se \ d^{(s)} = +1 \\ -\frac{1}{\|w_o\|}, \ se \ d^{(s)} = -1 \end{cases}$$

Seja ρ o valor ótimo da margem de separação entre as duas classes que constituem o conjunto de treinamento, então:

$$\rho = 2r = \frac{2}{\|w_o\|}$$

Isso significa que maximizar a margem de separação entre as classes é o mesmo que minimizar a norma euclidiana do vetor w.

Então, dada uma amostra de treinamento $\{(x_i, d_i)\}_{i=1}^{N}$, é preciso encontrar os valores ótimos do vetor de peso w e do *bias* b, de modo que satisfaça as restrições

$$d_i(w^T x_i + b) \geq 1 \text{ para } i = 1, 2, \cdots, N$$

e o vetor de peso minimize a função de custo

$$\Phi(w) = \frac{1}{2} w^T w$$

Esse problema de otimização restrito é chamado de problema primordial (HAYKIN, 2001). Ele possui duas características: a função de custo $\Phi(w)$ é uma função convexa de w e as restrições são lineares em relação a w. Por isso, pode-se resolver esse problema utilizando o método dos multiplicadores de Lagrange. A função lagrangiana é construída da seguinte forma:

$$J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i [d_i(w^T x_i + b) - 1]$$

Onde as variáveis não-negativas α_i são chamadas de multiplicadores de Lagrange. A solução para o problema de otimização se dá no ponto de mínimo da função lagrangiana, que deve ser minimizada em relação a w e a b e maximizada em relação a α. Calculando as derivadas parcials de J em relação a w e a b e igualando os resultados a zero, tem-se:

$$\text{Condição 1:} \quad \frac{\partial J(w,b,\alpha)}{\partial w} = 0$$

$$\text{Condição 2:} \quad \frac{\partial J(w,b,\alpha)}{\partial b} = 0$$

Após a manipulação das equações acima, se obtém:

$$w = \sum_{i=1}^{N} \alpha_i d_i x_i$$

$$\sum_{i=1}^{N} \alpha_i d_i = 0$$

O vetor solução w, é definido em uma expansão que envolve os N termos de treinamento, e embora essa solução seja única devido a convexidade da função, os coeficientes de Lagrange α_i não são. Então, o meio de se encontrar os valores dos coeficientes α_i é formulando um novo problema, chamado de problema dual, que possui a mesma solução ótima do problema primordial (HAYKIN, 2001). Esse problema pode ser descrito da seguinte forma:

- Se o problema primordial tem uma solução ótima, então o problema dual também tem uma solução ótima com os mesmos valores
- Para que w_o seja uma solução ótima e α_o também seja uma solução ótima, é necessário e suficiente que w_o seja realizável no problema primordial e:

$$\Phi(w_o) = J(w_o, b_o, \alpha_o) = \min_{w} J(w, b_o, \alpha_o)$$

O desenvolvimento do problema dual não será mostrado aqui, pela complexidade matemática e por sair do foco do objetivo principal, então ao pular para a sua solução, deve-se encontrar os multiplicadores de Lagrange que maximizam a função objetivo

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j$$

sujeita as restrições

- $\sum_{i=1}^{N} \alpha_i d_i = 0$
- $\alpha_i \geq 0$ para $i = 1, 2, \cdots, N$

Assim, a função $Q(\alpha)$ que deve ser maximizada depende apenas dos padrões de entrada na forma de um conjunto de produtos escalares. Ao se determinar os multiplicadores de Lagrange ótimos, dados por $\alpha_{o,i}$, pode-se calcular o vetor peso ótimo w_o da seguinte maneira:

$$w_o = \sum_{i=1}^{N} \alpha_{o,i} d_i x_i$$

E para se calcular o *bias* ótimo, b_o, pode-se usar a equação relativa ao vetor de suporte positivo e escrever:

$$b_o = 1 - w_o^T x^{(s)} \text{ para } d^{(s)} = 1$$

Existe também o caso em que os padrões não podem ser separados por um hiperplano. Desse modo, em vez de simplesmente calcular um hiperplano que separe as classes de forma ótima, é preciso que se calcule esse hiperplano de modo que o erro de classificação seja mínimo. Se diz que a margem de separação entre as classes é suave se um ponto de dado (x_i, d_i) violar a seguinte condição:

$$d_i(w^T x_i + b) \geq +1, \ i = 1, 2, \cdots, N$$

E isso pode acontecer de duas formas:

- O ponto se encontra dentro da região de separação e no lado correspondente a sua classe.
- O ponto se encontra no lado errado da superfície de decisão.

Para esse caso de dados, é introduzido um novo conjunto de variáveis escalares não-negativas, denotadas por $\{\xi_i\}_{i=1}^{N}$, na definição do hiperplano de separação, da seguinte forma:

$$d_i(w^T x_i + b) \geq 1 - \xi_i,\ i = 1, 2, \cdots, N$$

Essas variáveis ξ_i são chamadas de variáveis soltas e medem o desvio de um ponto de dado da condição ideal de separação. Para $0 \leq \xi_i \leq 1$, o ponto se encontra dentro da região de separação e no lado correto da classe. Para $\xi_i > 1$, ele se encontra no lado errado. Analogamente ao caso em que as classes são linearmente separáveis, se pode chegar ao problema dual em que dada a amostra de treinamento $\{(x_i, d_i)\}_{i=1}^{N}$, é necessário encontrar os multiplicadores de Lagrange $\{\alpha_i\}_{i=1}^{N}$ que maximizam a função objetivo

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j$$

sujeita as restrições

- $\sum_{i=1}^{N} \alpha_i d_i = 0$
- $0 \leq \alpha_i \leq C$ para $i = 1, 2, \cdots, N$

onde C é um parâmetro positivo especificado pelo usuário.

A função objetivo $Q(\alpha)$ a ser maximizada é a mesma em ambos os casos e a diferença entre o caso separável e o não-separável é a restrição mais rigorosa $0 \leq \alpha_i \leq C$. A solução ótima para o vetor peso w é dada por:

$$w_o = \sum_{i=1}^{N_s} \alpha_{o,i} d_i x_i$$

Onde N_s é o número de vetores de suporte. Para determinar o b_o, pode-se tomar qualquer ponto de treinamento (x_i, d_i) para o qual $0 \leq \alpha_{o,i} \leq C$ e com isso se tem que $\xi_i = 0$, e usá-lo na equação:

$$\alpha_{o,i}[d_i(w^T x_i + b) - 1] = 0$$

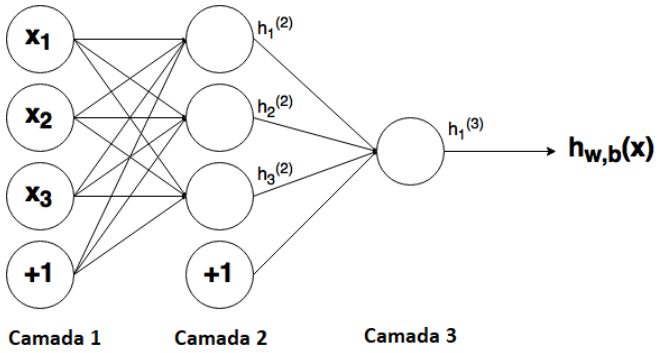
Porém, é melhor se utilizar o valor médio de todos os b_o resultantes desses dados específicos (HAYKIN, 2001).

2.5 REDES NEURAIS ARTIFICIAIS

2.5.1 Definição

Uma rede neural artificial (RNA) é um processador cuja arquitetura tenta imitar a do sistema nervoso humano, de modo que ela possa resolver problemas complexos do mesmo modo que o sistema nervoso resolve. Ela faz isso ao se utilizar um conjunto de unidades de processamento chamadas de neurônios, conectadas por ligações chamadas de sinapses, que conseguem se ”fortalecer” ou ”enfraquecer” para armazenar conhecimento em um processo chamado de aprendizagem (HAYKIN, 2001). A estrutura de uma RNA é representada na Figura 25.

Figura 25 – Estrutura de uma rede neural artificial



Fonte: Adaptado de THOMAS (2017).

No caso específico dessa imagem, a RNA possui 4 neurônios, 3 camadas, 3 entradas e 1 saída. A primeira camada, chamada de camada de entrada, possui as entradas da rede, denotadas por x_1, x_2 e x_3, e um valor constante +1, chamado de *bias*. A segunda camada possui 3 neurônios, e as saídas dos neurônios são denotadas por $h_1^{(2)}$, $h_2^{(2)}$ e $h_3^{(2)}$ e outro *bias*. Essa camada é chamada de camada escondida, pois ela não representa nenhuma entrada nem nenhuma saída da rede. A última camada, chamada de camada de saída, possui apenas um neurônio, denotado por $h_1^{(3)}$. A saída da rede é denotada por $h_{w,b}(x)$. As linhas que ligam a entrada e os neurônios de uma camada e da próxima são as sinapses. Uma RNA pode ter tantos neurônios, camadas escondidas, entradas e saídas quanto se queira.

Em uma RNA, cada sinapse possui um valor chamado de peso sináptico, que

atribui mais ou menos valor a cada saída da camada anterior para a entrada de cada neurônio da camada seguinte. Também, cada neurônio possui uma função chamada de função de ativação. Essa função calcula a saída do neurônio em função das entradas, após serem ponderadas pelos pesos sinápticos. Como pode ser percebido pela Figura 25, as saídas dos neurônios da última camada correspondem às saídas da rede.

Existem vários meios de se classificar uma RNA. Elas podem ser recorrentes ou alimentadas adiante, parcialmente ou totalmente conectadas, com treinamento supervisionado ou não-supervisionado, entre outras classificações (HAYKIN, 2001). Neste trabalho, serão utilizadas apenas RNAs alimentadas adiante, totalmente conectadas e com treinamento supervisionado. Isso quer dizer que o fluxo de propagação das informações na rede é unidirecional, ou seja, não há realimentação. Além disso, cada neurônio de uma camada está conectado a todos os neurônios da próxima camada e o processo de aprendizagem é feito com dados rotulados, ou seja, existe uma resposta esperada para cada dado de entrada do conjunto de dados de treinamento.

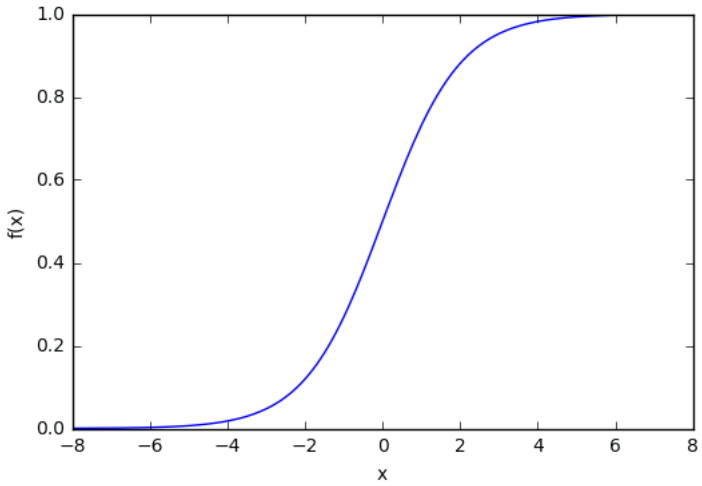
2.5.2 Modelo matemático

Inicialmente, deve-se definir uma função de ativação para os neurônios. O trabalho dessa função é de ”ativar” o neurônio quando a entrada excede certo valor (THOMAS, 2017). Algumas funções que fazem esse trabalho são a função de limiar, a função linear por partes e a função sigmoide. A função sigmoide é a mais utilizada, por ser a única das três que possui derivada para todo o domínio da entrada e isso é de grande utilidade para o processo de treinamento da rede. Ela é denotada por:

$$f(z) = \frac{1}{1+exp(-z)}$$

O gráfico dessa função é apresentado na Figura 26.

Figura 26 – Gráfico da função sigmoide



Fonte: THOMAS (2017).

Pode-se notar pelo gráfico que a partir de um certo valor a saída da função é sempre 1, desse modo o neurônio foi ”ativado”.

O valor de entrada de um neurônio é a soma de todas as entradas que chegam a esse neurônio pelas sinapses, multiplicadas pelo peso sináptico. Dado um neurônio i da camada $l+1$ com entradas $x_1, x_2, \cdots, x_n$, os pesos $w_{i1}^{(l)}, w_{i2}^{(l)}, \cdots, w_{in}^{(l)}$ da camada l, em que $1, 2, \cdots, n$ é o índice do neurônio da camada l que está ligado ao neurônio i da camada $l+1$, e $b_i^{(l)}$ o peso do *bias* da camada l que está ligado a esse mesmo neurônio, então a entrada do neurônio é dada por:

$$z_i^{(l+1)} = w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{in}x_n + b_i^{(l)} = \sum_{j=1}^{n} w_{ij}^{(l)} x_j + b_i^{(l)}$$

O valor da saída desse neurônio, denotada por $h_i^{(l+1)}$, é dada então por:

$$h_i^{(l+1)} = f(z_i^{(l+1)})$$

As entradas x_j correspondem às saídas dos neurônios da camada anterior $h_j^{(l)}$.

Generalizando, seja a matriz $Z^{(l+1)}$ a matriz coluna formada pelas entradas z_i dos neurônios da camada $l+1$, a matriz $W^{(l)}$ a matriz formada pelos pesos w_{ij} da camada l, a matriz $B^{(l)}$ a matriz coluna formada pelos pesos do *bias* da camada l e a matriz $H^{(l)}$ a matriz coluna formada pelas saídas $h_i^{(l)}$ da camada l, então:

$$Z^{(l+1)} = W^{(l)} \times H^{(l)} + B^{(l)}$$

$$H^{(l+1)} = f(Z^{(l+1)})$$

Essas duas funções mais a função de ativação descrevem todas as saídas de todos os neurônios da rede. Além disso, deve-se notar que, seja a matriz Y a matriz coluna formada por todas as saídas da rede e a matriz X a matriz coluna formada por todas as entradas da rede, então:

$$H^{(0)} = X$$

$$H^{(N)} = Y$$

com N sendo o número de camadas da rede.

2.5.3 Treinamento

O treinamento consiste em encontrar os valores dos pesos W e B de cada camada. Para isso utiliza-se um conjunto de dados de treinamento, que são um conjunto de amostras de pares entrada-saída, para que os pesos da rede sejam ajustados de modo que as entradas gerem as saídas correspondentes.

Uma técnica de treinamento muito utilizada é conhecida como gradiente descendente. Nela, se utiliza uma função de custo em função dos pesos e eles são ajustados de acordo com o gradiente dessa função, de modo que se encontre o ponto onde o custo é mínimo, que é o ponto em que o gradiente é nulo ou pelo menos está abaixo de um limiar escolhido, da seguinte forma:

$$W_{novo} = W_{antigo} - \alpha \bigtriangledown Erro$$

$$B_{novo} = B_{antigo} - \alpha \bigtriangledown Erro$$

Onde α é uma constante, chamada de taxa de aprendizado, definida pelo projetista da rede.

Seja $X^{(k)}$ o vetor de entrada da amostra k e $Y^{(k)}$ o vetor de saída esperado, então se denota a função:

$$J(W, B, X^{(k)}, Y^{(k)}) = \frac{1}{2}\|Y^{(k)} - H^{(N)}(X^{(k)})\|^2$$

onde $H^{(N)}$ é a saída estimada da rede para a entrada $X^{(k)}$ usando o conjunto de pesos W e B. Essa é o valor da função de custo de um único par de amostras. Para se calcular o erro em relação às m amostras, se faz uma média aritmética dos erros, logo:

$$J(W, B) = \frac{1}{m}\sum_{k=1}^{m}\frac{1}{2}\|Y^{(k)} - h^{(n)}(x^{(k)})\|^2$$

$$= \frac{1}{m}\sum_{k=1}^{m} J(W, B, X^{(k)}, Y^{(k)})$$

Desse modo:

$$W^{(l)}_{novo} = W^{(l)}_{antigo} - \alpha\left[\frac{1}{m}\sum_{k=1}^{m}\frac{\partial}{\partial W^{(l)}}J(W, B, X^{(k)}, Y^{(k)})\right]$$

$$B^{(l)}_{novo} = B^{(l)}_{antigo} - \alpha\left[\frac{1}{m}\sum_{k=1}^{m}\frac{\partial}{\partial B^{(l)}}J(W, B, X^{(k)}, Y^{(k)})\right]$$

Os valores de W e B são calculados repetidamente até que a função de custo satisfaça um limiar ou que um certo número de iterações seja antigido.

2.6 REDES NEURAIS CONVOLUCIONAIS

Redes neurais convolucionais (RNC) são um tipo de rede neural projetado especificamente para reconhecer dados bidimensionais, como imagens, com um alto grau de invariância a translação, escalamento, rotação e outras formas de distorção (HAYKIN, 2001).

O grande problema de se utilizar RNAs convencionais para se trabalhar com imagens é que os pixels devem ser ordenados para serem utilizados como entrada. Isso significa que um simples deslocamento da imagem faria com que a rede entendesse que ela fosse uma imagem totalmente diferente, porque os pixels estariam deslocados de posição e os seus valores seriam inseridos de forma errada na rede. As RNCs resolvem esse problema ao se fazer uma etapa de convoluções na imagem para se extrair *features* dela, de modo que, uma vez extraídos, suas posições exatas

se tornam irrelevantes, desde que suas posições relativas se mantenham a mesma em relação às outras (HAYKIN, 2001).

Essa etapa de convoluções é dividida em camadas, em que cada camada possui inúmeros filtros de pequeno tamanho (*kernels*) em que cada um realiza uma convolução para extrair uma *feature* diferente. Os valores desses filtros são determinados no treinamento da rede, para que sejam encontradas as principais *features* que influenciam no resultado final da rede.

Cada saída de cada filtro é passada por uma função de ativação, como a sigmoide, ou no caso de RNCs, é mais comum se utilizar uma função conhecida por ReLU (*rectified linear unit* - - do inglês, unidade linear retificada). Essa função é dada por:

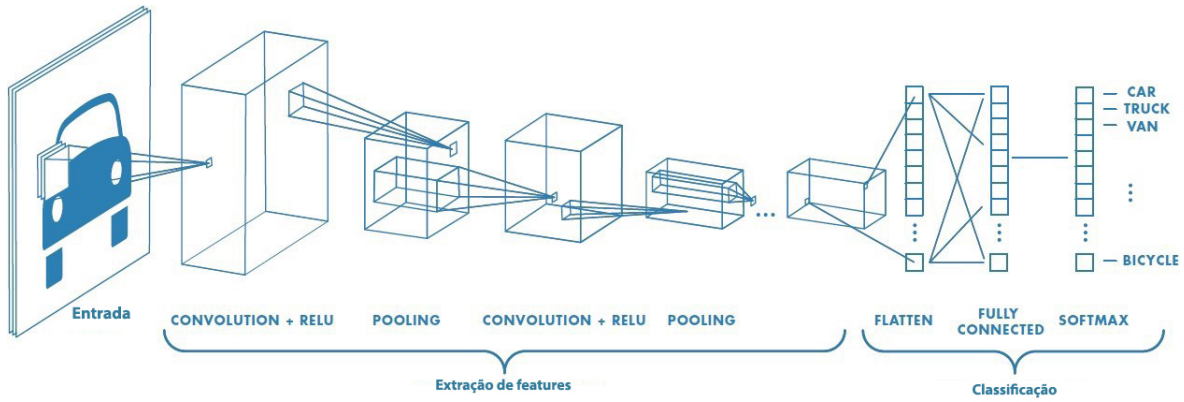
$$f(x) = max(0, x)$$

Portanto, todos os valores menores que zero produzem a saída 0, e todos maiores que zero produzem a própria entrada como saída, daí o nome retificador linear.

Cada saída da função de ativação para cada filtro é armazenada em um mapa separado e em seguida, cada mapa é subamostrado, reduzindo assim sua resolução. Essa subamostragem é feita ao se fazer uma função estatística local, como a média, a mediana ou o valor máximo. Essa etapa é chamada de *pooling*. Ela tem como objetivo reduzir a sensibilidade do mapa de características em relação a diversas formas de distorção (HAYKIN, 2001).

Após realizar essas operações, podem ser feitas novas convoluções ao se inserir mais camadas de convolução. Ao fim das operações de convolução, todos os mapas de *features* gerados são alinhados em um vetor, e esse vetor é utilizado como entrada de uma RNA totalmente conectada. Essa estrutura completa pode ser vista na Figura 27.

Figura 27 – Estrutura de uma rede neural convolucional



Fonte: Adaptado de SAHA (2018).

O treinamento da RNC consiste então de encontrar os valores para cada *kernel* e os pesos da camada totalmente conectada, de modo que a função de custo seja minimizada em relação a esses valores e as combinações dos dados de treinamento.

2.7 HISTOGRAMA DOS GRADIENTES ORIENTADOS

2.7.1 Definição

O histograma dos gradientes orientados (HOG - do inglês, histogram of oriented gradients) é um descritor de histogramas locais de gradientes orientados de uma imagem proposto por Dalal e Triggs em 2005. A ideia proposta por eles é que a aparência e o formato de objetos podem ser bem descritos pela direção dos gradientes de uma imagem e eles são utilizados em conjunto de um classificador para fazer a identificação de objetos (ou qualquer forma) em imagens.

O gradiente de um pixel representa a variação da intensidade dos pixels vizinhos a ele e esse gradiente é denominado orientado porque possui, além do valor de intensidade, um ângulo que determina a sua direção. A Figura 28 mostra como seria a representação gráfica dos gradientes de uma imagem. Como diferentes imagens de um mesmo objeto podem apresentar diferentes valores de intensidade dos pixels devido a fatores como iluminação, posição ou diferentes características do objeto em si que não o determinam, como a cor ou a textura, os tipos de abordagem que dependem disso para classificação de objetos não são muito eficazes. Ao se utilizar gradientes

orientados, esses fatores não influenciarão significativamente o descritor do objeto, pois as direções dos gradientes dependem fortemente do formato dele. Além disso, no trabalho desenvolvido por Dalal e Triggs, são aplicadas diversas técnicas que reduzem mais ainda a influência dessas condições no descritor final.

Figura 28 – Representação dos gradientes de uma imagem

Fonte: VONDRIC et al. (2013).

2.7.2 Algoritmo

O HOG é criado ao se dividir uma imagem em pequenas regiões, chamadas de células, e cada uma contém um histograma local das direções dos gradientes dos pixels que estão contidos naquela célula. O conjunto dos histogramas de todas as células formam o descritor (DALAL e TRIGGS, 2005). Para se obter uma melhor invariância a iluminação, eles sugerem que se faça uma normalização local do contraste das células por meio de um agrupamento de células chamado de bloco. Isso é feito ao se obter um valor médio da ”energia” dos histogramas de células dentro de um bloco e usando isso para normalizar todas as células daquele bloco.

A primeira coisa a se fazer é transformar a imagem RGB para uma representação em tons de cinza. Dalal e Triggs afirmam que utilizar a imagem no espaço de cores RGB melhorou o desempenho do classificador deles, porém não levam em conta

o custo computacional extra necessário para calcular os gradientes em todos os canais de cores.

Para cada pixel $P(x, y)$ são calculadas as componentes do gradiente nas duas direções, $H_x(x, y)$ e $H_y(x, y)$. Isso é feito ao se passar um filtro móvel G_x e um filtro G_y nas direções horizontal e vertical, respectivamente. Esses filtros são da forma:

$$G_x = [-1\ 0\ 1]$$

$$G_y = [-1\ 0\ 1]^T$$

Para cada par de componentes correspondentes a cada pixel (H_x, H_y), o módulo e o ângulo do gradiente são dados por:

$$H(x, y) = \sqrt{H_x^2(x, y) + H_y^2(x, y)}$$

$$\theta(x, y) = Tg^{-1}(\frac{H_y(x,y)}{H_x(x,y)})$$

O próximo passo é definir o histograma local para cada célula. O tamanho da célula pode ser definido como se queira, mas Dalal e Triggs mostraram que o melhor desempenho foi atingido quando o tamanho da célula escolhido foi 8 x 8 pixels. O histograma é definido por um conjunto de direções igualmente espaçadas em uma abertura de $180°$, para um descritor sem sinal (apenas direção), ou de $360°$ para um descritor com sinal (direção e sentido). Cada gradiente de cada pixel da célula contribui para duas direções, para evitar o *aliasing*. Os descritores sem sinal com 9 direções foram os que apresentaram o melhor desempenho no trabalho de Dalal e Triggs e por isso geralmente são utilizados esses valores para se calcular o HOG. Nesse caso, as direções são: $10°,\ 30°,\ 50°,\ 70°,\ 90°,\ 110°,\ 130°,\ 150°\ e\ 170°$. Para cada gradiente de cada pixel na célula, a magnitude é utilizada em uma função linear para calcular a contribuição de cada um no histograma da célula. Essa função é construída de modo que:

- No caso do ângulo do gradiente coincidir com um dos ângulos do histograma, a contribuição para aquela direção é maxima (igual ao valor absoluto do gradiente) e é nula para os outros ângulos.

- No caso do ângulo do gradiente ser o valor médio entre 2 direções consecutivas, a contribuição para cada uma das direções é metade do valor máximo (metade do valor absoluto do gradiente).

Dado um gradiente com ângulo $10° < \theta \leq 30°$ e magnitude H, pode-se construir duas funções lineares que calculem a contribuição do gradiente para as direções $10°$ e $30°$. Essas funções podem ser escritas como:

$$H_{10}(x, y) = \frac{30-\theta(x,y)}{20} H(x, y)$$

$$H_{30}(x, y) = \frac{\theta(x,y)-10}{20} H(x, y)$$

Analogamente, essas funções podem ser escritas para todos os intervalos entre as componentes do histograma seguindo as regras citadas anteriormente. Desse modo, a contribuição do gradiente de cada pixel é somada ao histograma da célula e assim os histogramas locais são formados.

Finalmente, é feita uma normalização das células utilizando a norma L2 ou a norma L2-Hys, sugeridas por Dalal e Triggs. Para isso, são utilizadas regiões chamadas de blocos, de tamanho 16 x 16 pixels. Esse tamanho também foi determinado por obter os melhores resultados no trabalho deles. Cada bloco possui uma separação (*stride*) de 8 x 8 pixels, sendo assim, cada célula é envolvida por mais de um bloco e por isso cada uma contribui para a normalização de mais de um bloco. Apesar de parecer redundante, isso aumenta o desempenho do sistema (DALAL e TRIGGS, 2005). Desse modo, cada bloco engloba 4 células. Os histogramas de cada célula são concatenados e formam um vetor v de direções para cada bloco. Então, cada componente do histograma do bloco é normalizada pela norma L2 do vetor v, de modo que:

$$v_{norm} = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}}$$

Em que $\|v\|_2$ é a norma L2 do vetor v e ϵ é uma pequena constante. A normalização L2-Hys pode ser utilizada no lugar da normalização L2. Ela consiste na normalização L2, seguida de uma delimitação do valor máximo por um limiar, geralmente 0,2, terminando com uma renormalização L2.

Após feita a normalização, todos os histogramas locais são concatenados e isso forma o descritor HOG.

2.8 ESTIMATIVA DOS PONTOS DE REFERÊNCIA DA FACE

2.8.1 Definição

A estimativa dos pontos de referência da face é feita por um algoritmo que detecta as posições dos olhos, das sobrancelhas, do nariz, da boca e do contorno do rosto ao se estimar a posição de pixels em uma imagem que indiquem essas regiões. A Figura 29 apresenta alguns exemplos de pontos de referência das faces de alguns indivíduos.

Figura 29 – Exemplos de pontos de referência da face de alguns indivíduos

Fonte: KAZEMI e SULLIVAN (2014).

Existem inúmeras publicações que apresentam formas de se estimar esses pontos, mas a técnica utilizada neste trabalho é baseada no trabalho de Kazemi e Sullivan (2014). Eles propôem nesse trabalho uma sequência de árvores de regressão para se estimar esses pontos.

2.8.2 Algoritmo

Primeiramente, define-se que o vetor $S = (x_1^T, x_2^T, \cdots, x_p^T \in \mathbb{R}^{2p})$, chamado de contorno, é o vetor que contém todas as p coordenadas dos pontos de referência da imagem I. Cada elemento $x_i \in \mathbb{R}^2$ é um par de coordenadas (x, y).

Em cada iteração, um regressor $r_t(\cdot, \cdot)$ é utilizado para prever o próximo vetor

contorno estimado, denotado por $\hat{S}^{(t+1)}$, dado que o atual contorno estimado é $\hat{S}^{(t)}$, da seguinte forma:

$$\hat{S}^{(t+1)} = \hat{S}^{(t)} + r_t(I, \hat{S}^{(t)})$$

O regressor r_t é calculado baseado em *features* de I, como a intensidade dos pixels, que são ordenadas baseadas no contorno estimado atual $\hat{S}^{(t)}$, de modo que isso dá uma certa invariância do formato final do contorno.

Para se calcular o regressor r_t, primeiro denota-se os dados de treinamento por $(I_1, S_1), \cdots, (I_n, S_n)$, em que I_i é uma imagem de uma face e (S_i) é o conjunto de pontos de referência já conhecidos dessa imagem. Para se encontrar o primeiro regressor r_0 são criadas *triplets* dos dados, que consistem em uma imagem da face, um contorno estimado inicial (que pode ser o contorno médio das faces centralizadas) e um passo de incremento, denotados por $(I_{\pi_i}, \hat{S}_i^{(0)}, \Delta S_i^{(0)})$, onde:

$$\pi_i \in \{1, \cdots, n\}$$

$$\hat{S}_i^{(0)} \in \{S_1, \cdots, S_n\} \backslash S_{\pi_i} \text{ e}$$

$$\Delta S_i^{(0)} = S_{\pi_i} - \hat{S}_i^{(0)}$$

para $i = 1, \cdots, N$ e $N = nR$, onde R é o número de vezes que cada imagem de treinamento é utilizada no processo de treinamento. Cada regressor r_t é calculado da seguinte forma:

1. Inicializa-se a função:

$$f_0(I, \hat{S}^{(t)}) = \underset{\gamma \in \mathbb{R}^{2p}}{arg\,min} \sum_{i=1}^{N} \|\Delta S_i^{(t)} - \gamma\|^2$$

2. para $k = 1, \cdots, K$:

   - Calcula-se para $i = 1, \cdots, N$:

$$r_{ik} = \Delta S_i^{(t)} - f_{k-1}(I_{\pi_i}, \hat{S}_i^{(t)})$$

- Treina-se uma árvore de regressão para as saídas r_{ik} tendo como entrada um vetor v, calculado a partir de *features* da imagem e do contorno estimado atual por uma função $g_k(I, \hat{S}_i^{(t)})$.
- Se calcula o novo f_k da seguinte forma:

$$f_k(I, \hat{S}^{(t)}) = f_{k-1}(I, \hat{S}^{(t)}) + \nu \; g_k(I, \hat{S}^{(t)})$$

onde ν é uma constante definida para a taxa de aprendizado.

3. O regressor r_t é dado por:

$$r_t(I, \hat{S}^{(t)}) = f_K(I, \hat{S}^{(t)})$$

Com o regressor r_t calculado, se calcula o próximo contorno estimado e o próximo passo de incremento da seguinte forma:

$$\hat{S}_i^{(t+1)} = \hat{S}_i^{(t)} + r_t(I_{\pi_i}, \hat{S}_i^{(t)})$$

$$\Delta S_i^{(t+1)} = S_{\pi_i} - \hat{S}_i^{(t+1)}$$

Com esses valores se pode calcular o próximo regressor r_{t+1}. Esse processo é repetido até que as estimativas dos contornos $\hat{S}_i^{(t)}$ estejam próximas dos valores de S_i de acordo com alguma função de erro.

3 METODOLOGIA

No capítulo anterior, foi definido que o método de biometria utilizado no sistema de autenticação seria o reconhecimento facial. Neste capítulo são dados detalhes de como foi implementado o algoritmo de reconhecimento facial e como foi medido o desempenho do sistema. Além disso, são referenciadas as ferramentas utilizadas e as bases de dados que foram utilizadas para o desenvolvimento do sistema.

3.1 ETAPAS DO ALGORITMO

Segundo Geitgey (2016), para se desenvolver um algoritmo que faça o reconhecimento facial de pessoas em imagens, é necessário se resolver quatro problemas básicos:

- Identificar as faces na imagem para selecionar a região de interesse (ROI - do inglês, region of interest)
- Padronizar as faces para que a posição do rosto (olhos, nariz, boca, etc) nas ROIs de todas as imagens seja a mesma
- Extrair características únicas da face que possam diferenciar um indivíduo dos outros
- Identificar a pessoa com base nas características extraídas

Ainda, de acordo com Geitgey, o segundo problema não é necessariamente um problema, mas uma etapa que facilita bastante a resolução do terceiro problema e da exatidão do resultado final do sistema. No caso do último tópico, para este trabalho não é necessário identificar uma pessoa e sim verificar a sua identidade, pois como já foi dito, em um sistema de autenticação é feita a verificação de que um indivíduo é ou não quem ele afirma ser.

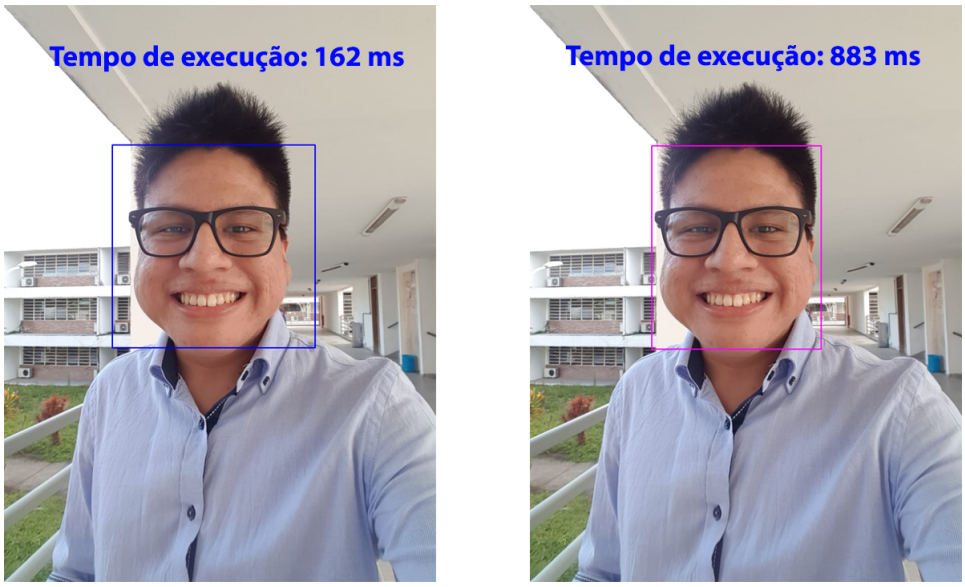
Assim, as próximas seções tratarão de explicar como cada um desses problemas foi resolvido e os algoritmos utilizados. Ao se resolver todos os problemas o algoritmo de reconhecimento facial final é simplesmente a combinação dos algoritmos utilizados em cada etapa.

Todos os algoritmos foram desenvolvidos na linguagem de programação Python, versão 3.6.5, e para cada etapa serão referenciados os pacotes utilizados e a explicação do funcionamento das funções utilizadas desses pacotes.

3.1.1 Identificação de faces em uma imagem

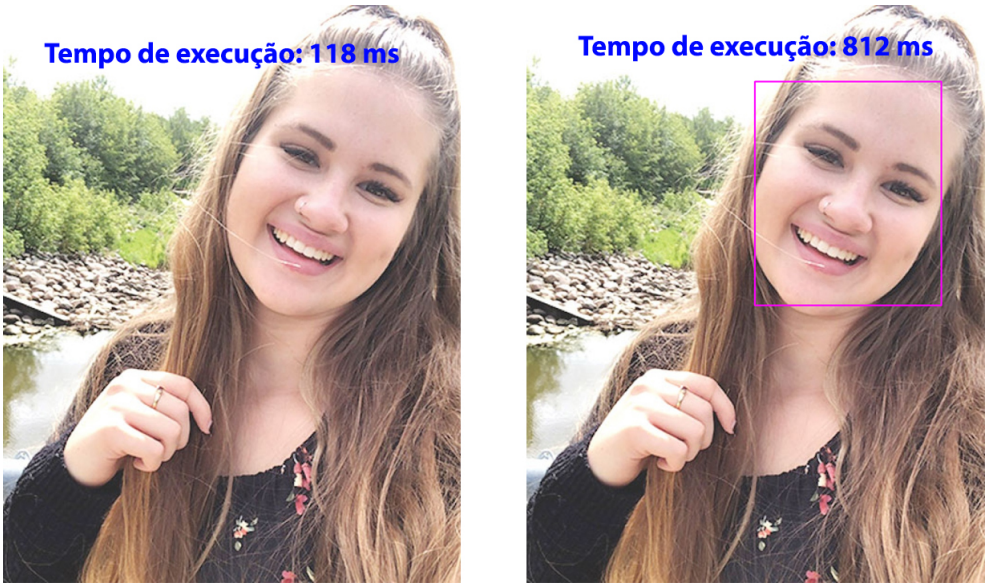
Há diversos trabalhos produzidos para identificar faces em uma imagem e as abordagens utilizadas podem ser divididas naquelas baseadas em *features* e naquelas baseadas na própria imagem (AL-KHALIDI et al., 2018). Os dois métodos mais famosos de detecção de face são o que utiliza detectores de Viola-Jones, propostos por Paul Viola e Michael Jones em 2001, e o que utiliza HOG. Os detectores de Viola-Jones possuem um tempo de execução muito pequeno, aproximadamente 10 vezes mais rápido que o método com HOG (CRUZ et. al, 2015), porém o último produz resultados mais confiáveis (GEITGEY, 2016) o que é muito importante em um sistema de autenticação. Portanto, neste trabalho foi utilizado o método baseado em HOG. As Figuras 30 e 31 apresentam um comparativo entre os dois métodos de detecção de face em relação ao tempo de execução do algoritmo e ao sucesso na detecção.

Figura 30 – Comparativo 1 entre o detector de Viola-Jones (à esquerda) e o detector por HOG (à direita)



Fonte: Autoria própria.

Figura 31 – Comparativo 2 entre o detector de Viola-Jones (à esquerda) e o detector por HOG (à direita)



Fonte: Adaptado de MERCURY (2018).

Para se utilizar o método baseado em HOG, primeiramente é necessário um *dataset* de imagens de faces e de imagens que não sejam faces. Em seguida, é calculado o descritor HOG de todas essas imagens e finalmente se treina um classificador para classificar se uma imagem é uma face ou não. O classificador escolhido foi a SVM, porque ela é um classificador binário, possui uma alta taxa de acerto em relação a outros classificadores quando a dimensão dos dados de entrada é grande (GAHUKAR, 2018) e a complexidade computacional é baixa, pois o cálculo feito para a classificação é uma combinação linear das entradas e apenas uma parte delas é utilizada na classificação, ao contrário de outros classificadores.

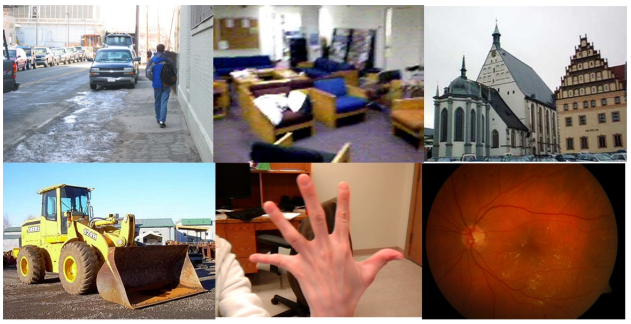
O *dataset* utilizado para as faces foi o FaceScrub, que é um *dataset* criado no trabalho de Ng e Winkler (2014). Esse *dataset* contém cerca de 47.000 imagens de atores e atrizes famosas em diferentes posições e diferentes expressões faciais e cerca de 47.000 imagens correspondentes às faces extraídas das imagens originais, totalizando quase 100.000 imagens. A Figura 32 apresenta algumas imagens desse *dataset*. O *dataset* utilizado para as imagens de não faces foi adaptado do LabelMe, do MIT. Esse *dataset* foi criado no trabalho de Russel et al. (2008) e contém cerca de 200.000 imagens de cenários diversos e ele foi adaptado de modo que foram removidas as imagens que continham faces para evitar problemas de classificação da SVM.

A Figura 33 apresenta algumas imagens desse *dataset*.

Figura 32 – Algumas imagens do dataset FaceScrub

Fonte: Autoria própria.

Figura 33 – Algumas imagens do dataset LabelMe



Fonte: Autoria própria.

Para esta etapa, foram utilizadas os pacotes ***Pickle***, ***OpenCV***, ***NumPy***, ***scikit-learn*** e ***Time***. O pacote ***Pickle*** é um pacote que utiliza protocolos binários para serialização e desserialização de objetos. Ele é utilizado principalmente para salvar dados, variáveis e objetos em arquivos binários para carregá-los posteriormente. O pacote ***OpenCV*** é uma biblioteca de código livre para o desenvolvimento de aplicações em visão computacional. Ele possui diversas funções que auxiliam na manipulação e no processamento de imagens. O pacote ***NumPy*** é um pacote de computação scientífica. Ele é utilizado principalmente para se fazer cálculos com vetores e matrizes e possui diversas funções de operações matemáticas com foco em operações matriciais. O pacote ***scikit-learn*** é um pacote que possui ferramentas para ML, análise e mineração de dados. Ele pode ser usado em problemas de classificação, regressão, agrupamento entre outras aplicações de ML. O pacote ***Time*** serve para acessar o tempo do sistema

e é utilizado unicamente para se medir o tempo de processamento do código.

Inicialmente, todas as imagens do *dataset* foram transformadas para escala de tons de cinza e redimensionadas para o tamanho 96 x 96 pixels, para padronizar os dados. Esse tamanho foi escolhido para que o tamanho do descritor HOG não seja tão grande e ao mesmo tempo as imagens não percam tanto a resolução de modo que características do rosto sejam perdidas.

Em seguida, foi criada uma função que calcula o descritor HOG de uma imagem. O tamanho dos blocos escolhidos foi 16x16 pixels, o tamanho das células 8x8 pixels, o número de direções do histograma 9, o sentidos não foram considerados (ângulos de 0 a 180^o), e o fator ϵ do cálculo da normalização L2 foi 0,01. Também foi utilizado o descritor HOG do pacote OpenCV. Para se calcular o HOG utilizando as funções desse pacote, primeiramente se cria um objeto de HOG utilizando o construtor ***cv2.HOGDescriptor()*** e depois se calcula o descritor de uma imagem utilizando o método ***HOGDescriptor.compute()***, como é mostrado na Figura 34.

Figura 34 – Cálculo do HOG utilizando a classe do OpenCV

```
image = cv2.imread('test_image.jpg', 0)

HOG_object = cv2.HOGDescriptor(winSize,blockSize,blockStride,cellSize,nbins,
derivAperture,winSigma,histogramNormType,L2HysThreshold,gammaCorrection,nlevels)

image_HOG = HOG_object.compute(image,winStride,padding)
```

Fonte: Autoria própria.

A função ***cv2.imread()*** carrega um arquivo de imagem na forma matricial. O primeiro argumento é caminho do arquivo a ser lido e o segundo é a forma como a imagem será representada, que pode ser colorida, em tons de cinza ou a imagem como ela é vista, incluindo o canal alfa. No construtor do objeto do HOG, o argumento ***winSize*** é o tamanho da janela que será calculado o HOG (pois se pode calcular vários HOGs na mesma imagem e concatená-los no final), ***blockSize*** é o tamanho de cada bloco, ***blockStride*** é a distância entre dois blocos vizinhos, ***cellSize*** é o tamanho das células, ***nbins*** é o número de direções dos histogramas locais, ***derivAperture*** é uma constante que multiplica o filtro do gradiente, ***winSigma*** é o desvio padrão do filtro gaussiano, ***histogramNormType*** é o tipo de normalização que será utilizado, ***L2HysThreshold*** é

o limiar utilizado na normalização L2-Hys, caso ela seja utilizada, ***gammaCorrection*** é um valor que especifica se a correção de gama será utilizada ou não e ***nlevels*** é o número máximo de aumentos da janela de detecção.

Um código foi escrito para a medição do tempo de cálculo do HOG pela função criada para compará-lo com o tempo de cálculo do HOG utilizando o pacote ***OpenCV***. Foi verificado que o tempo de cálculo do descritor HOG do ***OpenCV*** é em média 25 vezes menor que o da função criada. Esse resultado provavelmente foi originado por otimizações de código e cálculos matriciais que são feitos pelo descritor do ***OpenCV*** e que não foram implementados na função criada. Portanto, o descritor HOG do ***OpenCV*** passou a ser utilizado como padrão para os próximos algoritmos.

Todas as imagens de face do *dataset* foram embaralhadas e todas as imagens de não faces também, porém os dados de classes diferentes não foram misturados entre si para que fosse mais fácil de rotulá-los, tanto para o treinamento, como para a medição do desempenho da SVM. Foi calculado o HOG de todas as 242.002 imagens e armazenado o resultado em listas de tamanho 10.000 x 2, com cada linha sendo composta por um descritor HOG e um valor binário 1 ou 0, sendo atribuído o valor 1 para os descritores de imagens que são faces e 0 para os descritores de imagens que não são faces. Cada uma dessas listas foi salva em um arquivo binário com auxílio do pacote Pickle.

Para se criar e treinar a SVM, foi utilizada a classe ***LinearSVC*** do pacote ***scikit-learn.svm***. Os objetos dessa classe são máquinas de vetores de suporte. O método ***LinearSVC.fit()*** faz o treinamento da SVM utilizando os parâmetros de inicialização utilizados na construção do objeto. A Figura 35 exemplifica a criação de uma SVM e o seu treinamento utilizando a classe ***LinearSVC***.

Figura 35 – Criação e treinamento de uma SVM utilizando o pacote scikit-learn.svm

```
clf = LinearSVC(random_state=0, tol=1e-5, C = 0.5, max_iter=2000)
clf.fit(X, y)
```

Fonte: Autoria própria.

O parâmetro ***random_state*** seleciona o *seed* que será usado para embaralhar os dados de treinamento e nesse caso foi escolhido o mesmo *seed* utilizado pelo pa-

cote numpy.random. O parâmetro ***tol*** é a tolerância máxima que se deseja obter para a função de custo que é usada como critério de parada. O parâmetro ***C*** é a constante do termo de erro da função objetivo e o parâmetro ***max_iter*** é o número máximo de iterações que se deve fazer. Com o objeto criado, se utiliza o método ***LinearSVC.fit()*** para treinar a SVM. O parâmetro ***X*** é uma lista contendo os *features* de entrada de cada uma das amostras de treinamento e o parâmetro ***y*** é a saída alvo para cada amostra correspondente.

Foram feitos alguns testes iniciais e foi verificado que os melhores parâmetros de treinamento dentre os testados são os que foram utilizados na figura anterior. No primeiro treinamento foram selecionadas 2.000 imagens de faces e 10.000 de não faces arbitrárias. No segundo, foi feito um processo chamado de *hard mining*, em que se utiliza a SVM treinada para classificar as amostras de teste e encontrar falsos-positivos e então essas imagens são adicionadas ao conjunto de dados de treinamento para ser feito um novo treinamento e diminuir o erro de classificação. No terceiro, foram selecionados novos dados arbitrários, totalizando 10.000 amostras positivas e 10.000 amostras negativas. Em cada um desses treinamentos foi feito um teste de desempenho ao se fazer um teste de classificação da SVM treinada com todas as 242.002 imagens do *dataset*. O teste de desempenho foi feito utilizando o método ***LinearSVC.score()***, que calcula a taxa de acerto da SVM para um conjunto de dados de teste, para cada 10.000 amostras. A sintaxe desse método é mostrada na Figura 36.

Figura 36 – Método utilizado para o teste de desempenho da SVM

```
result = lsvm.score(test_input, test_output)
```

Fonte: Autoria própria.

Em que o parâmetro ***test_input*** é uma lista com as *features* de todos os dados de teste e o parâmetro ***test_output*** é uma lista com as saídas esperadas correspondentes. Foi escolhido o modelo de SVM com a menor média das taxas de erro.

Finalmente, para se encontrar as faces, ou uma face, em uma imagem é utilizada uma janela móvel que varre a imagem em busca de faces. Em cada posição da janela a região contida nela é redimensionada para o tamanho 96x96 e é calcu-

lado o HOG dessa janela. Esse HOG, então, é utilizado como entrada na SVM para a janela ser classificada como face ou não face. As coordenadas das janelas classificadas como face são guardadas, porque elas representam as coordenadas das faces na imagem.

Essa abordagem parece ser bem promissora, porém há alguns problemas relacionados a ela.

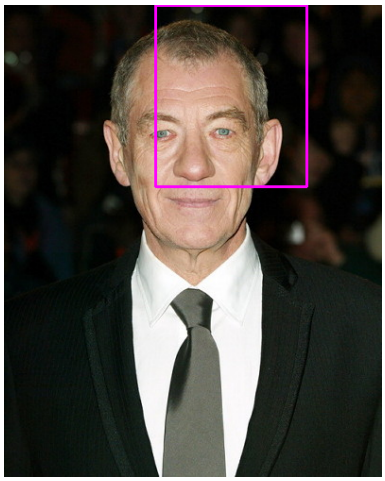
O primeiro problema é que se a janela tiver um tamanho fixo a SVM pode não fazer uma classificação muito boa ou nem mesmo conseguir detectar a face na imagem, pois, a depender da distância entre o indivíduo e a câmera, o rosto dele representará uma porcentagem diferente do tamanho total da imagem e o tamanho da janela pode ser muito grande para a SVM fazer uma boa classificação. Logo, a janela em vez de varrer a imagem uma única vez, deve fazer essa varredura e diminuir de tamanho e repetir esse procedimento até um limite de reduções de tamanho. Esse problema é retratado na Figura 37.

Figura 37 – Primeiro problema de detecção de face

Fonte: Autoria própria.

O segundo problema é que, se o passo passo de varredura for muito grande, a janela pode ”pular” uma face ou a face pode estar mal posicionada (não centralizada) na janela e novamente a SVM fará uma classificação ruim. Esse problema é apresentado na Figura 38.

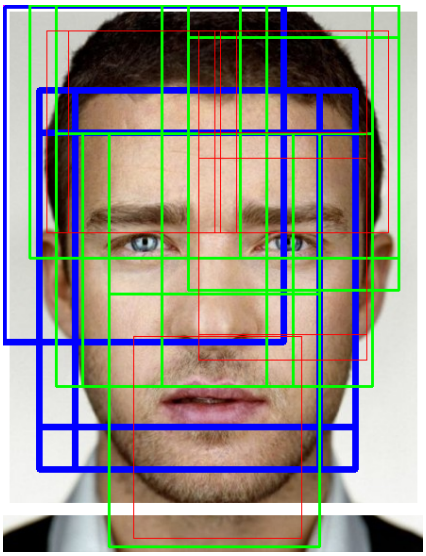
Figura 38 – Segundo problema de detecção de face



Fonte: Adaptado de MARVEL MOVIES (2008).

O terceiro problema é exatamente o oposto do problema anterior. Se o passo de varredura for muito pequeno, mais de uma janela detectará a mesma face e isso resultará em várias regiões sobrepostas contendo ela. Esse problema pode ser visto na Figura 39.

Figura 39 – Terceiro problema de detecção de face



Fonte: Adaptado de LOFT965 (2013).

Todos esses problemas levam a um *trade-off* de qualidade, custo e velocidade. A qualidade seria o desempenho do algoritmo de achar uma janela com a face mais

centralizada possível e de achar uma face independente do tamanho dela na imagem. O custo seria a quantidade de recursos computacionais (cálculos) necessários para se encontrar o local da face. A velocidade está relacionada ao tempo de execução necessário para fazer essa detecção. Quanto menor o passo de varredura e mais redimensionamentos da janela forem feitos, maior o número de cálculos necessários, menor a velocidade do algoritmo, e maior a qualidade da detecção.

Foram encontrados os parâmetros ideais de forma empírica ao se mudar os valores de forma manual, testar o detector em diversas imagens e fazer uma análise qualitativa da região de interesse calculada. Assim, foi encontrado um equilíbrio entre tempo de execução e desempenho do detector. Porém, para isso ser obtido foi necessário que fossem impostas duas restrições ao detector. A primeira é que há uma distância máxima que o indivíduo deve estar da câmera para que sua face seja detectada. A segunda é que o detector só consegue encontrar uma única face na imagem. Essa última restrição não é um problema para um sistema de autenticação, pois geralmente só uma pessoa é verificada por vez.

Para o problema de múltiplas regiões contendo a mesma face, foi utilizado o método ***LinearSVC.decision_function()***, que calcula a distância euclidiana do vetor calculado até o hiperplano de decisão. Foi verificado que essa distância pode ser utilizada como um indicativo do grau de centralização da face na janela. Foi definido um limiar de distância mínimo para que algumas regiões que não contêm faces, como podem ser vistas na Figura 39, sejam eliminadas do algoritmo de decisão da janela final. Para se escolher a região final, dado um número *N* de regiões detectadas como face, foram utilizadas as seguintes regras:

- Se N for igual a 1, a região escolhida é a única região detectada
- Se N for maior que 1 e a distância de um vetor ao hiperplano se sobressai em relação à média das distâncias de todos os vetores ao hiperplano de um certo limiar, então a região que origina aquele vetor é a região escolhida
- Se N for maior maior que 1 e a distância de nenhum vetor ao hiperplano se sobressai a média das distâncias, então a região escolhida é uma média aritmética de todas as regiões detectadas

Com essas informações, foi criada a função de detecção de faces, que utiliza

quatro parâmetros de entrada: a imagem que se deseja detectar a face, um descritor HOG, um modelo treinado de SVM e um limiar utilizado para reduzir o número de regiões indesejadas. Essa função retorna quatro variáveis, que são as coordenadas dos limites de um retângulo que delimita a face detectada.

3.1.2 Correção da posição das faces

A correção da posição das faces é um passo importante porque o objetivo é tentar manter as faces sempre na mesma posição na imagem, de modo que isso aumente o desempenho da rede neural que será utilizada no próximo passo e diminua as taxas de erro do resultado final do sistema de autenticação. As Figuras 40 e 41 mostram uma imagem com a face original desalinhada e a imagem da face correspondente já alinhada, respectivamente.

Figura 40 – Imagem original com a face desalinhada

Fonte: Adaptado de NNDB (2014).

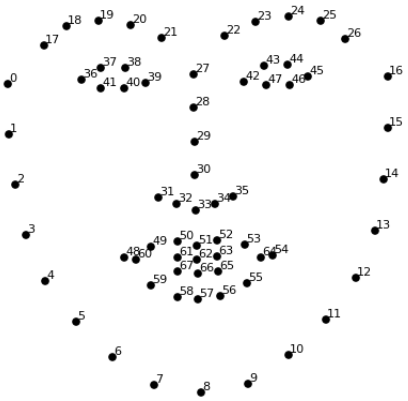
Figura 41 – Imagem com a posição da face corrigida

Fonte: Adaptado de NNDB (2014).

Para isso, o objetivo é fazer com que os olhos estejam sempre na mesma posição em todas as imagens e como consequência disso as faces também ficarão sempre na mesma posição. Para se chegar a esse objetivo, foi utilizado o método mostrado na seção 2.8 para se encontrar 68 pontos de referência da face. A Figura 42 mostra a

distribuição desses pontos. Esses pontos podem ser encontrados em todas as faces, mesmo que elas não estejam alinhadas ou centralizadas. Eles marcam o local das sobrancelhas, dos olhos, do nariz, dos lábios e da linha que contorna o rosto.

Figura 42 – 68 pontos de referência das faces



Fonte: GEITGEY (2016).

Ao se detectar esses pontos em uma face, eles podem ser utilizados para realizar transformações afins na imagem (redimensionamento, rotação e translação) de modo que se chegue ao objetivo proposto.

Para se encontrar os pontos na face, foi utilizada a função ***face_landmarks()*** do pacote ***face_recognition***. Essa função utiliza um modelo de regressores já treinados pelo desenvolvedor do pacote para encontrar os 68 pontos de referência. Ela é utilizada da seguinte forma:

Figura 43 – Sintaxe da função utilizada para detectar os 68 pontos de referência da face

```
landmarks = fr.face_landmarks(img)
```

Fonte: Autoria própria.

A variável ***img*** é uma imagem carregada pelo ***OpenCV*** e o retorno da função é um dicionário contendo os componentes do rosto (sobrancelha esquerda, sobrancelha, direita, olho esquerdo, etc) como chaves e uma lista contendo as posições dos pontos para cada uma dessas chaves.

Com o conhecimento desses pontos, então foi criada uma classe de transformação com um método utilizado para transformar uma imagem de entrada. O construtor dessa classe utiliza como argumentos de entrada a nova posição relativa que se deseja para posicionar o olho esquerdo na imagem transformada e as dimensões da imagem transformada. O método criado para alinhar a imagem utiliza como argumentos de entrada a imagem a ser alinhada e os pontos de referência da face.

Para o alinhamento da face, como mostrado na Figura 44, primeiro se obtém os pontos de referência dos dois olhos e é calculada a posição do ponto médio de cada olho.

Figura 44 – Primeira parte do código do algoritmo de alinhamento da face

```
24      def align(self, image, face_landmarks):
25
26          # extract the left and right eye (x, y)-coordinates
27          leftEyePts = np.array(face_landmarks['right_eye'])
28          rightEyePts = np.array(face_landmarks['left_eye'])
29
30          # compute the center of mass for each eye
31          leftEyeCenter = leftEyePts.mean(axis=0).astype("int")
32          rightEyeCenter = rightEyePts.mean(axis=0).astype("int")
```

Fonte: Autoria própria.

A sintaxe ***np.array()*** cria um objeto da classe ***numpy.array()*** a partir do seu argumento, que no caso são os pontos correspondentes aos dois olhos. O método ***np.array.mean()*** calcula o ponto médio dos elementos do array, tanto no eixo X, como no eixo Y.

Em seguida, é calculado o ângulo formado entre o eixo horizontal da imagem e um eixo que liga os pontos médios de cada olho, como mostrado na Figura 45.

Figura 45 – Segunda parte do código do algoritmo de alinhamento da face

```
34          # compute the angle between the eye centroids
35          dY = rightEyeCenter[1] - leftEyeCenter[1]
36          dX = rightEyeCenter[0] - leftEyeCenter[0]
37          angle = np.degrees(np.arctan2(dY, dX)) - 180
38          desiredRightEyeX = 1.0 - self.desiredLeftEye[0]
```

Fonte: Autoria própria.

A função ***np.degrees()*** converte um ângulo em radianos para graus e a função ***np.arctan2()*** calcula o arco cuja tangente é o valor do primeiro argumento dividido pelo segundo argumento.

Em seguida, calcula-se a escala da imagem transformada em relação a imagem original, baseado na distância entre os olhos na imagem original e a distância entre os olhos na imagem desejada, como mostrado na Figura 46.

Figura 46 – Terceira parte do código do algoritmo de alinhamento da face

```
44          dist = np.sqrt((dX ** 2) + (dY ** 2))
45          desiredDist = (desiredRightEyeX - self.desiredLeftEye[0])
46          desiredDist *= self.desiredFaceWidth
47          scale = desiredDist / dist
```

Fonte: Autoria própria.

A função ***np.sqrt()*** calcula a raíz quadrada do seu argumento.

Então, se calcula o ponto médio entre os dois olhos para ser utilizado como o centro de rotação da imagem e ele é utilizado em conjunto com o ângulo e a escala para se definir uma matriz de transformação utilizando a função ***cv2.getRotationMatrix2D()*** como é mostrado na Figura 47.

Figura 47 – Quarta parte do código do algoritmo de alinhamento da face

```
eyesCenter = ((leftEyeCenter[0] + rightEyeCenter[0]) // 2, (leftEyeCenter[1] + rightEyeCenter[1]) // 2)

# grab the rotation matrix for rotating and scaling the face
M = cv2.getRotationMatrix2D(eyesCenter, angle, scale)
```

Fonte: Autoria própria.

Finalmente, se atualiza a componente de translação da matriz de transformação e se utiliza a função ***cv2.warpAffine()*** para fazer a transformação da imagem. Os argumentos dessa função são: a imagem a ser transformada, a matriz de transformação, o novo tamanho da imagem e uma flag indicando o método de interpolação a ser utilizado. A Figura 48 mostra esses dois últimos passos.

Figura 48 – Quinta parte do código do algoritmo de alinhamento da face

```
56          # update the translation component of the matrix
57          tX = self.desiredFaceWidth * 0.5
58          tY = self.desiredFaceHeight * self.desiredLeftEye[1]
59          M[0, 2] += (tX - eyesCenter[0])
60          M[1, 2] += (tY - eyesCenter[1])
61
62          # apply the affine transformation
63          (w, h) = (self.desiredFaceWidth, self.desiredFaceHeight)
64          output = cv2.warpAffine(image, M, (w, h), flags=cv2.INTER_CUBIC)
65
66          # return the aligned face
67          return output
```

Fonte: Autoria própria.

3.1.3 Extração de *features* do rosto

Todos os passos descritos até aqui foram pré-processamentos para preparar a imagem para ser utilizada na resolução do problema principal, que é encontrar um método de diferenciar o rosto de uma pessoa de todas as demais. Ao se pensar em técnicas de ML para resolver esse problema, a primeira ideia que vem à mente é de utilizar a imagem obtida no último passo em um classificador, como uma RNA ou uma RNC, e utilizar uma enorme quantidade de imagens das mesmas pessoas para fazer o treinamento. O principal problema dessa abordagem é que fatores como iluminação, expressão facial e até mesmo a presença/ausência de pêlos ou objetos no rosto, como óculos, teriam uma grande influência no processo de identificação. Por isso, seriam necessárias centenas ou milhares de imagens da mesma pessoa, em diferentes poses, com diferentes iluminações, para se formar dados de treinamento para essa rede. Além disso, cada vez que uma nova pessoa fosse cadastrada seria necessário se retreinar a rede e adicionar mais uma saída, ou de se treinar uma rede individual para cada pessoa. Para um sistema com poucas pessoas cadastradas, como um sistema

de autenticação residencial, por exemplo, essa abordagem não teria tantos problemas, pois o tempo para se retreinar a rede seria relativamente pequeno e não seriam necessários tantos dados de treinamento. Mas para um sistema com muitas pessoas, como um banco ou uma empresa, essa abordagem seria ineficiente, pela quantidade de imagens de cada pessoa que seria necessária, como dito anteriormente.

Então, a abordagem que se utiliza é de, em vez de se utilizar a imagem do rosto para identificar as pessoas, extrair-se *features* dessa imagem para que elas sejam utilizadas como um código único (uma senha) para identificá-las. Essas *features* são uma analogia de características do rosto, como cor dos olhos, distância entre eles, formato do rosto, tamanho da boca, etc. Tais características poderiam ser utilizadas como *features*, porém ao se utilizar técnicas de *deep learning* (do inglês, aprendizado profundo), podem ser definidas *features* que possuam maior impacto na diferenciação do rosto de uma pessoa para outra de forma autônoma pelo processador.

Para se definir as *features* é utilizada uma técnica baseada no trabalho de Schroff et al. (2015). A ideia geral dessa técnica é de se treinar uma Rede Neural Convolucional Profunda (RNCP), que é uma RNC com muitas camadas, com uma etapa extra de treinamento que consiste em uma função de perda baseada em um conjunto de *triplets*. Em cada iteração dessa etapa, três imagens (um *triplet*) são selecionadas: uma âncora, um exemplo positivo e um exemplo negativo. A âncora é uma imagem de uma face de uma pessoa, o exemplo positivo é outra imagem da mesma pessoa e o exemplo negativo é uma face de outra pessoa. Schroff et al. afirmam que a escolha dos *triplets* influencia bastante no tempo de treinamento e no resultado final da RNCP, de modo que é de grande ajuda que os exemplos positivos sejam bem parecidos com as âncoras e os exemplos negativos sejam totalmente diferentes. A função de perda busca modificar os pesos da RNCP de modo a minimizar a distância entre os valores gerados da imagem âncora e do exemplo positivo e maximizar a distância entre os valores gerados da imagem âncora e do exemplo negativo. A Figura 49 apresenta essa ideia de forma gráfica.

Esse processo de se transformar *features* de uma imagem em valores numéricos é chamado de *embedding* (GEITGEY, 2016) e esses valores são chamados de *encodings*. O que cada um desses valores significa pouco importa, mas os vetores formados por eles em diferentes pessoas são muito distantes entre si, uma vez que a

Figura 49 – Etapa de treinamento da RNCP baseado em triplets

Fonte: SCHROFF et al. (2015).

rede é treinada.

Para fazer esse treinamento é necessário um grande número (milhões) de dados de treinamento e ainda levaria um tempo enorme. De fato, é mencionado no trabalho de Schroff et al. que, utilizando um *cluster* de cpus, cada treinamento demorou de 1000 a 2000 horas. Porém, não foi necessário ser feito isso, pois o projeto *OpenFace* publicou diversas RNCPs treinadas para fazerem o *embedding*. Nesta etapa, foi utilizado o pacote ***face_recognition***, que utiliza uma dessas redes publicadas pelo *OpenFace*.

Utilizando a função ***fr.face_encodings()***, que utiliza como parâmetros a imagem que se quer extrair os valores e a localização das faces na imagem, se extrai 128 valores de uma face com o modelo de RNCP utilizado. A sintaxe é mostrada na Figura 50.

Figura 50 – Sintaxe da função que extrai 128 medidas da face de uma pessoa

```
encodings_input = fr.face_encodings(img_aligned, [(0, face_width_o, face_height_o, 0)])
```

Fonte: Autoria própria.

3.1.4 Verificação dos indivíduos

Esse é o passo com menor complexidade computacional. Primeiro, são obtidos previamente os *encodings* da face de um indivíduo e armazenadas em uma memória. Para se fazer a verificação, são comparados os *encodings* de uma nova leitura desse indivíduo com os *encodings* previamente armazenados, e se eles forem próximos o indivíduo é quem diz ser.

Isso é feito ao ser calculada a distância euclidiana entre os dois *encodings* e utilizar um limiar para separar uma verificação positiva de uma verificação negativa. Esses dois cálculos podem ser feitos com uma única função do pacote face_recognition, que é a ***fr.compare_faces()***. A sintaxe dessa função é mostrada na Figura 51.

Figura 51 – Sintaxe da função que compara dois conjuntos de medidas da face

```
verification = fr.compare_faces(known_encodings, encodings_input[0])
```

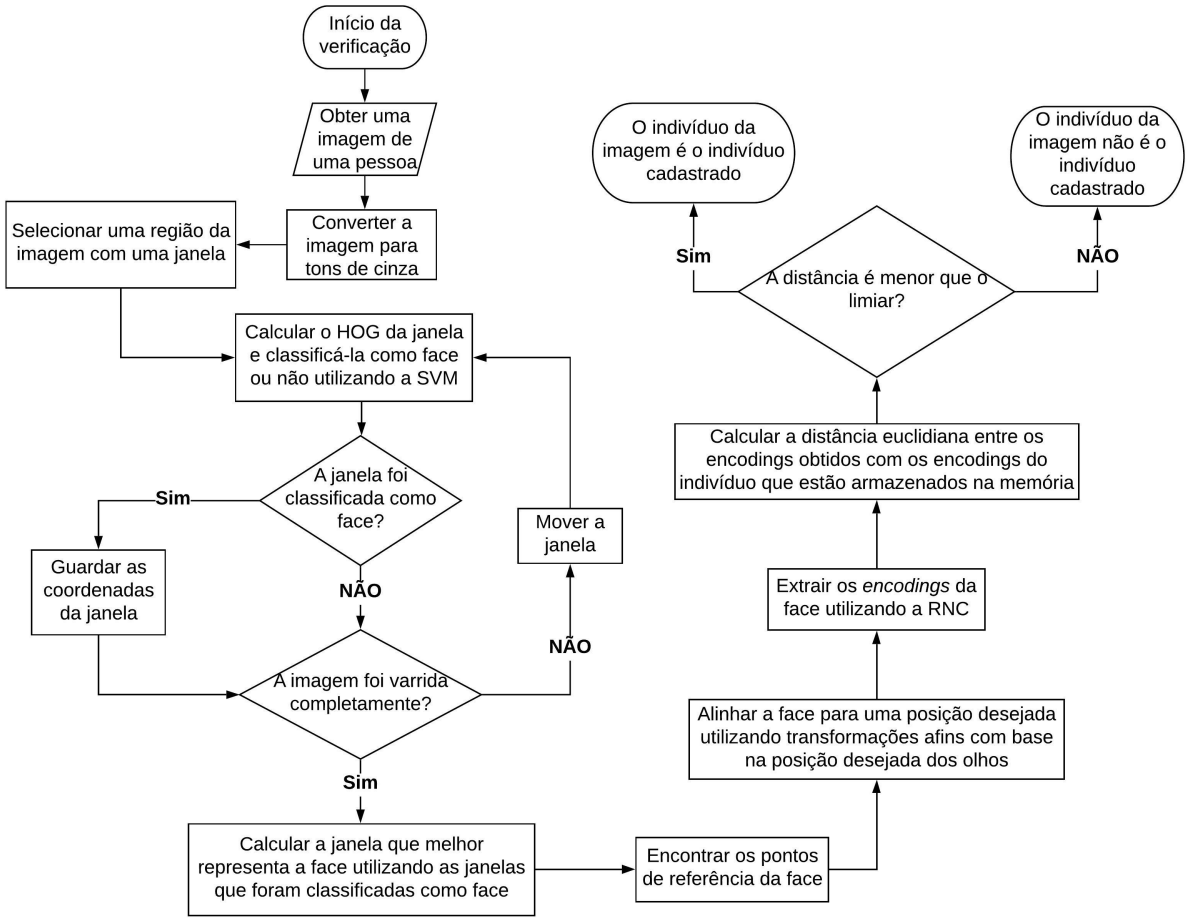
Fonte: Autoria própria.

Essa função tem como parâmetros de entrada uma lista de medidas conhecidas e um conjunto de medidas a ser verificado. Também pode ser escolhido o limiar de decisão, mas o valor padrão é 0,6, então ele pode ser omitido. A saída é uma lista de booleanos que são resultado da comparação da distância euclidiana com o limiar do conjunto de medidas novo com os conjuntos já conhecidos.

3.1.5 Algoritmo final

O fluxograma mostrado na Figura 52 apresenta o algoritmo completo ao se juntar os passos descritos anteriormente.

Figura 52 – Fluxograma do algoritmo para reconhecimento de face



Fonte: Autoria própria.

3.2 OBTENÇÃO DOS RESULTADOS

Para definir o melhor modelo de SVM para a detecção de faces, como já foi dito, foram feitos três treinamentos e medidas as taxas de acerto de cada modelo treinado. O modelo com as menores taxas foi o escolhido para ser utilizado.

Para o detector da região da face, foram variados os parâmetros de passo de varredura, tamanho das janelas, quantidade de diminuições das janelas e limiar de distância da SVM e foram apresentados resultados qualitativos das mudanças desses parâmetros para algumas imagens, além do tempo de execução.

Para os testes de desempenho do sistema, foram selecionados 20 atores e 20 atrizes arbitrários do *dataset*, além do próprio autor, para se medir a FAR e a FRR do

sistema. Foram medidos esses valores ao utilizar 1, 3, 5 e 10 *encodings* conhecidos de cada indivíduo e ao se adicionar ou remover a etapa de alinhamento da imagem. A verificação é positiva se a comparação dos *encodings* obtidos na autenticação com pelo menos um dos encodings armazenados na memória gerar uma verificação positiva. Também foi medido o tempo de execução do programa para cada variação desses parâmetros.

Foi feito um teste qualitativo do sistema, utilizando o conceito que foi chamado de decisão difícil. Foram selecionadas algumas imagens de pares de pessoas famosas que possuem o rosto muito parecido e foi feito o cadastro delas no sistema. O objetivo desse teste foi de ajustar o limiar de decisão da distância euclidiana máxima entre os *encodings* das imagens cadastradas e da nova imagem, para que o sistema consiga diferenciar de forma correta pessoas com o rosto parecido.

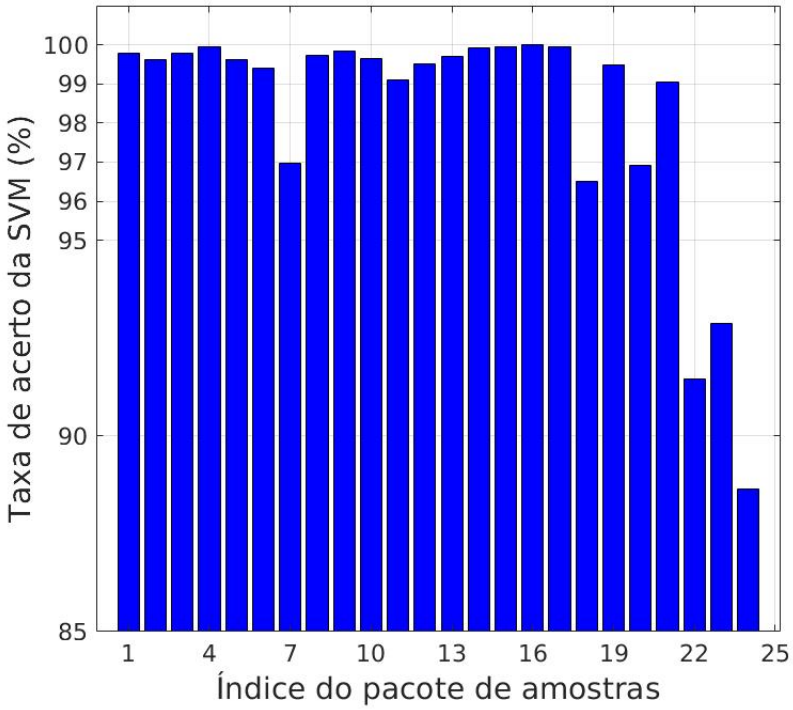
No final, foi feita uma demonstração do funcionamento do sistema de autenticação, utilizando imagens dos *datasets* e imagens obtidas pela webcam interna do Notebook Dell-7567. Não se tem certeza qual tipo de sensor é utilizado por esse webcam, mas por conhecimentos prévios se pode inferir que é CMOS, porque a resolução é baixa, a imagem perde a nitidez quando há um movimento brusco entre os *frames* e porque é o tipo de sensor mais utilizado em webcams de notebooks. A resolução de captura utilizada foi de 640 x 480 pixels e a taxa de captura de 30 *frames* por segundo. O algoritmo é executado em 1 *frame* a cada 15, ou 0,5 segundos. O processador utilizado para rodar o sistema foi o i7-7700HQ Quad Core, utilizando apenas uma *thread*, e a linguagem de programação utilizada foi Python versão 3.6.5.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

4.1 RESULTADOS DOS TREINAMENTOS DA SVM

Para o modelo de SVM treinado com 2.000 imagens de faces e 10.000 imagens de não faces, foi feito o teste a cada 10.000 amostras e medida a taxa de acerto da SVM. Os resultados são mostrados na Figura 53.

Figura 53 – Taxa de acerto da primeira SVM treinada para cada pacote de amostras

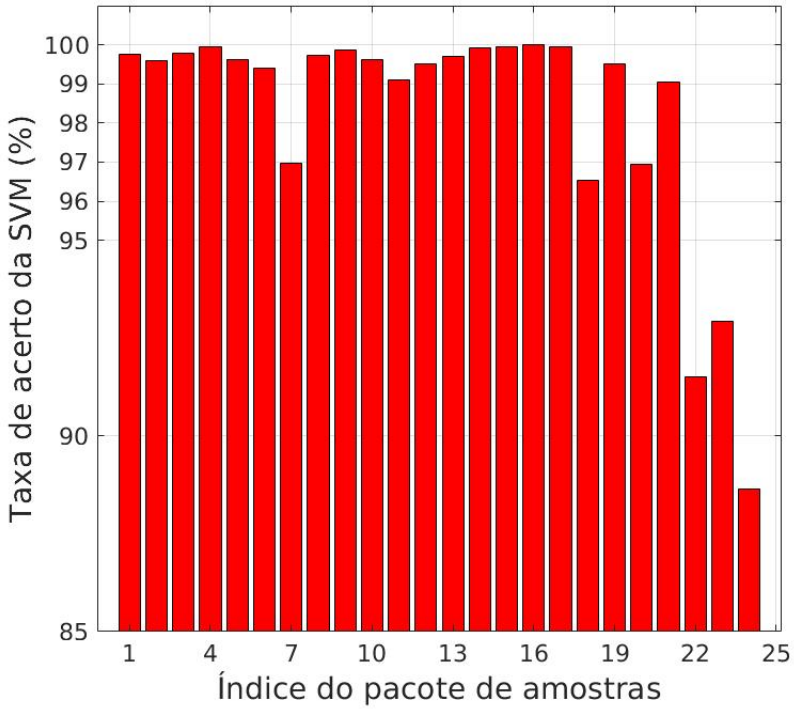


Fonte: Autoria própria.

A média da taxa de acerto para esse treinamento é de 98,226%.

Para o modelo retreinado com o processo de *hard mining*, foram obtidos os resultados mostrados na Figura 54.

Figura 54 – Taxa de acerto da SVM retreinada com hard mining para cada pacote de amostras



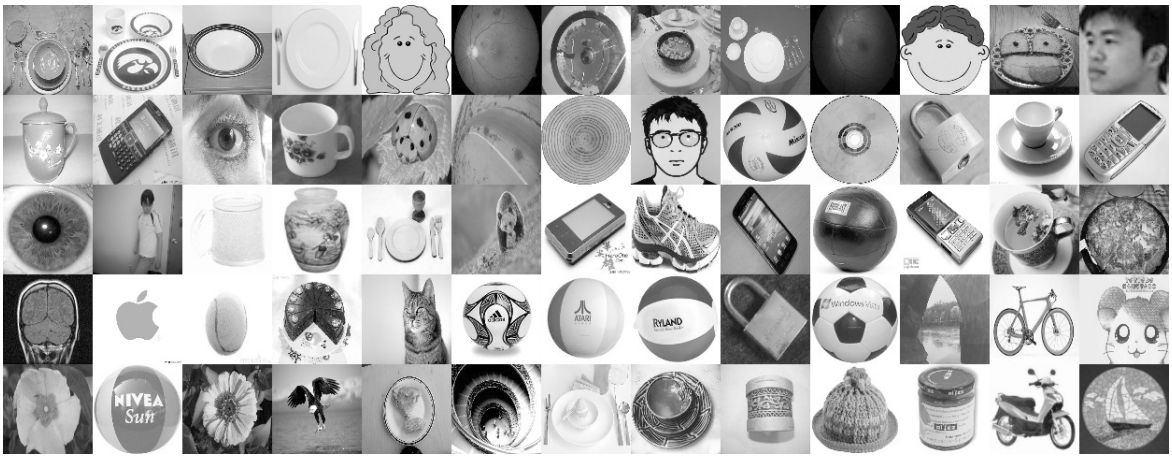
Fonte: Autoria própria.

A média da taxa de acerto para esse treinamento foi de 98,233%. Esse resultado mostrou que o processo de *hard mining* não melhorou significativamente o resultado.

Pode-se notar pelos dois resultados que os últimos 3 pacotes de dados apresentaram uma taxa de acerto bem menor que os outros. Foi feita uma investigação sobre as imagens desses pacotes que foram classificadas de forma errada e percebeu-se um padrão entre algumas delas. A maioria das imagens que foram classificadas de forma errada possuía alguma forma elíptica ou circular predominante, como planetas, bolas e pratos. Além disso, foi percebido que algumas imagens de faces foram rotuladas de forma errada e foram classificadas corretamente, porém geraram resultados errados por esse motivo. Também, algumas imagens de objetos eletrônicos como calculadoras e celulares foram classificadas como faces. Além desses 3 padrões de dados, houveram algumas imagens que não se parecem com faces, mas foram classificadas como faces e não há um padrão determinado entre elas. As taxas de acerto mais baixas nesses pacotes ocorreram também porque haviam diversas cópias das mesmas

imagens que foram classificadas de forma errada. A Figura 55 apresenta algumas das imagens que foram classificadas de forma errada.

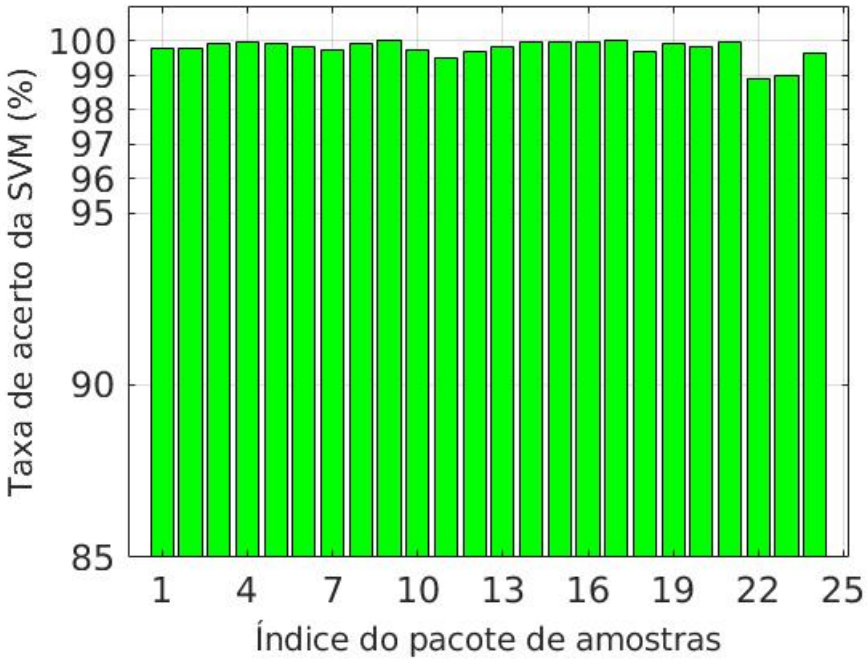
Figura 55 – Algumas imagens dos pacotes 22, 23 e 24 que foram classificadas de forma errada



Fonte: Autoria própria.

Finalmente, no terceiro treinamento foram utilizadas novos conjuntos de imagens, e um total de 10.000 imagens de faces e 10.000 imagens de não faces. Os resultados são mostrados na Figura 56.

Figura 56 – Taxa de acerto da terceira SVM treinada para cada pacote de amostras



Fonte: Autoria própria.

A média de acerto dessa SVM foi de 99,75%. Essa foi a maior taxa de acerto de todos os treinamentos e também se pode notar pelo gráfico que o desvio padrão diminuiu. Pode-se sugerir por esse resultado que o método de *hard mining* feito no treinamento anterior não produziu resultados melhores porque a pequena quantidade de dados utilizadoss influenciou mais no resultado do que o *hard mining*, e isso explicaria os dois resultados anteriores estarem bem próximos. Esse resultado foi considerado suficientemente bom e não foi feito um *hard mining* para tentar melhorá-lo, portanto essa foi a SVM utilizada no sistema.

4.2 RESULTADOS DA MUDANÇA DE PARÂMETROS DO DETECTOR DE FACES

Os parâmetros do detector de face foram alterados diversas vezes de forma empírica. Os resultados a seguir mostram o que acontece com o tempo de execução do programa e com a região escolhida ao se mudar alguns parâmetros. As imagens escolhidas para os testes são mostradas na Figura 57.

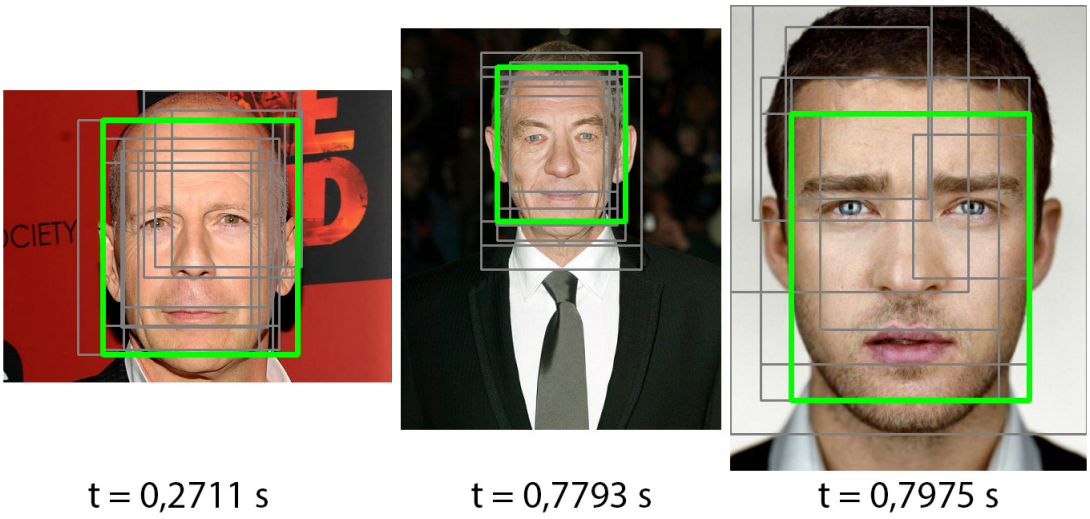
Figura 57 – Imagens usadas como exemplo de mudança de parâmetros do detector de face

Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

4.2.1 Mudança do limiar de distância do hiperplano

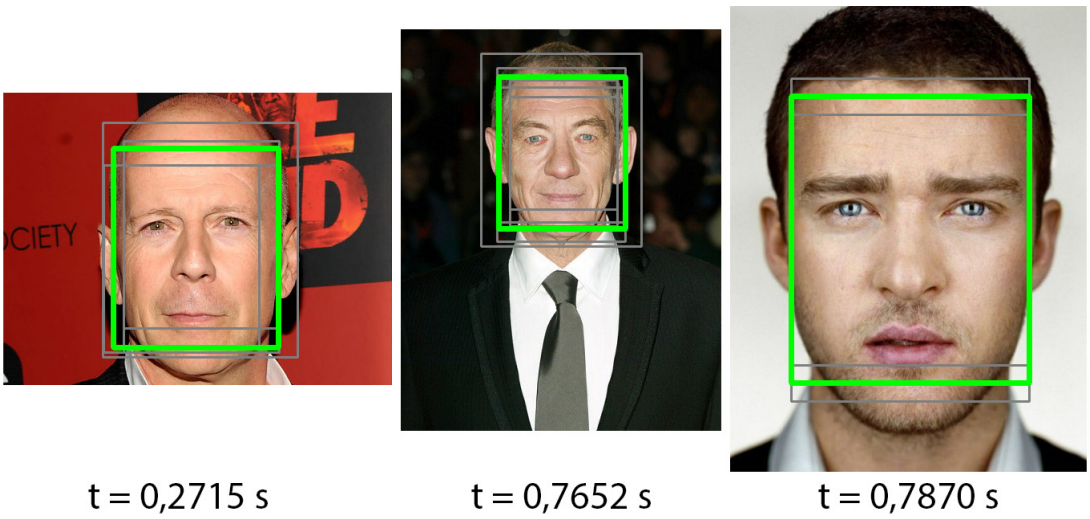
O limiar de distância do hiperplano é um parâmetro utilizado para se definir um valor mínimo de distância do vetor ao hiperplano de decisão da SVM, para se confiar na classificação dada por ela. Um limiar pequeno considera a decisão de vetores muito próximos do hiperplano corretas e um limiar grande só considera a decisão de regiões que seguramente são uma face. A Figura 58 mostra as regiões detectadas em cinza e as regiões escolhidas em verde para um limiar de 0,5 e a Figura 59 mostra essas regiões para um limiar de 1,5. Os tempos de execução do programa são indicados abaixo de cada figura.

Figura 58 – Regiões detectadas para um limiar de decisão de 0,5



Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

Figura 59 – Regiões detectadas para um limiar de decisão de 1,5



Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

Nota-se que a mudança de limiar não afeta o tempo de execução do programa, pois a quantidade de janelas varridas se mantém a mesma, porém pode-se notar pelas imagens que mudar esse parâmetro muda a quantidade de regiões detectadas.

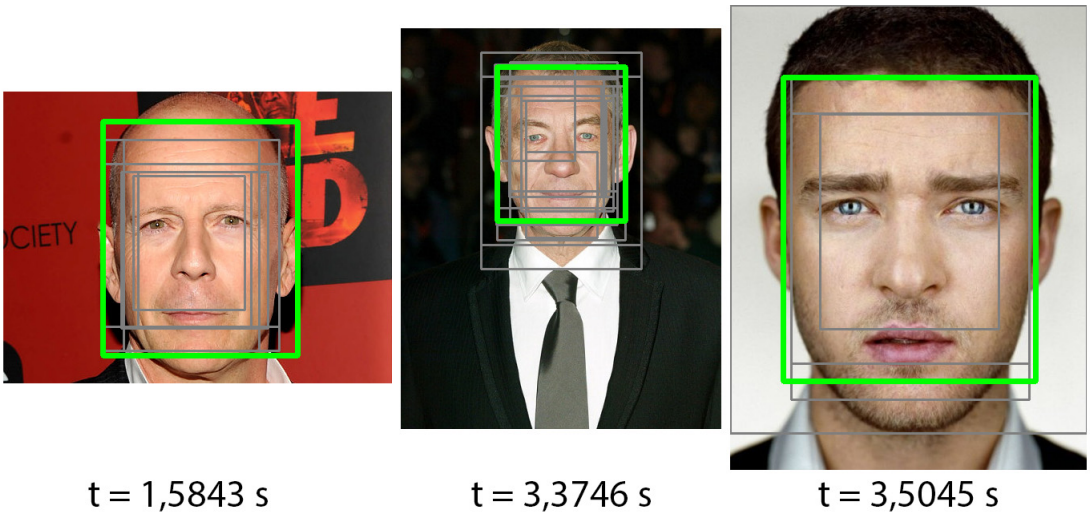
4.2.2 Mudança do número de redimensionamentos da janela

Esse parâmetro afeta o número de vezes que a janela será diminuída para encontrar faces de pessoas mais distantes da câmera. Uma janela menor fará mais varreduras na imagem. A Figura 60 mostra o resultado das detecções para um número de 2 redimensionamentos e a Figura 61 mostra o resultado das detecções para 8 redimensionamentos, assim como os respectivos tempos de execução do código.

Figura 60 – Regiões detectadas para 2 redimensionamentos da janela de detecção

Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

Figura 61 – Regiões detectadas para 8 redimensionamentos da janela de detecção



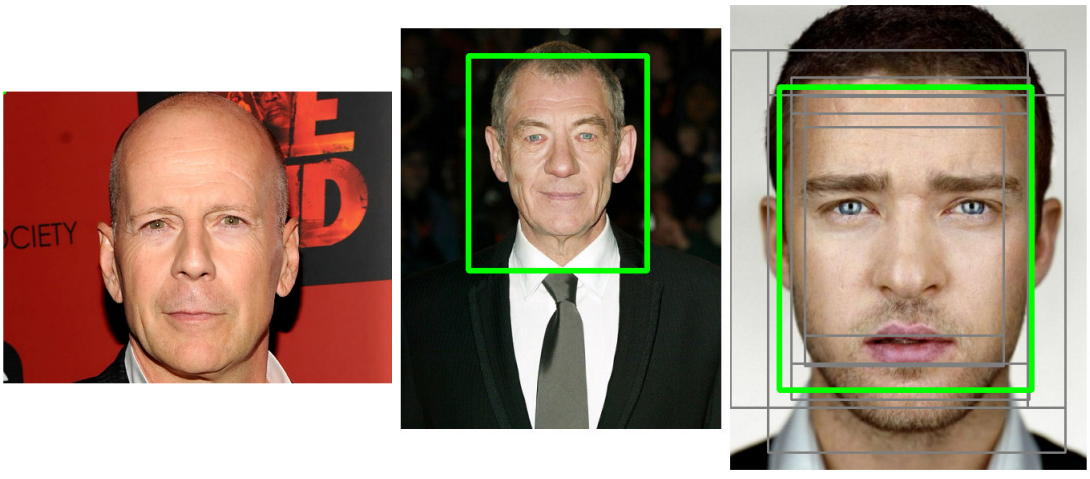
Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

A primeira coisa a se notar é a grande mudança no tempo de execução do algoritmo, que chega a ser aproximadamente 500% de diferença para a primeira imagem. Era esperado que um número maior de redimensionamentos afetasse o tempo de execução de forma não linear, pois cada janela menor executa mais varreduras que a anterior. A segunda coisa a se notar é que para 2 redimensionamentos não foram detectadas as faces das duas primeiras imagens, pois as janelas são muito grandes e isso combinado com um limiar de distância do hiperplano não muito pequeno não permite que o detector considere essas regiões como faces.

4.2.3 Mudança da taxa de redução do tamanho da janela

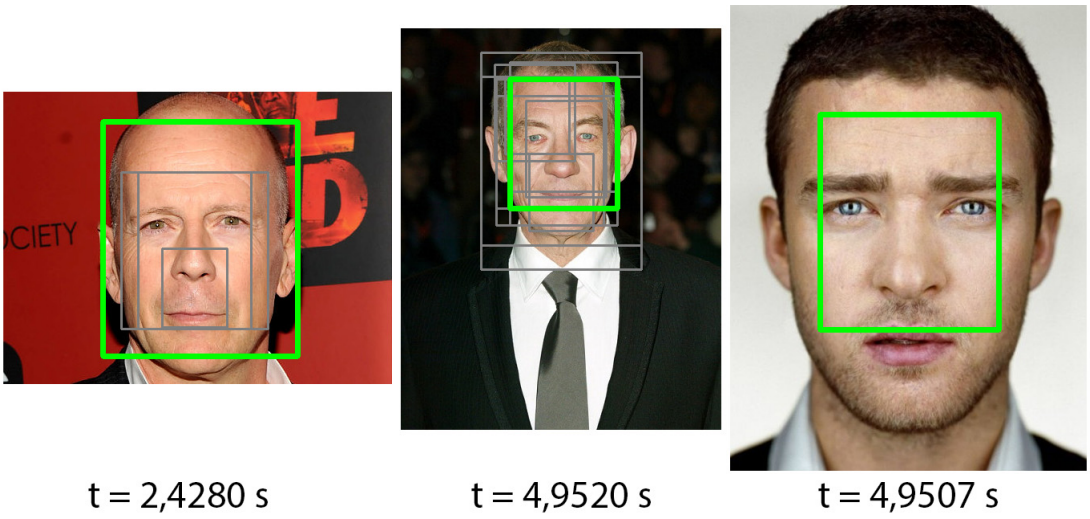
Essa taxa afeta o quão a próxima janela será menor que a anterior. Quanto maior esse número, menor o tamanho da janela em relação à anterior. A Figura 62 indica as regiões detectadas para uma taxa de redução de 0,3 e a Figura 63 indica as regiões para uma taxa de redução de 1, assim como os tempos de execução do algoritmo.

Figura 62 – Regiões detectadas para uma taxa de redimensionamento da janela de 0,3



Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

Figura 63 – Regiões detectadas para uma taxa de redimensionamento da janela de 1



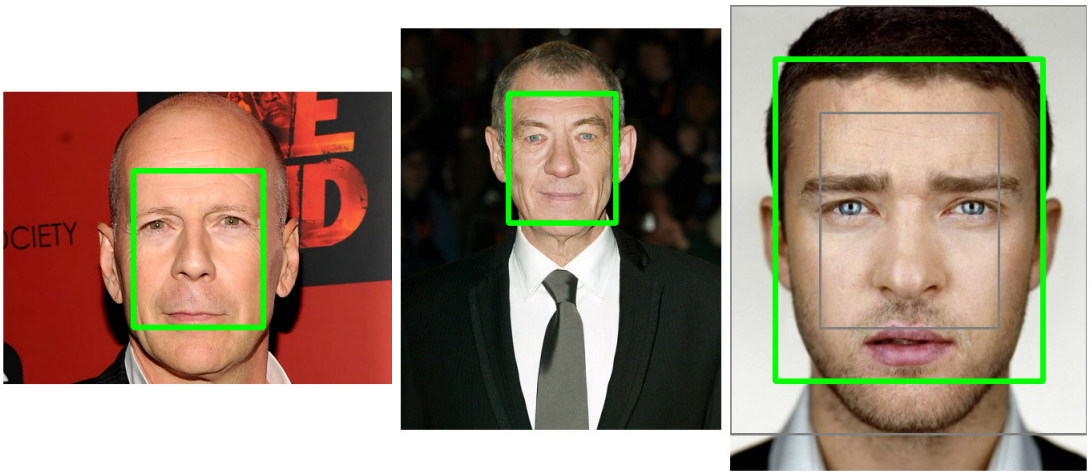
Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

Novamente os tempos de execução são muito afetados, pois para janelas menores, serão feitas mais varreduras, logo mais cálculos. Uma coisa a importante a se notar é que para a taxa de 0,3 a face da primeira imagem não foi detectada, provavelmente porque a redução da primeira janela foi tão pequena que não havia uma janela pequena o suficiente para detectar aquela face, e para a taxa de 1 a face da última imagem foi detectada de uma forma não tão precisa, provavelmente pelo problema oposto ao anterior: não havia uma janela grande o suficiente para detectar a face completa.

4.2.4 Mudança do passo de varredura da janela

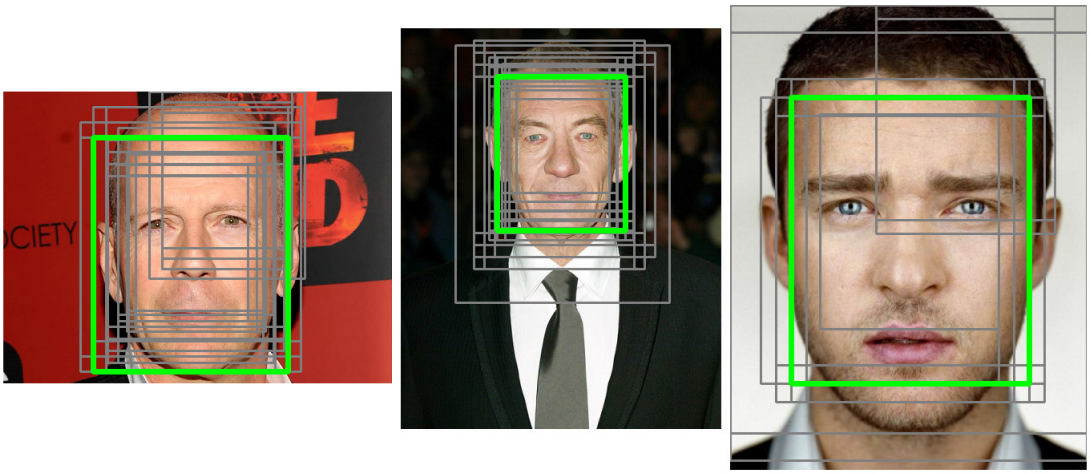
O passo de varredura da janela, chamado de *stride*, indica a quantidade de pixels que a janela deve se mover cada vez. O maior passo possível é o próprio tamanho da janela e o menor passo possível é de 1 pixel. Quanto maior for esse valor, menos cálculos serão feitos, porém a janela pode falhar em detectar faces que estão localizadas entre duas janelas consecutivas. A Figura 64 e a Figura 65 indicam as regiões detectadas para um passo de 1/2 e 1/16 do tamanho da janela, respectivamente, assim como os tempos de execução do algoritmo.

Figura 64 – Regiões detectadas para um stride de 1/2 do tamanho da janela



Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

Figura 65 – Regiões detectadas para um stride de 1/16 do tamanho da janela



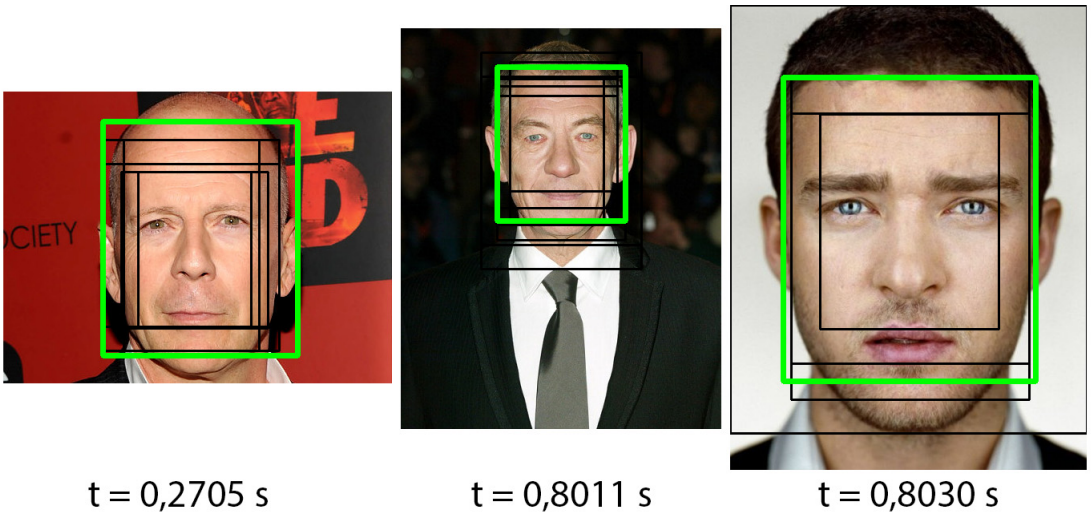
Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

O tempo de execução novamente é muito afetado, porque com um passo menor são feitos mais cálculos por haverem mais janelas. Outra coisa que é afetada por isso é que várias janelas consecutivas detectam a mesma face, por estarem muito próximas umas das outras, portanto a quantidade de regiões detectadas é maior.

4.2.5 Parâmetros ótimos

Foram feitas diversas variações nesses parâmetros e mesmo que alguns problemas não tenham aparecido nas imagens utilizadas como exemplo, eles ocorreram em outras imagens do conjunto de imagens utilizados para testar essa variações. Se chegou a conclusão que houve um bom equilíbrio do tempo de execução do código e da qualidade das regiões detectadas ao se utilizar os seguintes parâmetros: limiar de distância = 0,8, número de redimensionamentos da janela = 5, taxa de redução do tamanho da janela = 0,5 e passo de varredura = 1/8 do tamanho da janela. A Figura 66 mostra as regiões de detecção e os tempos de execução para as imagens utilizadas de exemplo utilizando esses parâmetros.

Figura 66 – Regiões detectadas paras as imagens de exemplo utilizando os parâmetros ótimos



Fonte: Adaptado de LOVEKIN (2013), MARVEL MOVIES (2008) e LOFT965 (2013).

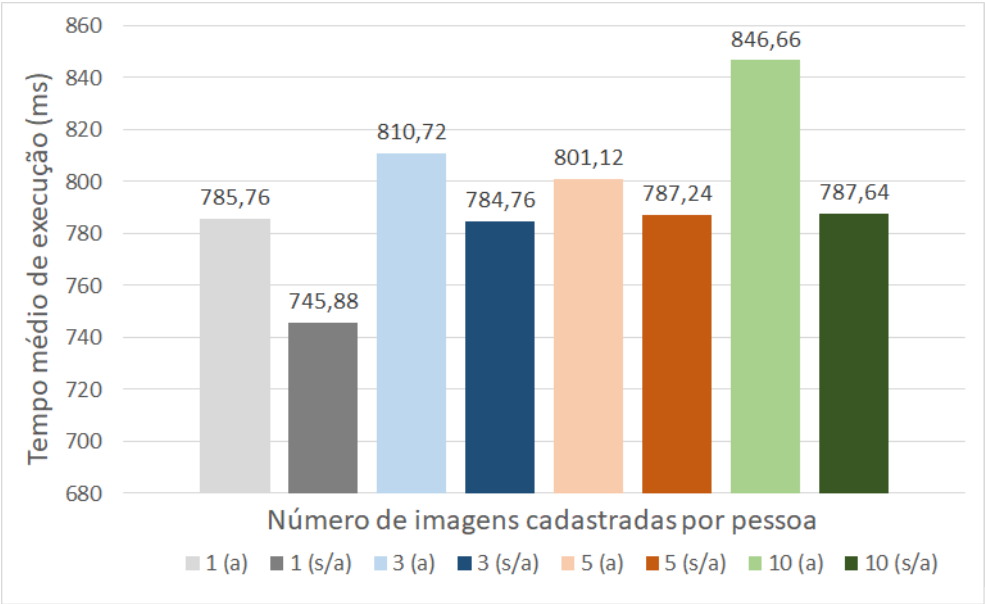
4.3 TESTES DE DESEMPENHO DO SISTEMA

Como dito na metodologia, foram selecionadas 20 atores e 20 atrizes arbitrários do *dataset* para que suas imagens fossem usadas para medir o desempenho do sistema. Foram medidas a FRR, que é a porcentagem de decisões negativas incorretas, a FAR, que é a porcentagem de decisões positivas incorretas e o tempo de execu-

ção médio do algoritmo. Deve-se notar que uma decisão positiva incorreta causa um impacto negativo muito maior que uma decisão negativa incorreta em um sistema de autenticacão, pois isso está relacionado diretamente ao nível segurança.

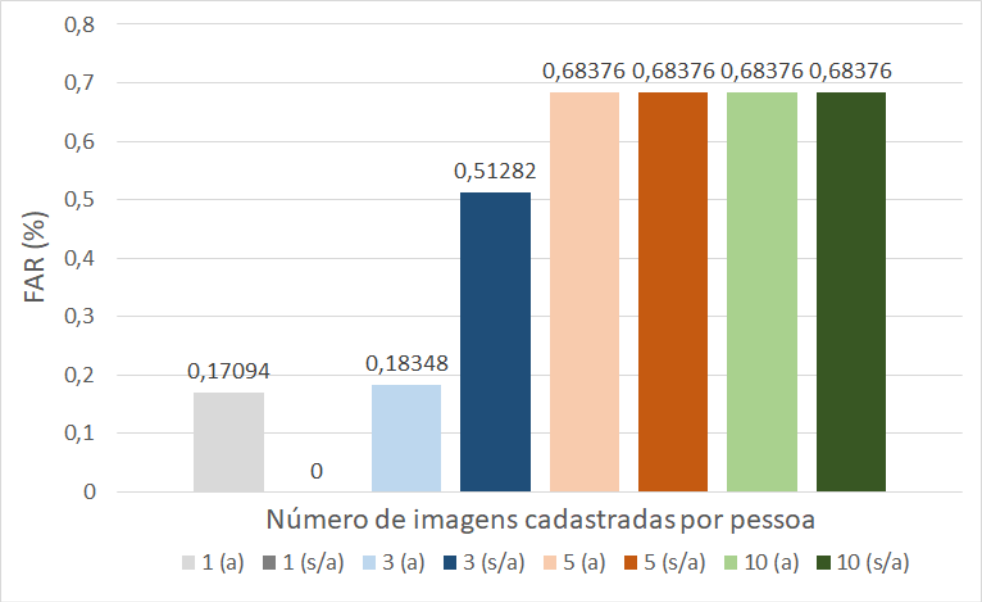
Devido ao tempo de realização dos testes, não foi possível fazer um teste para todas as pessoas, por isso foram selecionadas 5 pessoas arbitrárias das 40 e realizado um teste individual para cada uma delas. Para cada teste foi medida a FAR, a FRR e o tempo de execução do algoritmo e para cada pessoa foram utilizadas entre 20 a 60 imagens da mesma e cerca de 60 imagens das outras 39 pessoas. Foi feita uma média dos 5 resultados de cada medida para se obter o desempenho final do sistema. Esse desempenho foi medido para 1, 3, 5 e 10 imagens cadastradas de cada indivíduo e com a utilização ou não da etapa de reposicionamento da face, totalizando 8 conjuntos de testes. As Figuras 67, 68 e 69 apresentam o tempo médio de execução de uma verificação, a FAR e a FRR, respectivamente, em função do número de imagens pré-cadastradas de cada indivíduo. A marcação (a) significa que foi utilizada a etapa de alinhamento da imagem e a marcação (s/a) indica que ela não foi utilizada.

Figura 67 – Tempo de execução médio em função do número de imagens cadastradas do indivíduo



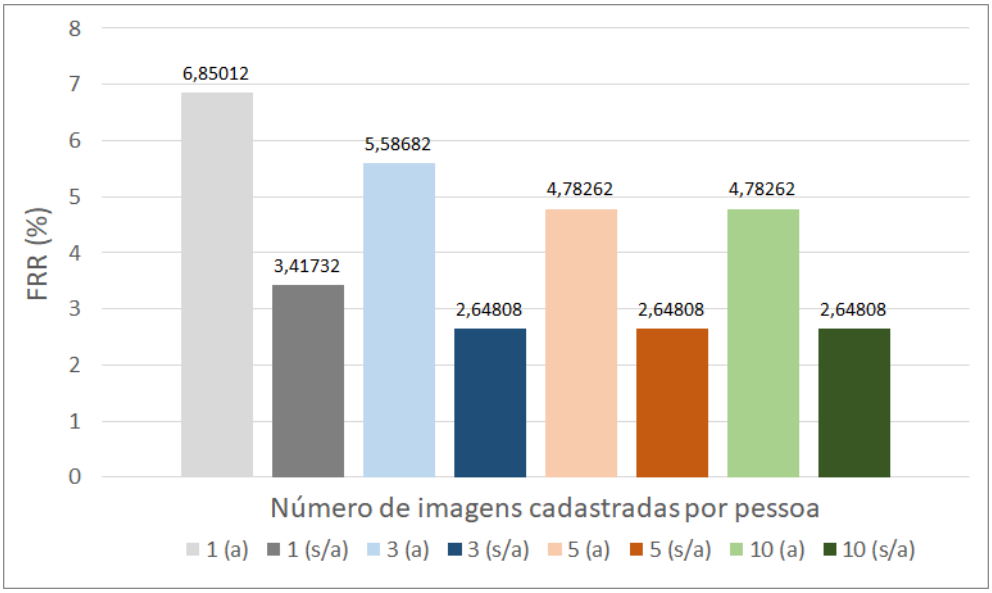
Fonte: Autoria própria.

Figura 68 – Taxa de falsa aceitação (FAR) em função do número de imagens cadastradas do indivíduo



Fonte: Autoria própria.

Figura 69 – Taxa de falsa rejeição (FRR) em função do número de imagens cadastradas do indivíduo



Fonte: Autoria própria.

Pelos gráfico dos tempos de execução, ocorre algo que já era esperado, que é a diminuição do tempo com a ausência da etapa de alinhamento da face. Também se pode notar que o aumento do número de faces cadastradas possui um impacto muito

pequeno no tempo de execução médio. Portanto, pode-se afirmar que o tempo de execução do código é afetado de forma desprezível por essas escolhas.

Pelo gráfico da FAR se pode notar que ela é maior quanto mais imagens pré cadastras são utilizadas, mas para 5 e 10 o valor é o mesmo. O ideal seria utilizar o menor valor possível que é com 1 imagem e sem a etapa de alinhamento. De acordo com Thakkar (2017), o valor médio dos sistemas de reconhecimento facial comerciais é de 0,1%.

Pelo gráfico da FRR é possível se afirmar que ela é sempre menor quando se omite a etapa de alinhamento e e que a partir de 3 imagens faces esse valor não diminui mais. O valor médio da FRR dos sistemas comerciais é de 6% (THAKKAR, 2017).

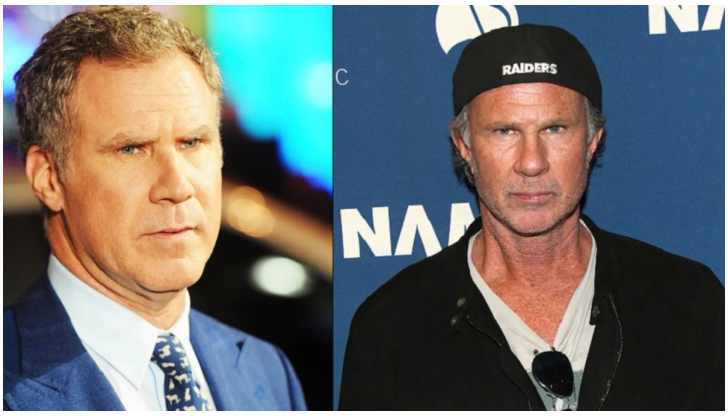
Pelo resultado dos gráficos a melhor escolha possível seria se utilizar apenas 1 imagem cadastrada e omitir o alinhamento da face. O sistema foi testado manualmente diversas vezes para ser avaliado o desempenho de forma qualitativa e foi verificado que apesar dos resultados estatísticos levarem à afirmação anterior, o valor ideal de imagens cadastradas está entre 3 e 5. Isso acontece porque ter imagens do indivíduo em mais de uma pose, com e sem óculos e com diferentes expressões faciais contribui bastante para o desempenho do sistema. O resultado anterior não foi comprovado porque a análise foi feita unicamente de forma estatística e foram utilizados poucos dados de validação. Isso é sustentado ao se notar que foram obtidos resultados de FAR e FRR melhores que nos sistemas comerciais. Um resultado melhor provavelmente pode ser obtido ao se fazer uma análise qualitativa e quantitativa. Isso pode ser feito ao se selecionar as imagens de validação de forma minuciosa de modo que se tenha garantia que o rótulo da imagem é o correto, que o detector facial possa identificar a face na imagem, que a face da pessoa não esteja coberta por algum objeto como uma máscara ou um óculos, que se tenha uma única pessoa na imagem (para satisfazer a restrição do detector facial), que o rosto da pessoa verificada não tenha envelhecido em relação às imagens cadastradas, entre outros fatores. Após o banco de imagens ser selecionado, a abordagem estatística utilizada anteriormente poderia apresentar melhores resultados.

4.4 REFINAMENTO DO SISTEMA POR DECISÕES DIFÍCEIS

Para melhorar mais ainda o desempenho do sistema, foram feitos testes qualitativos, que aqui são chamados de decisões difíceis, para se encontrar o limiar ideal de distância entre as amostras conhecidas e as amostras novas que será utilizado na classificação.

Uma decisão difícil é uma verificação feita pelo sistema entre duas pessoas com o rosto muito parecido, a critério do autor. Um exemplo dessas pessoas é apresentado na Figura 70.

Figura 70 – Par de indivíduos que caracterizam uma decisão difícil



Fonte: HUFF (2017).

Foram testados 10 pares dessas pessoas, com cerca de 5 imagens de cada pessoa, de forma que o limiar escolhido fosse ajustado manualmente para minimizar a FAR desse conjunto de amostras. A lista dos pares escolhidos pode ser vista no Apêndice 1. Ao final dessa etapa, foi encontrado um limiar de 0,5 para a distância euclidiana entre os encodings previamente gravados e os novos. As Figuras 71, 72 e 73 mostram o resultado de algumas decisões difíceis após o ajuste do limiar.

Figura 71 – Exemplo 1 de resultados de uma decisão difícil após o ajuste do limiar

Fonte: Adaptado de COUNTESS (2015).

Figura 72 – Exemplo 2 de resultados de uma decisão difícil após o ajuste do limiar

Fonte: Adaptado de 4EVER (2006).

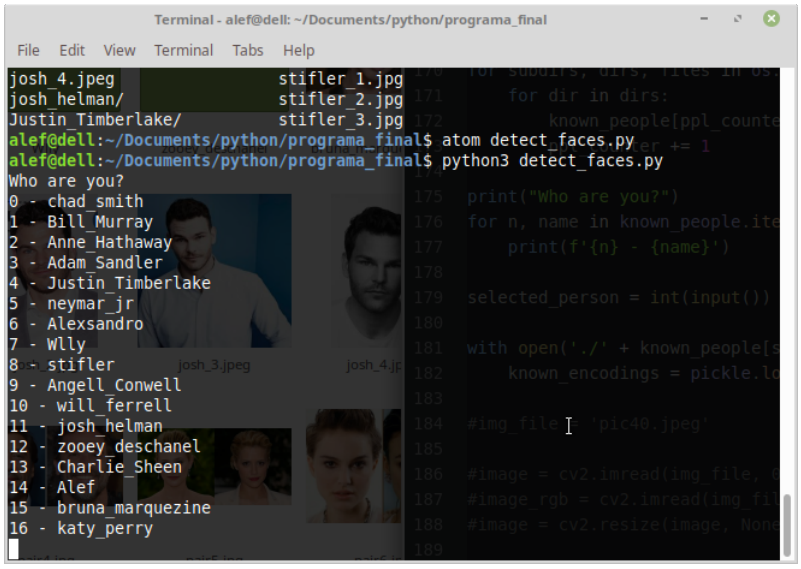
Figura 73 – Exemplo 3 de resultados de uma decisão difícil após o ajuste do limiar

Fonte: Adaptado de VAGALUME (2012) e BUKLEY (2006).

4.5 FUNCIONAMENTO DO SISTEMA

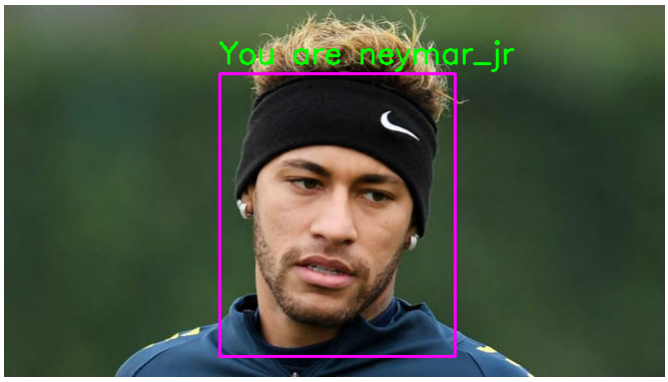
Previamente são feitos os cadastros de todos os usuários, utilizando de 3 a 5 fotos de cada um deles. O usuário então ao rodar o programa deve declarar quem ele é, dentre os usuários cadastrados, como é demonstrado na Figura 74. O sistema então verifica se a declaração do usuário é verdadeira ou falsa. As Figuras 75, 76 e 77 mostram o resultado da verificação de imagens de algumas pessoas famosas.

Figura 74 – Tela do programa onde o usuário declara quem ele é



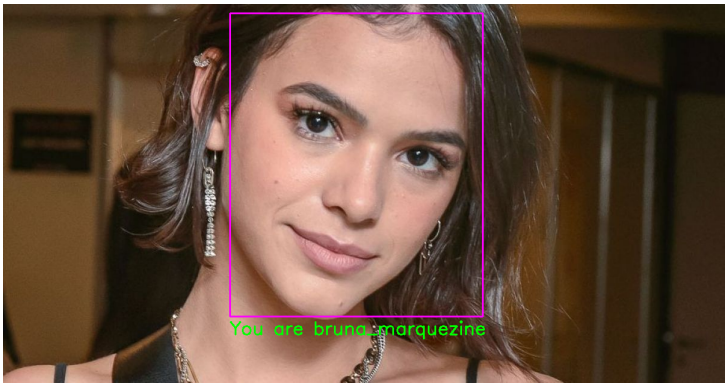
Fonte: Autoria própria.

Figura 75 – Exemplo 1 de funcionamento do sistema com imagens de pessoas famosas



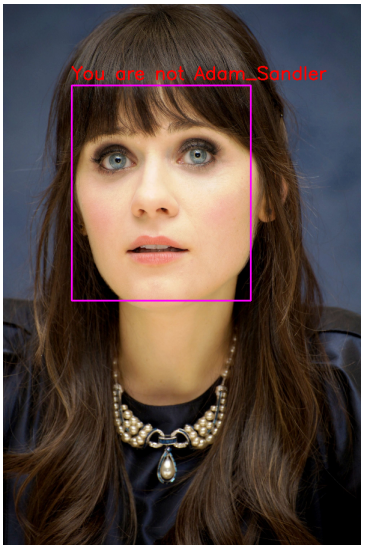
Fonte: Adaptado de EFE (2018).

Figura 76 – Exemplo 2 de funcionamento do sistema com imagens pessoas famosas



Fonte: Adaptado de CEPEDA (2019).

Figura 77 – Exemplo 3 de funcionamento do sistema com imagens de pessoas famosas

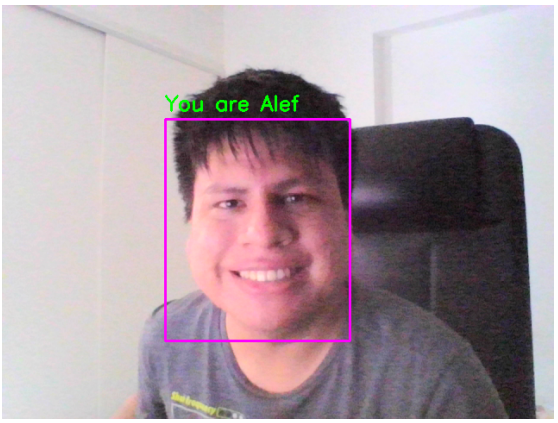


Fonte: Adaptado de THE PLACE (2014).

Pode-se notar que mesmo com os rostos não estando totalmente retos nas imagens o sistema ainda faz a verificação de forma correta.

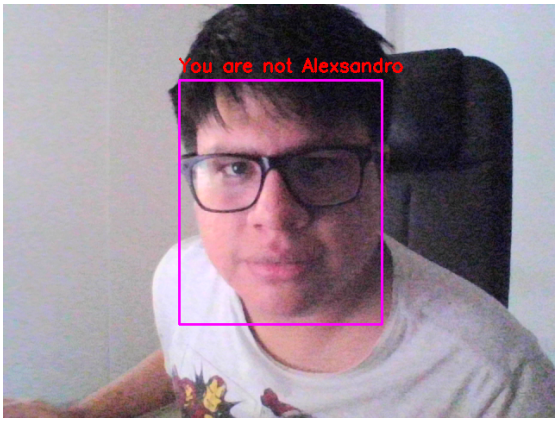
As Figuras 78, 79 e 80 mostram, respectivamente, uma pessoa sendo verificado para a sua própria entrada, para a entrada de outro usuário, e para a própria entrada com o rosto levemente virado. Essas imagens foram obtidas pela webcam.

Figura 78 – Exemplo 1 de funcionamento do sistema com a webcam



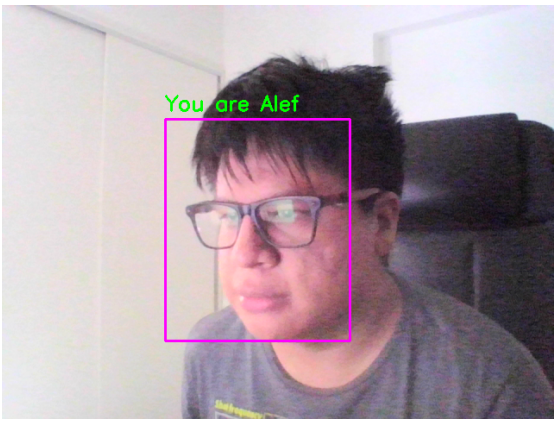
Fonte: Autoria própria.

Figura 79 – Exemplo 2 de funcionamento do sistema com a webcam



Fonte: Autoria própria.

Figura 80 – Exemplo 3 de funcionamento do sistema com a webcam



Fonte: Autoria própria.

Deve-se destacar que mesmo com o usuário usando óculos, fazendo uma expressão facial não-neutra, com rosto levemente virado e com a qualidade das imagens obtidas pela webcam muito inferior à das imagens do *dataset*, o sistema ainda faz a verificação de forma correta. Isso mostra a importância do número e da variedade das imagens utilizadas no cadastramento do usuário.

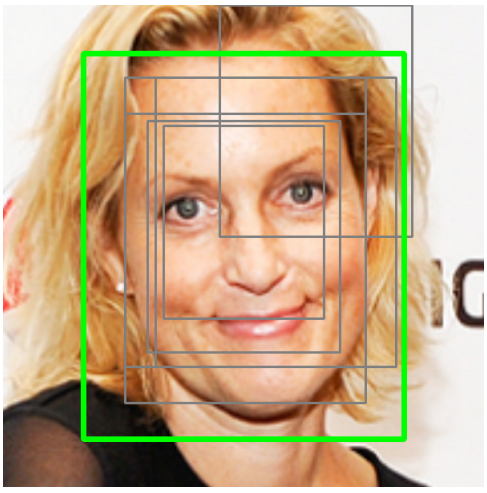
O tempo de execução médio medido para o programa fazer a autenticação ao se utilizar as imagens obtidas pela webcam foi de 2,7 s.

4.6 PROBLEMAS DO SISTEMA E SUGESTÕES DE CORREÇÃO

4.6.1 Detector de face

O primeiro problema do sistema é o detector de face projetado. Ele possui um tempo de detecção de, em média, 430 ms para as imagens obtidas pela webcam. Esse valor é muito alto em relação aos detectores no estado da arte que chegam até 200 ms se for utilizada uma CPU para o processamento. Isso se dá porque o descritor HOG calculado das imagens possui um tamanho muito grande, de 39204 dimensões. Além disso, como mostrado na Figura 81, há ainda algumas regiões que não são face que são detectadas como face e dependendo de quantas regiões dessa são detectadas, isso leva a uma decisão final de má qualidade.

Figura 81 – Problema de regiões detectadas erroneamente como face pelo detector



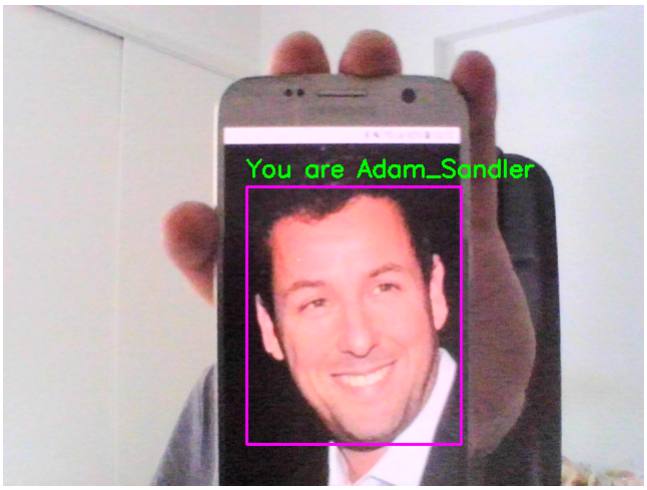
Fonte: Adaptado de GETTY IMAGES (2014).

Para se resolver o problema do tempo de detecção, a sugestão é diminuir o tamanho do descritor HOG, e isso é feito ao se diminuir o tamanho da imagem que é utilizada para calculá-lo. O tamanho atualmente é 96 x 96, e caso fosse reduzido para 48 x 48, o descritor HOG teria 1/4 do tamanho atual. Se isso for feito, a SVM deve ser treinada novamente para poder se utilizar esse novo descritor como entrada. E para se resolver o problema das regiões detectadas erroneamente, a solução é melhorar a SVM, fazendo um treinamento com uma quantidade e qualidade maior dos dados. Com um treinamento melhor, a taxa de erro será menor e menos regiões serão detectadas como face erroneamente.

4.6.2 Imagem estática

Um problema muito comum em sistemas de reconhecimento facial, e que também ocorre no que foi desenvolvido neste trabalho, é que uma imagem estática pode ser reconhecida também. Para um sistema de autenticação real isso não é bom, pois uma pessoa com uma foto de outra poderia enganar o sistema, como ocorre na Figura 82, onde uma foto mostrada no celular pode ser utilizada para burlar o sistema.

Figura 82 – Detecção de imagem estática



Fonte: Autoria própria.

Para se resolver esse problema, há dois caminhos possíveis: correção por *hardware* ou por *software*. A correção por *hardware* gera um custo financeiro maior do sistema e a solução por *software* gera um custo computacional maior.

Uma possível correção por *hardware* seria utilizar um sensor de profundidade no rosto da pessoa. Ele funciona ao se projetar pontos de luz infravermelhos a sua frente e se utilizar uma câmera infravermelha para capturar esses pontos. O princípio básico é que em regiões mais próximas os pontos capturados pela câmera são maiores do que os pontos capturados de regiões mais distantes, assim se pode mapear a distância de cada local da imagem até a câmera. Esse funcionamento é mostrado na Figura 83. Esse sensor pode ser utilizado para diferenciar um rosto estático, que é plano, de um rosto real, que possui uma profundidade.

Figura 83 – Funcionamento de um sensor de profundidade

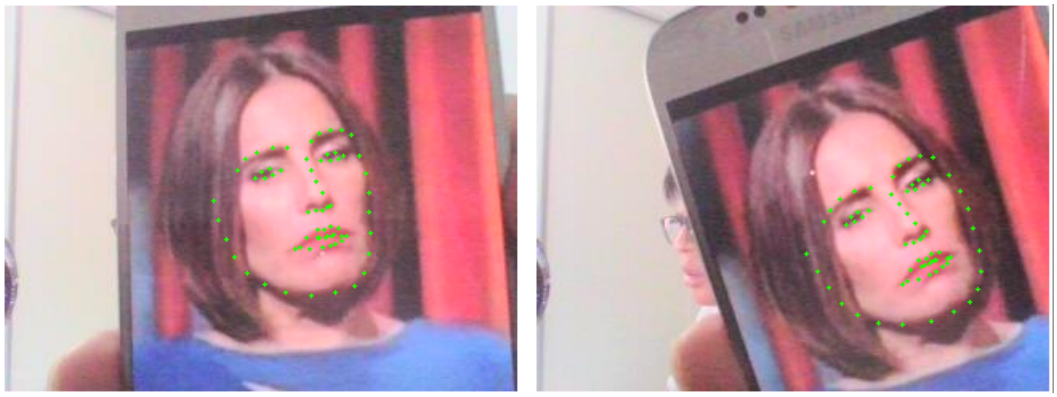
Fonte: FISHER (2014).

Uma possível abordagem por *software* é solicitar que o usuário vire a cabeça levemente e depois se encontrar os pontos de referência da face em dois *frames* diferentes do vídeo. Se a posição relativa entre os pontos for a mesma nos dois *frames* significa que a face é estática e se ela mudar significa que é uma pessoa de verdade na frente da câmera. As Figuras 84 e 85 apresentam essas duas possibilidades.

Figura 84 – Pontos de referência da face de uma pessoa real

Fonte: Autoria própria.

Figura 85 – Pontos de referência da face de uma imagem estática



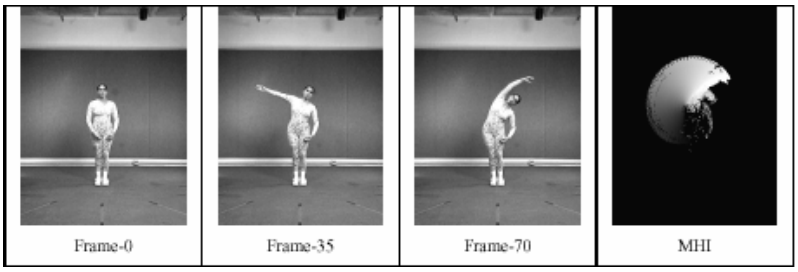
Fonte: Autoria própria.

Nota-se que na imagem estática, mesmo havendo uma rotação e translação, os pontos de referência se mantêm nas mesmas posições relativas no dois *frames*, enquanto na face real as posições mudam ao haver um movimento da cabeça para o lado.

Outra possível abordagem por *software* é solicitar que o usuário faça algum movimento com o rosto, como bocejar ou piscar os olhos, e se criar uma imagem do histórico de movimento (MHI) dessa ação para depois ser verificado se houve o movimento ou não. Para garantir que a pessoa que está sendo autenticada é a mesma que fez o movimento, se captura os frames para a verificação facial durante o movimento. Uma imagem MHI é uma imagem que mostra a variação do pixels ao longo do tempo,

sendo que as variações mais antigas são mostradas em intensidades menores e as variações recentes são mostradas em intensidades maiores. A Figura 86 mostra um exemplo de MHI.

Figura 86 – Exemplo de imagem do histórico de movimento



Fonte: Llorens et al. (2002).

5 CONCLUSÕES E TRABALHOS FUTUROS

Foi apresentada a ideia de que formas clássicas de autenticação são sujeitas a serem fraudadas por falhas humanas e por isso foi sugerido que o uso de reconhecimento biométrico não estaria sujeito a essas falhas. Então, um sistema de autenticação utilizando biometria, neste caso o reconhecimento facial, foi proposto e desenvolvido no trabalho.

Uma das principais dificuldades encontradas durante o desenvolvimento do sistema foi a otimização computacional, que é encontrar meios para ajustar o algoritmo de modo que a mesma tarefa seja feita em um tempo menor. Outro ponto que foi decisivo no desempenho final do sistema está relacionado às limitações dos recursos de *hardware* que estavam à disposição, que não permitiram que fosse utilizado um número maior de dados no treinamento da SVM e que fossem gerados resultados de forma automática, isto é, a escolha de parâmetros ótimos em cada etapa do algoritmo na maioria das vezes teve que ser feita de forma manual e por isso não se pôde garantir que o desempenho máximo encontrado é o desempenho máximo possível. Podem ser feitos estudos sobre os métodos utilizados em cada etapa do algoritmo para a melhoria geral do desempenho do sistema.

Foram feitos dois tipos de análise para se medir o desempenho do sistema: uma quantitativa e uma qualitativa. A quantitativa não gerou resultados confiáveis, porque ela foi feita de forma puramente estatística e o número de amostras utilizadas foi muito pequena. A qualitativa gerou melhores resultados, porém eles não puderam ser traduzidos em valores numéricos. Foi sugerido que um teste de desempenho ideal deveria ser feito ao se combinar os dois tipos de análise, mas as limitações de tempo e *hardware* não permitiram que esse teste fosse feito.

Apesar do objetivo inicial de desenvolver um sistema de autenticação imune à falhas humanas ter sido alcançado, foi verificado que ele pode ser fraudado por uma falha de máquina e foram feitas algumas sugestões de correções para essa falha.

Por fim, diversas técnicas de processamento digital de sinais e *machine learning* foram explicadas e apresentadas no trabalho, e essas mesmas técnicas podem ser utilizadas para resolver diversos outros problemas e se fazer diversas outras aplicações.

6 REFERÊNCIAS

4EVER. Steve Stifler. **4ever**, 2019. Disponível em: <http://pictures.4ever.eu/tag/3409/steve-stifler>. Acesso em: 4 de mai. de 2019.

AL-KHALIDI, Farah Qais; SADIQ, Abdulalla Asmaa; HAMEED, Shaimaa. A survey of human face detection methods. **Journal of AL-Qadisiyah for computer science and mathematics**, vol. 10, n. 2, pp. 108-117, 2018.

AMOROSO, Danilo. O que é autenticação?. **Tecmundo**, 2009. Disponível em: <https://www.tecmundo.com.br/seguranca/1971-o-que-e-autenticacao-.htm>. Acesso em: 4 de mai. de 2019.

ARAÚJO, L. et al. Oficina virtual. **Meninas de Diplomática**, 2015. Disponível em: <http://meninasdediplomatica.blogspot.com/2015/11/oficina-virtual.html>. Acesso em: 4 de mai. de 2019.

BBC. Como funciona o 'Big Brother' da China, com 170 milhões de câmeras que fazem identificação visual. **BBC**, 2017. Disponível em: <https://www.bbc.com/portuguese/internacional-42361047>. Acesso em: 4 de mai. de 2019.

BERWICK, R. An idiot's guide to support vector machines. **Advanced Computer Vision**. **University of Central Florida**, 2003. Disponível em: <http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf>. Acesso em: 4 de mai. de 2019.

BOLLE, Ruud M. et al. **Guide to biometrics**. New York: Springer Science & Business Media, 2013.

BOM DIA RIO. Câmeras com reconhecimento facial são instaladas em Copacabana durante o Carnaval. **G1**, 2019. Disponível em: <https://g1.globo.com/rj/rio-de-janeiro/noticia/2019/03/01/cameras-com-reconhecimento-facial-sao-instaladas-em-copacabana-durante-o-carnaval.ghtml>. Acesso em: 4 de mai. de 2019.

BUKLEY, S. Zooey Deschanel – Fotografía editorial de stock. **Depositphotos**, 2006. Disponível em: <https://mx.depositphotos.com/16459621/stock-photo-zooey-deschanel.html>. Acesso em: 4 de mai. de 2019.

BUNTON, Cam. Samsung Galaxy Note 7 iris scanner: What is it and how does it work?. **Pocket-lint**, 2016. Disponível em: <https://www.pocket-lint.com/phones/news/samsung/138335-samsung-galaxy-note-7-iris-scanner-what-is-it-and-how-does-it-work>. Acesso em: 4 de mai. de 2019.

CANTARINO BRASILEIRO. 79% dos internautas brasileiros usam aplicativos de bancos . **Cantarino Brasileiro**, 2017. Disponível em: <http://cantarinobrasileiro.com.br/blog/79-dos-internautas-brasileiros-usam-aplicativos-de-bancos/>. Acesso em: 4 de mai. de 2019.

CEPEDA, Francisco. Bruna marquezine relata espancamento de colega gay e se posiciona: ”vai além da política”. **Quem**, 2018. Disponível em: <https://revistaquem.globo.com/QUEM-News/noticia/2018/10/bruna-marquezine-relata-espancamento-de-colega-gay-e-se-posiciona-vai-alem-da-politica.html>. Acesso em: 4 de mai. de 2019.

COUNTESS, Jemal. Here’s what the ”Mad Max”cast looks like before and after. **BuzzFeed**, 2015. Disponível em: <https://www.buzzfeed.com/juliapugachevsky/heres-what-the-mad-max-fury-road-cast-looks-like-before-and?utm_term=.ffrP1eJX5&sub=3784526_5778075>. Acesso em: 4 de mai. de 2019.

CRUZ, Juliano E. C.; SHIGUEMORI, Elcio H.; GUIMARÃES, Lamartine N. F. A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery. **Journal of Computational Interdisciplinary Sciences**, vol. 6, n. 3, pp. 121-136, 2015.

DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: 2005 IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 1., 2005, San Diego. **Anais...** San Diego: IEEE Computer Society, 2010. pp. 886-893.

DAS, Swagatam. How electret microphones work – full tutorial and diagram. **Home-made Circuit Projects**, 2019. Disponível em: <https://www.homemade-circuits.com/how-electret-microphone-works/>. Acesso em: 4 de mai. de 2019.

DAUGMAN, John; DOWNING, Cathryn. Epigenetic randomness, complexity and singularity of human iris patterns. **Proceedings of the Royal Society of London. Series B: Biological Sciences**, vol. 268, n. 1477, pp. 1737-1740, 2001.

DEVELLE, Yuji. The forgotten origin of passwords. **LinkedIn Pulse**, 2016. Disponível em: <https://www.linkedin.com/pulse/forgotten-utility-passwords-what-we-need-learn-yuji-develle?trk=prof-post>. Acesso em: 4 de mai. de 2019.

EFE. Barcelona not ruling out Neymar return. **Marca**, 2018. Disponível em: <https://www.marca.com/en/football/barcelona/2018/11/18/5bf08a17e5fdea62088b4627.html>. Acesso em: 4 de mai. de 2019.

FERREIRA, Elnatan Chagas. Conversão AD e DA – Técnicas. **Eletrônica Digital**. **Universidade de Campinas**, 2018. Disponível em: <http://www.demic.fee.unicamp.br/~elnatan/ee610/19a%20Aula.pdf>. Acesso em: 4 de mai. de 2019.

FISHER, Matthew. Kinect. **Matt's Webcorner**, 2014. Disponível em: <http://graphics.stanford.edu/~mdfisher/Kinect.html>. Acesso em: 4 de mai. de 2019.

G1. Facebook completa 15 anos com 2,3 bilhões de usuários. **G1**, 2019. Disponível em: <https://g1.globo.com/economia/tecnologia/noticia/2019/02/04/facebook-completa-15-anos-com-23-bilhoes-de-usuarios.ghtml>. Acesso em: 4 de mai. de 2019.

GAHUKAR, Gaurav. Classification Algorithms in Machine Learning. **Medium**, 2018. Disponível em: <https://medium.com/datadriveninvestor/classification-algorithms-in-machine-learning-85c0ab65ff4>. Acesso em: 4 de mai. de 2019.

GAMA, João. Árvores de decisão. **Universidade do Porto**, 2003. Disponível em: <https://www.dcc.fc.up.pt/~ines/aulas/MIM/arvores_de_decisao.pdf>. Acesso em: 4 de mai. de 2019.

GEITGEY, Adam. Machine learning is fun! Part 4: modern face recognition with deep learning. **Medium**, 2016. Disponível em: <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>. Acesso em: 4 de mai. de 2019.

GETTY IMAGES. Ali Wentworth, 50, reveals Lena Dunham inspired her to get Botox after the Girls star called her ABC anchor husband George Stephanopoulos 'sexual'. **Mail Online**, 2015. Disponível em: <https://www.dailymail.co.uk/tvshowbiz/article-3117392/Ali-Wentworth-50-reveals-Lena-Dunham-inspired-Botox-gushed-sexual-icon-husband-George-Stephanopoulos.html>. Acesso em: 4 de mai. de 2019.

GUO, Zhenhua; ZHANG, Lei; ZHANG, David. Feature band selection for multispectral palmprint recognition. In: 2010 20TH INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 2010, Istanbul. **Anais...** Instanbul: IEEE, 2010. pp. 1136-1139.

HAYKIN, Simon. **Redes neurais: princípios e prática**. 2. ed. Porto Alegre: Bookman, 2001.

HORN, John. Forgot your password? Most people do. **Fiserv**, 2019. Disponível em: <https://www.fiserv.com/en/about-fiserv/the-point/forgot-your-password-most-people-do.html>. Acesso em: 4 de mai. de 2019.

HUFF, Steve. Watch Red Hot Chili Peppers drummer Chad Smith'S hilarious reaction after a heckler calls him 'Will Ferrell'. **Maxim**, 2017. Disponível em: <https://www.maxim.com/entertainment/chad-smith-response-will-ferrell-2017-11>. Acesso em: 4 de mai. de 2019.

IDENTYTECH SOLUTIONS. Palm vein recognition. **IDENTYTECH SOLUTIONS**, 2016. Disponível em: <https://identytech.com/palm-vein-recognition/>. Acesso em: 4 de mai. de 2019.

JAIN, Hardik. **Using morphable face model to improve stereo reconstruction and visualising the model on a smartphone**. 2016. Dissertação (Master of Technology in Electrical Engineering) - Department of Electrical Engineering, Indian Institute of Technology Roorkee, Roorke, 2016.

JIANG, H. et al. Vein pattern extraction based on the position-gray-profile curve. In: 2009 2ND INTERNATIONAL CONGRESS ON IMAGE AND SIGNAL PROCESSING, 2009, Tianjin. **Anais...** Tianjin: IEEE, 2009. pp. 1-4.

KAZEMI, Vahid; SULLIVAN, Josephine. One millisecond face alignment with an ensemble of regression trees. In: 2014 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2014, Columbus. **Anais...** Columbus: IEEE, 2014. pp. 1867-1874.

LIMA, Mariana. Brasil já tem mais de um smartphone ativo por habitante, diz estudo da FGV. **Link - Estadão**, 2018. Disponível em: <https://link.estadao.com.br/noticias/geral,brasil-ja-tem-mais-de-um-smartphone-ativo-por-habitante-diz-estudo-da-fgv,70002275238>. Acesso em: 4 de mai. de 2019.

LLORENS, Faraón et al. **Working with OpenCV and Intel image processing libraries. Processing image data tools**. 2002. Disponível em: <https://www.researchgate.net/publication/39436187_Working_with_OpenCV_and_Intel_Image_Proccessing_Libraries_Proccessing_image_data_tools>. Acesso em: 4 de mai. de 2019.

LOFT965. Justin timberlake readying new album for 2013. **Loft965**, 2013. Disponível em: <https://loft965.wordpress.com/2013/01/10/justin-timberlake-readying-new-album-for-2013/>. Acesso em: 4 de mai. de 2019.

LOURIDAS, Panos; EBERT, Christof. Machine learning. **IEEE Software**, vol. 33, pp. 110-115, 2016.

LOVEKIN, Stephen. Bruce Willis: I'm against new gun control laws that could infringe on Second Amendment rights. **Daily News**, 2013. Disponível em: <https://www.nydailynews.com/entertainment/gossip/bruce-willis-new-gun-control-laws-article-1.1257099>. Acesso em: 4 de mai. de 2019.

MARQUES FILHO, Paulo C.; LOPES, Hedibert F. Árvores de classificação e regressão. **Insper**, 2017. Disponível em: <http://hedibert.org/wp-content/uploads/2018/06/CART.pdf>. Acesso em: 4 de mai. de 2019.

MARVEL MOVIES. Ian McKellen. **Marvel Movies**, 2008. Disponível em: <https://marvel-movies.fandom.com/wiki/Ian_McKellen>. Acesso em: 4 de mai. de 2019.

MERCURY, Estevan. City adds to Selfie Spot inventory. **Estevan Mercury Publications**, 2018. Disponível em: <https://www.estevanmercury.ca/news/local-news/city-adds-to-selfie-spot-inventory-1.23476995>. Acesso em: 4 de mai. de 2019.

MIRANDA, James. Câmera de segurança detecta pessoas usando reconhecimento facial do Google. **Reconhecimento Facial**, 2017. Disponível em: <http://reconhecimentofacial.com.br/2017/06/01/camera-de-seguranca-detecta-pessoas-usando-reconhecimento-facial-do-google/>. Acesso em: 4 de mai. de 2019.

MOURA, José. What is signal processing?. **IEEE Signal Processing Magazine**, vol. 26, n. 6, p. 6, 2009. IEEE, 2009.

NG, Hong-Wei; WINKLER, Stefan. A data-driven approach to cleaning large face datasets. In: 2014 IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 2014, Paris. **Anais...** Paris: IEEE, 2014. pp. 27-30.

NNDB. Neve Campbell. **NNDB**, 2014. Disponível em: <https://www.nndb.com/people/749/000025674/>. Acesso em: 4 de mai. de 2019.

PANASONIC. Panasonic to launch face recognition server software using deep learning technology. **Panasonic**, 2018. Disponível em: <https://security.panasonic.com/news/archives/686>. Acesso em: 4 de mai. de 2019.

RAMOS, Maurício Peres. **Controle de acesso biométrico**. 2012. Trabalho de conclusão de curso (Especialização em Desenvolvimento de Produtos Eletrônicos) - Departamento Acadêmico de Eletrônica, Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Florianópolis, 2012.

RAUT, Shriram D.; HUMBE, V. T.; MANE, Arjun V. Development of biometric palm vein trait based person recognition system. In: 2017 1ST INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS AND INFORMATION MANAGEMENT, 2017, Aurangabad. **Anais...** Aurangabad: IEEE, 2017. pp. 18-21.

RIBEIRO, Igor. Twitter cresce receita e usuários “monetizáveis”. **Meio&mensagem**, 2019. Disponível em: <https://www.meioemensagem.com.br/home/midia/2019/02/07/twitter-cresce-receita-e-usuarios-monetizaveis.html>. Acesso em: 4 de mai. de 2019.

ROSEBROCK, Adrian. Histogram of oriented gradients and object detection. **Pyimagesearch**, 2014. Disponível em: <https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/>. Acesso em: 4 de mai. de 2019.

RUSSELL, B. C. et al. LabelMe: a database and web-based tool for image annotation. **International Journal of Computer Vision**, vol. 77, n. 1-3, pp. 157-173, 2008.

SAHA, Sumit. A comprehensive guide to convolutional neural networks — the ELI5 way. **Towards Data Science**, 2018. Disponível em: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Acesso em: 4 de mai. de 2019.

SARKAR, Ishani; ALISHEROV, Farkhod; BHATTACHARYYA, Debnath. Palm vein authentication system: a review. **International Journal of Control and Automation**, vol. 3, n. 1, pp. 27-33, 2010.

SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: 2015 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2015, Boston. **Anais...** Boston: IEEE, 2015. pp. 815-823.

SHARKAS, Maha. A neural network based approach for iris recognition based on both eyes. In: 2016 SAI COMPUTING CONFERENCE, 2016, London. **Anais...** London: IEEE, 2016. pp. 253-258. IEEE, 2016.

SHIN, Wu. Conversor digital-analógico. **Laboratório de Micro e Minicomputadores: Hardware**. **Universidade de Campinas**, 2012. Disponível em: <http://www.dca.fee.unicamp.br/courses/EA079/1s2012/complemento/conversaoDA.pdf>. Acesso em: 4 de mai. de 2019.

SHURE. Microphones: transducer types (dynamic, condenser, ribbon). **Shure**, 2016. Disponível em: <https://www.shure.eu/musicians/discover/educational/transducer-types>. Acesso em: 4 de mai. de 2019.

SILVA, Luiza Maria Oliveira da. **Uma aplicação de árvores de decisão, redes neurais e KNN para a identificação de modelos ARMA não-sazonais e sazonais**. 2005. Tese (Doutorado em Engenharia Elétrica) - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

SILVEIRA, Daniel. Brasil ganha 10 milhões de internautas em 1 ano, aponta IBGE. **G1**, 2018. Disponível em: <https://g1.globo.com/economia/tecnologia/noticia/2018/12/20/numero-de-internautas-cresce-em-cerca-de-10-milhoes-em-um-ano-no-brasil-aponta-ibge.ghtml>. Acesso em: 4 de mai. de 2019.

SMITH, Steven W. **The scientist and engineer's guide to Digital Signal Processing**. 2. ed. San Diego: California Technical Publishing, 1999.

SRIHARI, Sargur N.; SRINIVASAN, Harish; FANG, Gang. Discriminability of fingerprints of twins. **Journal of Forensic Identification**, vol. 58, n. 1, pp. 109-127, 2008.

THAKKAR, Danny. Top five biometrics: face, fingerprint, iris, palm and voice. **Bayometric**, 2017. Disponível em: <https://www.bayometric.com/biometrics-face-finger-iris-palm-voice/>. Acesso em: 4 de mai. de 2019.

THAKKAR, Danny. Fingerprint sensors face-off: capacitive sensor vs. optical sensor. **Bayometric**, 2018. Disponível em: <https://www.bayometric.com/capacitive-vs-optical/>. Acesso em: 4 de mai. de 2019.

THE PLACE. Zooey Deschanel. **The Place**, 2014. Disponível em: <https://www.theplace2.ru/photos/Zooey-Deschanel-md1406/pic-368276.html>. Acesso em: 4 de mai. de 2019.

THOMAS, Andy. Neural networks tutorial – a pathway to deep learning. **Adventures in machine learning**, 2017. Disponível em: <https://adventuresinmachinelearning.com/neural-networks-tutorial/>. Acesso em: 4 de mai. de 2019.

TRADER, John. Common misunderstandings between iris recognition and retina scanning. **M2SYS**, 2015. Disponível em: <http://www.m2sys.com/blog/scanning-and-efficiency/common-misunderstandings-between-iris-recognition-and-retina-scanning/>. Acesso em: 4 de mai. de 2019.

VAGALUME. Zooey Deschanel se sente lisonjeada ao ser confundida com Katy Perry. **Vagalume**, 2012. Disponível em: <https://www.vagalume.com.br/news/2012/11/20/zooey-deschanel-se-sente-lisonjeada-ao-ser-confundida-com-katy-perry.html>. Acesso em: 4 de mai. de 2019.

VINCENT, James. Chinese police are using facial recognition sunglasses to track citizens. **The Verge**, 2018. Disponível em: <https://www.theverge.com/2018/2/8/16990030/china-facial-recognition-sunglasses-surveillance>. Acesso em: 4 de mai. de 2019.

VIOLA, Paul; JONES, Michael. Rapid object detection using a boosted cascade of simple features. In: 2001 IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2001, Kauai. **Anais...** Kauai: IEEE, 2001 pp. 511-518.

VONDRICK, Carl et al. Visualizing object detection features. **International Journal of Computer Vision**, vol. 119, n. 2, pp. 145-158, 2016.

WAKKA, Wagner. Instagram bate marca de 1 bilhão de usuários ativos. **CanalTech**, 2018. Disponível em: <https://canaltech.com.br/redes-sociais/instagram-bate-marca-de-1-bilhao-de-usuarios-ativos-116344/>. Acesso em: 4 de mai. de 2019.

ZUBEN, Fernando J. Von; ATTUX, Romis R. F. Árvores de decisão. **Redes neurais II**. Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação, **Universidade de Campinas**, 2010. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia004_1s10/notas_de_aula/topico7_IA004_1s10.pdf>. Acesso em: 4 de mai. de 2019.

ZUIN, Lidia. Câmeras com reconhecimento facial e vigilância pública: da China para o Brasil. **Medium**, 2019. Disponível em: <https://medium.com/up-future-sight/c%C3%A2meras-de-reconhecimento-facial-e-vigil%C3%A2ncia-p%C3%BAblica-da-china-para-o-brasil-40c61a83a166>. Acesso em: 4 de mai. de 2019.

APÊNDICE 1 – PARES DE PESSOAS ESCOLHIDAS PARA AS DECISÕES DIFÍCEIS

America Ferrara e Jordin Sparks

Javier Bardem e Jeffrey Dean Morgan

Julia Stiles e Ansel Elgort

Isaac Mizrahi e JJ Abrams

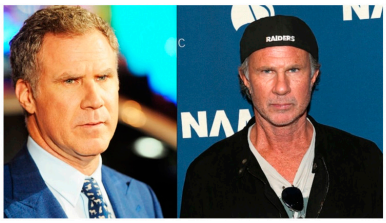
Brittany Murphy e Lili Reinhart

Natalie Portman e Keira Knightley

Sienna Miller e Mollie King

Katy Perry e Zooe Deschanel

Will Ferrell e Chad Smith



Josh Helman e Seann William Scott