UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

UMA ANÁLISE DAS BIBLIOTECAS *OPENFACE* E *FACE RECOGNITION* PARA RECONHECIMENTO DE PESSOAS COM OCLUSÃO PARCIAL

THAÏS DE SOUZA SIMÕES

Orientadora
GEIZA MARIA HAMAZAKI DA SILVA

RIO DE JANEIRO, RJ – BRASIL
MAIO DE 2020

Catalogação informatizada pelo autor

de Souza Simões, Thais
d593 UMA ANÁLISE DAS BIBLIOTECAS OPENFACE E FACE RECOGNITION PARA RECONHECIMENTO DE PESSOAS COM OCLUSÃO PARCIAL / Thais de Souza Simões. -- Rio de Janeiro, 2020.
83

Orientadora: GEIZA MARIA HAMAZAKI DA SILVA.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2020.

1. Reconhecimento facial. 2. Desaparecidos. 3. Oclusão parcial. I. HAMAZAKI DA SILVA, GEIZA MARIA, orient. II. Título.

UMA ANÁLISE DAS BIBLIOTECAS *OPENFACE* E *FACE RECOGNITION* PARA RECONHECIMENTO DE PESSOAS COM OCLUSÃO PARCIAL

THAÏS DE SOUZA SIMÕES

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

_____

GEIZA MARIA HAMAZAKI DA SILVA (UNIRIO)

_____

MORGANNA CARMEM DINIZ (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

MAIO DE 2020

Agradecimentos

Esse trabalho foi uma montanha-russa de emoções, toda vez que eu achava que estava finalmente mais calma e perto do final, tinha mais uma descida louca acontecendo. Nesse meio tempo, tiveram três pessoas que aturaram meus choros e meus gritos desesperados pedindo para sair: meus pais maravilhosos que não posso agradecer apenas por me apoiaram e levantaram a cada segundo desse processo, mas a todo o esforço e dedicação que eles tiveram comigo durante a vida e a Dra. Geiza Maria Hamazaki da Silva, que além de minha orientadora nessa jornada, foi uma amiga incrível nos bons e maus momentos.

Eles, obviamente, não foram os únicos que estiveram comigo nessa jornada (apesar de terem sido os que mais me ouviram chorar). São tantas pessoas, tantos nomes e rostos que eu espero conseguir me lembrar de todos.

À minha falecida vózinha, minha madrinha, minhas tias Andrea e Claudia, meu primo Luca e minha Lilian e tio David, muito obrigada por todo apoio que vocês me deram durante a vida. É difícil demais mensurar em palavras a importância que cada um de vocês têm pra mim.

Às pessoas do trabalho, principalmente Cesar Barbosa, Ramirez Alves, Ronald Campbell, Andre Mattos, Arthur dos Santos e todo mundo da SUSEG, minha genuína gratidão pela ajuda nos códigos, tabelas e suporte emocional.

Aos meus amigos, Matheus Costa, Gabriella Silva, Marina Coronel, Arthur Ferreira, Leonardo Meirelles, Giulia Alzuguir, Gustavo Escansetti, Carina Ezequiel, Gabriel Barroso, Rayanne Sanson, Julia Gomes, Eva Pencak, Cassio Coelho, aos Roots, o grupo de amigos mais heterogêneo que eu já conheci, e ao Wormhole, o melhor buraco de minhoca que eu poderia pedir, por terem me ouvido reclamar a cada segundo que se passava, por terem me ajudado a me distrair ou por terem me sacudido quando foi preciso. Separei um cantinho especial aqui para Thaíla Gomes, minha amiga incrível e maravilhosa que puxou minha orelha, me mandou respirar e disse: “vixe, calma que eu te ajudo”. Mulher, você não tem noção do que você fez por mim. E para o Guilherme Caeiros que é o ser humano mais paciente de toda a história! Ele ouviu meu lamúrios, me explicou como as coisas funcionavam e me ajudou ao longo de todo o percurso.

Aos meus grandes amigos Raphael Bueno, Rafael Longhi, Gabriel Raj Garcia, Alan Carpilovski e Laura Gaio. O Japão uniu o que o Brasil jamais teria conseguido.

Ao Stavale, Sabrina Lapa, Cassio Cidrini, Davi dos Anjos, Igor Castro, Yuri Pamplona, Ana Beatriz Valentina, Julia Viana, Lawrence Tavares e tantos outros por terem feito minha passagem pela universidade memorável e cheia de histórias para contar.

Não menos importante, à UNIRIO e todo o corpo docente maravilhoso que esteve comigo nessa formação. Aos professores que já foram e aos que ainda estão aqui, eu não tenho palavras para agradecer o tempo e o carinho de vocês dedicaram todas as vezes que eu tive problemas e dificuldade, relacionadas ou não a faculdade. Obrigada por descontruírem a minha visão de distanciamento, por mostrarem humanidade e criarem um ambiente tão aconchegante quanto o que eu vivi durante a minha vida universitária.

Por fim, obrigada a mim por não me ouvir quando quis desistir e ter tomado a decisão de cursar esse curso. A BSI é uma experiência a ser vivida e me orgulho por ter feito parte dessa história.

RESUMO

O crescimento acelerado da tecnologia nos últimos anos fez com que algumas tendências surgissem no mercado da computação, como a ciência de dados e a inteligência artificial. No mundo globalizado, a automação e a capacidade de fazer com que o computador consiga realizar tarefas tão bem quanto humanos se tornou quase vital para a indústria e impulsionou o crescimento de áreas como o reconhecimento facial.

Com aplicações atuantes na área de comércio, segurança, marketing, entre outras diversas, o reconhecimento facial tem ganhado atenção nos diversos tipos de papéis que ele pode exercer. Tendo isto em mente, a utilização desta tecnologia na segurança pública também tem tido bastante apelo nos últimos tempos.

Dado o número de desaparecimento de pessoas no Rio de Janeiro, a polícia civil precisa aumentar o número de frentes capazes de realizar parte do processo de reconhecimento. Tendo em vista a disponibilização de um aplicativo capaz de oferecer acesso às informações do banco de dados de desaparecidos, ponderou-se a utilização de reconhecimento facial como um método de agilização do processo. Contudo, em meio às considerações, questiona-se se a tecnologia atual está apta para reconhecer pessoas que tenham algum tipo de cobertura parcial do rosto.

Neste contexto, este trabalho se propõe a analisar tempo, precisão e acuidade de duas bibliotecas de reconhecimento facial em conjunto a fotografias voltadas para estudos de oclusão parcial do rosto.

Palavras-chave: desaparecidos, reconhecimento facial, aplicativo, oclusão parcial

ABSTRACT

The rapid growth of technology in the last years created some trends in the information technology market, such as Data Science and Artificial Intelligence. In the globalized world, the automation and the ability to make a computer work as a human were almost vital to the industry and promoted the growth of areas such as facial recognition.

Having applications working in commerce, security, marketing, among others, the facial recognition market has been getting attention in the variety of roles it can play. The use of such technology in the public security field has gained a lot of appeal in the last couple of years.

With many people getting missing every day, the police need to increase the number of those capable of doing part of the recognition process. Having in mind the availability of an app able to provide access to the police's missing people database, the use of face recognition to speed up the process has started to be taken into consideration. However, amidst the considerations, the question arose whether it would be able to recognize people who have some type of partial coverage of the face.

In this context, this work aims to analyze the time, precision and accuracy of two facial recognition libraries using photographs aimed at studies of partial occlusion of the face.

Keywords: missing people, face recognition, app, partial occlusion

Lista de Abreviaturas e Siglas

CNN	Convolutional Neural Network
DDPA	Delegacia de Descoberta de Paradeiros
GPU	Graphics Processing Unit
Qtde	Quantidade
ReLU	Unidade Linear Retificada
TanH	Tangente Hiperbólica
UNIRIO	Universidade Federal do Estado do Rio de Janeiro
API	Application Programming Interface
CV Dazzle	Computer Visual Dazzle Camouflage

Índice

Índice de Quadros

Índice de Figuras

Índice de Equações

1 Introdução

Apesar de se tratar de um trabalho relativamente simples para um ser humano, o ato de reconhecer um rosto através de uma máquina é um desafio que vem sendo estudado há décadas e que está se aproveitando dos avanços tecnológicos para ser aperfeiçoado.

De acordo com NEGRÃO, SANTOS e SOARES (2018), na década de 1960, foi criado um sistema para reconhecimento de face automatizado, pois já nesta época surgia a necessidade de encontrar mecanismos de identificação com rapidez e precisão.

Naquela época, segundo a Sinfic[1], o reconhecimento de uma face exigia que o seu administrador localizasse características nas fotografias (por exemplo, olhos, orelhas, nariz e boca) antes do sistema calcular distâncias e rácios para um ponto de referência comum.

Hoje, 60 anos depois, as técnicas de reconhecimento facial estão mais robustas e avançadas como quando comparamos com as suas origens anos atrás. A tecnologia já é capaz de fazer o reconhecimento completamente automatizado e de maneira bastante ágil, podendo ser bastante precisa com condições favoráveis.

Contudo, ainda existem barreiras que dificultam seu aperfeiçoamento. Por exemplo, dependendo do tipo de sistema utilizado, a taxa de erro aumenta significativamente quando a iluminação ambiente não é muito favorável, ou quando a face da pessoa a ser reconhecida é captada de perfil e não frontalmente, ou ainda se as expressões faciais não forem neutras (BRAGA, 2013). Isto é conhecido como oclusão parcial do rosto.

Por causa da praticidade conferida pela automatização através do uso de máquinas, o reconhecimento facial está atraindo atenção de vários mercados, sendo aplicado para funções de marketing[2], automatização e segurança, por exemplo.

A motivação deste trabalho está ligada diretamente a área de segurança pública e a utilização do reconhecimento facial para atuar como tecnologia auxiliar na busca de pessoas desaparecidas.

[1] http://www.sinfic.pt/SinficWeb/displayconteudo.do2?numero=24923 (Acesso: Jun/2020)
[2] http://bibliotecadigital.fgv.br/ocs/index.php/clav/clav2017/paper/view/6152/1806 (Acesso: Jun/2020)

O aplicativo Desaparecidos-RJ (LIMA, 2016) tem como proposta fornecer um sistema de cadastramento de pessoas desaparecidas para a Delegacia de Descoberta de Paradeiros (DDPA), no Rio de Janeiro. Esse sistema seria capaz de ter sua base de dados acessada por um aplicativo móvel que seria distribuído para autoridades e pessoas atuantes na área médica e social, como bombeiros, enfermeiros e profissionais do serviço social, com a intenção de diminuir uma etapa no processo de reconhecimento que hoje é feita apenas por policiais autorizados.

Diante deste quadro, o reconhecimento facial tornou-se atrativo por ser um facilitador na busca por resultados. Porém, considerando os possíveis problemas que poderiam causar erros no reconhecimento de uma pessoa, este trabalho tem por objetivo analisar empiricamente duas bibliotecas de reconhecimento facial, *Face Recognition* e *OpenFace*. Ambas as bibliotecas podem ser usadas em caso sem oclusão, contudo, a fim de verificar a precisão de seus resultados em casos com oclusão parcial, este trabalho está primariamente focado em casos de rostos parcialmente cobertos.

A avaliação foi realizada através do uso de uma base de dados especializada para estudos em oclusão parcial e os resultados de comparação tiveram como parâmetro a eficiência e a acuidade de cada biblioteca.

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo II: Referencial Bibliográfico – Apresenta aspectos técnicos sobre as abordagens conhecidas para o desenvolvimento de um algoritmo de reconhecimento facial.

- Capítulo III: Aplicações de reconhecimento facial – Aborda a importância do reconhecimento facial nos dias de hoje, com explicações de diferentes aplicações possíveis para esta tecnologia.

- Capítulo IV: Ferramentas e tecnologias – Descreve brevemente as tecnologias usadas neste trabalho e os principais componentes utilizados pelos algoritmos testados.

- Capítulo V: Testes - Explica como os testes foram realizados, tecendo comentários sobre o desenvolvimento de cada parte dos códigos. Também é apresentado uma breve citação das limitações encontradas no trabalho.

- Capítulo VI: Resultados – Traz uma análise completa dos resultados escolhidos para serem avaliados.
- Capítulo VII: Conclusões – Reúne as considerações finais referentes as análises.

2 Referencial Bibliográfico

Para o reconhecimento computacional de um rosto, a imagem deve ser processada até chegar a um resultado. Este capítulo apresenta os conceitos necessários para o entendimento do processo de reconhecimento facial, trazendo aspectos técnicos, uma revisão da visão computacional e alguns métodos utilizados hoje em dia.

2.1. Visão computacional

A visão computacional é uma área que busca simular o poder da visão humana em um computador. Segundo Marengoni et al (2009), a visão computacional é quando o computador consegue interpretar completamente, ou parcialmente, uma imagem de entrada dada pelo usuário.

O processamento de imagens consiste em transformar a imagem em um conjunto de números de forma a permitir que uma máquina consiga interpretar uma foto chegando próxima a uma conclusão dada por um ser humano. Em outras palavras, citando ALBUQUERQUE e ALBUQUERQUE (2002), processar uma imagem consiste em um processo de transformações sucessivas com o objetivo de extrair de forma eficiente a informação nela presente.

O processamento de imagens pode ser divido em três níveis: baixo, intermediário e alto.

No nível baixo, encontram-se tarefas voltadas a compactação, isto é, remoção de redundância de informações para reduzir a quantidade de dados que serão processados e algumas pequenas correções, tais como: escala de cores, retirada de bordas e filtros, por exemplo. No fim, este nível engloba grande parte das tarefas realizadas, relativo ao pré-processamento de imagens.

O pré-processamento de imagens vai desde correção de cores e seu redimensionamento, à segmentação de suas partes importantes e remoção de ruídos, que são condições que atrapalham a interpretação da imagem (marcas d'água, filtros, iluminação, etc.).

O nível intermediário é responsável pelo reconhecimento de pequenos padrões. Essas características encontradas ajudam o mapeamento de formas geométricas que serão

usadas como facilitadores para as tarefas do nível alto. Esses padrões incluem a detecção de características físicas e do próprio rosto.

No nível alto acontece a interpretação da imagem e a classificação de um objeto, ou seja, onde é realizado o reconhecimento. É o nível mais próximo ao sistema de visão humano.

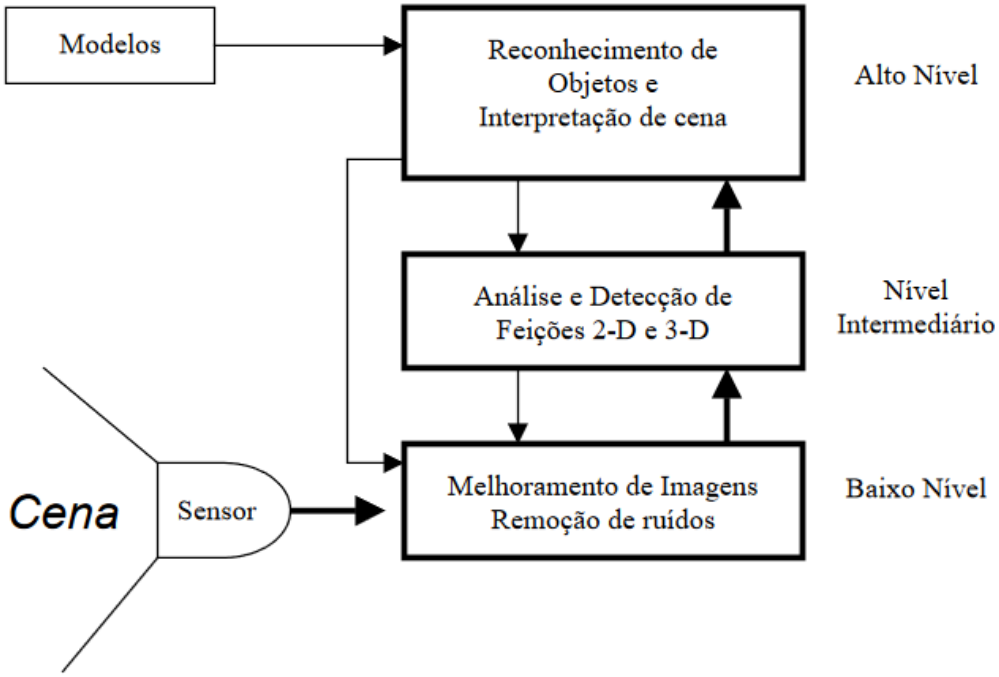


Figura 1- Níveis de reconhecimento facial
Fonte: https://www.lume.ufrgs.br/bitstream/handle/10183/2783/000326081.pdf?sequence=1&isAllowed=y (Jun/2020)

Como observado na figura 1, o reconhecimento facial passa por todos os níveis, tendo, cada um, uma função importante para o resultado. No nível mais baixo, é realizado o pré-processamento para reduzir a quantidade de redundâncias. Nela ocorre o corte inicial do rosto e o descarte das outras informações da imagem. Em seguida, no nível intermediário, serão encontradas as características faciais e a detecção de um rosto, fazendo mais um recorte na imagem, selecionando apenas o que será analisado pelo algoritmo. Por fim, serão feitos o reconhecimento e a classificação do rosto, no nível alto.

2.2. Reconhecimento facial

2.2.1. O que é?

O reconhecimento facial é uma das técnicas biométricas[3] que estão presentes em vários aspectos da vida moderna. Pode ser utilizada para algo simples como o desbloqueio da tela de um celular, assim como em questões mais críticas, como na República Popular da China, onde o reconhecimento facial em massa está sendo utilizado para calcular pontos de cidadania que garantem direitos e privilégios aos chineses.

> O governo utiliza a inteligência artificial para prestar serviços públicos e monitorar a cidade, e as pessoas que nela vivem. Já é possível que o cidadão chinês pague suas contas, saque dinheiro e acesse demais serviços públicos apenas com seu rosto (LOPES, 2018. p. 92)

Porém, em 2020, nos Estados Unidos, está sendo discutida as questões éticas do reconhecimento para vigilância em massa como vem sendo utilizada na China. Empresas multibilionárias, tais como IBM, Amazon e Microsoft, estão se recusando a vender soluções e programas de reconhecimento facial enquanto não existir uma lei baseada em direitos humanos que proteja as minorias e a população em geral.[4]

O movimento contra o reconhecimento facial não vem só de empresas, pessoas estão desenvolvendo maquiagens, usando uma técnica conhecida como *CV Dazzle* (*Computer Visual Dazzle Camouflage*)[5] para driblar a detecção de rostos. Essas maquiagens pintam pontos de chaves do rosto, fazendo o papel de uma oclusão parcial, que será abordado logo abaixo. A figura 2 exemplifica alguns tipos comuns de maquiagem utilizando a técnica *CV Dazzle.*

[3] Biometria é o estudo que se baseia em aspectos biológicos e únicos do ser humano para identificar um indivíduo. Graças ao seu poder de distinção de pessoas, atualmente ela vem ganhando espaço em diversas áreas, principalmente àquelas voltadas para segurança.
[4]https://www.uol.com.br/tilt/noticias/afp/2020/06/12/microsoft-se-une-aos-rivais-e-veta-uso-de-reconhecimento-facial-a-policia.amp.htm (Acesso: Jun/2020)
[5] https://cvdazzle.com/ (Acesso: Jun/2020)

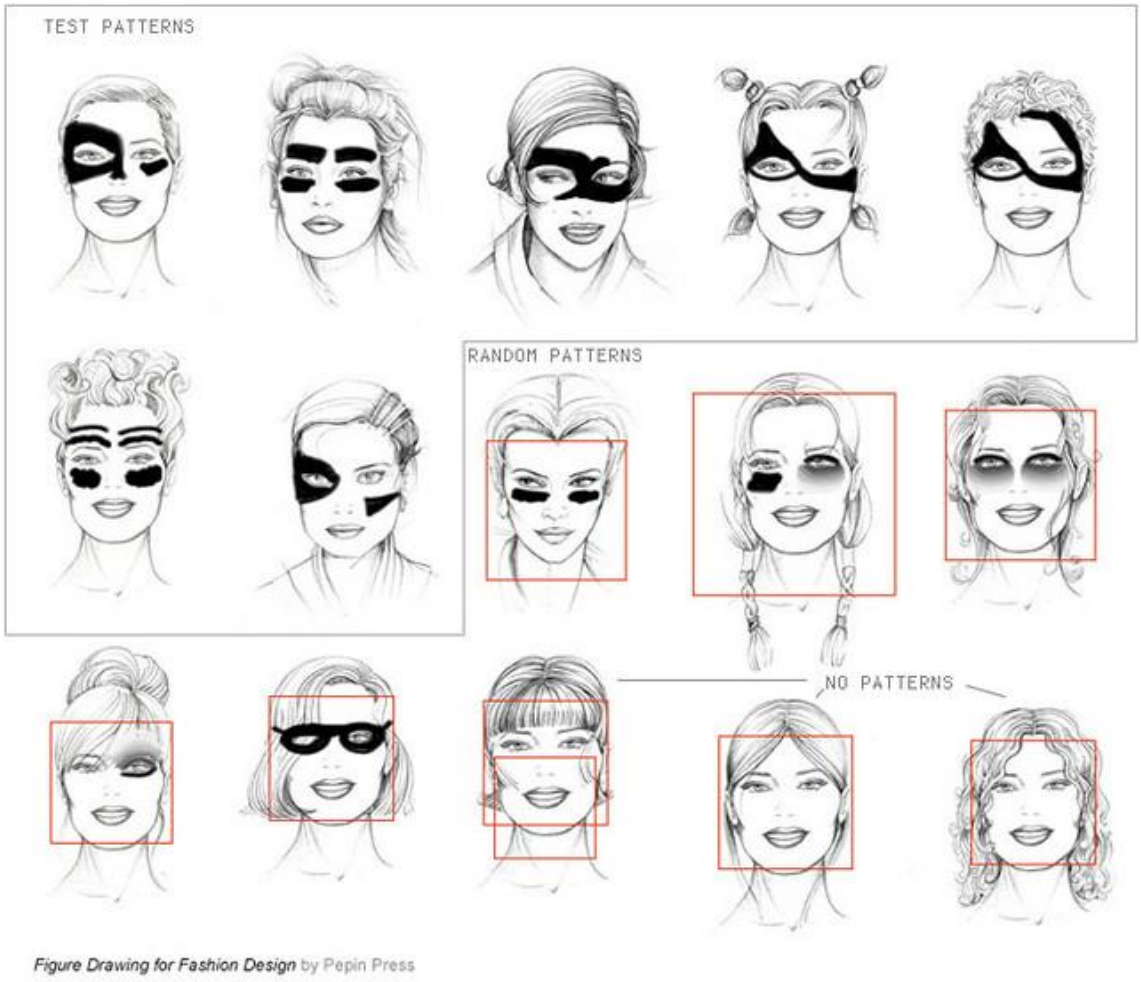


Figura 2 - Exemplo de maquiagem CV Dazzle
Fonte: *https://live.staticflickr.com/2726/4479432753_00ab74932e_z.jpg* (Acesso: Jun/ 2020)

Tratando-se de técnicas, existem dois modos de abordar o tema reconhecimento facial: o reconhecimento facial com profundidade ou sem profundidade.

O reconhecimento facial sem profundidade trata de fotografias comuns onde o dado de entrada é bidimensional. Em modelos em duas dimensões, muitas análises são realizadas primariamente contando com o posicionamento de certos elementos, como: olhos, nariz, boca, sobrancelhas etc. Muitas técnicas voltadas para imagens bidimensionais, ao tratar de imagens frontais, procuram por esses elementos em posições previamente mapeadas. Deste modo, ele é capaz de localizar e afirmar na imagem a existência de um rosto que deve ser reconhecido. Porém, ao rotacionar a cabeça em uma imagem bidimensional, como em uma foto em que apareça uma pessoa de perfil, o posicionamento destes elementos não será o mesmo uma vez que você terá a visão de apenas uma parte da face.

O reconhecimento facial com profundidade trata apenas rostos com três dimensões, ou seja, o dado de entrada é uma imagem tridimensional. Estes modelos 3D podem ser obtidos de modos diferentes: através de um *software* com capacidade de transformar imagens bidimensionais em tridimensionais, ou através de escaneamento do rosto que está sendo modelado. Apesar de sofrer com ruídos (movimentação, problemas no escaneamento etc.), utilizar esse tipo de dados de entrada possui vantagens em relação ao utilizado na análise bidimensional, podendo tratar com uma melhor precisão problemas posicionamento do rosto, reconhecimento de características etc.

Diferentemente dos modelos em três dimensões por escaneamento, onde usualmente não é reproduzido o ambiente em que a pessoa está inserida, as imagens 2D, possuem a preocupação de detectar um rosto em uma imagem, uma vez que a fotografia pode ter sido tirada em uma rua, em um quarto ou em um concerto musical, por exemplo. Para remover os ruídos de uma fotografia, o computador ser capaz de distinguir uma face na imagem que está sendo analisada, isso reduzirá o processamento a um segmento específico da foto.

A detecção de faces pode ser implementada de diversos modos. Destacando-se:

a) Método baseado em conhecimento: utiliza-se de regras pré-estabelecidas para identificar elementos que são comuns a todas as faces. Por exemplo, sabe-se que todas as pessoas possuem um nariz, localizado no meio do rosto. O mesmo não pode ser dito do desenho das sobrancelhas, uma vez que são elemento que podem estar ou não presentes.
b) Método baseado em *template*: este método avalia a existência de um rosto comparando-o com um modelo pré-existente. Este modelo pode ser uma forma geométrica, por exemplo.
c) Métodos baseados na aparência: diferente dos outros métodos, este não possui regras ou conhecimentos pré-estabelecidos. Segundo (BRAGA, 2013), têm como base aprender características sobre a face humana utilizando algoritmos de aprendizagem de máquina com uma vasta quantidade de imagens tanto de faces quanto de outros objetos.

A detecção de um rosto pode apresentar erros ou problemas, dado a existência de ruídos. Nesta fase, ruídos são conhecidos como oclusão e representam um dos maiores desafios para os algoritmos de reconhecimento facial. A oclusão pode ser classificada como: inexistente, parcial ou completa.

a) A oclusão inexistente é um rosto frontal, bem iluminado e sem partes cobertas, como mostrado na figura 03.a. Quando não existe oclusão, o rosto está completamente exposto e pode ser facilmente detectado pelo algoritmo.
b) A oclusão parcial pode ser de diversos tipos, mas não se limitando a: um acessório que cubra parte do rosto, como ilustrado na figura 03.b, uma foto de perfil com o rosto parcialmente virado ou uma iluminação ruim. A oclusão parcial pode gerar resultados positivos ou negativos, dependendo do algoritmo de detecção implementado.
c) A oclusão completa, não há reconhecimento de rosto, pois ele se encontra totalmente encoberto, como na figura 03.c.

Figura 3 - a. Oclusão inexistente (foto frontal), b. Oclusão parcial (foto perfil com acessórios) e c. Oclusão completa (foto de rosto completamente coberto pela máscara)
Fonte: (a) https://pxhere.com/pt/photo/1413323 (b) https://www.pexels.com/pt-br/foto/agua-areia-atraente-beira-mar-300968/ (c) https://www.soldaeletrica.com.br/imagens/mpi/mascara-de-solda-automatica-01.jpg (Acesso: Jul/2019)

Atualmente, existem pesquisas dedicadas ao estudo e aprimoramento de técnicas para superar a barreira que a oclusão parcial representa. Há algoritmos privados que são grades casos de sucesso, como o *DeepFace* do *Facebook*, que consegue reconhecer com uma alta quantidade de oclusões diferentes, possuindo uma precisão maior que 97%[6].

Esse trabalho trata da análise de fotografias (2D), uma vez que objetiva ser incorporado a um aplicativo de celular.

[6] https://www.forbes.com/sites/amitchowdhry/2014/03/18/facebooks-deepface-software-can-match-faces-with-97-25-accuracy/#5ad68a1954fc (Acesso: Out/2019)

2.2.2. Processo para o reconhecimento facial

Para o entendimento das abordagens de classificação dos algoritmos, é importante uma noção do funcionamento de um sistema de reconhecimento facial clássico. Abaixo é explicitado os elementos em comum em técnicas utilizando imagens bidimensionais.

O desenvolvimento de uma biblioteca de reconhecimento facial é organizado em alguns passos, sendo eles: entrada de dados, pré-processamento, extração de características e classificação, como esquematizado na figura 04.

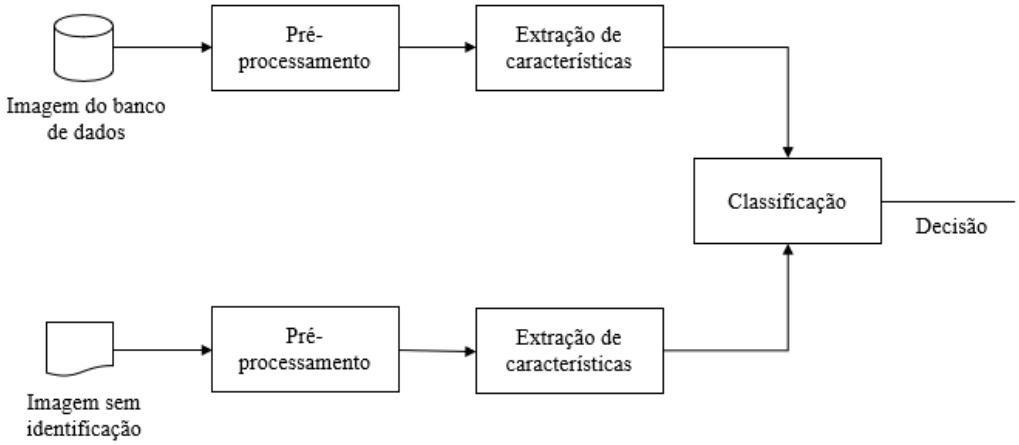


Figura 04 – Etapas de um processo de reconhecimento facial
Fonte: Elaboração própria

Como visto na figura 04, tanto a imagem conhecida quanto a foto da pessoa não identificada vão passar por todas as etapas até chegar à classificação. Elas precisam estar em condições semelhantes para que o algoritmo seja capaz de compará-las no final.

Após a entrada dos dados, o primeiro passo é conhecido como pré-processamento. Esta etapa é feita com a intenção de ajustar cada imagem para que o processamento seja feito de maneira menos onerosa, podendo envolver a correção de cores, retirada de marcas d´água etc.

A figura 05 mostra a diferença de uma imagem após passar pelo processo de pré-processamento, sendo: (A) a imagem original, (B) a mesma imagem após a aplicação de um filtro de redução de ruídos e (C) a utilização de um filtro de realce de contornos.

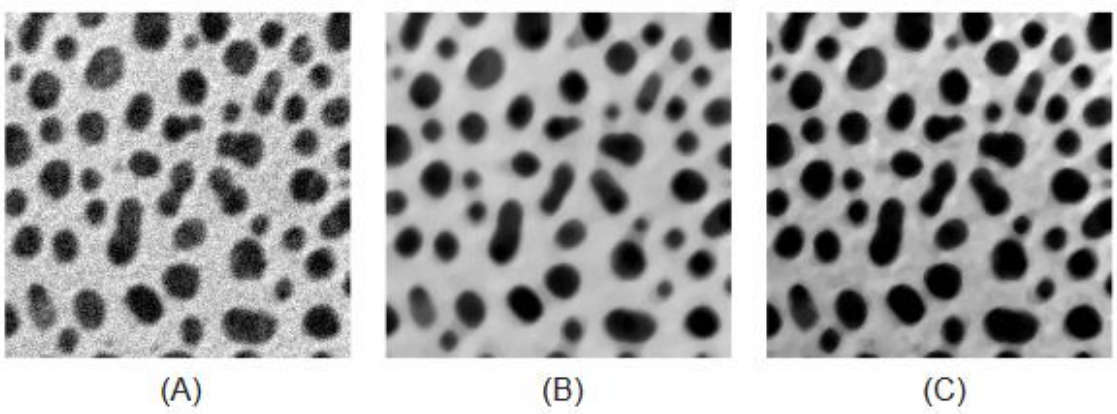


Figura 05 – Exemplo de pré-processamento de imagem (a) imagem original (b) imagem com filtro de detecção de ruídos e (c) imagem com realce de contornos
Fonte: http://www.cbpf.br/cat/pdsi/pdf/cap3webfinal.pdf (Acesso: Jun/2020)

Ainda referente ao pré-processamento, é bastante comum que as dimensões das imagens e quantidade da escala cores sejam reduzidas, de modo que diminua a carga de processamento e facilite o reconhecimento de estruturas relevantes na imagem, como visto na figura 06.

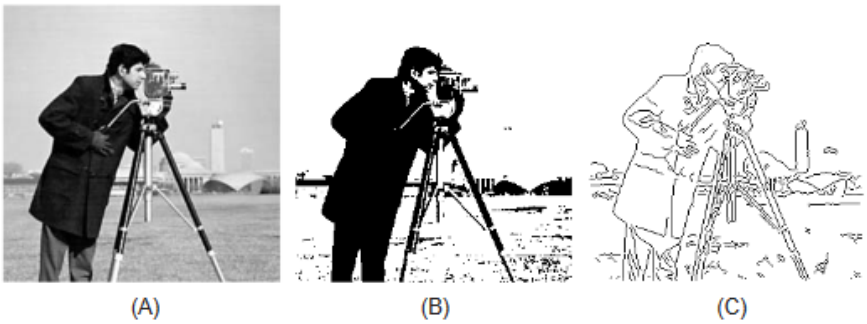


Figura 6 - Exemplo de redução de escala de cores. (a) imagem original (b) imagem em preto e branco, sem tons de cinza (c) imagem com detecção de borda
Fonte: http://www.cbpf.br/cat/pdsi/pdf/cap3webfinal.pdf (Acesso: Jun/2020)

Destaca-se que, em determinados casos, é possível remover alguns tipos de oclusão, tal como problemas de iluminação, que seria prejudicial à fase de extração de características que efetuada em seguida.

A próxima etapa é a extração de características, apresentada na figura 04. Nesta fase é feita a detecção de um rosto e/ou de características físicas de uma pessoa na foto pré-processada. Nela, o algoritmo deve ser capaz de localizar se há alguém ou não presente na imagem que sendo analisada.

A oclusão parcial é considerada um desafio para os estudos do reconhecimento facial, pois o algoritmo não consegue encontrar determinadas características que são consideradas essenciais, fazendo com que muitas vezes um rosto não seja detectado.

Seguindo o fluxograma apresentado na figura 04, a última etapa, a classificação, é onde as informações extraídas das duas imagens vão ser processadas juntas. Para tanto, é realizado comparações à base de cálculos matemáticos dos dados extraídos de ambas as fotos na fase de extração de características. Essas comparações terão um resultado único que será utilizado para a tomada de decisão.

A decisão é onde de fato haverá a identificação da pessoa da imagem de entrada, retornando se as pessoas comparadas são, de fato, a mesma.

Na próxima seção será abordada a ideia por detrás dos métodos de reconhecimento facial utilizados atualmente.

2.2.3. Métodos

O reconhecimento facial pode ser feito de diversas formas, com ideias e enfoques diferentes. Na figura 07, são ilustradas 3 abordagens que, neste momento, abrangem a área reconhecimento facial: abordagem holística, abordagem baseada em características e abordagem híbrida.

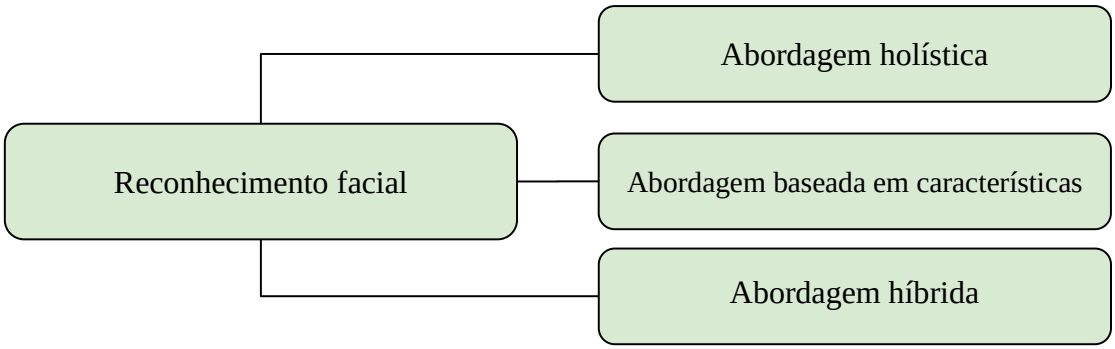


Figura 07 – Algumas abordagens de reconhecimento facial para imagens bidimensionais
Fonte: https://www.researchgate.net/publication/281838833_FACE_RECOGNITION_TECHNIQUES_AND_APPROACHES_A_SURVEY (Acesso: Abr/2019)

Abordagem holística

Na abordagem holística, método utilizado neste trabalho, o rosto é tratado como um objeto de análise único, não se preocupando separadamente cada característica individual da face, ou seja, não tratando isoladamente os olhos, a boca, o nariz etc.

O processamento da imagem baseia-se no reconhecimento de pequenos aspectos do rosto inteiro, tomando cuidado ao analisar cada pixel da imagem e calcular a variância entre eles para fazer a classificação.

Segundo (NAEEM; AZAM, 2015), os métodos que utilizam essa abordagem são as análises estatísticas e a inteligência artificial.

Abordagem baseada em características (ou métodos geométricos)

A abordagem baseada em características consiste na ideia de mapear as características individuais do rosto como, olhos, boca e nariz, para fazer o reconhecimento. Em seguida, o processamento da imagem é realizado através da relação geométrica entre os pontos criados.

A estrutura de dados denominada grafo é utilizada para fazer esse mapeamento e representar a relação entre as características, como na figura 08.

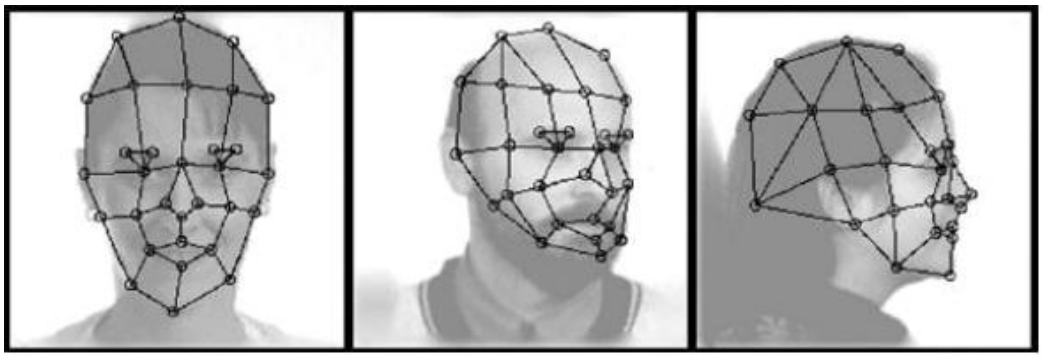


Figura 08 – Grafos mapeando o método por características
Fonte: https://www.researchgate.net/publication/274521637_A_Review_Of_Face_Recognition_Methods (Acesso: Jul/2019)

Abordagem híbrida

Como o próprio nome já diz, a abordagem híbrida é uma mistura de técnica utilizadas na abordagem holística e na abordagem baseada em características. É o que mais se aproxima do conceito humano, uma vez que as pessoas são capazes de identificar umas às outras por características individuais ou pelo rosto inteiro. A ideia é conseguir aproveitar o melhor das duas técnicas e criar algoritmos mais eficientes e robustos.

Após o entendimento do funcionamento do reconhecimento facial, percebe-se que sua utilização pode ir muito além da área de segurança, expandido para outros mercados, como marketing e automatização de tarefas, como será visto na seção 3.2. O próximo capítulo discute a maneira que aplicações de reconhecimento facial podem ser encontradas no mercado e como as organizações estão se utilizando dela.

3 Aplicações de Reconhecimento Facial

Este capítulo tem por objetivo introduzir algumas aplicações de reconhecimento facial, destacando principalmente, porém não exclusivamente, a área de segurança pública. A abordagem é separada em duas seções: reconhecimento facial em aplicações, a qual aborda as formas em que a tecnologia pode ser encontrada, e reconhecimento facial no mercado, seção que traz uma revisão de que tipo de aplicações se aproveitam da tecnologia nos dias de hoje.

3.1. Reconhecimento facial em aplicações

Reconhecimento facial por vídeo

Diferente de uma fotografia que pode ser tirada em condições ideais: uma foto frontal, com o rosto livre de acessórios e com o fundo menos ruidoso possível, o reconhecimento facial em vídeo, não costuma dispor de uma imagem nessas condições.

> Apesar de os atuais sistemas de reconhecimento terem alcançado certo nível de maturidade, seu sucesso é limitado pelas condições impostas por muitas aplicações reais, como por exemplo, a dificuldade de reconhecimento de imagens de faces adquiridas em um ambiente externo com mudanças de iluminação e ou de pose. Isso se deve ao fato de a grande maioria dos identificadores faciais serem construídos para reconhecimento de faces frontais e com iluminação uniforme, uma vez que a maioria das bases de dados disponíveis conta somente com imagens nessa pose fotografadas num ambiente controlado (com iluminação corretiva). (BONFÁ, 2013. p. 15)

Para ter uma visão melhor dos arredores, as câmeras costumam estar localizadas em uma altura elevada, possuindo uma visão ligeiramente inclinada das pessoas. Além disso, questões como iluminação, qualidade de vídeo, posicionamento de face e vestimentas/acessórios podem impedir o algoritmo de reconhecer um rosto. Em casos de ruas muito movimentadas, o sistema deve ser capaz de reconhecer dezenas, ou centenas, de transeuntes ao mesmo tempo.

O processo de reconhecimento em vídeo, porém, não é muito diferente do reconhecimento de uma fotografia, uma vez que o algoritmo verifica cada quadro (*frame)* como se fosse uma imagem estática. Após o reconhecimento de uma pessoa, para que a marcação do identificador seja realizada de maneira constante em um vídeo, como na Figura 09, o algoritmo aplicará o classificador em todos os *frames* seguintes até o rosto sair do alcance da câmera. Para tratar da análise de vários rostos ao mesmo tempo, é preciso que o algoritmo seja multi-instanciado de modo que consiga rodar paralelamente para cara rosto na imagem.

Figura 09 – Pessoas na China tendo seus rostos reconhecidos por algum algoritmo de reconhecimento facial em tempo real.
Fonte: https://www.socialismocriativo.com.br/reconhecimento-facial-e-tecnologia-para-vigilancia-nacional/ (Acesso: Set/2019)

Apesar do grande desafio que é reconhecer um rosto em tempo real, inúmeras bibliotecas gratuitas, disponibilizados na internet, já são capazes de fazer reconhecimento em vídeo como, por exemplo, o *Face Recognition*, de Adam Geitgey, e o *OpenFace*, ambos estudados neste trabalho.

Reconhecimento facial em imagens estáticas

Não há diferença relevante entre o algoritmo que deve ser aplicado, tanto para vídeos, quanto para imagens, dado que o primeiro sempre processará uma imagem por vez. A diferença está no propósito de utilização.

Para o caso de busca de pessoas desaparecidas, por exemplo, Moisés et al (2015) utiliza fotografias ao invés de vídeos. A utilização de fotos implica que a aplicação deva ser alimentada com uma imagem por vez, seja de forma manual ou automatizada, para que o banco seja percorrido fazendo as devidas comparações.

Assim como em vídeo, é possível reconhecer vários rostos ao mesmo tempo se o algoritmo estiver sendo executado em instâncias paralelas. Logo, se a foto comparada, por exemplo, é uma foto de família, é possível reconhecer todos os rostos de uma vez se o programa for capaz de ser instanciado para cada rosto detectado.

Na seção 3.2. será abordado o uso da tecnologia de reconhecimento facial em aplicações no mercado e na segurança pública, trazendo exemplos de como a técnica vem sendo utilizada pelas empresas e organizações.

3.2. Reconhecimento facial no mercado

Reconhecimento facial para segurança pública

Atualmente existe uma grande quantidade de câmeras espalhadas por centros com grande circulação. Essa constante vigilância, aliada ao poder do reconhecimento facial pode se tornar um grande facilitador na hora da busca por pessoas.

Segundo a Secretaria Estadual de Desenvolvimento Social e Direitos Humanos do Rio de Janeiro (agosto/2019)[7], o estado contabiliza em torno de 15 pessoas desaparecidas por dia, o que soma mais de 400 pessoas em um mês. No país inteiro, os valores chegam a 8 pessoas desaparecidas por hora.[8]

Com números tão expressivos, iniciativas de aliar tecnologia à polícia surgiram por todo o país em forma dos mais diversos tipos de aplicação, por exemplo, a criação de um protótipo de um aplicativo para auxiliar o gerenciamento de relatórios da polícia

[7] http://agenciabrasil.ebc.com.br/direitos-humanos/noticia/2019-08/estado-do-rio-tem-15-desaparecidos-por-dia-informa-secretaria (Acesso: Set/2019)

[8] https://www.redebrasilatual.com.br/cidadania/2019/04/a-cada-hora-brasil-registra-8-pessoas-desaparecidas/ (Acesso: Set/2019)

militar do Paraná[9] e aplicações como as delegacias virtuais de Minas Gerais[10] e o do Rio de Janeiro[11].

Além de pesquisas que envolvem reconhecimento facial, tais como: o aplicativo proposto por Vinícius Rodrigues (2017), no Rio de Janeiro, e a aplicação proposta por Claudio Augusto e Moisés Henrique (2015), em Minas Gerais, para a consultar os bancos de dados da polícia e encontrar pessoas desaparecidas.

A utilização do reconhecimento facial vai além de buscas por pessoas desaparecidas, também sendo utilizada para a captura de foragidos. Segundo VETTORAZZO e PITOMBO (2019) citado por CONCEIÇÃO, NUNES e ROCHA (2019), o reconhecimento facial foi utilizado no carnaval de Salvador em 2019, retirando de circulação pessoas que estavam em débito com a justiça. Essa ferramenta também foi utilizada no carnaval do Rio de Janeiro e na Copa América

Na China, um forte sistema de reconhecimento facial vem sendo implementado pelo governo há anos, e atualmente consegue encontrar crianças desaparecidas em todo o país, através das câmeras de segurança. A base de dados conta com mais de 20 (vinte) redes sociais disponíveis e já foi capaz de encontrar mais de 6 (seis) mil crianças ao longo de três anos.[12]

Reconhecimento facial para outros segmentos

O reconhecimento facial pode ser utilizado para questões que vão além da temática de segurança, muitas vezes buscando melhorar ou automatizar alguma tarefa já existente.

O *PresentEye* (MATTOS, 2017), por exemplo, é uma aplicação que se utiliza de câmeras em salas de aula para verificação automática de presença de alunos. Através do uso de reconhecimento facial, a aplicação consegue identificar todos os alunos de uma vez, acelerando o processo de checagem, que é realizado manualmente, para cada estudante presente.

[9] http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/2488/1/CT_TECJAVMOV_I_2012_04.pdf
[10] https://repositorio.ufmg.br/bitstream/1843/BUBD-A8SP2P/1/tcc_lucas_gon_alves_santa_rita.pdf
[11] https://play.google.com/store/apps/details?id=com.app.DelegaciaPCERJ&hl=pt_BR
[12] https://epocanegocios.globo.com/Tecnologia/noticia/2019/07/reconhecimento-facial-e-usado-para-encontrar-criancas-desaparecidas-na-china.html (Acesso: Out/2019)

Muitas companhias se utilizam de ferramentas de reconhecimento facial para validação de transações financeiras, como a *MasterCard* com o seu sistema *Identity Check Mobile*[13]. Nele, o usuário dos cartões de crédito da bandeira consegue finalizar transações com a digital ou com uma foto do rosto tirada na hora, ao invés da utilização tradicional de senhas.

Profissionais da área de marketing também estão usufruindo da facilidade proporcionada pela técnica para direcionarem melhor suas campanhas. Em São Paulo, uma das linhas metrô passou a identificar as reações das pessoas à comerciais através de portas interativas instaladas nos trens[14].

Aliado a outros tipos de tecnologias, como por exemplo, a realidade aumentada, o reconhecimento facial começa a revolucionar a maneira de se fazer atividades comerciais. Existem estudos que procuram unir os óculos inteligentes à tecnologia. A ideia é utilizar o reconhecimento facial para identificar o cliente e automaticamente trazer à tela dos óculos o histórico de compras do consumidor, suas reclamações etc., a fim de personalizar ao máximo a experiência do cliente com a empresa[15].

O propósito do reconhecimento facial está se expandindo sendo inserido gradualmente no dia a dia das pessoas.

No próximo capítulo, serão abordadas as ferramentas aplicadas neste trabalho, bem como a principal base tecnológica por detrás delas.

[13] https://newsroom.mastercard.com/latin-america/pt-br/press-releases/mastercard-disponibiliza-autenticacao-de-pagamento-por-selfie-no-brasil-e-no-mexico/ (Acesso: Nov/2019)
[14] https://www.citylab.com/design/2018/05/the-metro-stations-of-sao-paulo-that-read-your-face/559811/ (Acesso: Out/2019)
[15] https://www.forbes.com/sites/sap/2017/11/08/face-recognition-technology-set-to-transform-retail/#357852574877 (Acesso: Out/2019)

4 Ferramentas E Tecnologias

Com o objetivo de avaliar a viabilidade da utilização do reconhecimento facial, foram testados empiricamente dois algoritmos disponibilizados gratuitamente.

Para comparar sua eficiência, algumas ferramentas foram ser utilizadas de forma direta ou indireta. As escolhas foram motivadas, pelas linguagens de programação, que poderiam ser utilizadas e pelo licenciamento, dando preferência às ferramentas gratuitas. Avaliações do *GitHub* foram essenciais na decisão das bibliotecas também.

Neste capítulo serão apresentadas essas ferramentas, a linguagem de programação e o ambiente de desenvolvimento utilizado para os testes.

4.1. Python

Python é uma linguagem interpretada, conhecida pela sua sintaxe simples e amigável, o que lhe concedeu fama e rápido reconhecimento. É utilizada principalmente para trabalhar com *frameworks* para criação de servidores para páginas *web* e com a área de ciência de dados. [16]

Esta linguagem foi escolhida para este trabalho por ser utilizada no desenvolvimento do servidor do aplicativo Desaparecidos-RJ, além de possuir uma variedade de bibliotecas de reconhecimento facial.

4.2. *PyCharm - Professional* (versão licenciada para estudantes)

PyCharm é um ambiente de desenvolvimento para a linguagem *Python* e seus *frameworks* desenvolvido pela empresa *JetBrains*. Este ambiente foi escolhido por se tratar de uma IDE completa, com recursos de ponta, e por ser altamente recomendado pela comunidade de programadores de *python*.[17]

[16] https://www.cetax.com.br/blog/linguagem-python-por-que-aprender/ (Acesso: Jun/2020)
[17] https://oestatistico.com.br/10-ides-para-programar-em-python/ e https://blog.geekhunter.com.br/ides-e-editores-de-codigo-em-python-para-2020/ (Acesso: Mai/2020)

Por ser um *software* pago, em sua versão completa, foi requisitada a licença especial de estudantes. Não obstante, este trabalho também poderia ser desenvolvido utilizando a sua versão gratuita, *Community*, disponível na sua página *web*.[18]

4.3. Redes Neurais Convolucionais

Redes Neurais são modelos de aprendizado de máquina que se propõe a simular o funcionamento de um sistema nervoso humano. *Convolutional Neural Networks* (CNNs – traduzido do inglês como Redes Neurais Convolucionais) são provavelmente o modelo de rede *Deep Learning* mais conhecido e utilizado atualmente. (PONTI e COSTA, 2017, p.74).

Muito comum no tratamento de imagens, sua criação tem a intenção de se aproximar dos aspectos biológicos da visão. O funcionamento dos neurônios de uma rede neural convolucional é inspirado na parte do cérebro responsável pela visão humana, atribuindo pesos e valores para elementos de uma imagem. Segundo Vargas, Carvalho e Vasconcelos (2016, p.1): "uma CNN é capaz de aplicar filtros em dados visuais, mantendo à relação de vizinhança entre os pixels da imagem ao longo do processamento da rede". Deste modo, ele consegue interpretar a imagem como um todo e não apenas como pequenas unidades separadas.

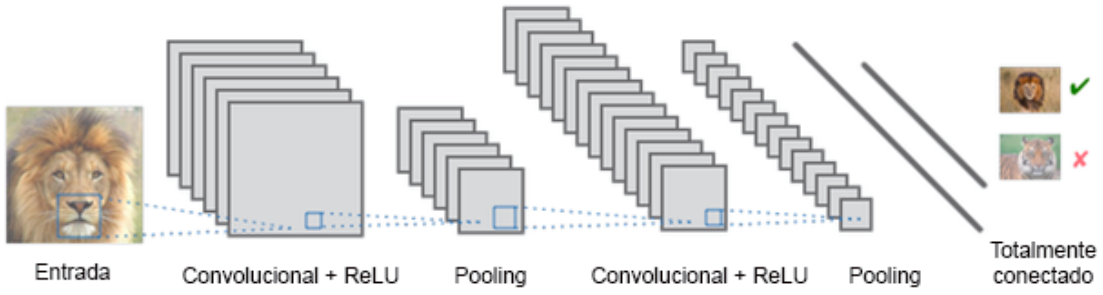


Figura 10 – Exemplo de uma rede neural convolucional.
Fonte: http://www.electricalelibrary.com/2018/11/20/o-que-sao-redes-neurais-convolucionais/ (Acesso em: Jan/2020)

A figura 10 apresenta o funcionamento de uma rede neural convolucional, dado uma imagem de entrada. Na primeira etapa, a de convolução (Convolutional + ReLU – Figura 10), haverá uma série de neurônios responsáveis pela aplicação de filtros em pedaços específicos da imagem. Em sua essência, um filtro é um vetor ou uma matriz

[18] https://www.jetbrains.com/pt-br/pycharm/download/#section=windows

pequena que percorre a matriz de entrada, fazendo cálculos que transformam o dado de entrada em uma matriz menor, ou vetor.

Os filtros são responsáveis pela redução de ruídos visuais, os quais costumam ser responsáveis pelo aumento da carga computacional. A exemplo, filtros de detecção de borda são utilizados para destacar o contorno dos objetos da imagem, facilitando a separação de características consideradas principais para o reconhecimento. Existem diversos filtros diferentes, como mostra a figura 11.

Figura 11 – Tipos de filtros de imagem
Fonte: http://www.electricalelibrary.com/2018/11/20/o-que-sao-redes-neurais-convolucionais/ (Acesso em: Jan/2020)

Aplicados os filtros, a próxima etapa vista na figura 10 é a aplicação de uma função de ativação. Esta etapa tem como objetivo retirar a linearidade dos resultados

obtidos anteriormente, trazendo uma maior capacidade interpretativa a rede neural, ou seja, ela consegue fazer pequenas alterações nos fatores de decisão da rede para que ela consiga aprender e gerar melhores resultados.

Existem diversas funções de ativação, tais como: Sigmoide, Tangente Hiperbólica (TanH) e Unidade Linear Retificada (ReLU).

Ainda observando a figura 10, após a fase de convolução, é aplicada a associação, responsável pela extração de apenas características importantes e redução do tamanho da entrada de dados. Tratando-se de matrizes, ele irá aplicar um filtro para subdividir a matriz ou vetor de dados vindo da fase anterior, podendo este filtro ser diferente em cada rede neural. Alguns exemplos de filtros são: maior valor, soma e média.

A Figura 12 apresenta o processo de associação com filtro de maior valor. Para funcionar, ele dividirá a matriz de entrada em seções iguais e percorrerá cada uma delas encontrando o maior valor associado àquela seção. Em seguida, ele passará este valor para uma matriz menor que, ao fim, conterá o maior valor de cada subdivisão feita na matriz de entrada.

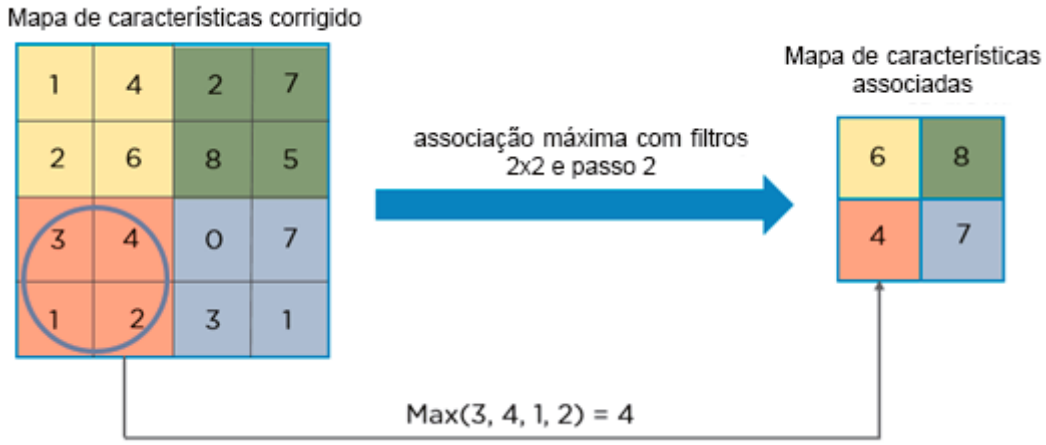


Figura 12 – Associação com filtro de maior valor
Fonte: http://www.electricalelibrary.com/2018/11/20/o-que-sao-redes-neurais-convolucionais/ (Acesso em: Jan/2020)

Essas etapas são importantes, pois reduzem a carga de processamento das redes neurais. Como pode ser observado na figura 10, a convolução e a associação fazem parte da seção de extração de características e ambas podem ocorrer duas ou mais vezes antes de ir para a seção de classificação. Isto garante a possibilidade de reaplicação de filtros para mais retirar ruídos ou a aplicação de novos filtros em cima daquela mesma etapa.

A seção de classificação, após a extração de características, possui a fase denominada como “Totalmente conectada”. Nesta se encontra a rede neural *perceptron*, responsável por pegar todos os dados que foram processados até o momento e passá-los por uma série de camadas que levarão até o resultado final.

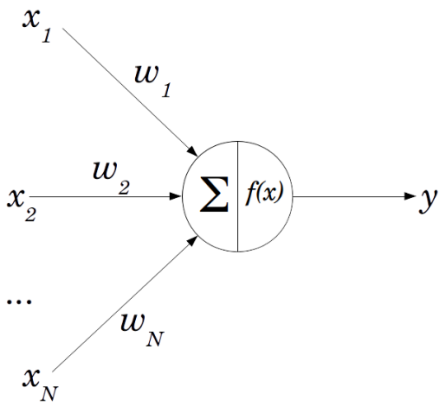


Figura 13 – Exemplo de perceptron
Fonte: https://lucidar.me/en/neural-networks/learning-rule-demonstration/ (Acesso: Jan/2020)

A figura 13 ilustra a ideia de um *perceptron*. As entradas x_1, x_2, ..., x_N vêm de posições diferentes e possuem pesos diferentes (w_1, w_2, ..., w_N). O neurônio que recebe essas informações, faz uma soma ponderada com os valores recebidos e compara o resultado com o limiar estabelecido para aquela camada. Ele identifica o nível de sucesso esperado e retorna um resultado y que será repassado ao(s) neurônio(s) da próxima camada. Este processo continua na estrutura até uma camada com um único neurônio que irá comparar o resultado da soma com o limiar e classificar a imagem de entrada.

As redes neurais convolucionais foram utilizadas na biblioteca *OpenFace* para treinar o modelo que foi aplicado, neste trabalho, para a detecção de rostos. Este modelo também é usado *Face Recognition* que foi a segunda alternativa testada no estudo realizado.

4.4. Base de dados

Para se trabalhar com reconhecimento facial, é necessário a existência de uma base de fotografias de indivíduos que possa ser classificada entre dois tipos de dado:

a) Banco de dados: Representa um subconjunto dos dados onde os elementos são identificáveis e serão usados para comparações com as imagens que estão sendo utilizadas como entrada.

b) Dados de entrada: É um subconjunto da base de dados original onde, em teoria, as pessoas não possuem identificação. Foram utilizados para simular a busca por pessoas sem identidade em um banco de dados de rostos conhecidos.

A base de dados criada pela Universidade de Milano-Bicocca[19] foi utilizada com para os testes. É uma base gratuita, porém de acesso restrito a universidades e pesquisadores, voltada para estudos de reconhecimento facial com oclusão parcial. São 590 imagens com oclusão parcial dos mais diversos tipos: cabelos, óculos de sol, chapéus, etc.

4.5. Biblioteca Dlib

Inicialmente idealizada por Davis King, hoje com mais de 100 colaboradores, Dlib é uma biblioteca gratuita, desenvolvida em C++ que objetiva servir como uma coleção de componentes de *framework* para serem usados em qualquer tipo de aplicação, fins comerciais ou acadêmicos[20]. Atualmente, disponibiliza módulos para suporte a Álgebra Linear, Processamento de imagens, Otimização, entre outros, todos com uma extensa documentação.

O site oficial da biblioteca oferece exemplos de classes desenvolvidas para detecção de face, alinhamento, clusterização etc.[21] Para detecção de rostos, a Dlib utiliza Redes Neurais Convolucionais.

Como exemplificado na figura 14, esta biblioteca é usada por ambas as bibliotecas utilizadas nos testes. O *OpenFace* utiliza apenas o módulo de detecção de rostos e o *FaceRecognition* o módulo de detecção de rostos e o modelo pré-treinado, com acurácia de 99.38%[22] no *dataset Labeled Faces in the Wild*[23].

[19] http://www.ivl.disco.unimib.it/minisites/umbdb/description.html (Acesso: set/2018)
[20] http://www.jmlr.org/papers/volume10/king09a/king09a.pdf
[21] http://dlib.net/ (Acesso: Abr/2019)
[22] https://github.com/ageitgey/face_recognition (Acesso: Set/2018)
[23] http://vis-www.cs.umass.edu/lfw/ (Acesso: Dez/2018)

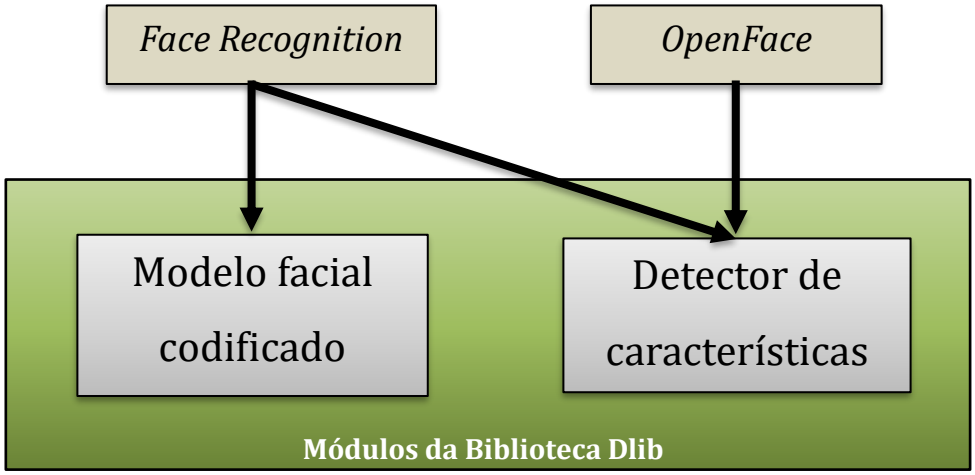


Figura 14 - Diagrama do uso da biblioteca Dlib nos testes
Fonte: Elaboração própria

4.6. Face Recognition

Criada por Adam Geitgey, *Face Recognition* é uma API (*Application Programming Interface*) de reconhecimento facial, baseada no estado da arte da biblioteca Dlib, desenvolvida em *Python* e disponibilizada gratuitamente no *GitHub* do próprio autor.[24] Foi escolhida pela sua ótima recomendação no *GitHub* e sua simplicidade de acesso, sendo muito fácil sua utilização.

4.7. FaceNet e a função Triplet Loss

FaceNet é uma rede neural convolucional que mapeia rostos utilizando o conceito de espaços Euclidianos para encontrar as similaridades dos rostos. Assim como a biblioteca Dlib, é eficiente com bases de dados de grande porte, como as do *YouTube*, onde atingiu 95,12% de acurácia[25].

A função *triplet loss* do *FaceNet* agrupa resultados similares de modo que estes permaneçam próximos uns dos outros, e que os diferentes fiquem mais distantes, como ilustrado na figura 15.

[24] https://github.com/ageitgey/face_recognition (Acesso: Set/2018)
[25] https://arxiv.org/pdf/1503.03832.pdf (Acesso: set/2018)

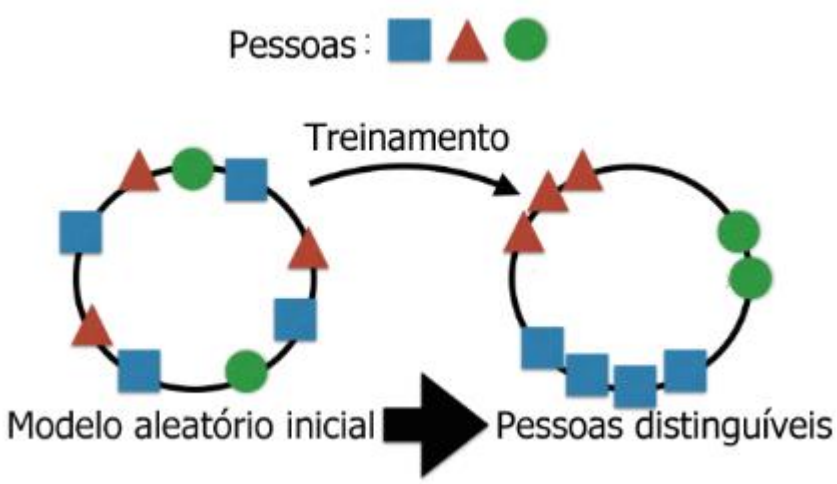


Figura 15 - Antes e depois da aplicação da função Triplet Loss do FaceNet
Fonte: http://reports-archive.adm.cs.cmu.edu/anon/anon/usr0/ftp/2016/CMU-CS-16-118.pdf (Acesso: set/2018)

Para obter este resultado é utilizada a fórmula apresentada na equação 01 e da codificação da face de três imagens que são classificadas como:

- Imagem âncora: Funciona como base comparativa. Ela é utilizada para criar um padrão de resultados.

- Imagem positiva: A imagem de uma mesma pessoa da imagem âncora. Os resultados serão bastante próximos do padrão esperado (o resultado da imagem âncora) e serão agrupados, para que resultados parecidos fiquem próximos.

- Imagem negativa: Imagem de uma pessoa diferente. Os resultados vão ser distantes do determinado como base pela imagem âncora.

$$Loss = \sum_{i=1}^{N} \left[\| f_i^a - f_i^p \|_2^2 - \| f_i^a - f_i^n \|_2^2 + \alpha \right]_+$$

Equação 01 – Fórmula da Triplet Loss
Fonte: https://medium.com/@mohitsaini_54300/train-facenet-with-triplet-loss-for-real-time-face-recognition-a39e2f4472c3 (Acesso: Jan/2020)

Esta função é utilizada para melhorar a performance do modelo, uma vez que irá agrupar as imagens das mesmas pessoas e separar em outros grupos as pessoas diferentes.

O *FaceNet* é aplicado em dois momentos na criação do modelo utilizado no *OpenFace*. Primeiramente, empregam o modelo pré-treinado do *FaceNet* para criar um vetor de 128 dimensões que representará um modelo genérico de rosto. Em seguida,

utilizarão a função *triplet loss* para a organização do modelo de modo que ele se torne mais eficiente.

4.8. OpenCV

A biblioteca *OpenCV* está dentre as mais famosas e utilizadas dentro da área de visão computacional. Sua intenção é servir como infraestrutura para o desenvolvimento de aplicativos voltados a visão computacional.

Também é utilizada pelo *OpenFace*, porém, na parte de normalização da imagem. Ela é responsável pela leitura da imagem e a padronização do esquema de cores.

4.9. Torch

Torch[26] é um *framework*, utilizado pelo *OpenFace* para melhorar a performance de sua rede neural na hora de extrair as características de um rosto. Ele é desenvolvido em lua, de código aberto, que busca prover uma estrutura para um melhor funcionamento de algoritmos que exijam grande poder computacional. É usado principalmente para fins científicos, trabalhando com otimização de numerais, álgebra linear, aprendizagem de máquina etc. Ele possui suporte para *Graphics Processing Unit* (*GPU*), tornando-se atrativo para desenvolvimento de códigos de visão computacional.

4.10. OpenFace

OpenFace[27] é uma biblioteca desenvolvida em *Python* com o propósito de ser utilizada em tecnologias móveis. Sua performance faz com que ela possa ser comparada com tecnologias e bibliotecas grandes, como a *DeepFace* do *Facebook.*

Utiliza-se de tecnologias como *FaceNet* (1), *Dlib* (2), *Torch* (3) e Scikit-learn (4)[28], como mostra a Figura 16. Ela foi escolhida para este trabalho por causa do número de recomendações no *GitHub* e ótima performance.

[26] http://torch.ch/ (Acesso: Jan/2020)
[27] https://github.com/cmusatyalab/openface (Acesso: Set/2018)
[28] https://scikit-learn.org/stable/

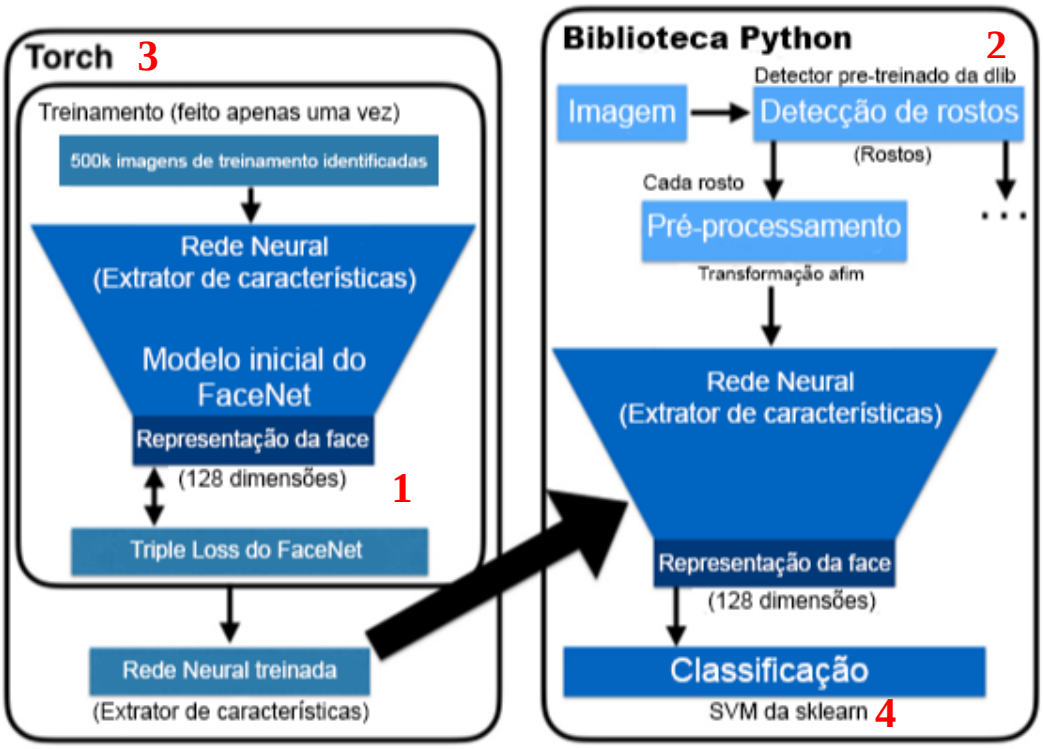


Figura 16 – Arquitetura do algoritmo OpenFace
Fonte: http://reports-archive.adm.cs.cmu.edu/anon/anon/usr0/ftp/2016/CMU-CS-16-118.pdf
(Acesso: Jul/2019)

4.11. Converseen

O Converseen é um Programa de edição de imagens gratuito para diversos sistemas operacionais. Este foi utilizado para converter as imagens da base de dados de seus formatos originais para que pudessem ser lidos propriamente pelos algoritmos utilizados nos testes.

Todas as imagens foram convertidas do formato .ppm para .png para que a qualidade da imagem não fosse perdida no processo.

Após a apresentação dos softwares utilizados para o desenvolvimento do teste e a arquitetura por detrás das bibliotecas, o próximo capítulo apresenta os testes realizados e como utilizar as bibliotecas *OpenFace* e *Face Recognition*.

5 Testes

Os testes foram realizados utilizando 120 fotos da base de dados da Universidade de Milano-Bicocca, criada especialmente para estudos de técnicas de reconhecimento facial com oclusão parcial.

Inicialmente foram selecionados 22 indivíduos, cada uma contendo entre 3 e 12 imagens de uma mesma pessoa, enumeradas de forma ordenada, seguindo o seguinte padrão: X-Y.png, onde X é o número de identificação do indivíduo e Y, o número identificador da imagem.

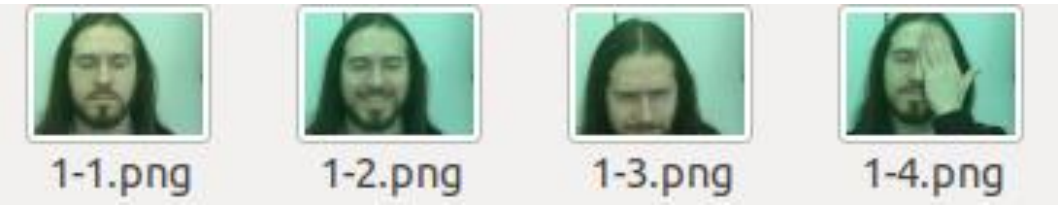


Figura 17 – Organização da base de dados
Fonte: Elaboração própria

Os experimentos consistem na comparação entre uma imagem frontal de cada indivíduo, similar a 1-1.png mostrada na figura 17, e todas as imagens restantes do banco. Foram analisados separadamente os seguintes pontos: imagens de um mesmo indivíduo ao comparar contra todas as imagens dele mesmo, as imagens de um indivíduo contra todo grupo em busca de falsos negativos, falsos positivos e tipos de oclusão com maior número de erros.

Para realizar os testes, é necessário fazer a comparação do valor adquirido através de uma fórmula matemática, observada na equação 02, com o *threshold*, um parâmetro de tolerância configurado no código que permitirá a classificação dos resultados como verdadeiros ou falsos.

- Valor > *Threshold:* O resultado do reconhecimento será negativo, classificando os dois rostos como de pessoas diferentes.
- Valor ≤ *Threshold:* O resultado do reconhecimento será positivo e classificará os dois rostos como a mesma pessoa.

Por isso, destaca-se a importância do ajuste correto do *threshold*, pois ele funciona como um limite na comparação entre as duas imagens, e, portanto, o valor determinado alterará os resultados.

Para os *thresholds* os seguintes valores foram adotados:

- *Face Recognition*: o valor 0.6.
- *OpenFace*: os valores 0,5, 0,6 e 0.9292[29]

O valor adotado para o *Face Recognition* foi a partir da recomendação do autor das bibliotecas. Para o *OpenFace* os testes foram feitos com três valores distintos, uma vez que não havia confirmação das medidas ideais. Estes foram escolhidos por diversos motivos:

- 0,9292: foi retirado de um documento de análise no *GitHub* da própria biblioteca *OpenFace*.
- 0,6: foi usado por ter o mesmo valor da biblioteca *Face Recognition*, a fim de testar se haveria similaridades de comportamento.
- 0,5: foi escolhido para servir de limiar inferior durantes os testes. O valor não é muito abaixo do *threshold* de valor médio, e foi usado para servir como um medidor em casos de pequenas alterações. Sabe-se que, caso o número escolhido fosse muito baixo, a diferença no número de acertos poderia ser maior do o que desejado.

O método de reconhecimento facial utilizado nesses testes se dá através da captura de 128 pontos de cada um dos rostos que estão sendo comparados. Os cálculos são feitos utilizando a Distância Euclidiana, ilustrada pela Equação 02. Para tanto, primeiro diminui-se o ponto A da fotografia 1 do ponto A da fotografia 2, obtendo um resultado. Eleva-se esse resultado ao quadrado. Repete-se todo o processo até o fim do cálculo da diferença de todos os 128 pontos das imagens. Soma-se os valores encontrados e, em seguida, calcula-se a raiz quadrada do resultado.

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \; = \; \sqrt{\sum_{i=1}^{n} (p_i - q_i)}$$

Equação 02 – Distância Euclidiana
Fonte: https://pt.wikipedia.org/wiki/Dist%C3%A2ncia_euclidiana (Acesso: Jul/2019)

[29] https://github.com/cmusatyalab/openface/blob/master/evaluation/lfw.nn4.small2.v1/accuracies.txt (Acesso: Abr/2020)

Desse modo, o *threshold* poderá variar o valor para ter mais rigidez (baixando-o) ou ser mais abrangente (aumentando-o).

5.1. Limitações

Os modelos pré-treinados disponibilizados por ambas as bibliotecas utilizam bases de dados públicas, onde o conteúdo varia de dezenas a centenas de milhares de fotos em uma grande variedade de posições. Elas, porém, não são próprias para análises de reconhecimento facial com oclusão parcial, que é o foco deste trabalho.

Ainda assim, devido a limitações computacionais, os modelos padrões utilizados como *default* foram mantidos durante os testes. Deste modo, os resultados esperados serão avaliados levando-se em consideração que não houve treinamento adequado para reconhecer todos os tipos de oclusão apresentados pela base de dados utilizada neste trabalho.

Ambos os algoritmos foram testados em uma máquina com sistema operacional Ubuntu 18.04.4 LTS, Processador Intel Core it-4200U CPU@ 1.60GHz x4 e 4GB de RAM.

5.2. Face Recognition

O algoritmo do *Face Recognition* possui pacote disponível para *download* no gerenciador de pacotes do *Python*, o pip. Após o *download*, é possível utilizá-lo dentro de qualquer código *Python* importando-o como um pacote normal.

Em seu repositório no *GitHub*, existe uma breve explicação de como utilizar os códigos disponíveis, contudo, o código apresentado no quadro 01 foi a principal parte utilizada neste trabalho.

Primeiramente, ele importa a biblioteca *face_recognition* no código. Em seguida, é endereçado a imagem de origem e a imagem a ser comparada para as variáveis *known_image* (imagem conhecida) e *unknown_image* (imagem desconhecida), respectivamente. Essas variáveis são utilizadas para gerar dois vetores de 128 pontos, um para cada imagem, que são usados para fazer comparações. É importante saber que esses pontos são gerados apenas se o algoritmo for capaz de reconhecer um rosto.

```
import face_recognition

known_image = face_recognition.load_image_file("biden.jpg")
unknown_image = face_recognition.load_image_file("unknown.jpg")

biden_encoding = face_recognition.face_encodings(known_image)[0]
unknown_encoding = face_recognition.face_encodings(unknown_image)[0]

results = face_recognition.compare_faces([biden_encoding], unknown_encoding)
```

Quadro 01 – Exemplo de como utilizar a biblioteca
Fonte: https://github.com/ageitgey/face_recognition (Acesso: Set/2018)

As variáveis *biden_enconding* (em homenagem a Joe Biden, ex-vice-presidente dos Estados Unidos, utilizado como exemplo pelo autor) e *unknown_encoding* armazenam os pontos gerados pela imagem conhecida e pela imagem desconhecida, respectivamente.

Ao fim, ele chama um método para comparar os dois vetores e estimar se há uma correspondência de resultados. Vale ressaltar que o detalhe mais importante deste pequeno trecho de código é a geração do vetor.

Em um teste de reconhecimento facial com foco em imagens com oclusão é importante que o vetor de pontos seja testado, uma vez que o algoritmo pode não ser capaz de compreender se existe um rosto na imagem. Da mesma forma, é importante que, ao descobrir um problema na geração de vetores, deve ser mapeado se o problema foi originado pela imagem de origem ou pela imagem desconhecida, quando não por ambas.

Sendo assim, os resultados possíveis dos testes são:

- [True], quando houver uma correspondência.
- [False], quando não houver uma correspondência.
- “Problema na imagem comparada”, quando não houver geração de vetor na imagem desconhecida.
- “Problema na imagem de origem”, quando não houver geração de vetor na imagem conhecida.
- “Problema nas duas imagens”, quando não houver geração de vetor em nenhuma das duas imagens.

5.2.1. Programa de Teste

Para a utilização deste código no teste realizado, foi desenvolvido um programa em *python*, separado em 4 (quatro) métodos:

Método de importação de endereços

Responsável pela criação de um dicionário de endereços onde as imagens estão localizadas.

Este é o primeiro método chamado pelo programa. Sua principal função é percorrer a pasta onde estão as imagens da base de dados e adicioná-las ao dicionário de indivíduos, como apresentado no algoritmo do quadro 02.

Entrada: O endereço E da pasta onde estão localizadas as imagens da base de dados

Criar um novo dicionário para armazenamento do endereço de imagens de cada indivíduo.
Para todos os arquivos em E faça:
    Para cada nome_arquivo em arquivos **faça:**
        arquivo := nome_arquivo.separar(-)
        Se arquivo[0] ⊄ dicionario.chaves então:
            Criar uma nova lista para armazenar o endereço daquele arquivo. Em seguida, adicionar lista de imagens a dicionario[arquivo[0]].
        Senão:
            Adicionar endereço de arquivo à lista de arquivos no dicionário[arquivo[0]].

Saída: Dicionário com endereço de todos as imagens, separadas para cada indivíduo

Quadro 02 – Pseudo-código do método de importação de endereços da base de dados
Fonte: Elaboração própria

A identificação de cada indivíduo no dicionário é feita através da chave, enquanto os endereços, de cada imagem do indivíduo, são armazenados numa lista encadeada que está relacionada a cada chave.

Método de comparação

Este faz a comparação de um indivíduo com todas as imagens do banco de dados. Este procedimento é executado em um laço até que todas as imagens tenham sido comparadas.

Este é o principal método do código. Através dele é realizado as comparações que chama os métodos de resultado e os exporta para o documento CSV.

Para comparar todas as imagens e retornar todas as variáveis necessárias para sua avaliação de desempenho é necessário percorrer o dicionário duas vezes para recuperar a lista de individuo_origem e individuo_comparado.

Entrada: Dicionário D de endereço de todas as imagens da base de dados

Para toda chave_origem, lista_imagem_origem em D faça:
Para toda chave_comparada, lista_imagem_comparada em D faça:
Para cada imagem_comparada em P_i, i={0, tamanho(lista_imagem_comparada)} faça:
Iniciar cálculo de tempo de cada reconhecimento

Chamar funções apresentadas na figura XX para fazer a detecção de rostos, utilizando os endereços para a primeira imagem_origem de cada indivíduo e das demais imagens para imagem_comparada.

Finalizar cálculo de tempo de cada reconhecimento

Chamar a função de cálculo de resultados e a função de exportação de resultados.

Saída: Documento CSV com os resultados finais.

Quadro 03 – Pseudo-código do método de comparação de imagens
Fonte: Elaboração própria

As listas são percorridas, preenchendo os endereços de cada imagem no código do quadro 03. Em seguida, o método de verificação de resultado é chamado para averiguar se não há erros de detecção de faces em nenhuma das duas imagens. Após isso, o método de exportação de resultados (Quadro 5) é chamado.

Método de verificação facial

Verifica se há algum erro de reconhecimento facial e geração de um vetor de pontos. Caso haja, ele retorna uma opção entre três tipos de erro:

- Problema na imagem de origem: quando a não detecção de um rosto se origina na imagem que está sendo usada como entrada para comparação
- Problema na imagem comparada: quando a não detecção de um rosto se origina na imagem que está no banco de dados e está sendo comparada à foto de entrada
- Problema nas duas imagens: quando ambas as fotos apresentam problemas na detecção de rosto.

Este método recebe os vetores de 128 pontos das duas imagens que estão sendo comparadas com a intenção de descobrir se o vetor está vazio ou não. Se estiver, significa que não foi possível detectar um rosto em pelo menos uma das imagens e retornará uma mensagem indicando o erro.

Caso contrário, ele fará o cálculo da distância euclidiana utilizando-se da fórmula apresentada na equação 02 e retornará um resultado *booleano*.

Entrada: Vetor V1 gerado pelo algoritmo ao detectar um rosto na imagem de origem e o vetor V2 também gerado pelo algoritmo ao detectar um rosto na imagem comparada.

Se tamanho(V1) == 0 e tamanho(V2)==0 então:
A variável resultado receberá o valor “Problema nas duas imagens”
Senão se tamanho(V1) == 0 então:
A variável resultado receberá o valor “Problema na imagem de origem”
Senão se tamanho(V2) == 0 então:
A variável resultado receberá o valor “Problema na imagem comparada”
Senão:
A variável resultado receberá o valor de comparação entre os dois vetores, podendo assumir os valores [True] ou [False]

Saída: O resultado do reconhecimento facial

Quadro 04 – Pseudo-código do método de verificação de resultados
Fonte: Elaboração própria

Método de exportação de resultados

Este método é responsável pela criação, abertura, acréscimo de dados e fechamento do arquivo onde os resultados deste teste estão sendo documentados. Ele recebe seis parâmetros que serão acrescentados ao final do documento, sendo eles: Indivíduo_Origem, Imagem_Origem, Indivíduo_Comparado, Imagem_Comparada, Resultado e Tempo.

A terminar, exporta o resultado de cada comparação feita no banco de dados em um arquivo .csv, como pode ser visto no quadro 05.

Entrada: Individuo Ind_Ori da imagem de origem, a imagem de origem Img_Ori, o indivíduo Ind_Com da imagem de comparação, sua imagem que está sendo comparada Img_Com, o Resultado final da comparação e o tempo de reconhecimento.

Abrir arquivo "resultado-face_recognition.csv"
Acrescentar à última linha cada variável na seguinte ordem: Ind_Ori, Img_Ori, Ind_Com, Img_Com, Resultado, tempo_reconhecimento. Quebrar a linha no final.
Fechar arquivo.
Saída: Arquivo CSV acrescido do resultado da última comparação feita.

Quadro 05 – Pseudo-código do método de exportação de resultados
Fonte: Elaboração própria

5.3. OpenFace

Para a instalação da *OpenFace* é aconselhado seguir os tutoriais apresentados no site do laboratório desenvolvedor, *cmusatyalab*[2], onde é possível encontrar diferentes métodos de instalação, sendo a opção mais fácil, a instalação através de *Docker*.

5.3.1. Programa de teste

Para esses testes, foi reaproveitado o modelo do código de demonstração disponibilizado no repositório do GitHub[30] junto à biblioteca. O código é simples, possui apenas chamadas de carregamento de bibliotecas e modelos, uma função para o processo de normalização e a iteração, onde são feitas as comparações entre as imagens de origem e as imagens comparadas.

[30] https://github.com/cmusatyalab/openface/blob/master/demos/compare.py

Contudo, para que os resultados pudessem ser obtidos de modo satisfatório foram necessárias algumas modificações. Para tanto, foi criada uma classe chamada ReconhecimentoFacial() com 4 (quatro) métodos, responsáveis por todo o processo de reconhecimento explicados abaixo:

Método de importação de endereços

Responsável pela criação de um dicionário de endereços de onde as imagens estão localizadas. Exatamente igual ao teste da *Face Recognition,* ver quadro 02.

Método utilizado para o processo de normalização

Este é responsável pela chamada das funções básicas do pré-processamento das imagens.

Entrada: Dicionário de imagens

Carregar a imagem E
Se imagem não pôde ser carregada:
    Levantar exceção de "Imagem não carregada"
Alterar esquema de cores de imagem para RGB

Encontrar o maior rosto detectado na imagem carregada
Se não pôde encontrar nenhum rosto:
    Levantar exceção de "Rosto não encontrado"

Transformar e alinhar rosto encontrado na imagem carregada
Se não pôde alinhar o rosto:
    Levantar exceção de "Incapaz de alinhar o rosto"

Alimentar a rede neural com a imagem transformada para extrair suas características
Saída: Vetor de características extraídas pela rede neural.

Quadro 06 – Método de exportação de resultados
Fonte: Elaboração própria

Esse método trata toda a parte principal de pré-processamento do código, conforme apresentado no quadro 06. Ele foi retirado da demonstração apresentada pelos criadores, com poucas modificações, deixando apenas o essencial para o funcionamento do código.

Ele faz correção de cores da imagem para padronização, encontra 68 pontos que são utilizados para detectar um rosto e, por último, transforma a imagem e a alinha.

Ao fim, o resultado final das transformações passa pela rede neural para extração das características do rosto que serão transformadas num vetor de 128 pontos.

Método de análise e extração de resultados

O método apresentado no quadro 07 é responsável pela comparação das imagens para extração de resultados.

Entrada: Dicionário D de endereço de todas as imagens da base de dados

Para toda chave_origem, lista_imagem_origem em D faça:
Para toda chave_comparada, lista_imagem_comparada em D faça:
Para cada imagem_comparada em P_i, i={0, tamanho(lista_imagem_comparada)} faça:
Iniciar cálculo de tempo
Imagem_origem = Endereço + indivíduo_origem + "-1.png"
Imagem1 = False
Imagem2 = False
Tentar:
A variável auxiliar da imagem origem tenta receber o vetor gerado pelo método de normalização, V1.
Exceção:
Imagem1 = True
Tentar:
A variável auxiliar da imagem origem tenta receber o vetor gerado pelo método de normalização, V2
Exceção:
Imagem2 = True
Finalizar cálculo de tempo
Se Imagem1 == False ou Imagem2== False, então:
Se Imagem1 == True e Imagem2==True, então:
A variável resultado receberá o valor "Problema nas duas imagens"
Senão se Imagem1 == True e Imagem2 ==False, então:
A variável resultado receberá o valor "Problema imagem de origem"
Senão se Imagem2 == True e Imagem1 ==False, então:
A variável resultado receberá o valor "Problema imagem comparada"
Senão:
Faz-se a diferença entre os vetores encontrados, V1 - V2, encontrando o resultado R. Em seguida, aplica-se um método de multiplicação de matrizes sobre R para elevá-lo ao quadrado.
Se R<=valor_threshold:
A variável resultado receberá o valor "True"
Senão:
A variável resultado receberá o valor "False"

Saída: Resultado da comparação das imagens.

Quadro 07 – Pseudo-código do método de comparação e extração de resultados
Fonte: Elaboração própria

Primeiro, ele tenta fazer a normalização de cada uma das imagens que estão sendo comparadas, separadamente. A divisão foi criada, de modo que, através de variáveis

booleanas, é possível fazer o controle e rastrear, em casos de falha, suas devidas origens. Para estes erros, os resultados seguem o mesmo padrão utilizado para os testes da *Facial Recognition*.

Em caso de sucesso na normalização das duas imagens, será aplicada a distância Euclidiana Quadrática, a qual segue a mesma ideia da Distância Euclidiana apresentada na equação 03, porém, sem a aplicação da raiz quadrada.

$$\sum_{i=1}^{n}(p_i - q_i)$$

Equação 03 – Fórmula da Distância Euclidiana Quadrática
Fonte:
http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Euclidean_and_Euclidean_Squared_Distance_Metrics.htm (Acesso: Mai/2019)

A razão de não se usar a Distância Euclidiana apresentada na equação 02, deve-se ao fato de que ao não executar a raiz quadrada, existe um ganho de tempo e, logo, uma maior eficiência nos cálculos de comparação.

Método de exportação de resultados

Exporta o resultado de cada comparação feita no banco de dados em um arquivo .csv. Exatamente igual ao teste da *Face Recognition*, apresentado no quadro 05.

Chamada do programa

Para chamar a classe ReconhecimentoFacial() com sucesso existe a necessidade de atribuir caminhos a algumas variáveis responsáveis pelo carregamento dos modelos e dados utilizados.

```
fileDir = os.path.dirname(os.path.realpath(__file__))
modelDir = os.path.join(fileDir, '..', 'models')
dlibModelDir = os.path.join(modelDir, 'dlib')
openfaceModelDir = os.path.join(modelDir, 'openface')
```

Quadro 08 – Código responsável pelo endereçamento das pastas
Fonte: https://github.com/cmusatyalab/openface/blob/master/demos/compare.py (Acesso: Set/2018)

Como visto no quadro 08, inicialmente, são atribuídos caminhos padrões a variáveis. Esses endereços são utilizados para encontrar arquivos em caminhos *default*,

ou seja, caso não haja alterações no código ou na organização das pastas, esses endereços serão usados como referência para recuperar os arquivos necessários para o funcionamento da classe.

```
parser = argparse.ArgumentParser()

parser.add_argument('--dlibFacePredictor', type=str, help="Path to dlib's face predictor.",
                    default=os.path.join(dlibModelDir, "shape_predictor_68_face_landmarks.dat"))

parser.add_argument('--networkModel', type=str, help="Path to Torch network model.",
                    default=os.path.join(openfaceModelDir, 'nn4.small2.v1.t7'))

parser.add_argument('--imgDim', type=int,
                    help="Default image dimension.", default=96)

args = parser.parse_args()
```

Quadro 09 – Código responsável pela parametrização de componentes utilizados na biblioteca
Fonte: https://github.com/cmusatyalab/openface/blob/master/demos/compare.py (Acesso: Set/2018)

Em seguida são importados os arquivos *default*, ou os passados por parâmetro, através da linha de comando, exemplificados no quadro 09.

Esses arquivos são fundamentais para o funcionamento do código, pois tratados modelos treinados para detecção e reconhecimento facial. Também pode ser alterado o parâmetro referente ao tamanho, em pixels, da imagem após o seu redimensionamento. Vale lembrar que quanto menor o tamanho da imagem, mais rápido é seu processamento.

Este código utiliza todos os valores considerados como *default* uma vez que não é chamado através da linha de comando.

```
align = openface.AlignDlib(args.dlibFacePredictor)
net = openface.TorchNeuralNet(args.networkModel, args.imgDim)
```

Quadro 10 – Código responsável pela inicialização das redes neurais utilizadas
Fonte: https://github.com/cmusatyalab/openface/blob/master/demos/compare.py (Acesso: Set/2018)

Por fim, as duas linhas apresentadas no quadro 10 são responsáveis pelo carregamento dos modelos e redes neurais a serem importadas e podem ser utilizados pelo método de normalização da classe ReconhecimentoFacial().

O próximo capítulo apresenta uma análise dos resultados gerados pelos testes descritos. Os códigos completos podem ser encontrados no repositório do *GitHub*[31].

[31] https://github.com/thaisssimoes/TCC

6 Resultados

Os resultados dos testes de reconhecimento facial utilizando as bibliotecas *Openface* e *Face Recognition* foram analisados tendo em vista sua utilização prática em um aplicativo.

Foram avaliados:

- Acertos – Imagens reconhecidas corretamente pelo algoritmo;
- Falsos Positivos – Imagens de indivíduos diferentes que foram reconhecidas;
- Falsos Negativos – Imagens de uma mesma pessoa que ao serem comparadas, retornaram resultados negativos, ou seja, não houve o reconhecimento;
- Tempo – O tempo médio para reconhecer cada rosto/levantar cada erro.
- Performance – Porcentagem de acertos dentro do universo de rostos reconhecidos como um indivíduo.
- Erros – Problemas de detecção de faces devido a alguma oclusão.

Em uma pré-avaliação das soluções obtidas, uma vez que foram realizados diversos testes com o mesmo algoritmo, foi determinado que os valores mínimos que os resultados deveriam possuir para serem avaliados era de 50% em cima da quantidade total de fotos da base de dados, semelhante aos valores apresentados no quadro 11.

Acertos	≥ 50%
Falsos Positivos	≤ 50%
Falsos negativos	≤ 50%

Quadro 11 – Pré-requisitos para avaliação
Fonte: Elaboração própria

Estes limitantes foram pensados levando em consideração o retorno esperado em uma busca por um aplicativo. Em uma pesquisa com reconhecimento facial, o ideal é que exista uma certeza de que o rosto procurado retorne como resultado.

O número de falsos positivos idealmente deve ser baixo, quando considerados os valores totais, pois pode comprometer a quantidade de respostas retornadas em uma busca.

A mesma medida se aplica aos falsos negativos, pois ele oculta a verdadeira quantidade de resultados considerados úteis para quem busca.

Portanto, foi estipulado o valor de 50% como limitante mínimo na pré-avaliação de resultados. Apenas o *Face Recognition* e o *OpenFace* com o *threshold* 0.6 alcançaram o objetivo.

Sendo assim, os resultados apresentados neste capítulo usaram os valores encontrados nestes dois testes para gerar uma avaliação.

Os resultados encontrados nos outros experimentos estão disponíveis no anexo 01.

6.1. Resultados

Acertos

Acertos em teste de reconhecimento facial significam que o algoritmo foi capaz de reconhecer o indivíduo A com ele mesmo. Porém, como o foco desse trabalho é reconhecimento facial com oclusão, os acertos também significam que os algoritmos são capazes de reconhecer a pessoa independentemente do tipo de oclusão da imagem.

Legenda do quadro 12:

- Indivíduo – Número identificador do indivíduo no banco de dados
- Qtde. Imagens – Quantidade de imagens que aquele indivíduo possui na base de dados
- *Face Recognition* – Número de acertos obtidos no teste com a biblioteca *Face Recognition*
- *OpenFace* – Número de acertos obtidos no teste com a biblioteca *OpenFace*

Indivíduo	**Qtde Imagens**	***Face Recognition***	***OpenFace***
1	7	7	5
2	4	1	1
3	4	2	2
4	4	4	3
5	4	3	2
6	5	4	3
7	3	3	2

8	4	3	3
9	4	3	3
10	4	3	3
11	5	5	4
12	9	8	6
13	6	5	3
14	6	5	4
15	4	2	2
16	5	3	3
17	6	3	3
18	4	2	2
19	12	12	1
20	8	5	4
21	6	4	2
22	6	6	2
Total:	**120**	**93**	**63**

Quadro 12 – Número de acertos dos algoritmos testados
Fonte: Elaboração própria

De acordo com os resultados apresentados no quadro 12, o algoritmo da *Face Recognition* possui uma margem de acerto bastante superior ao algoritmo da *OpenFace*. A razão para tal diferença pode estar em várias etapas do código. Alguns exemplos que poderiam causar isso: precisão do modelo, valor usado como *threshold*, problemas com a oclusão etc.

	0.5	**0.6**	**0.9292**
OpenFace	57	63	82

Quadro 13 – Resultado de acertos do algoritmo OpenFace
Fonte: Elaboração própria

Em uma comparação dos resultados apresentados nos três experimentos do algoritmo do *OpenFace,* explicitados no quadro 13, percebe-se que ao aumentar o valor do *threshold* aumenta-se também a quantidade de acertos.

Caso soose considerado apenas este ponto, a escolha por manter o valor limitante em um número alto seria a correta. Entretanto, em uma análise conjunta com os outros fatores decisivos, percebe-se que existem consequências, como o aumento considerável de falsos positivos, que prejudicam bastante o resultado final.

Ainda assim, foi observado que mesmo com o limiar mais alto, o *Face Recognition* ainda teve melhor desempenho, neste quesito, quando comparado com o outro algoritmo testado.

Falsos positivos

Falsos positivos são resultados positivos que não são verdadeiros. Em reconhecimento facial, um falso positivo é quando o algoritmo reconhece o indivíduo B, como indivíduo A, sendo A e B pessoas distintas. O quadro 14 ilustra a quantidade de falsos positivos de ambos os algoritmos.

Legenda do quadro 14:

- Indivíduo – Número identificador do indivíduo no banco de dados
- Qtde. Imagens – Quantidade de imagens que aquele indivíduo possui na base de dados
- *Face Recognition* – Número de falsos positivos obtidos no teste com a biblioteca *Face Recognition*
- *OpenFace* – Número de falsos positivos obtidos no teste com a biblioteca *OpenFace*

Indivíduo	**Qtde Imagens**	***Face Recognition***	***OpenFace***
1	7	6	-
2	4	1	-
3	4	-	3
4	4	1	1
5	4	-	7
6	5	7	1
7	3	5	5
8	4	-	2
9	4	1	-
10	4	2	1
11	5	3	3
12	9	-	-
13	6	1	1
14	6	-	1
15	4	5	-
16	5	6	2
17	6	5	4
18	4	2	1
19	12	2	5
20	8	6	-

21	6	-	-
22	6	8	-
Total:	**120**	**61**	**37**

Quadro 14 – Quantidade de falsos positivos dos algoritmos testados
Fonte: Elaboração própria

Apesar de apresentar um número superior de acertos, como visto no tópico anterior, a *Face Recognition* também obteve um maior número de falsos positivos. Isto significa que ao buscar por uma pessoa, ele vai trazer outras pessoas diferentes do indivíduo sendo buscado. A causa disso está diretamente ligada ao valor do *threshold* de cada algoritmo.

Após calcular o resultado da distância Euclidiana dos rostos, é feita uma comparação com *threshold.* Se o valor resultado do cálculo for menor ou igual ao limiar estabelecido, há grandes chances de serem o mesmo indivíduo. Portanto, quando o número de falsos positivos é alto, quer dizer que os resultados dos cálculos estão dentro do intervalo de valor que classifica aquele rosto como verdadeiro, por isso, existe a necessidade de ajuste do valor de tolerância.

Entretanto, o valor do *threshold* utilizado para o experimento com o *Face Recognition* foi escolhido utilizando a informação dada pelo autor da biblioteca, que em seus testes, o determinou como o parâmetro que obteve o melhor resultado. Quando levado em consideração, este fator demonstra que apesar da capacidade de reconhecimento ser melhor que a do *OpenFace*, é possível que o algoritmo tenha dificuldade em reproduzir resultados similares.

Por causa de retornos não desejados em uma pesquisa, ajustar a quantidade de falsos positivos é bastante importante.

Nos experimentos feitos com o *OpenFace*, conforme pode ser observado no anexo 01, nota-se que com o crescimento dos valores de corte, aumenta-se a quantidade de falsos positivos. É ideal que ele seja sempre o menor possível, sem comprometer o número de acertos.

Falsos Negativos

Os falsos negativos são as imagens de um indivíduo as quais o algoritmo falhou em reconhecer, ou seja, quando duas fotos comparadas são de uma mesma pessoa, mas o resultado diz que são pessoas diferentes.

Legenda do quadro 15:

- Indivíduo – Número identificador do indivíduo no banco de dados
- Qtde. Imagens – Quantidade de imagens que aquele indivíduo possui na base de dados
- *Face Recognition* – Número de falsos negativos obtidos no teste com a biblioteca *Face Recognition*
- *OpenFace* – Número de falsos positivos negativos no teste com a biblioteca *OpenFace*

Indivíduo	**Qtde Imagens**	***Face Recognition***	***OpenFace***
1	7	-	2
2	4	-	-
3	4	-	-
4	4	-	1
5	4	-	1
6	5	-	1
7	3	-	1
8	4	-	-
9	4	-	-
10	4	-	-
11	5	-	1
12	9	-	2
13	6	-	2
14	6	-	1
15	4	-	-
16	5	-	-
17	6	-	-
18	4	-	-
19	12	-	11
20	8	1	2
21	6	-	2
22	6	-	4
Total:	**120**	**1**	**31**

Quadro 15 – Quantidade de imagens que o algoritmo não foi capaz de reconhecer corretamente
Fonte: Elaboração própria

Os falsos negativos são importantes para medir a quantidade de imagens que não serão reconhecidas e retornadas em uma busca. Observando o quadro 15 é possível perceber que existe uma diferença de reconhecimento entre ambos os algoritmos.

Quanto maior o *threshold* utilizado, menores serão os valores dos falsos negativos. Considerando os três testes feitos com *OpenFace*, este não é o resultado mais baixo, mas, aquele que atingiu os resultados esperados dentro da pré-avaliação.

Nota-se que o valor utilizado pelo *Face Recognition* está no que seria considerado um "ponto ótimo", ou seja, onde o número de reconhecimento é alto e o de falsos negativos, baixo. Contudo, ao fazer uma análise dos resultados apresentados no quadro 12, percebe-se que existe uma enorme quantidade de rostos reconhecidos de maneira errada nos resultados, o que contrabalanceia a ideia de que o valor de tolerância usado para classificação está bom. Portanto, mesmo que o algoritmo atualmente pareça com o *threshold* em um "ponto ótimo", trata-se de uma falsa impressão.

Tempo

A métrica de tempo é bastante importante quando se trata de reconhecimento facial aplicado a um *software*. É preciso saber quanto tempo leva para percorrer a base de dados inteira, reconhecer um rosto ou retornar um erro. O quadro 16 apresenta os resultados de tempo médio aproximado de ambos os testes para cada categoria. Foram geradas 2640 linhas de resultado neste tempo.

Legenda do quadro 16:

- *Face Recognition* – Tempo utilizado em média para apurar cada categoria dos testes com a biblioteca *Face Recognition*.
- *OpenFace* – Tempo utilizado em média para apurar cada categoria dos testes com a biblioteca *OpenFace*.

	Face Recognition	***OpenFace***
Acertos	00:00:01	00:00:01
Falsos Positivos	00:00:01	00:00:01
Falsos negativos	00:00:01	00:00:01
Erros	00:00,8	00:00,9
Percorrer a base	0:01:58	0:02:10
Tempo total de teste	00:43:00	00:48:00

Quadro 16 – Quadro de medição de tempo dos algoritmos
Fonte: Elaboração própria

A diferença de rapidez entre os dois algoritmos é pequena. Levando em consideração que a diferença de tempo - que uma foto de 1 indivíduo demora para percorrer e classificar as 120 imagens da base - é de apenas 12 segundos ao comparar os dois algoritmos, pode-se dizer que ao utilizar bases pequenas, dificilmente o usuário sentirá diferença de entrega de resultado.

Performance

Também foi realizada uma análise performática do algoritmo, levando em consideração dois dos atributos avaliados: acertos e falsos positivos. A precisão do algoritmo é medida pela quantidade de acertos sobre a quantidade de imagens que foram e deveriam ser reconhecidas para aquele indivíduo, ou seja, a soma entre acertos, falsos positivos e falsos negativos. A equação 04 representa a fórmula da precisão.

Descrição das variáveis da equação 4:

- P – Precisão
- a – Quantidade de reconhecimentos corretos
- QR – Quantidade de reconhecimentos esperado e feito (acertos, falsos positivos e falsos negativos).

$$P = \frac{a}{QR}$$

Equação 04 – Fórmula usada para o cálculo de precisão
Fonte: https://revistas.unibh.br/dcet/article/view/1661/932 (Acesso: Jan/2020)

A performance foi medida para cada indivíduo em ambos os algoritmos afim de fazer uma análise comparativa.

O quadro 17 contém as porcentagens de acertos para cada pessoa estudada:

Legenda do quadro 17:

- Indivíduo – Identificação do indivíduo na base de dados
- *Face Recognition* – Porcentagem de acertos dos testes feitos utilizando a biblioteca *Face Recognition*

- *OpenFace* – Porcentagem de acertos dos testes feitos utilizando a biblioteca *OpenFace*

Indivíduo	***Face Recognition***	***OpenFace***
	Precisão	
1	0,538462	0,714286
2	0,5	1
3	1	0,4
4	0,8	0,6
5	1	0,2
6	0,363636	0,6
7	0,375	0,25
8	1	0,6
9	0,75	1
10	0,6	0,75
11	0,625	0,5
12	1	0,75
13	0,833333	0,5
14	1	0,666667
15	0,285714	1
16	0,333333	0,6
17	0,375	0,428571
18	0,5	0,666667
19	0,857143	0,058824
20	0,416667	0,666667
21	1	0,5
22	0,428571	0,333333
Média	**0,662812**	**0,581137**

Quadro 17 – Resultados de precisão de cada algoritmo
Fonte: Elaboração própria

Analisando a performance dos algoritmos, o primeiro ponto importante é a média de precisão dos algoritmos. A média é o resultado geral da performance de cada um e um dos principais fatores de avaliação na hora de decidir qual algoritmo deve ser utilizado. Entretanto, não devem ser esquecidas as análises individuais de cada atributo.

Com base no que foi avaliado anteriormente, nota-se que o grande número de falsos negativos trouxe a média de análise do *OpenFace* para baixo. Do mesmo modo

que, com o *Face Recognition*, os falsos positivos tiveram o mesmo efeito. E??? que conclusão temos?

Erros

Os erros acontecem quando o algoritmo não é capaz de detectar um rosto na imagem, isso é provocado por algum tipo de oclusão. Nenhum dos algoritmos testados tinham modelos voltados para tratamento para coberturas parciais, então eram esperados alguns problemas de detecção.

Legenda do quadro 18:

- Indivíduo – Identificação do indivíduo na base de dados
- *Face Recognition* – Número sequencial que identifica qual foto daquele indivíduo está sendo comparado.
- *OpenFace* – Número sequencial que identifica qual foto daquele indivíduo está sendo comparado.

Indivíduo	*Face Recognition*	*OpenFace*
2	2	2
	3	3
	4	4
3	2	2
	4	4
5	3	3
6	3	3
8	4	4
9	4	4
10	4	4
12	3	3
13	5	5
14	6	6
15	2	2
	4	4
16	3	3
	5	5
17	4	4
	5	5
	6	6

18	2	2
	4	4
20	5	5
	7	7
21	4	4
	6	6
Total	**26**	**26**

Quadro 18 – Imagens com problemas por indivíduo
Fonte: Elaboração própria

Por utilizarem o mesmo módulo de detecção de rostos da Dlib, não é surpresa que os erros encontrados sejam os mesmo em ambos, como pode ser visto no quadro 18. Porém, cabe uma análise sobre quais tipos de erro foram mais proeminentes durante os testes.

Legenda do quadro 19:

- Tipos de oclusão – Todos os tipos de oclusão encontrados na base de dados
- Quantidade – Quantidade de ocorrências de cada erro.

Tipos de oclusão	Quantidade
Chapéu	**2**
Máscara	11
Mãos	4
Ângulo	1
Cabelo	1
Careta	0
Óculos	0
Objetos	0
2 ou mais tipos	7
Total	**26**

Quadro 19 – Tipos de oclusão e quais erros foram encontrados
Fonte: Elaboração própria

A análise foi realizada separando as oclusões individualmente, ou seja, se um indivíduo possuísse mais de um tipo de cobertura, como chapéu e máscara, por exemplo, ele não estaria incluso na categoria chapéu e máscara, apenas na categoria ‘2 ou mais tipos’.

Conforme pode ser observado no quadro 19 a maior ocorrência de erros decorre de usuários utilizando máscaras. Esse tipo de oclusão possui cobertura praticamente

completa da parte inferior do rosto, cobrindo a boca, bochechas, nariz e parte do queixo, como visto na figura 18.

Figura 18 – Rostos cobertos por máscara que não foram reconhecidos
Fonte: http://www.ivl.disco.unimib.it/minisites/umbdb/description.html (Acesso: Ago/2018)

Observando cada erro pela sua quantidade individual, o segundo maior tipo foi gerado pela oclusão utilizando as mãos, cobrindo completamente, ou parcialmente, a lateral do rosto.

Figura 19 – Erros encontrados com oclusão de mão
Fonte: http://www.ivl.disco.unimib.it/minisites/umbdb/description.html (Acesso: Ago/2018)

Percebe-se que não é necessário que haja cobertura completa da lateral do rosto para que haja problemas de reconhecimento. A adição de um elemento "estranho" confunde o algoritmo, fazendo-o, como no exemplo da Figura 19, reconhecer a mão como se fosse um elemento que pertencesse naturalmente àquela posição. Ao comparar com o modelo, ele não verá semelhanças, uma vez que para ele se trata de objetos diferentes, como dois moldes que não se encaixam.

Os outros tipos de erro possuem uma incidência muito pequena ou são inconclusivos, uma vez que é difícil compreender as razões que causaram a oclusão. Outras imagens com os erros estão no anexo 02.

6.2. Análise

Devido a limitações de *hardware* este trabalho não pode ser implementado seguindo um modelo ótimo. Os resultados foram prejudicados pela falta de uma criação de modelo específico para casos de oclusão parcial, o que pode ter impactado negativamente questões como acurácia e quantidade de erros.

O *Face Recognition* possui a melhor performance nos resultados avaliados, como apresentado na seção anterior, podendo tornar o resultado dos reconhecimentos mais restritos ou abrangentes ao variar o *threshold*. Ao atribuir um valor menor, reduz-se o número de falsos positivos, porém, sempre considerando que isto também pode causar a diminuição da quantidade de rostos reconhecidos e aumentar a quantidade de falsos negativos.

Considera-se que este trabalho possui a intenção de testar algoritmos para serem implementados como mecanismos de busca. Portanto, a quantidade de falsos negativos possui uma relevância maior em análise, pois, significa que, em uma busca, o rosto procurado não seria reconhecido como o do indivíduo que deve ser localizado.

Uma pequena parcela de falsos positivos é ideal, mas não é crítica se os números de resultados retornados forem considerados baixos. Eles significam que apesar de retornar o rosto que você está procurando, ele também trará de pessoas diferentes, as quais ele considerou que fossem a mesma procurada. As avaliações mostram que, o maior indivíduo com falsos positivos, entre os dois algoritmos, possui apenas 8 reconhecimentos errados. Assumindo que apenas uma das imagens corretas retornará em uma busca, pois o banco de dados possuirá apenas uma imagem de cada indivíduo armazenada, o resultado será de 9 rostos, o que ainda é um número baixo em uma busca em um aplicativo.

Por utilizarem o mesmo detector de faces, não foi uma surpresa que houvesse os mesmos erros.

Se forem utilizados seguindo os mesmos moldes deste trabalho, é possível que exista complicações para reconhecer pessoas com curativos em pontos importantes do rosto, como: olhos, nariz, bocas e/ou bochechas. Ou, por exemplo, utilizando máscaras para auxílio respiratório.

As tabelas completas de cada algoritmo estão disponíveis no anexo 01 para fins de verificação. No próximo capítulo, serão apresentadas considerações finais do trabalho e quais os trabalhos futuros que poderiam ser feitos para melhorar o resultado dos testes.

7 Conclusão

Com a finalidade de encontrar uma biblioteca de reconhecimento facial para ser implementada em uma aplicação móvel para reconhecimento de pessoas desaparecidas, este trabalho se propôs a estudar a performance de duas bibliotecas disponíveis no GitHub, *OpenFace* e *Face Recognition*, em casos voltados a rostos com oclusão parcial.

As bibliotecas foram testadas sem quaisquer modificações além do valor do *threshold* e sem a criação de um modelo especializado em rostos parcialmente cobertos. Por isso, os resultados apresentados não tiveram uma performance que pudesse ser considerado ótima.

Com os resultados apresentados no quadro 20, percebe-se que *Face Recognition*, apesar de ter um alto número de acertos, também possui uma grande quantidade de falsos positivos. Enquanto o *OpenFace* e a variação de valores utilizados no *threshold* também não obtiveram o melhor desempenho. Quando número de acertos estava bom, os valores de falsos negativos e positivos estavam baixos e vice-versa.

	Face Recognition	***Open Face 0.5***	***Open Face 0.6***	***Open Face 0.9292***
Acertos	93	57	63	82
Falsos positivos	61	13	37	318
Falsos negativos	1	37	31	12
Erros	26	26	26	26

Quadro 20 - Resumo de resultados dos testes
Fonte: Elaboração própria

Levando-se em consideração as condições não ideais dos testes, a biblioteca *Face Recognition* possuiu uma performance levemente superior dentre as duas testadas, mesmo que ainda apenas com uma taxa de 77,5% de acerto.

Para o funcionamento ideal seria necessário realizar novos testes com um método de detecção de rostos alternativo que está disponível na própria Dlib. Ele se utiliza de técnicas de inteligência artificial que aumentam a capacidade de detecção e previne que haja um número de erros tão grande quanto os encontrados nos resultados atuais. Entretanto, por se tratar de um método que necessita de bastante poder computacional é recomendado a utilização de um ambiente em nuvem pela facilidade de configuração e

sua escalabilidade, ainda mais quando levado em consideração a implantação deste método em um aplicativo móvel.

Deste modo, conclui-se que existe uma eficiência parcial destes algoritmos para o que foi proposto. Em ocasiões com pequenas oclusões, não haverá problemas, pois nenhum dos dois algoritmos tiveram dificuldade para reconhecê-los. Porém, as grandes coberturas atrapalharam o reconhecimento de alguns rostos, mesmo que não todos, e poderão causar transtornos para os usuários do aplicativo de vez em quando.

Portanto, por possuir um número maior de acertos e menor de falsos negativos, ainda que com o mesmo número de erros, o *Face Recognition* foi considerado o melhor dentre os dois algoritmos testados, nas condições apresentadas. Desta forma, ele é o mais recomendado para ser implementado no aplicativo Desaparecidos – RJ.

Trabalhos futuros

Para trabalhos futuros é interessante refazer os testes levando-se em consideração as seguintes sugestões:

- Modelo: seria interessante fazer um modelo voltado para casos com oclusão para conferir se há alguma mudança positiva nos resultados.
- Detecção facial: A biblioteca Dlib possui dois tipos de detectores de rosto, este teste foi feito utilizando apenas um deles para ambos os algoritmos. Logo, é interessante tentar uma outra abordagem para conferir se os erros encontrados permanecem os mesmos.
- Base de dados: alguns erros encontrados se repetiram em fotos que não acusaram nenhum tipo de problema. Logo, para entender quais tipos de oclusão realmente causaram os erros, é interessante que os testes sejam refeitos utilizando uma nova base de dados.

Referências Bibliográficas

A. Ponti, M. and B. Paranhos da Costa, G. (2017). Como funciona o Deep Learning. In: V. Vieira, H. L. Razente and M. N. Barioni, ed., **TÓPICOS EM GERENCIAMENTO DE DADOS E INFORMAÇÕES 2017**, 1st ed. Uberlândia: Sociedade Brasileira de Computação – SBC, pp.63-88.

ALBUQUERQUE, M. P. ; ALBUQUERQUE, Marcelo Portes de . **Processamento de imagens: métodos e análises**. Rio de Janeiro: Editora da FACET, 2001 (Revista Científica).

B. Amos, B. Ludwiczuk, M. Satyanarayanan, "**Openface: A general-purpose face recognition library with mobile applications**," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

BEHAM, M. Parisa; ROOMI, S. Mohamed Mansoor. A REVIEW OF FACE RECOGNITION METHODS. **International Journal Of Pattern Recognition And Artificial Intelligence**, [s.l.], v. 27, n. 04, p. 1356005, jun. 2013. World Scientific Pub Co Pte Lt. http://dx.doi.org/10.1142/s0218001413560053.

BISSI, Thelry David. **Reconhecimento Facial com os algoritmos Eigenfaces e Fisherfaces.** 2018. 41 f. TCC (Graduação) - Curso de Bacharelado em Ciência da Computação, Universidade Federal de Uberlândia, Uberlândia, 2018.

BONFÁ, Cizenando Morello. **Um Sistema de Reconhecimento Facial em Vídeo Baseado em uma Implementação Multithread do Algoritmo TLD**. 2013. 102 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2013.

BRAGA, Luiz Filipe Zenicola. **Sistemas de reconhecimento facial**. 2013. 84 f. TCC (Graduação) - Curso de Curso de Engenharia Elétrica Com ênfase em Eletrônica, Universidade de São Paulo, São Carlos, 2013.

COLOMBO, C. Cusano, and R. Schettini, "**UMB-DB: A Database of Partially Occluded 3D Faces**," in *in Proc. ICCV 2011 Workshops*, pp. 2113-2119, 2011.

CONCEIÇÃO, Valdir Silva *et al*. O Reconhecimento Facial como uma das Vertentes da Inteligência Artificial (IA): um estudo de prospecção tecnológica. **Cadernos de Prospecção**, Salvador, v. 3, n. 3, p. 1-14, jun. 2020.

CORAL, Sergio et al. Estudo de caso de um Sistema de reconhecimento facial utilizando o OpenFace para identificação de faces em bancos de imagens. **Revista de Sistemas e Computação**, Salvador, v. 1, n. 10, p. 4-11, jan. 2020.

Data Science Academy. Deep Learning Book, 2019. Disponível em: <http://www.deeplearningbook.com.br/>. Acesso em: 10 janeiro. 2020.

De Lemos Ferreira Casimiro da Costa, B. (2018). **Practical Face Recognition: Building a Complete System Able to Identify Subjects in Real Time**. Mestre. Universidade do Porto.

DEFREITAS JURASZEK, G. (2014**). Reconhecimento De Produtos Por Imagem Utilizando Palavras Visuais E Redes Neurais Convolucionais**. Mestre. Universidade Do Estado De Santa Catarina – UDESC.

FACURE, Matheus. **Funções de Ativação**: entendendo a importância da ativação correta nas redes neurais.. Entendendo a importância da ativação correta nas redes neurais. Disponível em: https://matheusfacure.github.io/2017/07/12/activ-func/. Acesso em: 28 jan. 2019.

Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. **Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.** *University of Massachusetts, Amherst, Technical Report 07-49*, October, 2007.

JACKSON, Aaron S. et al. **3D Face Reconstruction from a Single Image.** 2017. Disponível em: <https://cvl-demos.cs.nott.ac.uk/vrn/>. Acesso em: 07 set. 2017.

KING, Davis E**.. Dlib-ml: A Machine Learning Toolkit**. Journal Of Machine Learning Research, Baltimore, v. 10, n. 10, p.1755-1758, jul. 09.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. **Convolutional networks and applications in vision. In: Circuits and Systems (ISCAS)**, Proceedings of 2010 IEEE International Symposium on. [S.l.: s.n.], 2010. p. 253–256.

LIMA, Vinícius Rodrigues. **UM SISTEMA DE INFORMAÇÃO PARA APOIO À BUSCA DE PESSOAS DESAPARECIDAS NO RIO DE JANEIRO**. 2016. 69 f. TCC (Graduação) - Curso de Bacharelado em Sistemas de Informação, Instituto de Informática, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2016.

LOPES, João Fabiano Martucci. **Governo Orientado por Dados.** 2018. 131 f. Tese (Doutorado) - Curso de Programa de Educação Continuada, Escola politécnica, Universidade de São Paulo, São Paulo, 2018.

MARENGONI, Maurício; STRINGHINI, Denise. Tutorial: Introdução à Visão Computacional usando OpenCV. **Revista de Informática Teórica Aplicada**, Porto Alegre, v. 01, n. 16, p. 01-36, jan. 2009.

MATTOS, Guilherme Caeiro de. **PresentEye: Sistema de Controle de Presença por Reconhecimento Facial.** 2017. 72 f. TCC (Graduação) - Curso de Bacharelado em Sistemas de Informação, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

MOLZ, Rolf Fredi. **Uma Metodologia para o desenvolvimento de aplicações de visão computacional utilizando um projeto conjunto de hardware e software.** 2001. 80 f. Tese (Doutorado) - Curso de Programa de Pós-graduação em Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2001.

Naeem, Muhammad & Qureshi, Imran & Azam, Faisal. (2015). FACE RECOGNITION TECHNIQUES AND APPROACHES: A SURVEY. Science International (Lahore). 27. 301-305.

QUEIROZ, J. C. L. ; SEGUNDO, M. P. . **Análise da acurácia de métodos de descrição 2D para o reconhecimento facial 3D**. 2015. (Apresentação de Trabalho/Conferência ou palestra).

RESENDE, Claudio Augusto de Paulo; PEREIRA, Moisés Henrique Ramos. VISÃO COMPUTACIONAL APLICADA EM RECONHECIMENTO FACIAL NA BUSCA POR PESSOAS DESAPARECIDAS. **Revista E-xacta**, Belo Horizonte, v. 2, n. 8, p. 95-107, nov. 2015.

SANTOS, Liliane Ribeiro dos *et al*. Reconhecimento de Face Aplicada ao Controle de Chamada de Classe. **E-rac**, Santos, v. 8, n. 1, p. 1-18, jan. 2018.

SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. FaceNet: a unified embedding for face recognition and clustering. : A unified embedding for face recognition and clustering. **2015 Ieee Conference On Computer Vision And Pattern Recognition (cvpr)**, [s.l.], v. 3, jun. 2015. IEEE. http://dx.doi.org/10.1109/cvpr.2015.7298682.

SILVA, Alex Lima; CINTRA, Marcos Evandro. **Reconhecimento de padrões faciais: Um estudo**. 2015. 8 f. - Universidade Federal Rural do Semi-Árido, Mossoró, 2015.

STROSKI, Pedro Ney. **O que são redes neurais convolucionais?** 2018. Disponível em: http://www.electricalelibrary.com/2018/11/20/o-que-sao-redes-neurais-convolucionais/. Acesso em: 28 jan. 2020.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. **Um Estudo sobre Redes Neurais Convolucionais e sua Aplicação em Detecção de Pedestres**. In: CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES, 29. (SIBGRAPI), 2016, São José dos Campos. **Proceedings...** Porto Alegre: Sociedade Brasileira de Computação, 2016. On-line. IBI: <8JMKD3MGPAW/3ME3L2P>. Available from: <http://urlib.net/rep/8JMKD3MGPAW/3ME3L2P>.

Anexo 01 – Quadros de avaliação

Face Recognition

Pessoa comparada	Qtde imagens	Acertos		Falsos Positivos		Falsos negativos		Erros*		Tempo total** (minuto/pessoa)
		Qtde	Tempo médio	Qtde	Tempo médio	Qtde	Tempo médio	Qtde	Tempo médio	
1	7	7	00:00:01	6	00:00:01	-	-	-	-	0:01:58
2	4	1	00:00:01	1	00:00:01	-	-	3	00:00,8	0:01:58
3	4	2	00:00:01	-	-	-	-	2	00:00,8	0:01:58
4	4	4	00:00:01	1	00:00:01	-	-	-	-	0:01:58
5	4	3	00:00:01	-	-	-	-	1	00:00,8	0:01:58
6	5	4	00:00:01	7	00:00:01	-	-	1	00:00,8	0:01:58
7	3	3	00:00:01	5	00:00:01	-	-	-	-	0:01:58
8	4	3	00:00:01	-	-	-	-	1	00:00,8	0:01:58
9	4	3	00:00:01	1	00:00:01	-	-	1	00:00,8	0:01:58
10	4	3	00:00:01	2	00:00:01	-	-	1	00:00,8	0:01:58
11	5	5	00:00:01	3	00:00:01	-	-	-	-	0:01:58
12	9	8	00:00:01	-	-	-	-	1	00:00,8	0:01:58
13	6	5	00:00:01	1				1	00:00,8	0:01:58
14	6	5	00:00:01	-	-	-	-	1	00:00,8	0:01:58
15	4	2	00:00:01	5	00:00:01	-	-	2	00:00,8	0:01:58
16	5	3	00:00:01	6	00:00:01	-	-	2	00:00,8	0:01:58
17	6	3	00:00:01	5	00:00:01	-	-	3	00:00,8	0:01:58
18	4	2	00:00:01	2	00:00:01			2	00:00,8	0:01:58
19	12	12	00:00:01	2	00:00:01	-	-	-	-	0:01:58
20	8	5	00:00:01	6	00:00:01	1	00:00:01	2	00:00,8	0:01:58
21	6	4	00:00:01	-	-	-	-	2	00:00,8	0:01:58
22	6	6	00:00:01	8	00:00:01	-	-	-	-	0:01:58
Total:	**120**	**93**	**00:00:01**	**61**	**00:00:01**	**1**	**00:00:01**	**26**	**00:00,8**	**00:43:16**

*** Em comparação com ela mesma**

*** * 120 imagens/indivíduo**

*** * valor aproximado**

OpenFace

0.5

Pessoa comparada	Qtde Imagens	Acertos		Falsos Positivos		Falsos negativos		Erros*		Tempo total** (minuto/pessoa)
		Qtde	Tempo médio	Qtde	Tempo médio	Qtde	Tempo médio	Qtde	Tempo médio	
1	7	4	00:00:01	0	00:00:01	3	00:00:01	-	-	00:02:10
2	4	1	00:00:01	0	00:00:01	0	00:00:01	3	00:00:01	00:02:10
3	4	2	00:00:01	1	00:00:01	0	00:00:01	2	00:00:01	00:02:10
4	4	3	00:00:01	0	00:00:01	1	00:00:01	-	-	00:02:10
5	4	2	00:00:01	1	00:00:01	1	00:00:01	1	00:00:01	00:02:10
6	5	3	00:00:01	0	00:00:01	1	00:00:01	1	00:00:01	00:02:10
7	3	2	00:00:01	2	00:00:01	1	00:00:01	-	-	00:02:10
8	4	3	00:00:01	0	00:00:01	0	00:00:01	1	00:00:01	00:02:10
9	4	3	00:00:01	0	00:00:01	0	00:00:01	1	00:00:01	00:02:10
10	4	3	00:00:01	1	00:00:01	0	00:00:01	1	00:00:01	00:02:10
11	5	3	00:00:01	1	00:00:01	2	00:00:01	-	-	00:02:10
12	9	5	00:00:01	0	00:00:01	3	00:00:01	1	00:00:01	00:02:13
13	6	3	00:00:01	0	00:00:01	2	00:00:01	1	00:00:01	00:02:10
14	6	4	00:00:01	0	00:00:01	1	00:00:01	1	00:00:01	00:02:09
15	4	2	00:00:01	0	00:00:01	0	00:00:01	2	00:00:01	00:02:09
16	5	3	00:00:01	0	00:00:01	0	00:00:01	2	00:00:01	00:02:10
17	6	2	00:00:01	3	00:00:01	1	00:00:01	3	00:00:01	00:02:09
18	4	2	00:00:01	1	00:00:01	0	00:00:01	2	00:00:01	00:02:10
19	12	1	00:00:01	3	00:00:01	11	00:00:01	-	-	00:02:11
20	8	2	00:00:04	0	00:00:01	4	00:00:01	2	00:00:01	00:02:21
21	6	2	00:00:01	0	00:00:01	2	00:00:01	2	00:00:01	00:02:19
22	6	2	00:00:01	0	00:00:01	4	00:00:01	-	-	00:02:13

*** Em comparação com ela mesma**

*** * 120 imagens/indivíduo**

*** * valor aproximado**

Total:	120	57	00:00:01	13	00:00:01	37	00:00:01	26	0:00:01	00:48:04

0.6

Pessoa comparada	Qtde Imagens	Acertos		Falsos Positivos		Falsos negativos		Erros*		Tempo total** (minuto/pessoa)
		Qtde	Tempo médio	Qtde	Tempo médio	Qtde	Tempo médio	Qtde	Tempo médio	
1	7	5	00:00:01	-	-	2	00:00:01	-	-	00:02:10
2	4	1	00:00:01	-	-	-	-	3	00:00,9	00:02:10
3	4	2	00:00:01	3	00:00:01	-	-	2	00:00,9	00:02:10
4	4	3	00:00:01	1	00:00:01	1	00:00:01	-	-	00:02:10
5	4	2	00:00:01	7	00:00:01	1	00:00:01	1	00:00,9	00:02:10
6	5	3	00:00:01	1	00:00:01	1	00:00:01	1	00:00,9	00:02:10
7	3	2	00:00:01	5	00:00:01	1	00:00:01	-	-	00:02:10
8	4	3	00:00:01	2	00:00:01	-	-	1	00:00,9	00:02:10
9	4	3	00:00:01	-	-	-	-	1	00:00,9	00:02:10
10	4	3	00:00:01	1	00:00:01	-	-	1	00:00,9	00:02:10
11	5	4	00:00:01	3	00:00:01	1	00:00:01	-	-	00:02:10
12	9	6	00:00:01	-	-	2	00:00:01	1	00:00,9	00:02:13
13	6	3	00:00:01	1	00:00:01	2	00:00:01	1	00:00,9	00:02:10
14	6	4	00:00:01	1	00:00:01	1	00:00:01	1	00:00,9	00:02:09
15	4	2	00:00:01	-	-	-	-	2	00:00,9	00:02:09
16	5	3	00:00:01	2	00:00:01	-	-	2	00:00,9	00:02:10
17	6	3	00:00:01	4	00:00:01	-	-	3	00:00,9	00:02:09
18	4	2	00:00:01	1	00:00:01	-	-	2	00:00,9	00:02:10
19	12	1	00:00:01	5	00:00:01	11	00:00:01	-	-	00:02:11
20	8	4	00:00:01	-	-	2	00:00:01	2	00:00:01	00:02:21
21	6	2	00:00:01	-	-	2	00:00:01	2	00:00:01	00:02:19
22	6	2	00:00:01	-	-	4	00:00:01	-	-	00:02:13

*** Em comparação com ela mesma**

*** * 120 imagens/indivíduo**

*** * valor aproximado**

Total:	120	63	00:00:01	37	00:00:01	31	00:00:01	26	0:00:01	00:48:04

0.9292

Pessoa comparada	Qtde Imagens	Acertos		Falsos Positivos		Falsos negativos		Erros*		Tempo total** (minuto/pessoa)
		Qtde	Tempo médio	Qtde	Tempo médio	Qtde	Tempo médio	Qtde	Tempo médio	
1	7	5	00:00:01	7	00:00:01	2	00:00:01	-	-	00:00:03
2	4	1	00:00:01	22	00:00:01	0	-	3	00:00:01	00:00:03
3	4	2	00:00:01	21	00:00:01	0	-	2	00:00:01	00:00:03
4	4	3	00:00:01	18	00:00:01	1	00:00:01	-	-	00:00:03
5	4	3	00:00:01	28	00:00:01	0	-	1	00:00:01	00:00:03
6	5	3	00:00:01	7	00:00:01	1	00:00:01	1	00:00:01	00:00:04
7	3	3	00:00:01	34	00:00:01	0	-	-	-	00:00:02
8	4	3	00:00:01	8	00:00:01	0	-	1	00:00:01	00:00:03
9	4	3	00:00:01	8	00:00:01	0	-	1	00:00:01	00:00:03
10	4	3	00:00:01	3	00:00:01	0	-	1	00:00:01	00:00:03
11	5	5	00:00:01	14	00:00:01	0	-	-	-	00:00:02
12	9	8	00:00:01	2	00:00:01	0	-	1	00:00:01	00:00:03
13	6	4	00:00:01	22	00:00:01	1	00:00:01	1	00:00:01	00:00:04
14	6	5	00:00:01	3	00:00:01	0	-	1	00:00:01	00:00:03
15	4	2	00:00:01	10	00:00:01	0	-	2	00:00:01	00:00:03
16	5	3	00:00:01	33	00:00:01	0	-	2	00:00:01	00:00:03
17	6	3	00:00:01	18	00:00:01	0	-	3	00:00:01	00:00:03
18	4	2	00:00:01	0	00:00:01	0	-	2	00:00:01	00:00:03
19	12	7	00:00:01	24	00:00:01	5	00:00:01	-	-	00:00:03
20	8	6	00:00:01	18	00:00:01	0	00:00:01	2	00:00:01	00:00:04

*** Em comparação com ela mesma**

*** * 120 imagens/indivíduo**

*** * valor aproximado**

21	6	4	00:00:01	4	00:00:01	0	00:00:01	2	00:00:01	00:00:04
22	6	4	00:00:01	14	00:00:01	2	00:00:01	-	-	00:00:03
Total:	**120**	**82**	**00:00:01**	**318**	**00:00:01**	**12**	**00:00:01**	**26**	**0:00:01**	**00:01:09**

Anexo 02 – Oclusões com erro

Indivíduo: 2 Identificador: 2

Tipo de oclusão: 2 ou mais tipos

Indivíduo: 2 Identificador: 3

Tipo de oclusão: 2 ou mais tipos

Indivíduo: 2 Identificador: 4

Tipo de oclusão: Máscara

Indivíduo: 3 Identificador: 2

Tipo de oclusão: Ângulo

Indivíduo: 3 Identificador: 3

Tipo de oclusão: Máscara

Indivíduo: 5 Identificador: 3

Tipo de oclusão: Máscara

Indivíduo: 6 Identificador: 3

Tipo de oclusão: Máscara

Indivíduo: 8 Identificador: 4

Tipo de oclusão: Máscara

Indivíduo: 9 Identificador: 4

Tipo de oclusão: 2 ou mais tipos

Indivíduo: 10 Identificador: 4

Tipo de oclusão: Cabelo

Indivíduo: 12 Identificador: 4

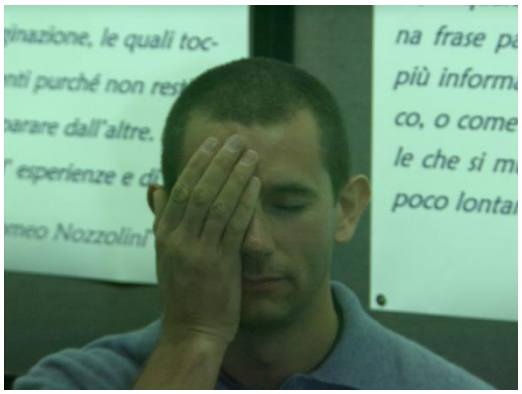
Tipo de oclusão: Mãos

Indivíduo: 13 Identificador: 5

Tipo de oclusão: Máscara

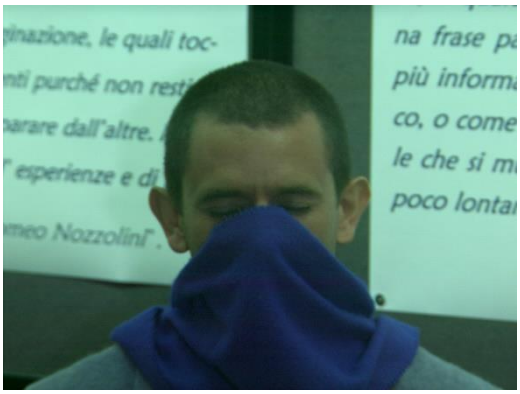
Indivíduo: 14 Identificador: 6

Tipo de oclusão: Máscara



Indivíduo: 15 Identificador: 2

Tipo de oclusão: Mãos



Indivíduo: 15 Identificador: 4

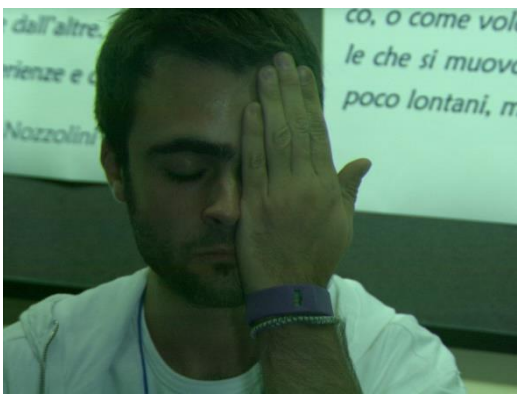
Tipo de oclusão: Máscara

Indivíduo: 16 Identificador: 3

Tipo de oclusão: Mãos

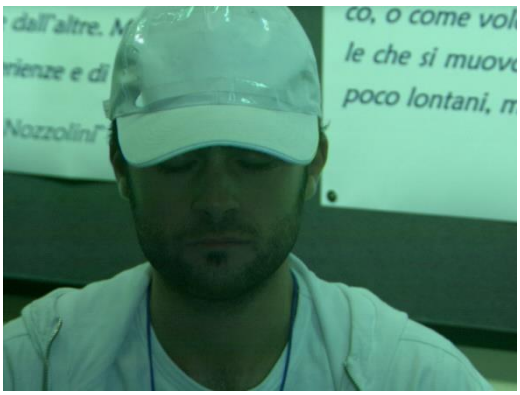
Indivíduo: 16 Identificador: 5

Tipo de oclusão: Máscara



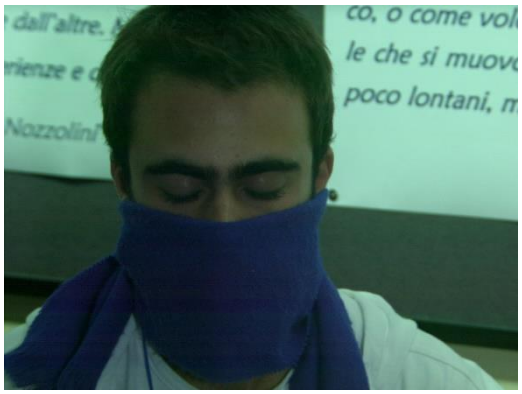
Indivíduo: 17 Identificador: 4

Tipo de oclusão: Mãos



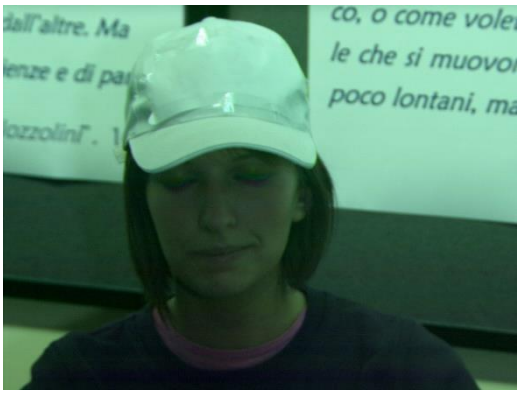
Indivíduo: 17 Identificador: 5

Tipo de oclusão: Chapéu



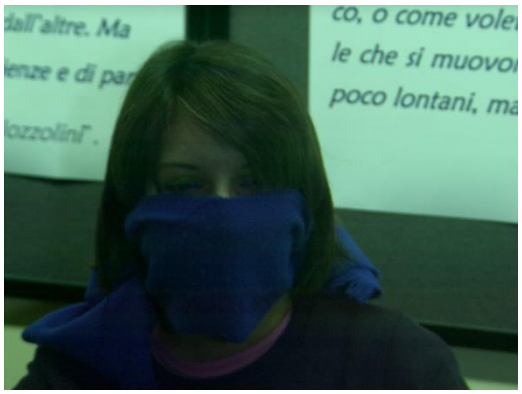
Indivíduo: 17 Identificador: 6

Tipo de oclusão: Máscara



Indivíduo: 18 Identificador: 2

Tipo de oclusão: Chapéu



Indivíduo: 18 Identificador: 4

Tipo de oclusão: 2 ou mais tipos

Indivíduo: 20 Identificador: 5

Tipo de oclusão: 2 ou mais tipos

Indivíduo: 20 Identificador: 7

Tipo de oclusão: 2 ou mais tipos

Indivíduo: 21 Identificador: 4

Tipo de oclusão: Máscara

Indivíduo: 21 Identificador: 6

Tipo de oclusão: 2 ou mais tipos