INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS

CAMPUS MANAUS DISTRITO INDUSTRIAL

BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

MISAEL MARQUES RICARDO

Análise Comparativa de Algoritmos de Reconhecimento Facial

MANAUS - AM

2022

MISAEL MARQUES RICARDO

Análise Comparativa de Algoritmos de Reconhecimento Facial

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Controle e Automação, do Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Campus Manaus Distrito Industrial – IFAM/CMDI

Orientador: Prof. Dr. Alyson de Jesus dos Santos

MANAUS - AM

2022

Dados Internacionais de Catalogação na Publicação (CIP)

R488a

Ricardo, Misael Marques.
Análise comparativa de algoritmos de reconhecimento facial / Misael Marques Ricardo. — Manaus, 2023.
50f.: il. (color.).

Monografia (Graduação) — Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito Industrial, Curso de Engenharia de Controle e Automação, 2023.
Orientador: Prof.º Alyson de Jesus dos Santos, Dr.

1. Visão computacional. 2. Reconhecimento facial. 3. Inteligência artificial. I. Santos, Alyson de Jesus dos. II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 629.89

Elaborada por Oziane Romualdo de Souza (CRB11/ nº 734)

ANEXO 7

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 17 dias do mês de Fevereiro, de 2023, às 10 h, o(a) discente Misael Marques Ricardo apresentou o seu Trabalho de Conclusão de Curso para avaliação da Banca Examinadora constituída pelos seguintes integrantes: Prof(a). Dr. Alyson de Jesus dos Santos (docente-orientador), Prof(a). Msc. Jaidson Brandão da Costa (Membro 1) e Prof(a). Msc. Gabriel Pinheiro Compto (Membro 2). A sessão pública de defesa foi aberta pelo(a) presidente da banca, que apresentou a Banca Examinadora e deu continuidade aos trabalhos, fazendo uma breve referência ao TCC, que tem como título Análise Comparativa de Algoritmos de Reconhecimento Facial. Na sequência, o(a) discente teve até 30 minutos para a comunicação oral de seu trabalho. Cada integrante da banca examinadora fez suas arguições após a defesa do mesmo. Ouvidas as explicações do(a) discente, a banca examinadora, reunida em caráter sigiloso, para proceder à avaliação final, deliberou e decidiu pela APROVAÇÃO com média final 9,5 (Nove, Cinco)

do referido trabalho.

Foi dada ciência ao(à) discente que a versão final do trabalho deverá ser entregue até o dia 16/03/2023, com as devidas alterações sugeridas pela banca. Nada mais havendo a tratar, a sessão foi encerrada às 11h 00 min, sendo lavrada a presente ata, que, uma vez aprovada, foi assinada por todos os membros da Banca Examinadora e pelo(a) discente.

Prof.(a) Orientador(a)/Presidente: Alyson de Jesus dos Santos

Prof.(a) Avaliador 1: Jaidson Brandão da Costa

Prof.(a) Avaliador 2: Gabriel Pinheiro Compto

Discente: Misael Marques Ricardo

AGRADECIMENTOS

Palavras não podem descrever o sentimento de gratidão perante a conclusão de mais uma jornada.

Agradeço a minha mãe Rebeca, heroína que me deu apoio, incentivo nas horas difíceis, de desânimo e cansaço e ao meu pai que apesar de todas as dificuldades me fortaleceu e que para mim foi muito importante.

Aos meus queridos amigos, quero agradecer o apoio, força, amor e assistência inabalável. A Secretaria do Curso, pela cooperação.

Agradeço ao meu orientador, Prof. Dr. Alyson de Jesus dos Santos, pela sabedoria e paciência com que me orientou, sendo bastante acessível para finalizar a jornada, e por sempre estar disposto a me ajudar.

Enfim, a todos os que de alguma forma contribuíram para a realização desta pesquisa e fim de mais esse ciclo.

‘‘O Naruto pode ser um pouco duro às vezes”.

Kakashi Hatake

RESUMO

O PRESENTE TRABALHO TEM POR OBJETIVO REALIZAR UM ESTUDO SOBRE O FUNCIONAMENTO DE ALGORITMOS DE RECONHECIMENTO FACIAL DESCRITOS NA LITERATURA. FORAM UTILIZADOS TRÊS DOS ALGORITMOS DE RECONHECIMENTO FACIAL DESCRITOS NA LITERATURA E PROPOSTO UM CONJUNTO DE TESTE DE MODO A AVALIAR O RESULTADO E VERIFICAR A POSSIBILIDADE DE USAR O ALGORITMO EM UM SISTEMA DE CONTROLE DE ACESSO A ESPAÇOS FÍSICOS. FORAM UTILIZADOS OS ALGORITMOS EIGENFACES, FISHERFACES E LBPH. PARA OS TESTES FORAM UTILIZADOS TRÊS BANCOS DE DADOS: O PRIMEIRO FOI CONSTITUÍDO COM FOTOS DO AUTOR E SEU FAMILIAR, O SEGUNDO FOI O BANCO DE DADOS DE PESSOAS COM E SEM MÁSCARAS E POR ÚLTIMO O BANCO DE DADOS CONSTITUÍDO DE PESSOAS COM E SEM ÓCULOS E OS RESULTADOS DA PRECISÃO DO RECONHECIMENTO FORAM RELATADOS E APRESENTADOS NESSE TRABALHO.

Palavras-chave: Visão Computacional. Reconhecimento Facial. Inteligência Artificial.

ABSTRACT

THE PURPOSE OF THIS PAPER IS TO CARRY OUT A STUDY ON THE OPERATION OF FACE RECOGNITION ALGORITHMS DESCRIBED IN THE LITERATURE. THREE OF THE FACIAL RECOGNITION ALGORITHMS DESCRIBED IN THE LITERATURE WERE USED AND A TEST SET IS PROPOSED IN ORDER TO EVALUATE THE RESULT AND VERIFY THE POSSIBILITY OF USING THE ALGORITHM IN AN ACCESS CONTROL SYSTEM TO PHYSICAL SPACES. THE EIGENFACES, FISHERFACES AND LBPH ALGORITHMS WERE USED. TO THE TESTS THREE DATABASES WERE USED: THE FIRST WAS CONSTITUTED WITH PHOTOS OF THE AUTHOR AND HIS FAMILY, THE SECOND WAS THE DATABASE OF PEOPLE WITH AND WITHOUT MASKS AND LASTLY THE DATABASE CONSISTED OF PEOPLE WITH AND WITHOUT GLASSES AND THE RESULTS THE ACCURACY OF RECOGNITION WERE REPORTED AND PRESENTED IN THIS PAPER.

Keywords: Computer Vision. Facial Recognition. Artificial Inteligence

LISTA DE ILUSTRAÇÕES

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A biometria é uma tecnologia cada vez mais presente na nossa vida diária, com o objetivo de aumentar a segurança de acesso a diferentes sistemas e espaços. Esta tecnologia consiste na utilização de recursos físicos individuais, como impressões digitais, reconhecimento facial e reconhecimento de áudio, de modo a garantir que somente indivíduos autorizados possam ter acesso às informações que desejam. A utilização desta tecnologia possibilita a descoberta de novas formas de identificação segura, diretamente a experiência do usuário e aumentando a segurança da informação. Além disso, a biometria permite a eliminação da necessidade de alertar senhas, aumentando ainda mais a segurança do sistema. Com a utilização de sensores de alta tecnologia, a biometria pode ser uma ferramenta precisa e eficiente na autenticação de usuários.

O reconhecimento facial é uma tecnologia que permite que computadores determinem de forma precisa a identidade de uma pessoa a partir de uma imagem. O reconhecimento facial se baseia em técnicas biométricas que consistem identificar padrões faciais tais como, rosto, formato da boca, distância entre olhos, entre outros (SILVA; CINTRA, 2015). Esta tecnologia pode usar algoritmos avançados para detectar e marcar características como padrões de luz e sombra, que então são usados para determinar a identidade de um indivíduo. O reconhecimento facial é um dos métodos mais eficazes de identificação.

As vantagens do reconhecimento facial incluem:

Facilidade: O reconhecimento facial é simples e rápido, permitindo que um computador identifique uma pessoa por meio de imagens de vídeo ou fotos.

Segurança: O reconhecimento facial fornece um nível maior de segurança para as aplicações. Pode ser usado para verificar a identidade de alguém em locais sensíveis como aeroportos, bancos etc.

Conveniência: O reconhecimento facial permite que os usuários acessem computadores, aplicativos, sites e serviços com facilidade, usando seu próprio face como uma chave de acesso. Isso permite que os usuários acessem seus serviços com facilidade e rapidez, sem a necessidade de lembrar senhas e outros detalhes de login.

Monitoramento: O reconhecimento facial pode ser usado para monitorar um ambiente, facilitando a detecção de intrusos ou pessoas desconhecidas.

Engajamento do usuário: O reconhecimento facial pode ser usado para personalizar e melhorar a experiência do usuário, criando sombras mais naturais e eficazes.

Reconhecimento de emoções: O reconhecimento facial também pode ser usado para analisar e detectar as expressões emocionais de alguém, permitindo a identificação de emoções e ações. Isso fornece insights valiosos para os usuários.

A pretensão deste trabalho é estudar e avaliar o comportamento de algoritmos de reconhecimento facial. Para isso serão estudados algoritmos que utilizam duas técnicas: PCA (*Principal Component Analysis*) e o LDA (*Linear Discriminant Analysis*). Ambas as técnicas consistem em reduzir a redundância ou mesmo para detectar padrões como faces e objetos em um conjunto de dados utilizando técnicas de álgebra linear e estatística aplicada.

Com base na técnica do PCA foi estudado um dos primeiros algoritmos criados quando o assunto é reconhecimento facial, o eigenfaces, que são os componentes principais extraídos de um conjunto de dados de imagens faciais usando PCA. Eles são usados como base para sistemas de reconhecimento facial e são compostos por componentes principais que representam as características mais importantes de uma face.

A ideia principal por trás dos eigenfaces é que eles capturam as características essenciais de um rosto, ignorando pequenos detalhes. Isso é feito criando uma decomposição de autovalor do conjunto de dados de face original, que identifica os recursos mais significativos do conjunto de dados. As autofaces resultantes podem então ser usadas para identificar com rapidez e precisão o rosto de um indivíduo.

O mesmo foi feito para o algoritmo Fisherfaces porém com uso da técnica LDA. Esta técnica reduz a dimensionalidade da imagem de entrada, mantendo assim as características mais importantes da imagem, enquanto elimina os ruídos irrelevantes. Depois de reduzir a dimensionalidade da imagem, a técnica Fisherfaces usa um algoritmo de classificação supervisionado para classificar e identificar as características da imagem.

Os Fisherfaces são usados em diversas aplicações, como identificação de crianças, detecção de rostos, sistemas de reconhecimento de voz e até mesmo detecção de crimes de identidade.

Também foi utilizado os Algoritmos LBPH *(Local Binary Pattern Histogram)* são usados para aplicações de reconhecimento facial. Ele usa um padrão de padrões binários locais para criar características independentes de imagens e determinar a similaridade entre caras. O algoritmo é significativamente menos complexo e rápido para implementar do que outros algoritmos de reconhecimento facial, como o Eigenface, e é bastante eficaz para trabalhar com fotos sujas, desfocadas e de baixa resolução. O algoritmo também é conhecido por ser robusto a variações na iluminação e na posição da face

Para validação dos algoritmos implementados neste trabalho foram utilizados criado um banco de dados com a imagem de dois usuários para reconhecimento facial, além da utilização de dois bancos de dados de imagens, o primeiro banco é formando por um conjunto de dados de vinte mil imagens de pessoas com e sem máscara, e no segundo banco é constituído por cinco mil imagens de faces de pessoas geradas por uma inteligência artificial com e sem óculos.

1.1 OBJETIVO GERAL

O objetivo do trabalho é usar ferramentas otimizadas de visão computacional e aprendizado de máquina supervisionado a afim de verificar o desempenho de algoritmos computacionais de reconhecimento facial com o intuito de indicar sua viabilidade no desenvolvimento de ferramentas de controle de acesso de indivíduos. Mais especificamente serão investigados três dos principais algoritmos de reconhecimento facial descritos pela literatura, a saber: o Eigenfaces, Fisherfaces e LBPH visando avaliar a robustez de cada algoritmo usando um conjunto de 3 bancos de dados.

1.2 OBJETIVOS ESPECÍFICOS

Para obter um resultado satisfatório listamos os seguintes objetivos específicos:

a) Capturar de diversas fotos dos utilizadores do sistema.
b) Construir do banco de dados com as fotos.
c) Dividir o banco em treinamento e teste.
d) Treinar as inteligências artificiais com as imagens de treino dos três bancos de dados
e) Utilizar modelo criado para teste.
f) Demostrar os resultados
g) Realizar comparações entre as técnicas utilizadas.

Nesse contexto, o trabalho mostrará como o estudo pode aplicado na área da tecnologia a fim de que se crie uma solução de segurança com mais

1.3 JUSTIFICATIVA

A capacidade de realizar análise automática e confiável de imagens possibilita um número muito grande de aplicações. Existem utilidades na área de telecomunicações, como a codificação e a melhoria da qualidade das imagens. A área de segurança e monitoramento poderia se beneficiar de uma interpretação mais inteligente das cenas, assim como as características faciais podem ser extraídas e suas posições localizadas (LINGGARD; MYERS; NIGHTINGALE).

O reconhecimento facial é um método não intrusivo, e a imagem do rosto é uma característica biométrica muito utilizada para reconhecimento de identidade. Neste sentido, o projeto abordado neste trabalho buscou apresentar uma solução de reconhecimento atuando sobre a face a ser reconhecida, com base na análise computacional da imagem.

Tal projeto exige conhecimentos de lógica de programação, linguagem Python, inteligência artificial, visão computacional, e processamento digital de imagens, reunindo dessa forma diversas disciplinas cursadas ao longo da formação

de Engenharia de Controle e Automação. É importante lembrar também que a integração de diferentes sistemas é função primordial do Engenheiro de Automação.

1.4 ESTRUTURA DO TCC

O restante deste TCC está estruturado da seguinte maneira. No Capítulo 1 é apresentado a introdução do TCC juntamente com a apresentação dos objetivos e justificativa, No Capítulo 2, é apresentada a fundamentação teórica com o propósito de mostrar uma explicação sobre a visão computacional e as técnicas utilizadas para captura das informações juntamente com a linguagem utilizada e o material. O Capítulo 3, apresenta os métodos e materiais utilizados a fim de mostrar as ferramentas computacionais utilizadas, descrição e implementação do trabalho proposto. No Capítulo 4 é apresentado os resultados e discussões. Por fim, no Capítulo 5 é apresentada a conclusão do trabalho e anexos.

2 REFERENCIAL TEÓRICO

Este capítulo tem por objetivo apresentar conceitos teóricos necessários para as discussões práticas deste TCC.

2.1 PYTHON

Segundo Borges, Python (Figura 1) pode ser definida como “uma linguagem de altíssimo nível (em inglês, *Very High Level Language*) orientada a objetos, de tipagem dinâmica e forte, interpretada e interativa” e Corrêa (2020, p.12) completa o pensamento afirmando:

> Python é uma linguagem de programação de propósito geral, o que significa que ela pode ser empregada nos mais diferentes tipos de projetos, variando desde aplicações Web até sistemas de inteligência artificial.

Figura 1 - Python Logo

Fonte: PYTHON.ORG (2022)

O Python foi criado por Guido Van Rossum, em 1990. É uma linguagem interpretada que possui uma sintaxe mais clara e dinâmica comparada com as outras linguagens, tornando uma das linguagens mais produtivas e que abrange um público maior entre os mais experientes e iniciantes no mundo da programação. Sendo construído como um código aberto que pode ser modificado por qualquer usuário, é possível baixar e instalar o interpretador Python gratuitamente e sua licença compatível com a *General Public License* (Licença Pública Geral), o que a torna bastante difundida entre grandes empresas como: Google, Microsoft, Yahoo e Disney. (CORREA, 2020)

A sua gratuidade e código aberto possibilita a criação de diversas bibliotecas por usuários em qualquer lugar do mundo, podendo ser facilmente expandida com a importação de diversos pacotes, atualmente existem milhares de pacotes disponíveis

no repositório central do Python, Python Package Index (PyPI). Muitos deles voltados para a ciência de dados, tais como os famosos, ‘NumPy’ (manipulação de vetores e matrizes), ‘SciPy’ (rotinas numéricas para resolver problemas de integração, equações algébricas e cálculo numérico, entre outras coisas), ‘pandas’ (importação e transformação de bases de dados), ‘Matplotlib’ (geração de gráficos) e ‘*scikit-learn’* (algoritmos de mineração de dados e aprendizado de máquina). No projeto será utilizado diversos pacotes que serão explicados mais à frente. (CORREA, 2020)

Outro ponto importante que tornam o Python produtivo e popular, se deve a sua tipagem da linguagem, definida como uma tipagem dinâmica, dispensando a obrigatoriedade do usuário definir o tipo de variáveis, apenas atribuir valor a ela que o interpretador se encarrega de definir o seu tipo a partir do que for atribuído a essa variável. Além disso o interpretador Python pode ser usado de forma interativa, onde cada linha de código é imediatamente traduzida e executada, oferecendo aos programadores uma forma simples e eficiente para examinar no instante em que escrevem os resultados intermediários obtido em qualquer passo de um processo de análise de dados. (CORREA, 2020)

2.2 VISUAL STUDIO CODE

Os codificadores não podem usar o Python sem um editor de código apropriado, também conhecido como ambiente de desenvolvimento integrado (do inglês, *Integrated Development Environment* (IDE), duas das IDE’s mais relevante hoje são o *Pycharm* da *JetBrains* e o *Visual Studio Code* (Figura 2) da Microsoft. (MILLER, 2021)

Figura 2 - *Visual Studio Code*

Fonte: VSCODE, 2022

Dentre as vantagem e desvantagens de ambos as Edes escolhemos o *Visual Studio Code* pois é mais um editor de código que emula a funcionalidade e a experiência de um IDE usando um conjunto de extensões. O VsCode é um produto de código aberto, gratuito para baixar e usar para iniciantes profissionais e experientes. (MILLER, 2021)

O Vscode é um programa mais enxuto que o *PyCharm* requer um download e instalação de 76,2 MB. O IDE consome apenas 40 MB de RAM durante a execução, ou um décimo do que o *PyCharm* precisa para um desempenho estável. (MILLER, 2021)

O gerenciamento de código é intuitivo com o VSCode. Inclui um recurso de destaque automático que marca possíveis erros diretamente no ambiente de trabalho. Ele também cria um resumo de erros e conflitos de código em uma guia separada para torná-los mais fáceis de solucionar. (MILLER, 2021)

O VSCode, como o PyCharm, possui um recurso robusto de preenchimento automático para ajudar os programadores a economizar tempo durante a codificação, também oferece suporte à refatoração após uma breve configuração. A lista de recursos de refatoração não é tão robusta quanto a do PyCharm. (MILLER, 2021)

2.3 INTELIGÊNCIA ARTIFICIAL

> A Inteligência Artificial é um área de estudo da ciência da computação que objetiva possibilitar que computadores realizem processos complexos, abstratos e lógicos tipica- mente humanos. Geralmente, os sistemas que são descritos como artificialmente inteligentes s˜ao sistemas que pensam como humanos, sistemas que agem como humanos, sistemas que pensam racionalmente e sistemas que agem racionalmente. (KOK, 2009)

Herbert Simon (Figura 3), cientista que contribuiu enormemente nos campos de economia, psicologia, ciência cognitiva, Inteligência Artificial e outros, afirma que assim como o pensamento humano, a Inteligência Artificial se baseia na procura de padrões em informaçoes, na aplicação desses padrões e na realização de inferências construídas a partir deles. (FRANTZ, 2003)

Figura 3 - Hebert Simon

Fonte: CARNEGIE MELLON UNIVERSITY, 2022

Alan Turing (Figura 4) foi um cientista que contribuiu nas áreas de lógica, matemática, biologia, filosofia e criptoanálise e foi responsável por desenvolver conhecimentos que se tornaram base do que hoje são a ciência da computação, ciência cognitiva, vida artificial e Inteligência Artificial. Em 1936, ele publicou um artigo que descrevia uma máquina algorítmica abstrata, princípio essencial para a futura concepção dos computadores modernos. Além disso, ele também foi responsável por quebrar a estratégia de criptografia baseada na máquina *Enigma* utilizada pela Alemanha nazista durante a Segunda Guerra Mundial. (COPELAND, 2004).

Figura 4 - Alan Turing

Fonte: COPELAND, 2004.

Em fevereiro de 1947, foi a primeira vez que Turing levou a público o seu conceito recentemente criado de Inteligência Artificial em uma palestra cujo nome era *"Lecture on the Automatic Computing Engine"*, em tradução livre "Palestra sobre o Motor Computacional Automático". Em 1950, ele publicou o primeiro texto que fazia referência a um intelecto de máquina, um artigo chamado *"Computing Machinery and Intelligence"*, em tradução livre "Máquina Computadora e Inteligência". Posteriormente, Turing desenvol- veu um teste que determinaria se uma máquina poderia ser considerada pensante ou não. Esse teste hoje é conhecido como "Teste de Turing" (COPELAND, 2004).

2.4 APRENDIZADO DE MÁQUINA

O aprendizado de máquina, do inglês *machine learning* (ML), consiste na criação de algoritmos com o fim de permitir a um computador reconhecer padrões e realizar inferências, sem que ele seja diretamente programado para isso. Um sistema baseado em algoritmos de aprendizagem é capaz de, pouco a pouco, melhorar os seus resultados à medida em que vai sendo exposto a dados e consegue acumular experiência a partir deles, algo similar ao que acontece no

aprendizado humano. (ALPAYDIN, 2020)

Em aprendizado de máquina, estatística e ciência de dados são fortemente representadas em processos de inferência edificados em amostras de dados. Essas são ferramentas importantes para otimizar algoritmos, já que os dados são a fonte que levará o código a desenvolver sua inteligência artificial. Assim como a qualidade dos livros didáticos influi o aprendizado humano, a qualidade dos dados também é importante para o aprendizado de máquina. (PIRES, LIMA, SILVA, 2020)

Os conhecimentos de ciência e engenharia de computação são utilizados para desenvolver códigos, sensores, máquinas e arquiteturas computacionais através do processo de ETL (*extract, transform, load*) aplicado aos dados, a fim de criar e alimentar o modelo de aprendizado. Além disso são usados na solução de problemas de otimização, junto a matemática, e para avaliar a inferência dos modelos de ML. Os algoritmos de aprendizado de máquina processam dados para aprender a computar uma função desejada, então, dependem diretamente dos dados utilizados para que a resposta obtida seja de boa qualidade. Dentro desse contexto, existem vários tipos de algoritmos diferentes e estratégias diferentes de aprendizado. Os principais tipos de aprendizado serão abordados a seguir. (PIRES, LIMA, SILVA, 2020)

2.5 TIPOS DE APRENDIZADO

Com a tecnologia se fazendo cada vez mais presente nos negócios, a utilização de ciência de dados e de inteligência artificial tem se tornado cada vez mais frequente. Conhecer os tipos de aprendizado de máquina pode ajudar a encontrar inovações que de fato atendam os requisitos do projeto.

Compreender quais os tipos de aprendizado de máquina é uma forma de ter certeza que o *software* atenderá a demanda. Caso seja necessário automatizar determinado processo e diminuir a atividade humana em cima dele, algumas opções devem ser evitadas.

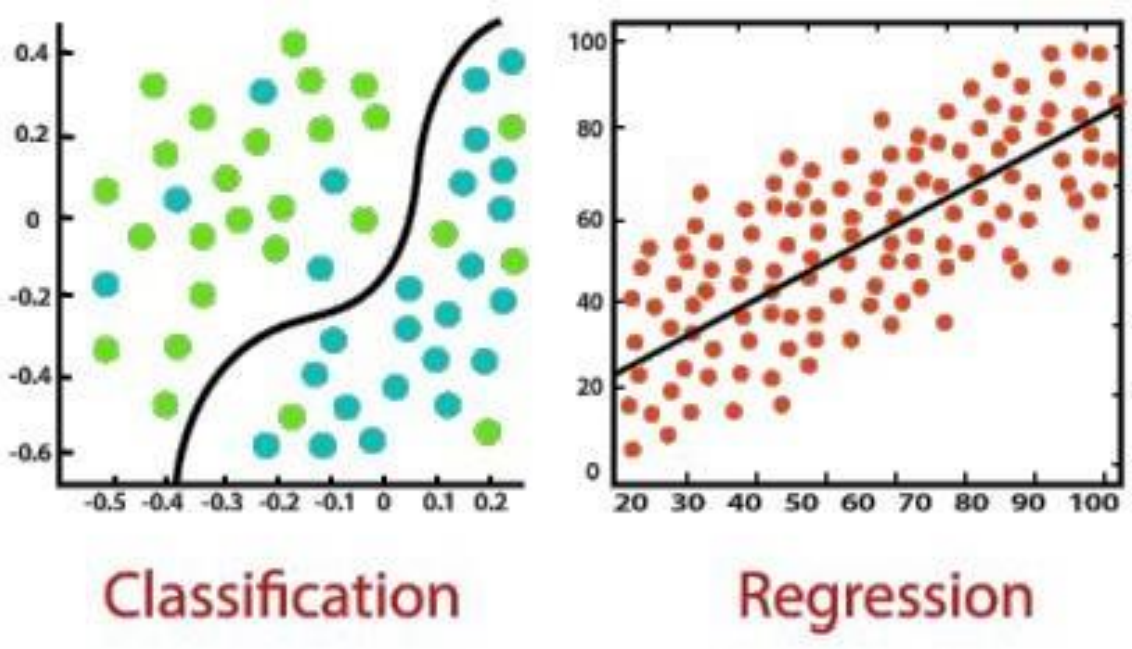
2.5.1 APRENDIZADO SUPERVISIONADO

No aprendizado supervisionado, o algoritmo aprende a partir da definição de um conjunto de variáveis, e de seu respectivo conjunto imagem com respeito a função que se deseja assimilar. Basicamente, o programa aprende observando casos em que a resposta é rotulada e fornecida a ele, sendo esses os dados de treino. Assim, detecta-se um padrão para a resposta de acordo com os valores das variáveis de entrada em cada caso e o objetivo do algoritmo é minimizar o erro de saída para casos não conhecidos, ou seja, inputs que não fizeram parte do treino do modelo. (PIRES, LIMA, SILVA, 2020)

As inferências de um modelo de Aprendizado Supervisionado são de grande importância para classificações e regressões (Figura 5). Elas tornam possível a previsão de casos futuros não presentes durante o treinamento e a detecção de fraudes a partir da percepção de *outliers*. (ALPAYDIN, 2020).

São exemplos de classificação: o reconhecimento de letras ou estilos de escrita, o reconhecimento de fala, diversos casos de diagnósticos médicos e a biometria. Já exemplos de regressões são a navegação de um carro e cinemática de robôs. Para este projeto, os modelos utilizados se encaixam no caso de classificação, em biometria.

Figura 5 - Regressão versus Classificação

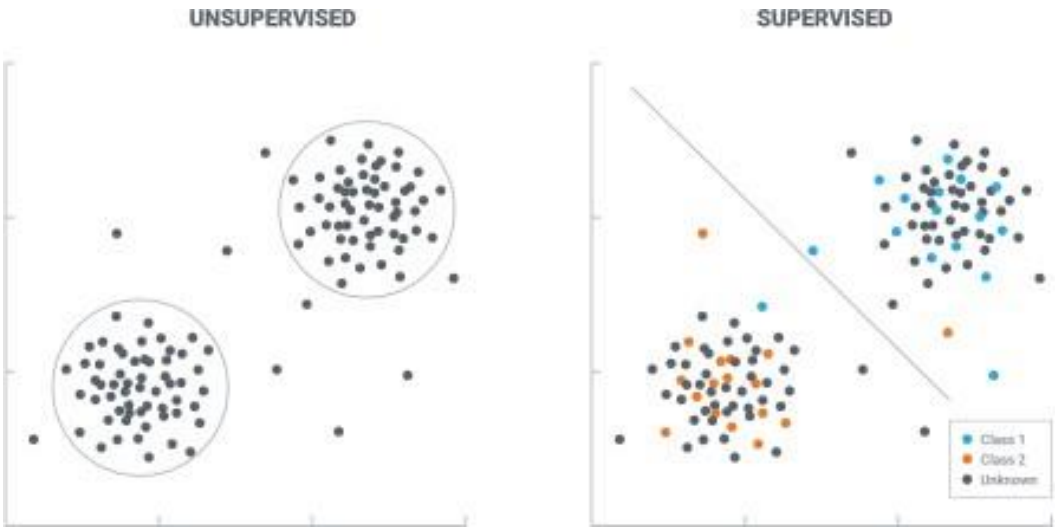


Fonte: JAVATPOINT, 2018

2.5.2 APRENDIZADO NÃO-SUPERVISIONADO

No aprendizado não supervisionado, nenhuma resposta ou rotulo são fornecidos ao algoritmo, o que limita o escopo de utilização da abordagem. Normalmente, esse método de aprendizado é adequado para detectar desvios em uma distribuição ou grupos de distribuições. Esse aprendizado tem como objetivo encontrar um padrão não reconhecido ou oculto em dados não rotulados. E geralmente aplicado em problemas de agrupamento (*clustering*). (PIRES, LIMA, SILVA, 2020)

Figura 6 - Não-Supervisionado versus supervisionado



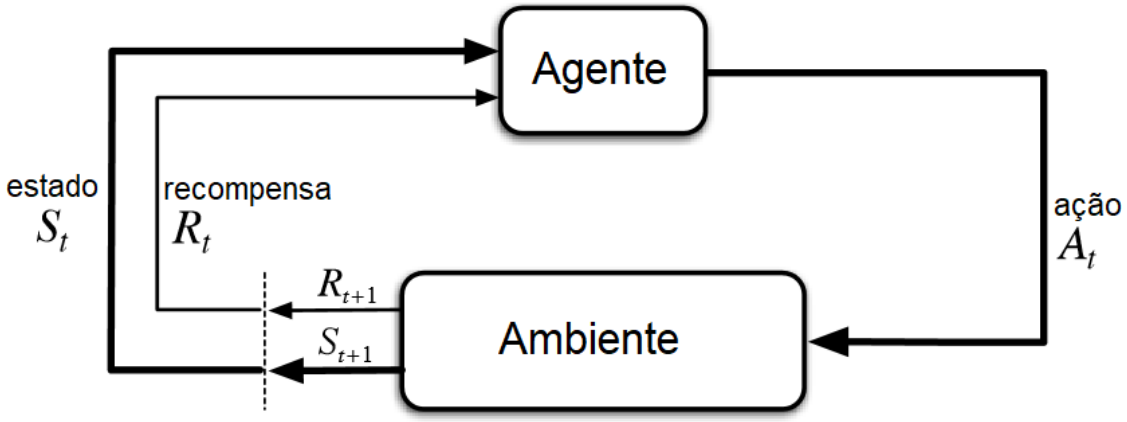
Fonte: PIRES, LIMA, SILVA, 2020

2.5.3 APRENDIZADO POR REFORÇO

A Aprendizagem por Reforço (AR) é um formalismo da Inteligência Artificial que permite que a um agente, com as interações com o ambiente onde ele está inserido possa realizar os seus aprendizados. (NERI, 2011).

O conceito central do aprendizado por reforço, é que as percepções são utilizadas não apenas para agir, mas paralelamente, melhorar a habilidade do agente para agir no futuro. A aprendizagem ocorre à medida que o agente observa suas interações com o ambiente e com os seus próprios processos de tomada de decisão, como mostra a Figura 7. Normalmente, o tipo de realimentação disponível para o aprendizado é um fator importante na determinação da natureza do problema (RUSSELL; NORVIG, 2004).

Figura 7 - Processo de tomada de decisão



Fonte: NEVES, 2020

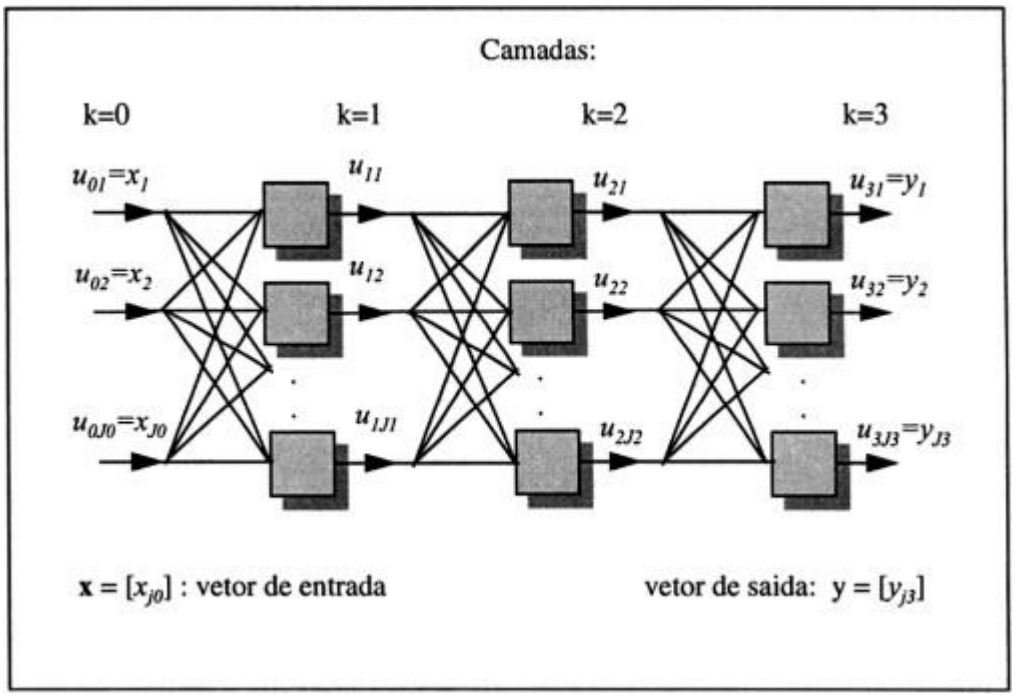
As recompensas servem para definir políticas ótimas em processos de decisão de Markov (PDMs). Uma política ótima é uma política que aumenta ao máximo a recompensa total esperada. A tarefa da aprendizagem por reforço consiste em usar recompensas para aprender uma política ótima. O aprendizado por reforço pode ser entendido com uma forma dos agentes aprenderem o que fazer, particularmente quando não existe nenhum professor dizendo ao agente que ação ele deve executar em cada circunstância. Para isso, a realimentação, chamada de recompensa ou reforço, informa ao agente recebendo algo bom quando ganha e recebendo algo ruim quando perde. (RUSSELL; NORVING, 2004)

.

2.5.4 REDES NEURAIS

No final da década de 1950, Rosenblatt na Universidade de Cornell criou uma genuína rede múltiplos neurônios do tipo descriminadores lineares e chamou esta rede de perceptron, sendo essa uma rede com uma topologia representada na figura 8, com os neurônios arranjados em diversas camadas. (KOVÁCS, 2006)

Figura 8 - Topologia de Rede Neural



Fonte: KOVÁCS, 2006

A camada de entrada é composta por neurônios que recebem diretamente as entradas da rede. Os neurônios que recebem que recebem como entrada as saídas dos neurônios da camada de entrada constituem a segunda camada e assim são construídas as camadas por diante até a camada de saída. As camadas que não internas da rede neural, que não são nem entrada nem saída, são chamadas de camadas ocultas. (KOVÁCS, 2006)

2.6 VISÃO COMPUTACIONAL

A Inteligência Artificial aplicada à visão computacional é um forte exemplo de avanço tecnológico. Ainda é fascinante perceber que a repetição de ações humanas bem como a tomada de decisões se tornou possível com o uso de softwares e hardwares. Algo que antes era pura ficção. (GRYFO, 2021)

A visão computacional é uma área da ciência da Inteligência Artificial (IA). Ela é responsável pelo desenvolvimento de recursos e conceitos que tem como objetivo

a obtenção de dados valiosos extraídos das imagens. Esse é um tema novo, mas seus resultados e melhorias já são sentidos em larga escala. (GRYFO, 2021)

Os sistemas de visão são extremamente importantes para se garantir a qualidade de um produto e a credibilidade da marca como mostra a Figura 9, onde é utilizada a visão computacional para assegurar a qualidade de línguas de gato de chocolate. A confiabilidade, competitividade e eficiência de toda a cadeia produtiva, não somente na inspeção acabam por crescerem com a inserção desses sistemas no ambiente fabril. (BAUMGARTEN, 2018)

Figura 9 - Sistema de inspeção de qualidade com visão computacional

Fonte: SNEF BRASIL, 2020

A explicação de como a Inteligência Artificial está aplicada à visão computacional é o próprio funcionamento da visão. As máquinas que operam com a

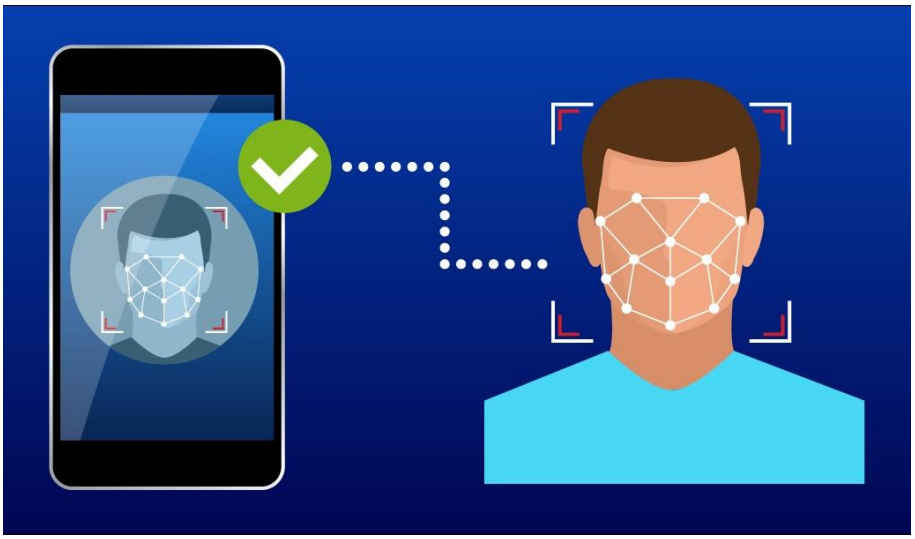
visão computacional só funcionam por conta da aplicação da IA, mais especificamente a *deep learning*, que nada mais é do que uma aprendizagem com milhões e milhões de dados. E, semelhantemente, o uso está ligado ao aprendizado da máquina (*machine learning*). É o treinamento e aprendizado de máquinas que são agentes da visão computacional, fazendo com que o computador busque a compreensão e interpretar dados do mundo visual. Como consequência, essas mesmas máquinas são capazes de reagir e agir de acordo com o que “viram” através do *deep learning.* (GRYFO, 2021)

A Inteligência Artificial treina as máquinas para atuar como humanos, mas não de maneira básica. O comportamento abrange reconhecimento de fala, mas também de imagem, classificação dessas imagens, previsões e ações. Para isso é preciso uma evolução na forma como os dados são organizados e utilizados. (GRYFO, 2021)

2.7 BIOMETRIA FACIAL

Uma das ramificações de aplicação da inteligência artificial juntamente com visão computacional é a biometria facial (Figura 10), de acordo com o livro “Our Biometric Future: Facial Recognition Technology and the Culture of Surveillance” (GATES, 2011), é uma tecnolofgia complexa que abrange, de forma automática, a identificação individual de acordo com as características faciais. Nem a detecção de expressões nem o reconhecimento facial precisam ser perfeitos para serem efetivos. Elas são tecnologias básicas mas difíceis de serem projetadas, para alcançar-se um nível superior de segurança social como um todo. O autor dedica parte do livro para explicar como o reconhecimento facial pode ser útil na prevenção de ataques terroristas e acesso a sistemas cr´ıticos, por exemplo. (COSTA, 2019)

Figura 10 - Biometria Facial



Fonte: BRAGA, 2020

2.8 TAREFAS DO RECONHECIMENTO FACIAL

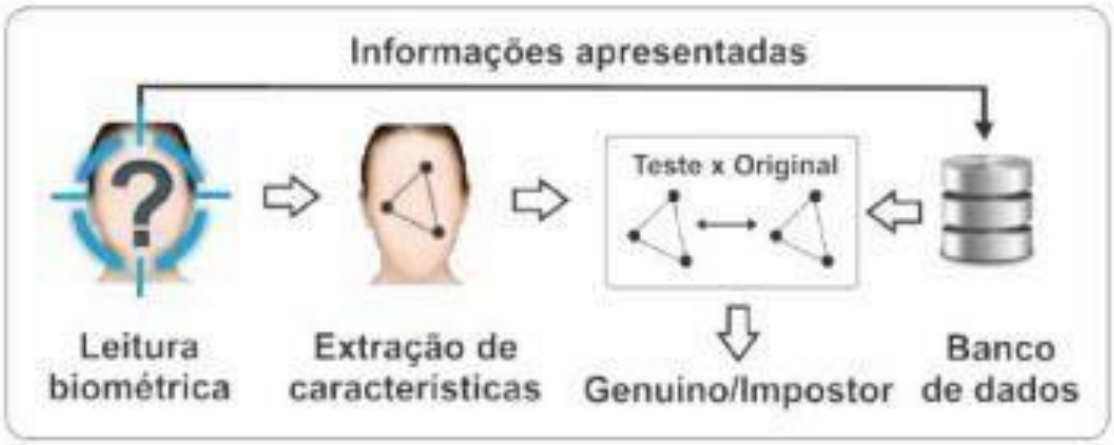
A verificação de características de uma face captura e outras face que possam estar presentes no banco executa-se o processo de reconhecimento facial, podendo ser executado de formas distintas para que requisitos específicos de um ambiente que necessite de avaliação biomédica ou um outro problema possa ser atendido. Assim, o procedimento de reconhecer faces pode ser divido em três tarefas específicas, que alteram a forma como o sistema reconhecedor vai atuar e processar as faces captadas (FAGERTUN, 2005). Essas tarefas específicas são: verificação (autenticação), identificação (reconhecimento) e observação (reconhecimento geral).

A tarefa de verificação (autenticação) trata o processo de reconhecimento facial como uma tarefa de identificação para acesso a um determinado local, onde o indivíduo é testado para comprovar se ele é quem realmente diz ser. O teste de verificação é feito separando os indivíduos em dois grupos distintos, os clientes, que são pessoas tentando obter acesso com sua própria identidade, e os impostores, que são as pessoas tentando obter acesso ao sistema através de uma identidade falsa. Na verificação existe o confronto de um para um entre as imagens para a identificação do usuário, ou seja, sua imagem facial é comparada com o objetivo de encontrar uma única imagem correspondente dentre as armazenadas no sistema. A Figura 11 mostra o processo de verificação facial, bem como o propósito da tarefa de verificar a

autenticidade de um indivíduo específico ou a presença de um impostor (FAGERTUN, 2005).

Duas métricas são utilizadas para avaliar a qualidade do sistema na verificação de indivíduos, a Taxa de Falsos Aceitos (TFA) e a Taxa de Falsos Rejeitos (TFR) (YANG, 2002). A TFA mede a porcentagem de impostores que ganham acesso e mostra quando o sistema possui falhas na montagem do banco de dados de faces, que permite que características semelhantes de um cliente e um impostor sejam identificadas como iguais. Já a TRF mede a porcentagem 30 de clientes que têm o acesso negado, que diz respeito a falhas no modo como o sistema capta os dados no momento do acesso, não permitindo que as características obtidas possam ser identificadas no banco de dados. Na tarefa de verificação, faz-se necessária a iteração do usuário com o sistema fornecendo a sua face para captação e comparação imediata em relação a sua identidade, como é feito geralmente em áreas com acesso restrito tais como em prédios e departamentos (YANG, 2002)

Figura 11 - Processo de Verificação Visual

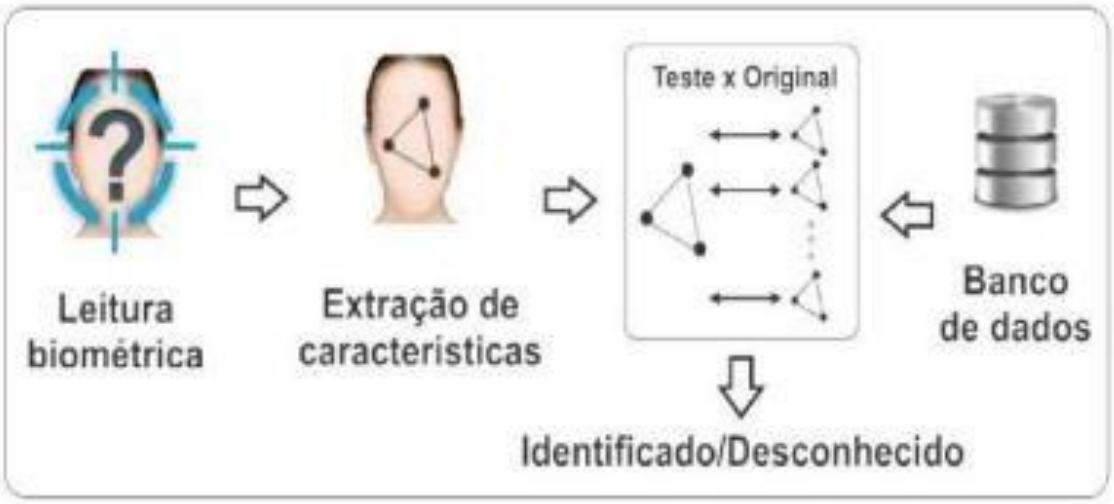


Fonte: SILVA, 2015

Entre os requisitos mais complexos em sistemas de reconhecimento facial está a atividade de vigilância, que consiste na verificação contínua de vários indivíduos e sua correspondência com relação as faces já conhecidas e armazenadas (PRADO, 2018). Essa atividade é feita pela tarefa de identificação (reconhecimento), que diferente da tarefa de verificação não exige iteração direta dos usuários, capturando de forma automática a face de todos os indivíduos que entram em contato

com uma câmera que monitora o ambiente. Aqui temos um confronto de um para muitos, onde a imagem de uma face capturada sem o conhecimento do usuário é comparada com várias armazenadas a fim de identificar o mesmo como conhecido ou desconhecido. Para realizar a identificação é necessário que o banco de dados com imagens contenha não só características faciais de um conjunto específico de pessoas, mas sim o máximo de faces possíveis para que seja possível fazer uma identificação de conhecido para qualquer indivíduo no ambiente (FAGERTUN, 2005). A Figura 12 mostra o procedimento da tarefa de identificação.

Figura 12 - Processo de Identificação



Fonte: SILVA, 2015

Na tarefa de identificação, o teste facial é feito com base na premissa de que todas as faces do teste são de pessoas conhecidas. Assim é comum o banco de dados conter imagens providas de diversas bases espalhadas em locais diferentes de onde o sistema se encontra operando. Quanto maior diversidade de amostras nas bases utilizadas maiores as chances de identificar um indivíduo como conhecido em um ambiente aberto e com muitas pessoas. Com isso, duas métricas são utilizadas para medir a porcentagem de faces que são identificadas de forma correta pelo sistema, a Taxa de Identificações Corretas (TIC) e Taxa de Identificações Falsas (TIF) (KRIG, 2014). Essas métricas são utilizadas para avaliar a performance do sistema de reconhecimento e para mostrar em qual aspecto ele possa estar apresentando menor

eficiência ou comportamento incorreto. Como exemplo, uma vez que a TIF está alta o sistema pode estar com dificuldades devido ao baixo número de amostras de faces no banco de dados.

A tarefa de observação (reconhecimento geral) é uma generalização da tarefa de identificação, que nesse caso inclui a possibilidade de considerar pessoas desconhecidas. Na observação existe a possibilidade de montagem de uma lista de observação, contendo faces desconhecidas a serem observadas em um determinado ambiente. Assim, é feita a verificação dos indivíduos que vão surgindo no ambiente a fim de descobrir se algum desses possui correspondência facial para a lista de observação. Nos testes de correspondência da tarefa de observação as métricas de TIC e TIF são utilizadas, mas é possível incluir as métricas de TFA e TRF para mensurar o quanto faces capturadas de indivíduos desconhecidos estão sendo marcados como correspondentes na lista de observação (KRIG, 2014)

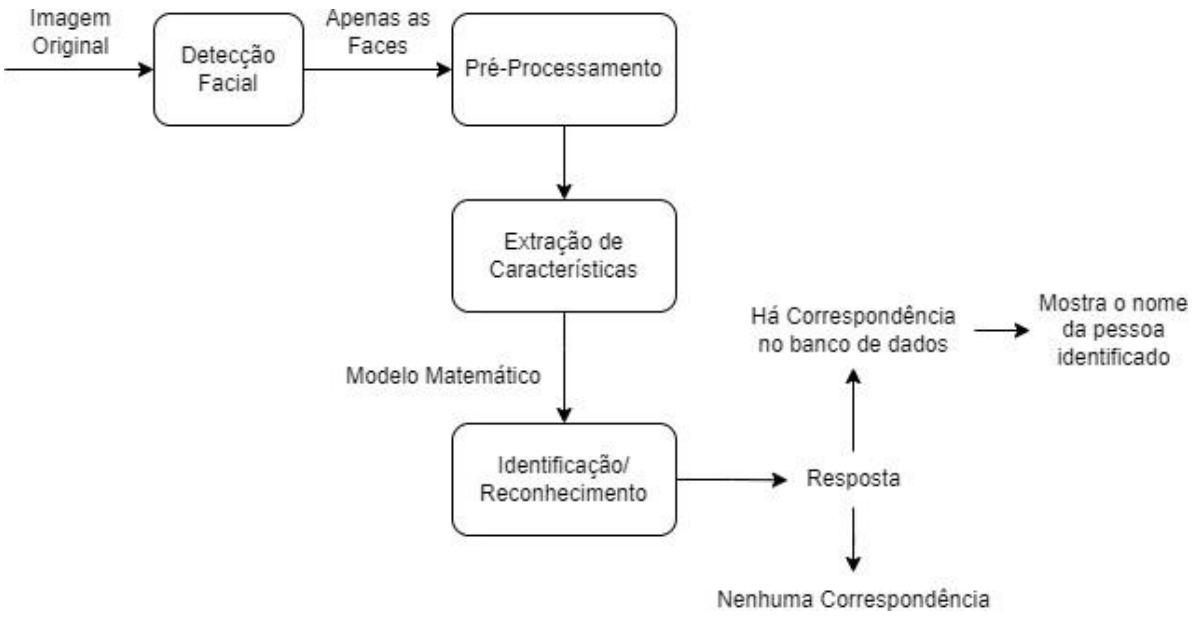
2.9 PROCESSO DE RECONHECIMENTO FACIAL

O uso de reconhecimento facial é uma atividade complexa em qualquer uma de suas possíveis tarefas, cuja execução depende de diversas etapas que são executadas em uma ordem específica definindo o processo necessário para uso e construção de um sistema reconhecedor de faces, que inicia com uma base de imagens de faces que juntamente com os algoritmos necessários será submetida junto a este processo.

A primeira atividade executada é a detecção de faces, que irá encontras nas imagens do banco de dados as faces dos indivíduos presentes. Em seguida é executado o pré-processamento, que recebe as faces encontradas na tarefa anterior e aplica técnicas que visam melhorar a qualidade da imagem e consequentemente a performance do reconhecedor (GALTON,1888). A próxima atividade é a extração de características responsável por encontrar os atributos mais relevantes na separação dos tipos de faces presentes nas imagens pré-processadas, além de reduzir a dimensionalidade do domínio de imagens faciais. Por fim, é executada a Classificação e Correspondência de faces, onde as faces dos indivíduos têm suas identidades verificadas e validadas, como legitimas ou não, por meio de algoritmos de

classificação de padrões (ZHAO et al, 2003). A Figura 13 exibe um diagrama com o fluxo das atividades que integram o processo de reconhecimento facial.

Figura 13 - Fluxo do Processo de Reconhecimento Facial



Fonte: AUTOR, 2022

2.10 NUANCES DO RECONHECIMENTO FACIAL

No processo de reconhecimento facial, manipulação dos dados de entrada e pequenas alterações em características, como ruídos, mudança do equipamento de obtenção de imagem digital e ambiente de captura podem interferir na classificação, alterando o desempenho do algoritmo e a confiabilidade do resultado.

2.10.1 VARIAÇÃO NA APARÊNCIA DA FACE

Diferentes tipos de alterações podem ocorrer em uma imagem facial, como iluminação, expressão fácil, pose facial, obstáculos na imagem, tipo de equipamento que fez a captura e a qualidade de captura dessa imagem (FAGERTUN, 2015) como mostrado nas Figuras 14 e 15 onde foram tiradas duas fotos com óculos em uma e na outra sem óculos e com a fonte de iluminação na parte inferior do rosto.

Figura 14 - Foto do Banco com Óculos

Fonte: AUTOR, 2022

Figura 15 - Foto do Banco sem Óculos com Iluminação Inferior

Fonte: AUTOR, 2022

Essas características influenciam no resultado dos algoritmos classificadores. Por serem características que demandam mais processamento dos algoritmos, estas podem gerar pontos estáticos e estratégicos para o conhecimento facial (SHARIF et al, 2017)

2.10.2 Análise da Face no Espaço da Imagem

Uma imagem é caracterizada pela sua dimensionalidade, que por sua vez é calculada com base no tamanho de suas linhas e colunas em quantidade de pixels. Uma imagem de entrada contém dimensão NxM, porém somente um subconjunto de coordenadas cotem uma face. Como exemplo, podemos escolher uma imagem com 32x32, linhas e colunas em pixels respectivamente, sua dimensionalidade será 1024 dimensões. O processo de extração de características baseados em características tradicionais em reconhecimento é facilitado efetuando a redução da dimensionalidade a um nível aceitável, reorganizando os pixels na imagem (levando em consideração sua vizinhança), a imagem permanecerá com as mesmas características, desde que todas as outras imagens sejam rearranjadas de forma idêntica. (TURK, 2001). Por outro lado, faces podem não ser detectadas caso estejam contidas em um subconjunto pequeno de uma imagem que contenha uma dimensionalidade muito grande, ou seja, ocupem uma pequena porção das dimensões. Nestes casos os algoritmos classificadores não conseguem aplicar suas técnicas de extração de características para o reconhecimento.

2.10.3 Gerenciar Porções Não Lineares e Convexa da Imagem.

Para a diminuição dos custos reconhecimento é feita a redução de uma imagem, é necessário que seja realizada a normalização da imagem, diminuindo a não linearidade e convexa da imagem. Isso pode ser obtido através das aplicações e dos algoritmos Análise de Componentes Principais (PCA) e Análise Linear Discriminante (LDA). As metodologias podem ser combinadas junto aos motores de classificação como Redes Neurais, *Support Vector Machine* (SVM) entre outros, utilizados para esse fim (FAGERTUN, 2015).

2.11 OPENCV

A biblioteca de visão computacional mais difundida no mercado chama-se OpenCV (open source Computer Vision) (Figura 16) de Código Aberto que em 1999, Gary Bradski, trabalhando na Intel Corporation, lançou o OpenCV com a esperança de acelerar o desenvolvimento na área de visão computacional e de inteligência artificial, fornecendo uma sólida infraestrutura para qualquer pessoa interessada trabalhar nesse campo. A biblioteca é escrita em C e C++ e pode rodar em Linux, Windows e Mac OS X. Existe desenvolvimento ativo nas interfaces para Python, Java, MATLAB, entre outras linguagens incluindo a portabilidade da biblioteca para Android e IOS para aplicações móveis. (KAEHLER; BRADSKI, 2017)

Figura 16 - OpeCV Logo

Fonte: OPENCV.ORG(2014)

O OpenCV tem recebido bastante suporte através dos anos da Intel como também do Google, mas especialmente da Itseez (recentemente adquirida pela Intel), que fez grande parte do desenvolvimento inicial. Finalmente Arraiy se juntou para manter o Opencv.org gratuito. O OpenCV foi projetado para eficiência computacional e com um forte foco em aplicativos de tempo real. Ele é escrito em C++ otimizado e pode tirar proveito dos processadores Multicores. (KAEHLER; BRADSKI, 2017)

Um dos objetivos do OpenCV é fornecer uma infraestrutura de visão computacional simples de usar que ajuda os desenvolvedores a criar aplicações de visão bastante sofisticadas rapidamente. A biblioteca do OpenCV contém atualmente mais de 500 funções que abrangem muitas das áreas de visão computacional,

incluindo inspeção em produtos de fábrica, imagens médicas, segurança, interface de usuário, calibração de câmeras e robótica. Como a visão computacional e o aprendizado de máquina costumam andar de mãos dadas, o OpenCV também contém uma biblioteca de aprendizado de máquina completa e de uso geral. (KAEHLER; BRADSKI, 2017)

Mesmo que existem outras tecnologias e bibliotecas para análise e reconhecimento de padrões, análise de movimento e rastreamento de objetos, o OpenCV ainda é a solução mais adequada para o pré-processamento e manipulação de imagens e por conta disso a comunidade mantém uma biblioteca atualizada. Essas informações podem ser verificadas por meio do repositório do OpenCV no GitHub. (KAEHLER; BRADSKI, 2017)

3 MÉTODO E MATERIAIS UTILIZADOS

A metodologia geral de pesquisa respeita as características da pesquisa aplicada, entendendo esse modelo como “[...] objetiva gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos”. (GERHARDT; SILVEIRA, 2009, p.35).

A essa pesquisa foi utilizada uma abordagem quantitativa, que esclarece Fonseca (2002, p.20):

> Diferentemente da pesquisa qualitativa, os resultados da pesquisa quantitativa podem ser quantificados. Como as amostras geralmente são grandes e consideradas representativas da população, os resultados são tomados como se constituíssem um retrato real de toda a população alvo da pesquisa. A pesquisa quantitativa se centra na objetividade. Influenciada pelo positivismo, considera que a realidade só pode ser compreendida com base na análise de dados brutos, recolhidos com o auxílio de instrumentos padronizados e neutros. A pesquisa quantitativa recorre à linguagem matemática para descrever as causas de um fenômeno, as relações entre variáveis etc. A utilização conjunta da pesquisa qualitativa e quantitativa permite recolher mais informações do que se poderia conseguir isoladamente.

Esse conceito se aplica ao TCC pois as aplicações do nosso trabalho podem ser quantificadas por meio da análise de métricas de desempenho dos algoritmos treinados.

Quanto à natureza da pesquisa aplicada, Gerhard e Silveira (2009, p.35) definem aquela cujo objetivo é o de “gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.” O que define a nossa pesquisa, tendo em vista que a implantação do tema, é necessária uma aplicação prática para solucionar o problema proposto.

3.1 WEBCAM HD LOGITECH C270

A C270 HD Webcam oferece chamadas em conferência nítidas e suaves (720p/30 qps) em formato *widescreen*. A correção automática de luz mostra cores naturais e realistas. Possui iluminação com correção automática de luz para a melhor imagem possível. Na Figura 17 é mostrada a câmera usada no protótipo.

Figura 17 - Logitech C270

Fonte: LOGITECH (2020)

3.2 CRIAÇÃO DO BANCO DE IMAGENS

Para a criação do banco de dados é necessário realizar diversas capturas de fotos dos usuários que utilizarão o sistema.

Nesse sistema serão capturadas 50 fotos de duas pessoas para exemplificação do funcionamento. Serão eles: Misael Marques Ricardo e Jefté Marques Ricardo.

Serão necessárias capturas de fotos das seguintes formas 3 vezes em cada modo.

a. Frontal.

b. Rosto levemente para a esquerda.

c. Rosto levemente para cima.

d. Rosto levemente para a direita.

e. Rosto levemente para baixo.

f. Expressão facial neutra.

g. Expressão facial feliz.

h. Expressão facial triste.

i. Rosto frontal com luz forte (lanterna) incidindo na frente.

j. Rosto frontal com luz forte (lanterna) incidindo pela esquerda.

k. Rosto frontal com luz forte (lanterna) incidindo por cima.

l. Rosto frontal com luz forte (lanterna) incidindo pela direita.

m. Rosto frontal com luz forte (lanterna) incidindo debaixo.

Para a avaliação do classificador é necessário realizar capturas de fotos dos usuários que utilizarão o sistema diferentes das usadas para treinamento.

Nesse sistema serão capturadas 10 fotos de duas pessoas para exemplificação do funcionamento. Serão eles: Misael Marques Ricardo e Jefté Marques Ricardo.

Serão necessárias capturas de fotos das seguintes formas 2 vezes em cada modo.

a. Piscando
b. Rindo
c. Com óculos de grau
d. Óculos Escuro
e. Espantado

3.3 SOFTWARE DE RECONHECIMENTO

Para a escolha do melhor método de reconhecimento facial foram estudados três algoritmos diferentes, sendo eles apresentados a seguir:

3.4 EIGENFACES

O método *Eigenfaces* busca um conjunto de características que não depende das formas geométricas da face (olhos, nariz, orelhas e boca), utilizando toda a informação da representação facial (KSHIRSAGAR; BAVISKAR; GAIKWAD, 2011). Seu funcionamento é similar ao funcionamento do PCA, entretanto é utilizada uma leve otimização para reduzir a matriz de covariância, reduzindo o processamento necessário para fazer o cálculo de seus auto-vetores e auto-valores. (BISSI, 2018)

Vale lembrar que no *Eigenfaces* a iluminação é um aspecto importante a ser considerado, uma vez que ele cria os *eigenvectors* baseado nas diferentes variações das faces, bem com as variações de luminosidade, portanto se tivermos muitas faces de uma mesma pessoa com variações faciais consideráveis seu resultado será melhor, porém com a variação de iluminação para cada face, ele usará essas

variações para determinar as *eigenfaces* (faces fantasmas) de forma que essas variações prejudiquem a eficiência do algoritmo. (BISSI, 2018)

Baseadas na Teoria da Informação, as *eigenfaces* buscam identificar um pequeno número de características que são relevantes para diferenciar uma face de outras faces. Essas características podem ser analisadas apenas com a variação dos valores assumidos pelos pixels, em um conjunto de imagens de faces. (DINIZ et al., 2013).

O nome *eigenfaces* é atribuído aos auto-vetores (*eigenvectors*) da matriz de covariância das imagens das faces do banco de faces de treinamento por possuírem aspectos de faces. No *eigenfaces* o conjunto de imagens de treinamento irão extrair as componentes mais relevantes da face humana e criar uma face média com base no conjunto de dados de imagens e com a variação dos valores dessas componentes é possível representar uma imensa gama de faces apenas fazendo multiplicações escalares utilizando os auto-valores, na verdade cada face poderá ser representada como combinação linear das diversas *eigenfaces* (BISSI, 2018). Na Figura 18 temos um exemplo das faces do nosso banco de dados.

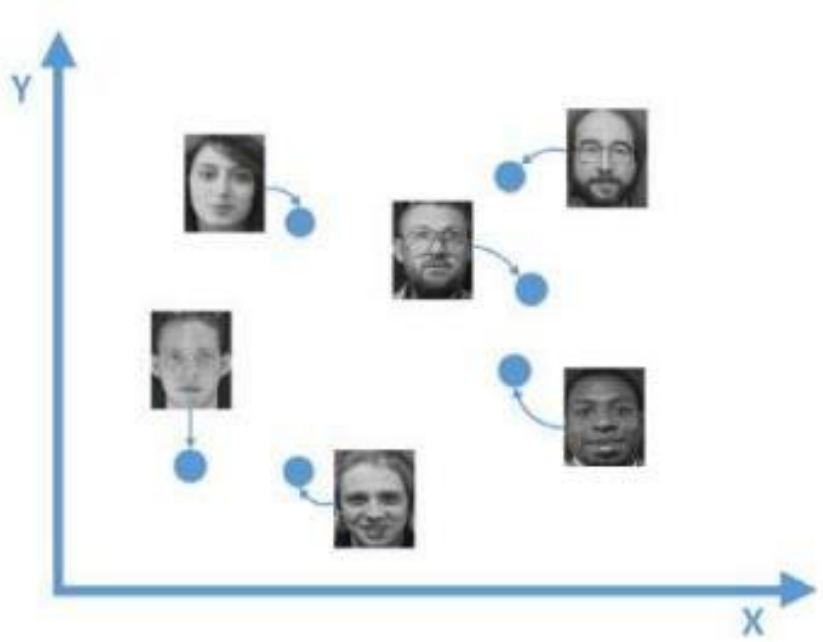
Figura 18 - Exemplo de Faces do Banco de Imagens

Fonte: AUTOR, 2022

Este método possibilita a classificação de faces a partir do cálculo da distância entre a imagem sendo analisada e a imagem sendo analisada projetada no novo espaço. Se o valor da distância estiver dentro de uma distância limite (*threshold*), a

imagem sendo analisada é considerada face, caso contrário é considerada como não face. O método *eigenfaces* é interessante por possibilitar, além da classificacão, a reconstrução e a compactação de imagens de faces. Para o cálculo da distância é escolhido as técnicas que mais se adaptem ao banco de imagens usado, sendo as mais usadas a distância euclidiana aplicado ao algoritmo KNN. Na Figura X temos a representação de um espaço de faces no plano cartesiano. (BISSI, 2018)

Figura 19 - Representação do Espaço de Faces no Plano Cartesiano

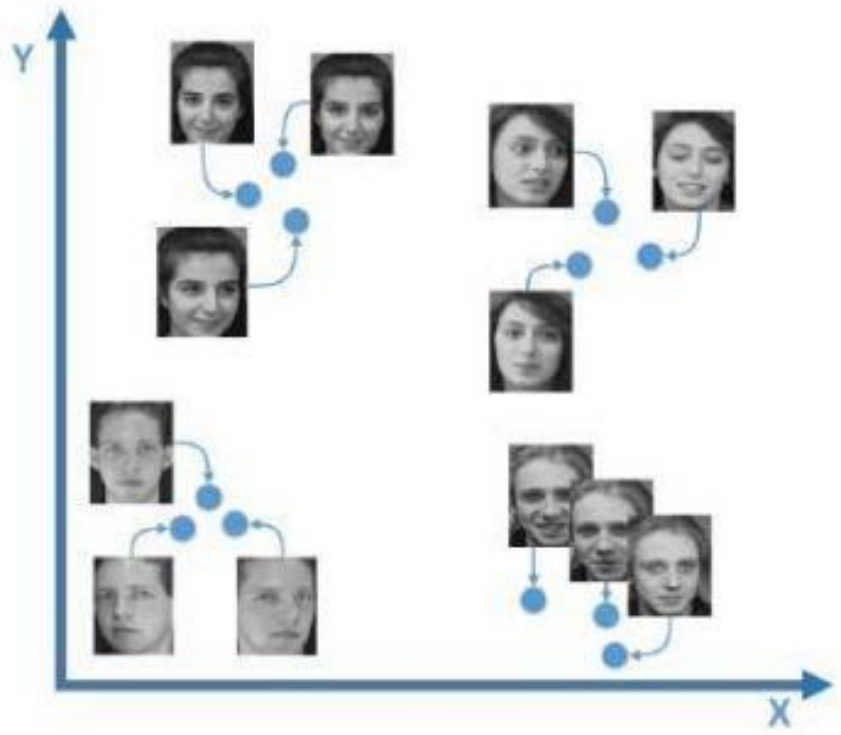


Fonte: SILVA, 2016

3.5 FISHERFACES

Similar *eigenfaces*, as *fisherfaces* podem ser visualizadas como imagens de características onde as características das *fisherfaces* são variações de aparência presentes nas imagens de cada indivíduo, tais como variações de luminosidade, poses e expressões faciais. Assim como as imagens no espaço de dados possuem um valor para cada atributo, os vetores e características possuem um valor para cada *fisherface*s. Na Figura 20 é mostrado a projeção das faces relativo ao *Fisherfaces*. (BISSI, 2018)

Figura 20 - Espaço de Faces Fisherfaces



Fonte: SILVA, 2016

Temos então que quanto mais diferente for uma pessoa da outra, mais distantes deverão estar suas projeções, e quanto mais parecidas forem as imagens de uma pessoa mais próximas deverão estar suas projeções. (BISSI, 2018)

O algoritmo *Fisherface* é composto por 8 etapas, sendo elas:

1. Cálculo da face média por classe: A face média por classe é o resultado da soma pixel a pixel de todas as imagens de uma mesma classe do conjunto de treinamento dividido pelo total de imagens da classe.

2. Cálculo de face média geral: A face média é o resultado da soma pixel a pixel de todas as imagens do conjunto de treinamento dividido pelo total de imagens.

3. Transformação das imagens em vetores: A transformação de uma imagem em um vetor consiste em concatenar as linhas da imagem, unindo o último pixel de cada linha com o primeiro pixel da linha seguinte, formando um vetor de pixels.

4. Construção da matriz de dispersão intraclasse: A matriz de dispersão intraclasse calcula o quanto os dados estão dispersos dentro da própria classe, isto é, o quanto as imagens de um mesmo indivíduo diferem umas das outras.

5. Construção da matriz de dispersão interclasse: A matriz de dispersão interclasse Sb calcula o quanto os dados estão dispersos entre as classes, isto é, o quanto as imagens de indivíduos distintos diferem umas das outras.

6. Cálculo das *fisherfaces*: As *fisherfaces* do *Linear Discriminant Analysis* (LDA) e seus valores associados são, respectivamente, os autovetores e autovalores.

7. Cálculo dos vetores de características: As imagens de face são representadas pela soma ponderada das *fisherfaces*. Os pesos dessa combinação foram os vetores de características das imagens de face. Para calcular o peso de cada *fisherfaces* na representação de uma imagem de face, multiplica-se a imagem de face, subtraída da média, por cada uma *fisherfaces*.

8. Cálculo da Similaridade: O reconhecimento pode ser realizado por meio do uso de um classificador ou de uma métrica de similaridade. Uma das métricas de similaridade mais comum é a distância euclidiana. (BISSI, 2018)

3.6 LOCAL BINARY PATTERNS HISTOGRAMS(LBPH)

Eigenface e *Fisherface* possuem uma abordagem holística, tratando os dados como um vetor em um espaço de alta dimensão. Como a alta dimensionalidade é ruim, um subespaço de menor dimensão é identificado, preservando somente as informações úteis. A abordagem Eigenface maximiza a dispersão total, podendo conduzir a problemas se a variância for gerada por uma fonte externa, por causa de componentes com um desvio máximo de mais de todas as classes não são necessariamente úteis para a classificação. Então para preservar algumas informações discriminativas foi aplicado a LDA. (OPENCV,2013).

Os Padrões binários locais descrevem apenas as características locais de um objeto, reduzindo a dimensionalidade, resumindo a estrutura local em uma imagem comparando os pixels com a sua vizinhança. Se a intensidade do pixel central é maior, igual ao seu vizinho, denota-se como 1 caso contrário 0. (WAGNER, 2014).

A representação proposta por Ahonen é dividir a imagem LBP em regiões locais e extrair um histograma cada. O vetor de características espacialmente reforçadas é obtido concatenando os histogramas locais. Estes histogramas são chamados *Local Binary Patterns Histograms* (LBPH). (OPENCV,2013).

3.7 BIBLIOTECAS

Biblioteca é uma coleção de subprogramas utilizados no desenvolvimento de software. Bibliotecas contém código e dados auxiliares, que provém serviços a programas independentes, o que permite o compartilhamento e a alteração de código e dados de forma modular.

3.7.1 CV2

OpenCV é uma biblioteca de programação, de código aberto possuindo mais de 500 funções, é usada para diversos tipos de análise em imagens e vídeos, reconhecimento facial, detecção e análise de textos entre outros. (CEDRO TECHNOLOGIES, 2018)

3.7.2 NUMPY

A biblioteca NumPy facilita trabalhar com arranjos, vetores e matrizes. Como vamos trabalhar com imagens digitais, que são representadas como matrizes de pixels, esse pacote é indispensável para aumentar a produtividade. (BARELLI, 2018)

3.7.3 PIL

A PIL *(Python Imaging Library)* adiciona recursos de processamento de imagens ao seu interpretador Python. Esta biblioteca fornece um amplo suporte e recursos de processamento de imagem bastante poderosos. (PILLOW, 2021)

3.7.4 HAAR CASCADE

É um método de detecção de objetos proposto por Paul Viola e Michael Jones. Esse método emprega várias funções retangulares ou senoidais na procura de

padrões. A biblioteca OpenCV possui vários modelos já treinado no reconhecimento de vários padrões, inclusive faciais em seu GitHub. (OPENCV, 2020)

3.8 KAGGLE

O aprendizado de Data Science pode ser um grande desafio para quem não conhece as melhores ferramentas e soluções. Para quem conhece, o caminho é mais fácil e intuitivo, com a ajuda de outros desenvolvedores. Ou seja, o aspecto comunitário da tecnologia é um dos fatores que mais ajudam os novatos. Uma prova disso é a importância de plataformas como o Kaggle, que servem como uma base para a evolução de muitos programadores.

O Kaggle é uma plataforma para aprendizado de ciência de dados. É também uma comunidade, a maior da internet, para assuntos relacionados com Data Science.

3.8.1 BANCO DE DADOS

As bases de dados disponíveis no Kaggle estão abertas para exploração e para resolução de problemas. São dados sempre limpos (sem erros, sem dados faltantes, dados padronizados e formatados), para que o cientista de dados foque mais em sua lógica e menos no tratamento dessas informações.

3.8.2 GLASSES OR NO GLASSES

Este conjunto de dados é de um projeto Kaggle do curso T81-855: *Applications of Deep Learning* na Washington University em St. Louis . Para esta competição, os alunos deveriam determinar se uma pessoa está usando óculos ou não. No entanto, seus assuntos de teste não são pessoas reais. Uma Rede Neural Adversarial Generativa (GAN) criou todas as pessoas que você vê neste banco. A rede GAN cria essas imagens usando um vetor latente de número 512. (Figura 21) A atribuição do Kaggle fornece os vetores latentes e as faces produzidas por esses vetores. Ambos podem ser úteis para você classificar se alguém está usando óculos ou não. (KAGGLE, 20020a)

Figura 21 - Imagens geradas com e sem óculos.

Fonte: KAGGLE, 2020a

Os óculos vêm em muitas formas diferentes, com óculos de sol, óculos de armações finas e mias grossas. A Figura 22 mostra uma amostra de alguns dos tipos de óculos contidos no conjunto de dados.

Figura 22 - Amostras com diferentes tipos de óculos

Fonte: KAGGLE, 2020b

3.8.3 FACEMASK DETECTION

Este conjunto de dados é uma versão editada, com imagens menores (224 × 224 em vez de 1024 × 1024 para fins de alimentá-lo no MobileNetV2) que foram convertidas em tons de cinza. (Figura A inspiração foi criar um detector de máscara facial para ajudar a identificar as pessoas que precisam tomar medidas mais preventivas nestes tempos difíceis com o Coronavírus. (KAGGLE, 20020b)

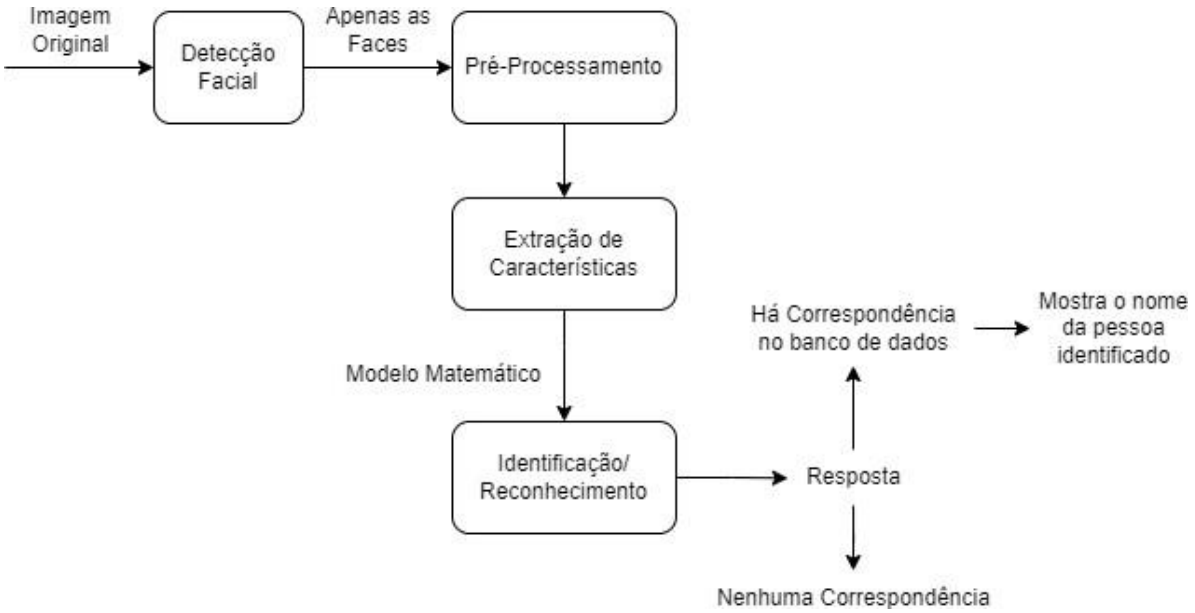
Figura 23 - Exemplo do *dataset* com máscara.

Fonte: KAGGLE, 2020b

4 RESULTADOS E DISCUSSÕES

Para realização do reconhecimento visual foram realizadas diversas etapas como mostrada na Figura 24, mostrando o fluxograma da programação.

Figura 24 - Fluxograma



Fonte: AUTOR, 2022

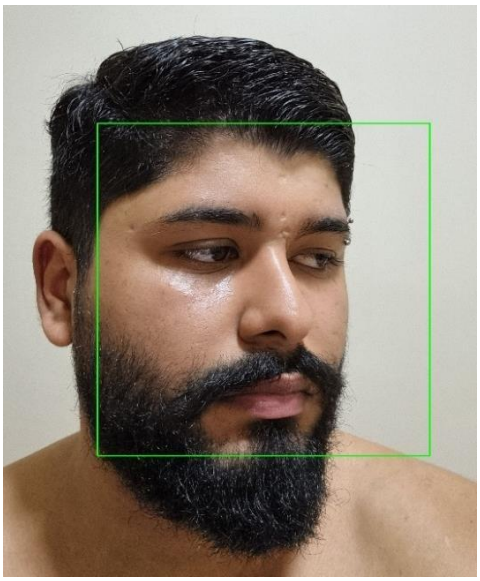
O primeiro passo é realizar as detecções das faces com a técnicas Haar Cascade, que retorna o posicionamento das faces detectadas, como mostra a Figura 25 e desenha um retângulo em volta delas (Figura 26), então recortamos a face e convertermos para a escala de cinza como mostra a face na Figura 27.

Figura 25 - Posicionamento das faces detectadas.

```
facesDetectadas = detectorFace.detectMultiScale(imagemCinza,
                  scaleFactor=1.2, minSize=(100,100))
for (x, y, l, a) in facesDetectadas:
        imagemFace = cv2.resize(imagemCinza[y:y + a, x:x + l],
                                (200, 200))
        imagemRed = cv2.rectangle(imagemRed,
                  (x, y), (x + l, y + a), (0,255,0), 2)
        cv2.imwrite("cor" + "Pessoa." +
        str(pessoa) + "." + str(num)+ ".jpg", imagemFace)
        cv2.waitKey(0)
```

Fonte: AUTOR, 2022

Figura 26 - Face Encontrada



Fonte: AUTOR, 2022

Figura 27 - Face recortada em cinza e redimensionada

Fonte: AUTOR, 2022

Após o recorte, utilizamos essas fotos para construir os reconhecedores faciais das três que estamos utilizando, *eigenfaces, fisherfaces, e* lbph com os devidos parâmetros que definimos conformes foram feitos os ajustes, como mostra a Figura 28.

Figura 28 - Construção dos Reconhecedores

```
eigenface = cv2.face.EigenFaceRecognizer_create(80, 36000)
fisherface = cv2.face.FisherFaceRecognizer_create(3, 2500)
lbph = cv2.face.LBPHFaceRecognizer_create(2, 2, 7, 7, 50)
```

Fonte: AUTOR, 2022

Então percorremos todas as imagens do arquivo do banco de treinamento, onde se encontram as 26 imagens de cada um dos usuários do sistema, retirando as identificações de cada foto para a realização do treinamento, as fotos são convertidas para Numpy, necessárias para o treinamento como mostra a Figura 29.

Figura 29 - Conversão de imagem e retirada de Identificação

```
def getImagemComId():
    caminhos = [os.path.join('foto_treinamento', f)
                for f in os.listdir('foto_treinamento')]
    faces = []
    ids = []
    for caminhoImagem in caminhos:
        imagemFACE = cv2.cvtColor(cv2.imread(caminhoImagem),
                                  cv2.COLOR_BGR2GRAY)
        imagemNP = np.array(imagemFACE, 'uint8')
        id = int(os.path.split(caminhoImagem)[-1].split(".")[1])
        ids.append(id)
        faces.append(imagemNP)
```

Fonte: AUTOR, 2022

Por fim é realizado o treinamento, como mostra a Figura 30.

Figura 30 - Treinamento dos reconhecedores

```
print("Treinando...")
eigenface.train(faces, ids)
eigenface.write('classificadorEigenAval.yml')

fisherface.train(faces, ids)
fisherface.write('classificadorFisherAval.yml')

lbph.train(faces, ids)
lbph.write('classificadorLBPHAval.yml')

print("Treinamento realizado")
```

Fonte: AUTOR, 2022

Agora seguimos para o algoritmo de reconhecimento facial, utilizamos a mesma estrutura para todos os reconhecedores, mudaremos somente qual arquivo é

iniciado, mudando assim o classificador, como mostra a Figura 31, onde foi iniciado o reconhecedor *eigeinface*.

Figura 31 - Inicialização do reconhecedor Eigenface

```
detectorFace = cv2.CascadeClassifier(cv2.data.haarcascades +
                    "haarcascade_frontalface_default.xml")
reconhecedor = cv2.face.EigenFaceRecognizer_create()
reconhecedor.read("classificadorEigenface.yml")
```

Fonte: AUTOR, 2022

O algoritmo percorre dessa vez todas as imagens do banco separadas para o teste. (Figura 32)

Figura 32 - Reconhecimento de faces

```
for caminhoImagem in caminhos:
    imagemCor = cv2.imread(caminhoImagem)
    imagemCinza = cv2.cvtColor(cv2.imread(caminhoImagem),
                               cv2.COLOR_BGR2GRAY)
    id = int(os.path.split(caminhoImagem)[-1].split(".")[1])
    ids.append(id)
    faces.append(imagemCinza)
    facesDetectadas = detectorFace.detectMultiScale(imagemCinza,
                        scaleFactor=1.2, minSize=(100,100))
```

Fonte: AUTOR, 2022

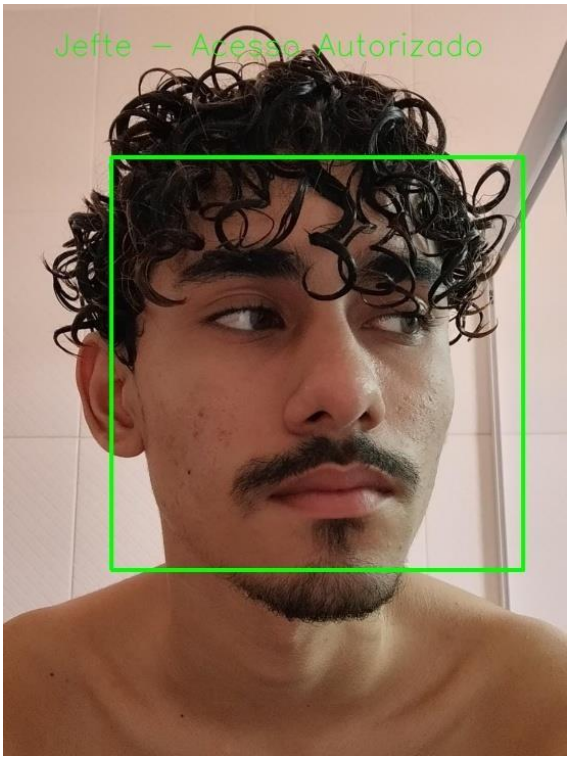
Após a nova detecção das faces, é realizado o processo de reconhecimento, retornando a identificação reconhecida com o nível de confiança do processo. (Figura 33). Então são substituídas as identificações pelos nomes dos que obterão acesso ao sistema. Na Figura 34 temos um exemplo usando *Eigenface*, Na Figura 32, usando *Fisherface,* e na Figura 36, usando LBPH.

Figura 33 - Inserção de Identificação

```
for (x, y, l, a) in facesDetectadas:
    imagemFace = cv2.resize(imagemCinza[y:y + a, x:x + l],
                            (200, 200))
    imagemCor = cv2.rectangle(imagemCor, (x, y), (x + l, y + a),
                              (0,255,0), 6)
    id, confianca = reconhecedor.predict(imagemFace)
    nome = ""
    if id == 1:
        nome = 'Misael'
    elif id == 2:
        nome = 'Jefte'
    imagemRed= cv2.resize(imagemCor, (552, 736))
    cv2.putText(imagemRed, nome + " - Acesso Autorizado",
        (50,50), font, 1, (0,255,0), 1, cv2.LINE_AA)
```

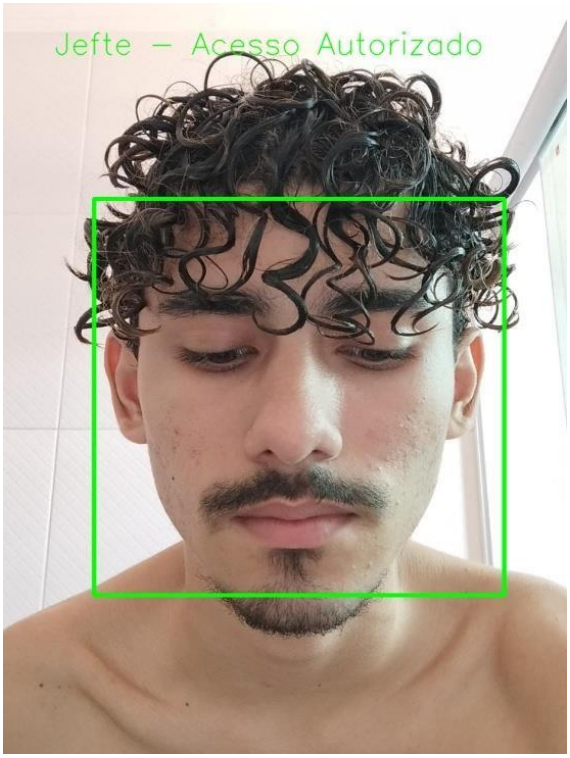
Fonte: AUTOR, 2022

Figura 34 - Reconhecimento com Eigenface



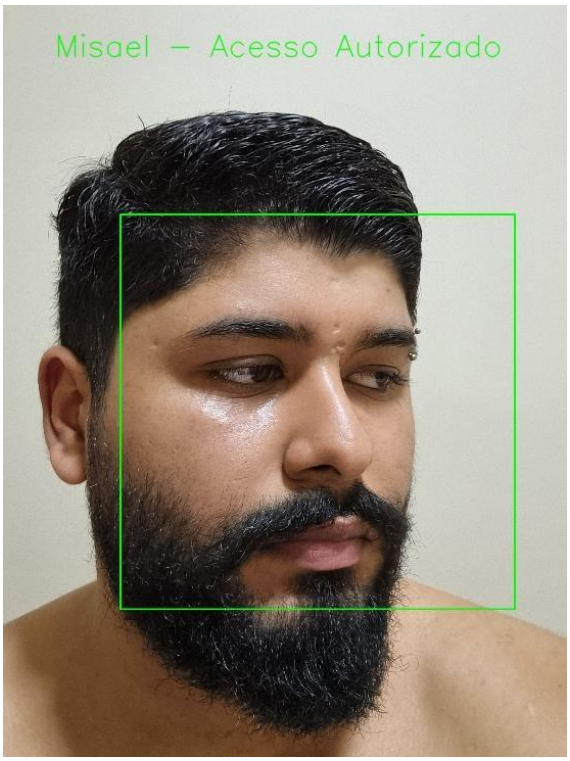
Fonte: AUTOR, 2022

Figura 35 - Reconhecimento com Fisherface



Fonte: AUTOR, 2022

Figura 36 - Reconhecimento com LPBH



Fonte: AUTOR, 2022

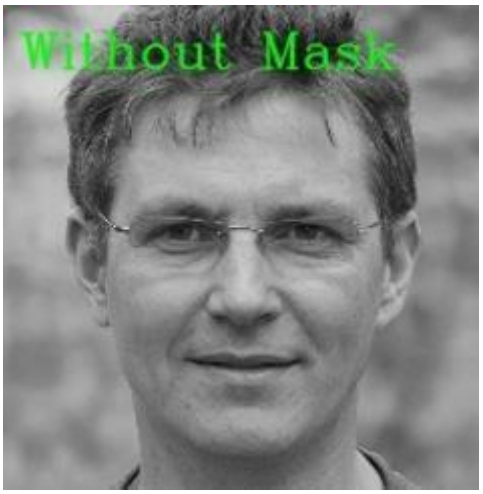
Os mesmos passos foram replicados para os outros bancos de imagens, tanto quanto o de óculos quanto o de máscaras. O banco de dados máscaras foi renomado e foi separado 150 imagens para treinamento sendo elas 75 com máscaras e 75 sem máscaras, e 50 imagens para teste, com mostra a Figura 37.

Figura 37 - Banco de treinamento

Fonte: AUTOR, 2022

Com as imagens de treinamento foi possível realizar o teste e classificar as imagens com e sem máscara como mostra a Figura 38 e 39.

Figura 38 - Imagem do teste sem máscara.



Fonte: AUTOR, 2022

Figura 39 - Imagem do teste com máscara.

Fonte: AUTOR, 2022

Seguindo os mesmos procedimentos treinamos outra inteligência utilizando o banco de dados das pessoas com óculos que foi renomado manualmente pois estava com a nomenclatura sem distinção no banco de dados, então foi separado 350 imagens para treinamento sendo elas 175 com óculos e 175 sem óculos, e 150 imagens para teste, com mostra a Figura 40.

Figura 40 - Banco de teste

Fonte: AUTOR, 2022

Com as imagens de treinamento foi possível realizar o teste e classificar as imagens com e sem óculos.

4.1 RESULTADOS GERAIS DOS TESTES NAS TRÊS MODALIDADES

Nas Tabelas 1 temos os resultados do Algoritmo de Eigenface para os três bancos da dados, na Tabela 2 temos os testes referentes para o Fisherface e por último temos a tabela dos testes para o LBPH.

Tabela 1 - Teste do Eigenface

	Eigenfaces		
Banco	Corretos	Errados	Acurácia
Faces	16	4	80%
Máscara	15947	3851	80.54%
Óculos	0	168	0%

Fonte: AUTOR, 2022

Tabela 2 - Teste do Fisherface

	Fisherface		
Banco	Corretos	Errados	Acurácia
Faces	17	3	85%
Máscara	19795	3	99.98%
Óculos	164	4	97.61%

Fonte: AUTOR, 2022

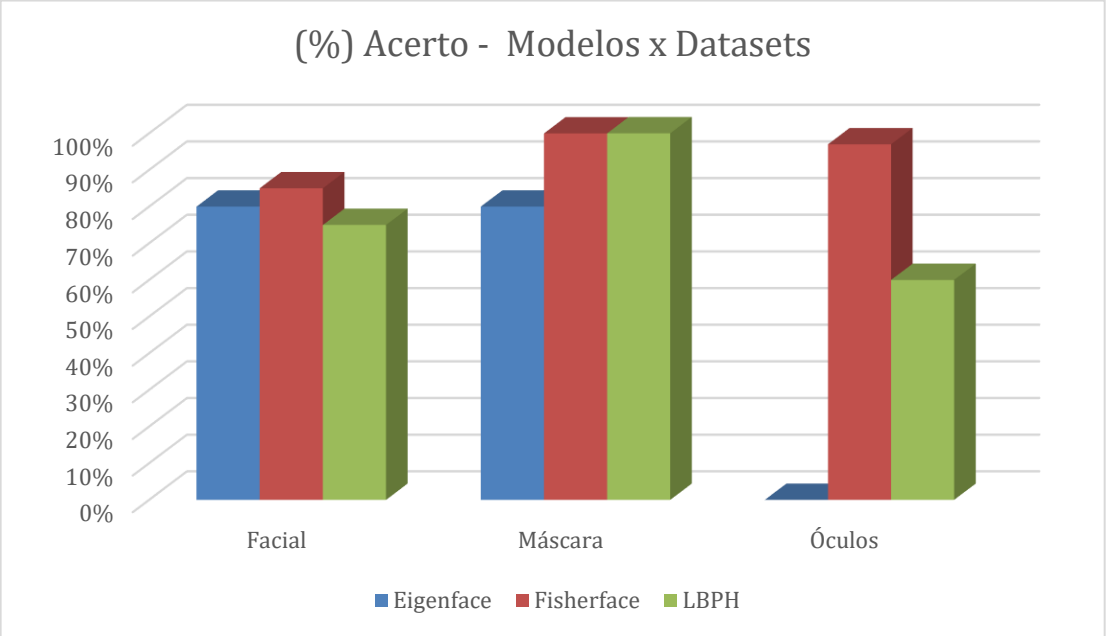
Tabela 3 - Teste do LBPH

	LBPH		
Banco	Corretos	Errados	Acurácia
Faces	15	5	75%
Máscara	19798	0	100%
Óculos	101	67	60.11%

Fonte: AUTOR, 2022

A Figura 41 mostra a porcentagem total de acerto do sistema treinados para reconhecer dois indivíduos diferentes, diferenciar entre pessoas com e sem máscaras e reconhecimento de pessoas com óculos nos três modelos.

Figura 41 - Taxa de Acerto do sistema



Fonte: AUTOR, 2022

A modalidade de *Fisherface* foi a que obteve maior sucesso para o banco de imagens criado. Depois de vários ajustes nos parâmetros de todas as três modalidades de reconhecedores faciais. Em segundo lugar veio o *Eigenfaces,* com a taxa de acerto de 80%, e por último a modalidade LBPH com a taxa de acerto de 75%. Mas mesmo com a diferença de porcentagem, ainda sim seria possível a utilização das três modalidades para utilização em um sistema de reconhecimento facial. Com algumas modificações nos parâmetros e um banco de imagens maior com sistema de iluminação ideal, a taxa de acerto pode alcançar valores maiores.

No segundo banco de dados constituído pelo banco de dados da máscara, o modelo de LBPH e Fisherface obteve maior sucesso de quase 100% de acurácia, por conta da máscara ser um diferencial forte, mesmo assim o Eigenface não se mostrou muito eficiente.

No último banco de dados dos óculos, o treinamento de Eigenface não foi conseguiu ser eficiente, o algoritmo não conseguiu classificar nenhuma imagem,

entretanto o modelo de Fisher face continuou sendo melhor entre os três, com mais de 90% de acerto.

5 CONCLUSÕES FINAIS

Neste trabalho desenvolveu-se um estudo para reconhecimento de faces, máscaras e óculos, através de interconexão de sistema, que são estes: sistema de aquisição de imagem, *software* de treinamento, e software de reconhecimento, após o treinamento foi feita a avaliação das melhores modalidades de reconhecimento facial.

Concluiu-se que para o banco utilizado com o algoritmo desenvolvido de reconhecimento, o algoritmo de *Fisherface* foi o que se saiu melhor nos testes, sendo ele o escolhido para uma futura aplicação em um sistema de reconhecimento facial mais completo.

O resultado dos testes pode melhorado através da ampliação do banco de imagens de treino, já que os testes foram realizados apenas com 26 imagens de cada usuário. O *software* desenvolvido foi realizado desde a visão teórica até a implementação prática sendo que todas as etapas intermediárias foram planejadas e elaborados após os estudos na área.

Com o sistema criado, fica planejado para futuros trabalhos a utilização desse sistema de reconhecimento integrado a um sistema composto por *front-end* e *back-end* conectado a um banco de dados, para a utilização em sistema de segurança.

5.1.1.1 REFERÊNCIAS BIBLIOGRÁFICAS

ALPAYDIN, Ethem. ***Introduction to Machine Learning.*** 4. Ed. MIT Press. Cambridge, 2020.

BARELLI, Felipe. **Introdução à visão computacional.** Casa do Código, 2018. 256 p.

BATTAGLIA, F., IANNIZZOTTO, G., BELLO, L. A. ***Person Authentication System Based on RFID Tags and a Cascade of Face Recognition Algorithms.*** IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, No. 8, August.

BAUMGARTEN, Gustavo. **Sistema de visão industrial: descubra tudo o que eles podem fazer por você.** Pollux. 9 out. 2018. Disponível em: <https://pollux.com.br/blog/ sistemas-de-visao-industriais-descubra-tudo-o-que-eles-podem-fazer-por-voce/>. Acesso em 30 out 2022.

BISSI, Thelry David. **Reconhecimento Facial com os algoritmos Eigenfaces e Fisherfaces**. Monografia, (Graduação) - Universidade Federal de Uberlândia, Minas Gerais, 2018.

BRAGA, Leonardo. **Reconhecimento facial Uber**. 2020, Disponível em: <https://motoristaonline.com/blog/reconhecimento-facial-uber/> Acesso em: 1 nov. 2022.

CEDRO TECHNOLOGIES. **OpenCV: Uma breve introdução à visão computacional com python.** 3 out. 2018.

CMU**, The Simon Initiative**, 2022. Disponível em: <https://www.cmu.edu/ simon/what-is-simon/herbert-a-simon.html>. Acesso em: 05 nov. 2022

COPELAND, Brian John. (2004). ***The Essential Turing***. Oxford University Press, 2004.

CORREA, Eduardo. **Meu Primeiro livro de Python.** 2. ed. 2020

COSTA, Vambaster José da. **Reconhecimento de Padrões Faciais:** Uma Síntese. Monografia, (Graduação) Universidade Tecnológica Federal do Paraná, Paraná. 2019

DINIZ, F. A. et al. **Redface:** um sistema de reconhecimento facial baseado em técnicas de análise de componentes principais e autofaces. Revista Brasileira de Computação Aplicada, v. 5, n. 1, p. 42-54, 2013.

FAGERTUN, Jens. **Face Recognition**. 2005. Dissertação (Mestrado). *Technical University of Denmark*, Lyngby, Denmark.

FONSECA, J. J. S. **Metodologia da pesquisa científica.** Fortaleza: UEC, 2002. Apostila.

Frantz, R. ***Herbert Simon.*** *Artificial Intelligence As A Framework For Understanding Intuition. Journal of Economic Psychology*. The Economic Psychology of Herbert A. Simon. 2003.

GALTON, F. *Personal identification and description*. Nature, 173-188, 1888.

Gates, K. ***Our Biometric Future****: Facial Recognition Technology and the Culture of Surveillance*. Critical Cultural Communication. NYU Press, 2011.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. **Métodos de Pesquisa**. – Porto Alegre: Editora da UFRGS, 2009

KAEHLER; Adrian; BRADSKI, Gary; ***Learning OpenCV 3:*** *Computer Vision in C++ with the OpenCV Library*. O´Reilly. 2017

KAGGLE. (2020a) Disponível em: <https://www.kaggle.com/code/lebegus/glasses-detection#1)-Data-cleaning:> Acesso em: 20 dez. 2022

KAGGLE. (2020b) Disponível em: < https://www.kaggle. com/datasets /pranavsingaraju/facemask-detection-dataset-20000-images:> Acesso em: 20 dez. 2022

KOK, Jost. (2009). ***ARTIFICIAL INTELLIGENCE****. Encyclopedia of life support systems*. Eolss Publishers Company. 2009.

KOVÁCS, Zsolts, **Redes Neurais Artificiais: Fundamentos e Aplicações: Um tecto básico** – 4 ed. – São Paulo: Editora Livraria da Fisica, 2006

KRIG, S. ***Computer Vision Metrics:*** *Survey, Taxonomy, and Analysis*. Apress, 2014

KSHIRSAGAR, V. P.; BAVISKAR, M. R.; GAIKWAD, M. E. ***Face recognition using Eigenfaces****.* – 3. ed. - IEEE, 2011. p. 302-306.

LOGITECH, **C270 HD WEBCAM.** Disponível em: < https://www.logitech.com /pt-br/products/webcams/c270-hd-webcam.960-000694.html>. Acesso em: 22 out 2022.

MILLER, April; ***PyCharm vs. VSCode:*** *Which Is the Better Python IDE?,* 3 Mar, 2022. Disponível em: <https://opendatascience.com/pycharm-vs-vscode-which-is-the-better-python-ide>. Acesso em: 10 nov. 2022

NERI, J. R. F, SANTOS, C. H. F E, FABRO, J. A. **Descrição Do Time GPR-2D.** Competição Brasileira de Robótica, 2011

NEVES, Enzo. **Aprendizado por Reforço #1**— Introdução. 23 fev. 2020. Disponível em: < https://medium.com/turing-talks/aprendizado-por-refor%C3%A7o-1-introdu%C3%A7%C3%A3o-7382ebb641ab> Acesso em: 07 nov. 2022.

OPENCV. **OPENCV**-*Open Source Computer Vision*. 2013 Disponível em: <http://opencv.org/>. Acesso em: 18 out. 2022

OPENCV.ORG. **Face recognition withOpenCV.** Disponível em: <http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html>. Acesso em: 18 out. 2022

PILLOW. **Pillow 9.2.0.** 2021. Disponível em:< https://pypi.org/project/Pillow/>. Acesso em 01 Jul 2021.

PIRES, C.A; LIMA, F.F; SILVA M. A; **O Reconhecimento facial aplicado a prevenção de fraudes** Monografia (Graduação), Universidade de São Paulo, São Paulo, 2020.

PYTHON, **HOME**. 2022. Disponível em: < https://www.python.org> Acesso em: 10 nov. 2022.

PRADO, Kelvin Salton do**. Comparação de técnicas de reconhecimento facial para identificação de presença em um ambiente real e semicontrolado**. Tese (Doutorado). Universidade de São Paulo, São Paulo. 2018.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**, - 3. ed. - Campus. 2004.

SHARIF, Muhammad et al. ***Face Recognition****: A Survey. Journal of Engineering Science & Technology Review*, v. 10, n. 2, 2017.

SILVA, Alex Lima; CINTRA, Marcos Evandro. **Reconhecimento de padrões faciais:** Um estudo. Encontro Nacional De Inteligência Artificial E Computacional, 2015. p. 224-231

SILVA, A. L. **Redução de características para classificação de imagens de faces.** Dissertação (Mestrado), Universidade do Estado do Rio Grande do Norte, Universidade Federal Rural do Semi-Árido, Mossoró. 2016.

SNEF BRASIL, **Sistema de Visão: Cabine e Técnica de Iluminação Adequadas para Células Robotizadas.** 2020. Disponível em: <https://snef.com.br/br/sistema-de-visao-cabine-e-tecnica-de-iluminacao-adequadas-para-celulas-robotizadas> Acesso em: 30 out. 2022

VCCODE. **Home,** 2022. Disponível em: <https://code.visualstudio.com/> Acesso em: 10 set. 2022.

WAGNER, P. ***Local binary patterns***. 2014. Disponível em: <http://www.bytefish.de/blog/local_binary_patterns/> Acesso em: 20 nov. 2022.

YANG, M. H.; AHUJA, N. & KRIEGMAN, D. ***Detecting Faces in Images:*** *A Survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 24, n. 1, pp. 34-58, 2002.

ZHAO, WENYI ET AL. **Reconhecimento facial:** uma pesquisa bibliográfica. Pesquisas de computação ACM (CSUR), v. 35, n. 4, p. 399-458, 2003.

5.1.1.2 APÊNDICE A – Algoritmo de redimensionar imagem

```
import cv2
largura, altura = 898, 1280
largura, altura = 449, 620
print(largura, altura)
detectorFace = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
print(detectorFace)

for pessoa in 1,2:
   for num in range(1,26):
      if pessoa == 1 and num == 20 :
            pass

      else:
            print("Pessoa" + str(pessoa) + "." + str(num)+ ".jpg")

            foto = ("Pessoa" + str(pessoa) + "." + str(num) )
            imagem = cv2.imread(foto+ ".jpg")
            imagemRedimensionada = cv2.resize(imagem, (largura, altura))
            imagemCinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
            facesDetectadas = detectorFace.detectMultiScale(imagemCinza,
                        scaleFactor=1.2, minSize=(100,100))
            for (x, y, l, a) in facesDetectadas:
                  imagemFace = cv2.resize(imagemCinza[y:y + a, x:x + l],
                                    (200, 200))
                  imagemRed = cv2.rectangle(imagemRed,
                        (x, y), (x + l, y + a), (0,255,0), 2)
                  cv2.imwrite("cor" + "Pessoa." +
                  str(pessoa) + "." + str(num)+ ".jpg", imagemFace)
                  cv2.waitKey(0)

            if cv2.waitKey(1) == ord('q'):
                  break
```

5.1.1.3 APÊNDICE B – Algoritmo de treinamento dos reconhecedores

```
import cv2
import os
import numpy as np
from PIL import Image

eigenface = cv2.face.EigenFaceRecognizer_create(40, 8000)
fisherface = cv2.face.FisherFaceRecognizer_create(3, 2000)
lbph = cv2.face.LBPHFaceRecognizer_create(2, 2, 7, 7, 50)

def getImagemComId():
    caminhos = [os.path.join('foto_treinamento', f) for f in os.listdir('foto_treinamento')]
    faces = []
    ids = []
    for caminhoImagem in caminhos:
        imagemFACE = cv2.cvtColor(cv2.imread(caminhoImagem),
cv2.COLOR_BGR2GRAY)
        id = int(os.path.split(caminhoImagem)[-1].split(".")[1])
        ids.append(id)
        faces.append(imagemFACE)

    return np.array(ids), faces


ids, faces = getImagemComId()


print("Treinando...")
eigenface.train(faces, ids)
eigenface.write('classificadorEigenface.yml')

fisherface.train(faces, ids)
fisherface.write('classificadorFisherface.yml')

lbph.train(faces, ids)
lbph.write('classificadorLBPH.yml')

print("Treinamento realizado")
```

5.1.1.4 APÊNDICE C – Algoritmo de reconhecimento eigenfaces

```
import cv2
import os

detectorFace = cv2.CascadeClassifier(cv2.data.haarcascades +
                    "haarcascade_frontalface_default.xml")
reconhecedor = cv2.face.EigenFaceRecognizer_create()
reconhecedor.read("classificadorEigenface.yml")
largura, altura = 220, 220
font = cv2.FONT_HERSHEY_SCRIPT_SIMPLEX

caminhos = [os.path.join('teste', f) for f in os.listdir('teste')]
faces = []
ids = []
i = 0
for caminhoImagem in caminhos:
    imagemCor = cv2.imread(caminhoImagem)
    imagemCinza = cv2.cvtColor(cv2.imread(caminhoImagem),
                               cv2.COLOR_BGR2GRAY)
    id = int(os.path.split(caminhoImagem)[-1].split(".")[1])
    ids.append(id)
    faces.append(imagemCinza)
    facesDetectadas = detectorFace.detectMultiScale(imagemCinza,
                        scaleFactor=1.2, minSize=(100,100))

    for (x, y, l, a) in facesDetectadas:
        imagemFace = cv2.resize(imagemCinza[y:y + a, x:x + l], (200, 200))
        imagemCor = cv2.rectangle(imagemCor, (x, y), (x + l, y + a), (0,255,0), 6)
        id, confianca = reconhecedor.predict(imagemFace)
        nome = ""
        if id == 1:
            nome = 'Misael'
        elif id == 2:
            nome = 'Jefte'
        imagemRed= cv2.resize(imagemCor, (552, 736))
        cv2.putText(imagemRed, nome + " - Acesso Autorizado",
            (50,50), font, 1, (0,255,0), 1, cv2.LINE_AA)

    cv2.imwrite("r-eigein-" + str(i) + ".jpg", imagemRed)
```

```
i = i+1

if cv2.waitKey(1) == ord('q'):
    break
```

5.1.1.5 APÊNDICE D – Algoritmo de reconhecimento fisherfaces

```
import cv2
import os
detectorFace = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
reconhecedor = cv2.face.FisherFaceRecognizer_create()
reconhecedor.read("classificadorFisherface.yml")
largura, altura = 220, 220
font = cv2.FONT_HERSHEY_SCRIPT_SIMPLEX

caminhos = [os.path.join('teste', f) for f in os.listdir('teste')]
faces = []
ids = []
i = 0
for caminhoImagem in caminhos:
    imagemCor = cv2.imread(caminhoImagem)
    imagemCinza = cv2.cvtColor(cv2.imread(caminhoImagem), cv2.COLOR_BGR2GRAY)
    id = int(os.path.split(caminhoImagem)[-1].split(".")[1])
    print(id)
    ids.append(id)
    faces.append(imagemCinza)
    facesDetectadas = detectorFace.detectMultiScale(imagemCinza, scaleFactor=1.2, minSize=(100,100))

    for (x, y, l, a) in facesDetectadas:
        imagemFace = cv2.resize(imagemCinza[y:y + a, x:x + l], (200, 200))
        imagemCor = cv2.rectangle(imagemCor, (x, y), (x + l, y + a), (0,255,0), 6)
        id, confianca = reconhecedor.predict(imagemFace)
        nome = ""
        if id == 1:
            nome = 'Misael'
        elif id == 2:
            nome = 'Jefte'
        imagemRed= cv2.resize(imagemCor, (552, 736))
        cv2.putText(imagemRed, nome + " - Acesso Autorizado", (50,50), font, 1, (0,255,0), 1, cv2.LINE_AA)
        #cv2.putText(imagemCor, str(confianca), (x,y + (a+70)), font, 4, (0,255,0))

    cv2.imwrite("r-fisher-" + str(i) + ".jpg", imagemRed)
```

```
i = i+1

if cv2.waitKey(1) == ord('q'):
    break
```

5.1.1.6 APÊNDICE E – Algoritmo de reconhecimento LBPH

```
import cv2
import os

detectorFace = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
reconhecedor = cv2.face.LBPHFaceRecognizer_create()
reconhecedor.read("classificadorLBPH.yml")
largura, altura = 220, 220
font = cv2.FONT_HERSHEY_SIMPLEX

caminhos = [os.path.join('teste', f) for f in os.listdir('teste')]
faces = []
ids = []
i = 0
for caminhoImagem in caminhos:
    imagemCor = cv2.imread(caminhoImagem)
    imagemCinza = cv2.cvtColor(cv2.imread(caminhoImagem), cv2.COLOR_BGR2GRAY)
    id = int(os.path.split(caminhoImagem)[-1].split(".")[1])
    print(id)
    ids.append(id)
    faces.append(imagemCinza)
    facesDetectadas = detectorFace.detectMultiScale(imagemCinza, scaleFactor=1.2, minSize=(100,100))

    for (x, y, l, a) in facesDetectadas:
        imagemFace = cv2.resize(imagemCinza[y:y + a, x:x + l], (200, 200))
        imagemCor = cv2.rectangle(imagemCor, (x, y), (x + l, y + a), (0,255,0), 6)
        id, confianca = reconhecedor.predict(imagemFace)
        nome = ""
        if id == 1:
            nome = 'Misael'
        elif id == 2:
            nome = 'Jefte'
        imagemRed= cv2.resize(imagemCor, (552, 736))
        cv2.putText(imagemRed, nome + " - Acesso Autorizado", (50,50), font, 1, (0,255,0), 1, cv2.LINE_AA)
        #cv2.putText(imagemCor, str(confianca), (x,y + (a+70)), font, 4, (0,255,0))
```

```
cv2.imwrite("r-lbph-" + str(i) + ".jpg", imagemRed)
i = i+1

if cv2.waitKey(1) == ord('q'):
    break
```

5.1.1.7 APÊNDICE F – Algoritmo de Avaliação

```
import cv2
import os
import numpy as np
from PIL import Image

detectorFace = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
reconhecedor = cv2.face.EigenFaceRecognizer_create()
reconhecedor.read("classificadorEigenAval.yml")
#reconhecedor = cv2.face.FisherFaceRecognizer_create()
#reconhecedor.read("classificadorFisherAval.yml")
reconhecedor = cv2.face.LBPHFaceRecognizer_create()
reconhecedor.read("classificadorLBPHAval.yml")

totalAcertos = 0
percentualAcerto = 0.0
totalConfianca = 0.0
faces = 0
caminhos = [os.path.join('foto_teste', f) for f in os.listdir('foto_teste')]
for caminhoImagem in caminhos:
    imagemFace = Image.open(caminhoImagem).convert('L')
    imagemFaceNP = np.array(imagemFace, 'uint8')
    facesDetectadas = detectorFace.detectMultiScale(imagemFaceNP, scaleFactor=1.1, minSize=(100,100))
    #facesDetectadas = detectorFace.detectMultiScale(imagemFaceNP)
    for (x, y, l, a) in facesDetectadas:
        faces = faces+1
        imagemFace = cv2.resize(imagemFaceNP[y:y + a, x:x + l], (200, 200))
        idprevisto, confianca = reconhecedor.predict(imagemFace)
        idatual = int(os.path.split(caminhoImagem)[-1].split(".")[1])
        print(str(idatual) + " foi classificado como " + str(idprevisto) + " - " + str(confianca))
        if idprevisto == idatual:
            totalAcertos += 1
            totalConfianca += confianca
        #cv2.rectangle(imagemFaceNP, (x, y), (x + l, y + a), (0, 0, 255), 2)
        #cv2.imshow("Face", imagemFaceNP)
        #cv2.waitKey(1000)
print(faces, "faces detectadas")
print(totalAcertos, "acertos")
```

```
percentualAcerto = (totalAcertos / faces) * 100
totalConfianca = totalConfianca / totalAcertos
print("Percentual de acerto: " + str(percentualAcerto))
print("Total confiança: " + str(totalConfianca))
```