UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS GRADUAÇÃO EM INFORMÁTICA

FELIPE CARMO CRISPIM

Reconhecimento Facial RGBD para Análise de Parentesco

Maceió-AL

Março de 2020

FELIPE CARMO CRISPIM

Reconhecimento Facial RGBD para Análise de Parentesco

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal de Alagoas.

Orientador: Tiago Figueiredo Vieira

Maceió-AL

Março de 2020

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

C932r Crispim, Felipe Carmo.
Reconhecimento facial RGBD para análise de parentesco / Felipe Carmo Crispim. – 2020.
72 f. : il.

Orientador: Tiago Figueiredo Vieira.
Dissertação (mestrado em Informática) - Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2020.

Bibliografia: f. 66-71.
Anexo: f. 72-73.

1. Parentesco - Percepção facial. 2. Biometria - Percepção facial. 3. Movimento de câmara. 4. Reconhecimento facial. I. Título.

CDU: 004.89

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Informática – PpgI
Instituto de Computação
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins
Maceió/AL – Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

Folha de Aprovação

Felipe Carmo Crispim

"Reconhecimento Facial RGBD para Análise de Parentesco."

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas e aprovada em 13 de março de 2020.

Banca Examinadora:

Prof. Dr. Tiago Figueiredo Vieira
Programa de Pós-graduação em Informática – UFAL
Orientador

Prof. Dr. Thales Miranda de Almeida Vieira
Programa de Pós-graduação em Informática – UFAL
Co-Orientador

Prof. Dr. Douglas Cedrim Oliveira
Instituto Federal Goiano
Examinador Externo

AGRADECIMENTOS

Agradeço primeiramente a Deus, que sempre me proporciona o que preciso e é meu maior guia.

Agradeço aos meus pais e irmãos, cujo empenho e carinho se traduzem nas minhas conquistas e sempre apoiam acreditam em mim, em todos os momentos.

A todos os meus amigos que sempre estão dispostos a me ajudar, pessoal e profissionalmente, mesmo os que moram distante.

Agradeço a todos os professores e orientadores por todos os conselhos, paciência e suporte que me ajudaram à trilhar o caminho que eu percorri na universidade.

À FAPEAL pelo apoio financeiro para realização deste trabalho de pesquisa e à UFAL pela infraestrutura fornecida.

“Lembre-se que as pessoas podem tirar tudo de você, menos o seu conhecimento.”
(Albert Einstein)

RESUMO

Este trabalho apresenta uma abordagem inédita de reconhecimento de parentesco baseada em Aprendizado Profundo aplicado a dados faciais de imagens coloridas e com informação de profundidade, i. e., RGBD. Para contornar a falta de uma base de dados 3D adequada com informações de parentesco, foi fornecida uma plataforma online onde os participantes podem submeter vídeos capturados com câmeras de *smartphones* comuns contendo a sua face e as de seus parentes. Em seguida, os vídeos são processados para a reconstrução 3D das faces gravadas, gerando um banco de dados normalizado batizado Kin3D. Nele, combinam-se informações de profundidade de reconstruções 3D normalizadas com imagens 2D, compondo o banco de dados RGBD de parentesco inédito na literatura. Seguindo as abordagens de trabalhos relacionados, imagens são organizadas em quatro categorias de acordo com suas respectivas relações de parentesco. Para a classificação foram utilizadas Redes Neurais Convolucionais (CNN) bem como Máquina de Vetores de Suporte para a obtenção de um *baseline*. A CNN foi testada em um banco de dados de parentesco 2D previamente consolidado na literatura científica, conhecido como KinFaceW-I e II, e em nosso Kin3D para comparação com trabalhos relacionados. Uma outra abordagem foi usada ao reunir todos os parentes de primeiro grau de uma vez e classificá-los de maneira binária. Resultados indicam que a adição de informação de profundidade aprimora a performance do modelo, aumentando a acurácia de classificação. Até o momento da escrita desse trabalho, este é o primeiro banco de dados contendo informação de profundidade para verificação de parentesco bem como a análise de técnicas do estado da arte para a obtenção do *benchmark*, fornecendo uma performance como ponto de partida para estimular ainda mais avaliações da comunidade de pesquisa.

Palavras-chaves: Verificação de Parentesco, Biometria facial, Estrutura a partir do Movimento, Reconstrução 3D.

ABSTRACT

This work presents a new approach to kinship recognition based on Deep Learning applied to facial data of color images with depth information, i. e., RGBD. To work around the lack of an adequate 3D database containing kinship information, an online platform was provided where participants can submit videos captured by common *smartphones* cameras containing their face and those of their relatives. Then, the videos are processed to generate the 3D reconstruction of recorded faces, resulting in a standardized database coined Kin3D. It combines depth information from normalized 3D reconstructions with 2D images comprising RGBD data with unprecedent kinship information. Following previous works, image files are segmented into four categories according to their respective kinship relationship. For the classification, Convolutional Neural Networks (CNN) were used, as well as a Support Vector Machines (SVM) to obtain a *baseline*. The CNN was tested in a 2D kinship database previously consolidated in the scientific literature, known as KinFaceW-I and II, and in our Kin3D for comparison with related works. Another approach was used by bringing all first-degree relatives together at once and classifying them in a binary way. Results have shown that the addition of depth information improves the performance of the model, increasing the classification accuracy. As of the writing of this work, this is the first database containing depth information for kinship verification as well as the analysis of state-of-the-art techniques for obtaining the *benchmark*, providing performance as a starting point to further stimulate evaluations from the research community.

Keywords: Kinship verification, Face Biometrics, Structure from Motion, 3D Reconstruction.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SFM	Structure From Motion
RGB	Red-Green-Blue
RGBD	Red-Green-Blue-Depth
CNN	Convolutional Neural Netowrks
SVM	Support Vector Machine
SGD	Stochastic Gradient Descent
VP	Verificação de Parentesco
PFo	Pai-Filho
PFa	Pai-Filha
MFo	Mãe-Filho
MFa	Mãe-Filha

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta o principal contexto sobre o desenvolvimento dessa dissertação, ao destacar as possibilidades de verificação de parentesco em visão computacional. Em seguida também é apresentada a abordagem que esta dissertação propõe para contribuir para pesquisas nessa área.

1.1 Motivação

O avanço em *hardware* e *software* das câmeras fotográficas e de *smartphones* juntamente com a redução de custo desses dispositivos possibilitaram a obtenção de fotos e vídeos em quantidades e qualidades superiores. Rehm (2018) mostra através de testes como as câmeras de *smartphones* evoluíram, considerando alguns critérios como: relação sinal-ruído, estabilização de imagem e vídeo, exposição, tons de pele, *bokeh*, entre outros. Como Kriss (2015) explica, esse avanço tecnológico tem impulsionado pesquisas na área de processamento de imagens e visão computacional e, uma vez que estão mais acessíveis, também a criação de bases de dados mais numerosas, por exemplo, usando imagens de rostos.

A verificação de parentesco (VP) via biometria tem como objetivo determinar se duas pessoas têm uma relação de parentesco a partir de imagens dos seus rostos. Pesquisas em psicologia (ALVERGNE et al., 2014; YAN; LU, 2017) têm demonstrado que é uma prática comum para seres humanos visualmente identificar parentes a partir dos rostos. A aparência facial fornece características suficientes para identificar pessoas e consequentemente seus parentes. Isso acontece porque, normalmente, pessoas que possuem um relacionamento biológico carregam mais similaridades do que pessoas que não possuem tal relacionamento.

As aplicações de VP em visão computacional podem ser encontradas quando se deseja (DUAN; ZHANG; ZUO, 2017): descobrir relações sociais humanas, encontrar crianças/pais desaparecidos, anotar imagens, criar uma árvore genealógica, etc. Ao descobrir as características chaves de parentesco, também é possível realizar operações como envelhecimento (SHU et al., 2016) e geração de filhos a partir de pais (Ertugrul; Dibeklioglu, 2017).

Yan e Lu (2017) afirmam que o uso do método de comparação de DNA (*Deoxyribonucleic Acid*) é os mais comum e preciso para validar a relação de parentesco entre duas pessoas. Contudo essa abordagem apresenta algumas limitações: (i) o custo do teste de DNA é alto e o laudo pode durar vários dias; (ii) os parentes que seriam submetidos ao teste podem não residir na mesma região, tornando o processo mais complicado.

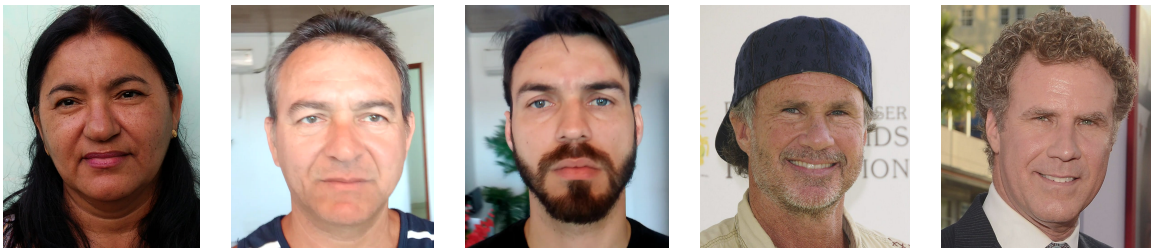
A VP usando imagens faciais pode contornar as desvantagens apresentadas pelo DNA (YAN; LU, 2017). Na tarefa de encontrar uma criança desaparecida dentre milhares, VP usando imagens poderia rapidamente identificar possíveis candidatos que possuem alta similaridade

facial. Depois disso, o teste de DNA pode ser aplicado em conjunto para obter uma validação mais precisa dessa pesquisa. Essas e outras aplicações estão levando um grande progresso através de pesquisadores nessa área como Vieira, Bottino e Islam (2013), Xu e Shang (2016) e Tidjani et al. (2018).

1.2 Justificativa

As técnicas disponíveis de reconhecimento facial e parental em 2D possuem performance satisfatória quando implementadas sob condições razoavelmente controladas durante treinamento e teste. Contudo, como explicam Guo, Ma e Lan (2018), muitas dificuldades surgem ao trabalhar com imagens faciais em cenários não controlados[1], pois a aquisição das imagens faciais podem sofrer influência por conta da iluminação, pose, expressão e *background*. Também existem variações nas categorias de gênero, idade e etnia. Lopez et al. (2018) citam que comparado com verificação facial, verificação de parentesco é ainda mais desafiante porque deve lidar com semelhanças entre não-parentes (o que pode reduzir a distância de similaridade entre classes) enquanto lida com parentes com aparências diferentes (que podem aumentar a distância intra classe), como ilustra a Figura 1, e ainda existe a possibilidade de meio-irmãos. Ao ter isso em mente, pode-se perceber que verificação de parentesco a partir de imagens de faces é um tópico de pesquisa amplamente aberto em visão computacional e biometria (GUO; MA; LAN, 2018).

Figura 1 – Desafios relacionados à Verificação de Parentesco - Pessoas em (a) e (b) são gêmeas (irmã e irmão, respectivamente). Indivíduos nas imagens (b) e (c) são pai e filho, respectivamente. Por outro lado, pessoas não relacionadas podem apresentar similaridades nas faces como pode ser visto nas semelhanças entre os atores Chad Smith (d) e Will Ferrell (e).



(a) (b) (c) (d) (e)

Fonte: Elaborada pelo autor.

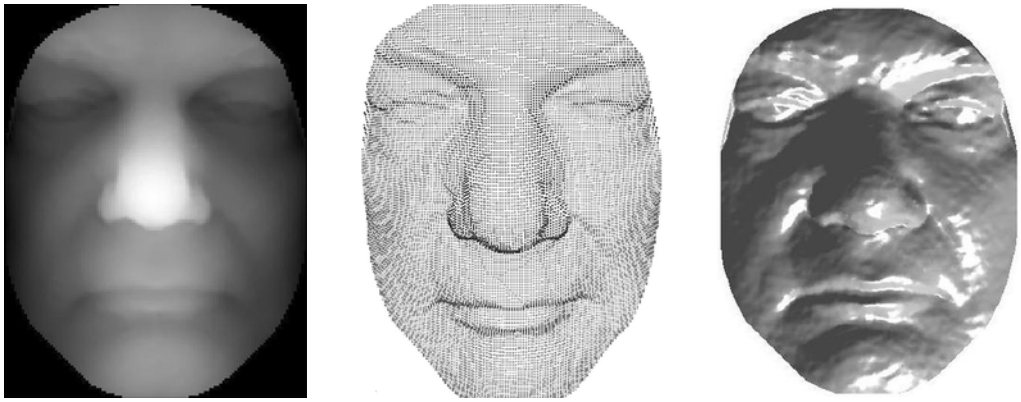
Como Priya et al. (2017) explicam, a profundidade geralmente pode ser percebida apenas pela visão humana. A partir da profundidade que os olhos observam, o cérebro é capaz de reconstruir a terceira dimensão a partir de projeções 2D. Através de técnicas e câmeras especiais, pesquisadores em visão computacional têm criado e aprimorado cada vez mais as reconstruções 3D (IOANNIDOU et al., 2017). Mesmo empresas como a Apple adicionaram iluminação infravermelha ativa para o reconhecimento de rosto através de seus *smartphones*, para que os usuários possam desbloquear seus dispositivos móveis usando a tecnologia FaceID. Erdogmus e

[1] Alguns autores também chamam de *in the wild*.

Marcel (2013) observaram que muitos sistemas de reconhecimento facial 2D podem apresentar inconsistências (geralmente conhecidos como mecanismos de falsificação de reconhecimento de rosto), o uso de informações 3D melhora a robustez do sistema, pois depende menos da interferência do ambiente.

Os pesquisadores Ganguly, Bhattacharjee e Nasipuri (2014) avaliaram como as faces 3D podem superar algumas desvantagens das faces 2D na tarefa de reconhecimento facial. Além disso, Zhou e Xiao (2018) explicam que reconhecimento facial 3D tem se tornado uma tendência a partir de imagens de profundidade e estruturas como nuvens de pontos e malhas, exibidos na Figura 2. Mesmo diante dos benefícios que eles vêm demonstrando e embasado pela revisão mais recente da literatura sobre o estado da arte em verificação de parentesco (QIN; LIU; WANG, 2019), pôde-se notar que nenhum outro pesquisador na área de visão computacional publicou algum trabalho sobre VP analisando o uso de características 3D com aprendizagem de máquina.

Figura 2 – Modelos com informações de profundidade.



(a) Profundidade (b) Nuvem de pontos (c) Malha

Fonte: Adotada de Zhou e Xiao (2018).

1.3 Objetivo

O objetivo desse trabalho é analisar a contribuição de informações relacionadas à profundidade, incorporadas à imagem rgb, nos estudos acerca de verificação de parentesco. Em geral, deseja-se melhorar e impulsionar novas descobertas em VP. Dada a aplicação de redes neurais em faces e parentesco (DUAN; ZHANG; ZUO, 2017), a seguinte questão de pesquisa foi definida:

QP: *Informações extraídas de modelos de faces 3D contribuem para precisão no reconhecimento de parentesco usando redes neurais profundas?*

1.4 Contribuição da pesquisa

As principais contribuições deste trabalho são as seguintes:

- É introduzido um banco de dados de parentesco com informações 2D e 3D, avaliado com métodos de visão computacional da literatura;
- Nós somos os primeiros pesquisadores à testar *features* relacionadas com profundidade em VP usando modelos de aprendizagem de máquina. Comparado com apenas imagens faciais, profundidade é mais discriminativa e melhora a acurácia de verificação;
- Publicação na conferência internacional *Advanced Concepts for Intelligent Vision Systems* com o trabalho *Verifying Kinship from RGB-D Face Data* (CRISPIM; VIEIRA; LIMA, 2020);
- Testamos novas topologias de rede neural e, diferente de outros trabalhos (QIN; LIU; WANG, 2019), foram usados todos os parentes para classificação em um modelo. Resultados experimentais demonstram que nossa abordagem proposta é competitiva com os métodos da literatura.

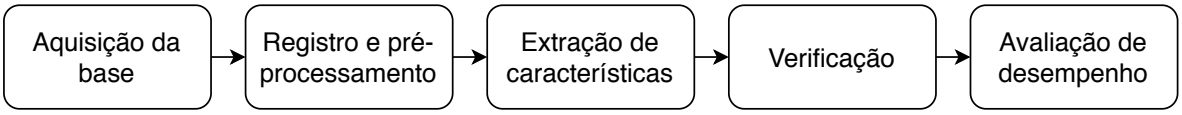
1.5 Estrutura da dissertação

Esta dissertação é estruturada da seguinte forma. O capítulo 2 apresenta o embasamento teórico de alguns conceitos envolvidos na construção e normalização da base de dados, as principais ferramentas e *frameworks* que serviram para desenvolvimento e suporte, e uma visão geral sobre redes neurais e formas de otimizá-la. O capítulo 3 descreve o estado da arte relacionado as bases de dados mais utilizadas, o de faces e os métodos utilizados para verificação de parentesco. O capítulo 4 trata da solução desenvolvida neste trabalho, detalhando a metodologia utilizada. O capítulo 5 apresenta os resultados e discussões sobre o estudo realizado. Por fim, o capítulo 6 apresenta a conclusão e os trabalhos futuros.

2 FUNDAMENTAÇÃO

Como ilustrado na Figura 3 (GANGULY; BHATTACHARJEE; NASIPURI, 2014), existem alguns passos iniciais que são seguidos quando se deseja criar e trabalhar com uma base de dados de faces. Este capítulo apresentará uma revisão sobre os principais assuntos envolvidos na execução desse fluxograma. No primeiro item são apresentados os fundamentos da técnica de estrutura a partir do movimento. No segundo item é discutido como são obtidas algumas características a partir de nuvens de pontos. O terceiro item refere-se as ferramentas e *frameworks* utilizados apontando suas principais características e métodos de funcionamento. O último item trata sobre aprendizagem profunda, incluindo formas de ajustamento da rede e métodos de avaliação.

Figura 3 – Diagrama de fluxo para verificação de parentesco.

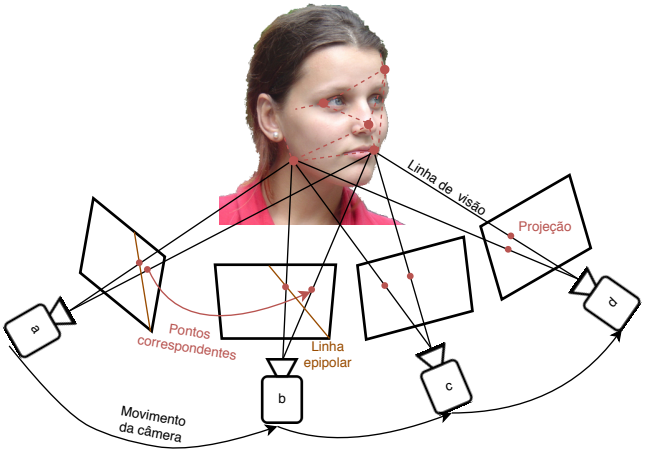


Fonte: Elaborada pelo autor.

2.1 Estrutura a partir do movimento

Estrutura a partir do movimento (do inglês: *structure from motion*[1] - SfM) é uma técnica de imagens no campo de fotogrametria usada há bastante tempo em trabalhos de visão computacional (WEI et al., 2019). O objetivo dessa técnica é recuperar uma estrutura tridimensional a partir de uma coleção de imagens estacionárias de uma cena ao estimar o movimento das câmeras correspondentes às imagens, como ilustrado na Figura 4.

Figura 4 – Estrutura a patir do movimento.



Fonte: Elaborada pelo autor.

[1] Também conhecido como *multiview structure from motion*.

Considere um caso simples com duas imagens ou as câmeras “a” e “b”, ilustradas na Figura 4. Segundo Ozyesil et al. (2017), o processo de SfM consiste das seguintes etapas:

1. Primeiramente, são extraídas as caraterísticas das imagens como pontos e linhas. Em seguida, são encontrados os pontos correspondentes ao fazer associação (do inglês: *matching*) dessas características ou ao fazer *tracking*[2] dos pontos da imagem da câmera “a” para a imagem da câmera “b”;

2. A posição relativa é estimada ao usar os pontos correspondentes do item anterior da câmera “b” em relação à “a”, então é computada a matriz fundamental que descreve a geometria epipolar das câmeras. Isso é obtido ao relacionar um ponto de uma câmera a uma linha epipolar da outra;

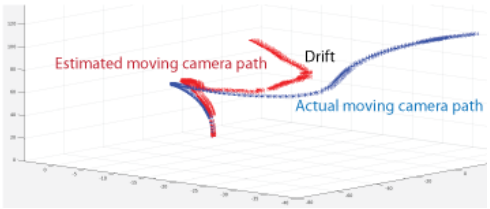
3. Usando a matriz fundamental, encontra-se a orientação e localização da segunda câmera em relação à primeira. Essa localização pode ser recuperada em *up to scale*[3].

4. Determina-se a localização 3D dos pontos associados usando triangulação e as matrizes das câmeras. Executa-se a reconstrução da estrutura 3D estimando os passos anteriores e minimizando o erro de reprojeção.

A abordagem anterior com 2 vistas (do inglês: *views*) pode ser estendida quando existem várias delas. Os pontos correspondentes são encontrados através das múltiplas imagens. Cada um desses pontos correspondem a um ponto 3D na cena. Os pontos 3D são computados usando triangulação para múltiplas vistas.

Ao longo desses procedimentos de estimação do movimento das câmeras, um erro também conhecido por *drift* se acumula ao longo das *views*, como ilustrado na Figura 5. Uma forma de reduzi-lo é usar um algoritmo de otimização não linear chamado de *bundle adjustment* (TRIGGS et al., 2000), usado como uma maneira de refinar o movimento.

Figura 5 – Erro acumulativo ao longo do movimento das câmeras - Em vermelho o caminho estimado e em azul o caminho real.



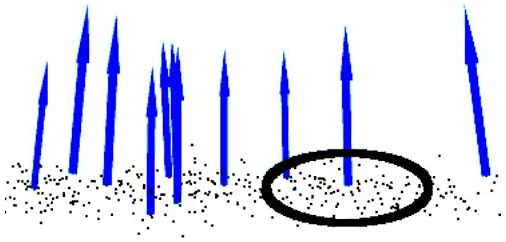
Fonte: Adotada de MATLAB (2019).

[2] O algoritmo de *tracking* Kanade-Lucas-Tomasi (KLT) pode ser usado quando as câmeras estão próximas.
[3] *Up to scale* significa que é possível reescalar a estrutura e a magnitude do movimento da câmera mantendo as observações.

2.2 Extração de características de nuvem de pontos

Segundo Kabbai, Abdellaoui e Douik (2018), características são atributos ou aspectos de alguma coisa pelo qual ele pode ser facilmente distinguível de um outro. Elas são consideradas como globais (considerando o objeto completo) ou locais (considerando uma parte do objeto). Em se tratando de nuvens de pontos, características locais 3D são representações em um certo ponto 3D ou posição no espaço que podem descrever padrões geométricos baseados em informações ao redor do ponto. O espaço de dados ao redor do ponto alvo geralmente é referido como uma k-vizinhança. A Figura 6 mostra um exemplo simples dessa vizinhança.

Figura 6 – **k**-vizinhança e alguns vetores normais sobre uma nuvem de pontos.



Fonte: Elaborada pelo autor.

Segundo Blomley et al. (2014), características 3D são bastante úteis na representação da forma geométrica local. Muitas abordagens empregam essas características derivadas da matrix de covariância local representando momentos invariantes de segunda ordem dentro das posições dos pontos. Essa matriz é calcula a partir de N observações $A_{1,2,3}$ da seguinte forma:

$$[mat]_{ij} = \frac{\sum_{l=1}^{N}(A_i - \overline{A_i}).(A_j - \overline{A_j})}{N},$$

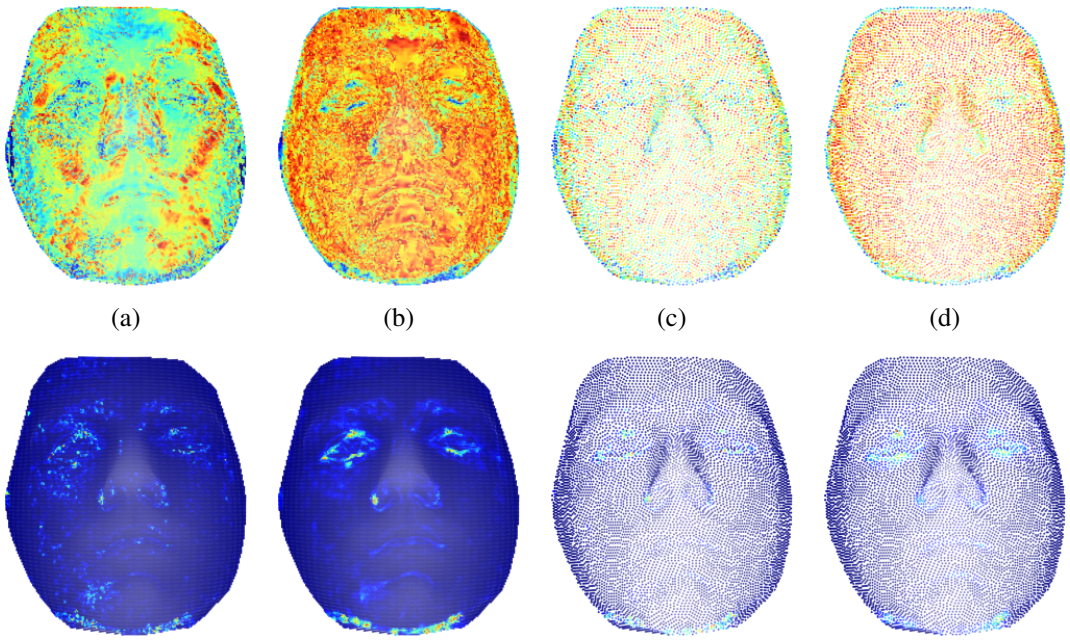
onde $i, j \in [1, 2, 3]$ e $\overline{A_i}$ mantém a média de todas as observações na respectiva dimensão. A análise dos componentes principais é usada para determinar momentos de segunda ordem linearmente não correlacionados em um espaço de autovetor ortogonal. Os autovalores $\lambda_{1,2,3}$ correpondentes são capazes de informar características locais relacionadas à dimensionalidade (linearidade, esfericidade e planaridade) e outras medidas como anisotropia, autoentropia e curvatura. Blomley et al. (2014) encontram esses autovalores ordenados como $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ e tendo $\lambda_1 + \lambda_2 + \lambda_3 = 1$, eles são referidos como características de covariância. A equação para obtenção dessas características são descritas a seguir e exemplos da planaridade e curvatura com variação de pontos e k vizinhos são ilustrados na Figura 7.

- Anisotropia: $A_\lambda = \frac{\lambda_1 - \lambda_3}{\lambda_1}$, descreve o grau em que uma certa estrutura varia em termos da direcionalidade;
- Curvatura: $C_\lambda = \frac{\lambda_3}{\sum_\lambda}$, descreve a quantidade na qual um objeto desvia ou deforma de ser plano ou reto;

- Autoentropia: $Au_\lambda = -\sum_{j=1}^{3} \lambda_j ln(\lambda_j)$, filtra os pontos com base na quantificação da desordem geométrica;
- Linearidade: $L_\lambda = \frac{\lambda_1 - \lambda_2}{\lambda_1}$, permite a detecção de estruturas de linhas;
- Planaridade: $P_\lambda = \frac{\lambda_2 - \lambda_3}{\lambda_1}$, tem a capacidade de discriminar estruturas planares;
- Esfericidade: $S_\lambda = \frac{\lambda_3}{\lambda_1}$, permite a exibição de pontos de alta curvatura.

Características geométricas estão relacionadas à cantos, bordas, variação de forma, entre outros. Elas são amplamente usadas em reconhecimento facial, sendo mais comum o uso de curvatura (ZHOU; XIAO, 2018). Além do mais, muitos trabalhos de verificação de parentesco lidam com informações desse tipo juntamente com informação de textura e demonstram que conseguem resultados ainda melhores (QIN; LIU; WANG, 2019).

Figura 7 – Características geométricas faciais - (a), (b), (c) e (d) representam a planaridade, onde, (a) e (c), (b) e (d) foram configurados com uma k-vizinhança igual à 5 e 15 respectivamente; além disso, (c) e (d) contém um terço da quantidade de pontos de (a) e (b). As figuras (e), (f), (g) e (h) representam a curvatura e estão ordenadas como as de planaridade.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fonte: Elaborada pelo autor.

2.3 Ferramentas e *frameworks*

Durante o desenvolvimento de um projeto de visão computacional e aprendizagem de máquina, diversas ferramentas podem ser utilizadas tanto para a construção como para a avaliação do mesmo. Alguns *frameworks* dão o suporte necessário para a realização dessas atividades,

que envolvem prover funcionalidades para limpeza, normalização, modelos de aprendizagem de máquina e validação.

A seguir serão apresentados alguns exemplos de *software* e *frameworks* utilizados no desenvolvimento dessa dissertação.

2.3.1 MeshLab

MeshLab (CIGNONI; CORSINI; RANZUGLIA, 2008) é um programa livre para processamento e edição de objetos 3D tais como nuvem de pontos e malhas. Ele permite editar, registrar, limpar, inspecionar, renderizar, criar texturas, entre outros. Ele foi projetado com três objetivos primários em mente: (i) ser fácil de usar, mesmo usuários sem grandes habilidades de modelagem 3D deveriam ser capazes de usá-lo (pelo menos as funcionalidades básicas); (ii) orientado a escaneamento 3D, o programa é focado no processamento e edição de malhas e nuvens de pontos; (iii) eficiência, o programa é capaz de lidar com milhões de primitivas que compõem objetos 3D.

No MeshLab (CIGNONI; CORSINI; RANZUGLIA, 2008), um conjunto de ferramentas prontas também estão disponíveis para diversas funcionalidades. Exemplos típicos seriam para remoção de duplicatas, vértices não referenciados, faces nulas, pequenos componentes isolados, espelhamento, entre outros. Diversos formatos de arquivos também são suportados (3ds, ply, stl, obg, ptx, xyz, pdb, wrl, etc.).

2.3.2 CloudCompare

O CloudCompare é um programa que disponibiliza diversas ferramentas para manipulação direta entre nuvens de pontos ou nuvem de ponto e malha. Inicialmente, seu principal objetivo era detectar rapidamente alterações entre nuvens de pontos de alta densidade. Contudo, o programa evoluiu suportando processamento de dados 3D mais avançados. No momento da escrita deste trabalho, CloudCompare está na versão 2.10.

Para comparação entre nuvens (CLOUDCOMPARE, 2015), CloudCompare realiza o cálculo de distâncias entre pontos pelo algoritmo do vizinho mais próximo (do inglês: *nearest neighbor*). Considerando cada ponto da nuvem a ser comparada, o programa busca o ponto mais próximo em relação a nuvem de referência e calcula a distância (euclidiana) entre eles. Uma vez que o espaçamento entre as nuvens é muito importante, quanto maior a proximidade entre os pontos maior a precisão do método.

O melhor procedimento para comparação seria entre uma nuvem e uma malha (CLOUDCOMPARE, 2015). Na ocasião em que não há uma malha, para representar com maior precisão uma superfície, e o método dos vizinhos mais próximos não é eficaz, o CloudCompare fornece uma outra abordagem chamada *local modeling*. Basicamente, o programa determina o ponto

mais próximo da nuvem de referência e modela localmente a superfície daquela nuvem através de um modelo matemático criado a partir de um ajustamento desse ponto e seus vizinhos.

CloudCompare também permite o alinhamento entre nuvens de forma automática e "manual". O primeiro pode ser feito através do método conhecido como *bounding-box centers matching*, que tenta centrar as entidades nos seus respectivos centros de gravidade. O segundo é realizado através da escolha de pelo menos 3 pontos das nuvens, a partir desses pontos o programa pode ajustar a escala e rotacionar a nuvem. Esse tipo de ajuste é muito importante pois alguns métodos automáticos podem considerar pontos mais externos e reajustar todos os outros pontos com base neles.

2.3.3 MATLAB

MATLAB (MATLAB, 2019) é uma plataforma de programação projetada especificamente para engenheiros e cientistas. No seu núcleo está a linguagem de programação MATLAB, uma linguagem de programação científica para implementar algoritmos complexos e analisar seu desempenho em formas de números e gráficos. Ela também é uma linguagem baseada em matriz o que permite a expressão mais natural da matemática computacional.

MATLAB (MATHWORKS, 2019) permite a execução de computações matemáticas complexas e possui várias funcionalidades que lidam com diversas tarefas na área de visão computacional, processamento de sinais, aprendizagem de máquina, entre outros. MATLAB também oferece *toolboxes* que podem ser usados em uma ampla gama de aplicações científicas e de engenharia, por exemplo, é possível lidar com diversas questões de processamento de imagem interativamente tornando o processo mais rápido e robusto do que desenvolver em linguagens de programação como Python ou C++. Isso é possível também graças à sua otimização, uma vez que suas operações matemáticas são distribuídas pelos núcleos do computador, as chamadas de biblioteca são altamente otimizadas e todo o código é compilado em tempo de execução (do inglês: *just-in-time*).

2.3.4 OpenCV

Open Source Computer Vision Library (OpenCV) (ITSEEZ, 2015) é uma biblioteca de funções de visão computacional, processamento de imagens e aprendizado de máquinas com código aberto. O OpenCV foi desenvolvido para fornecer uma infraestrutura comum em aplicações de visão computacional e acelerar o uso da percepção de máquinas em produtos comerciais. O pacote OpenCV possui licença BSD (Berkeley Source Distribution).

OpenCV conta com milhares de algoritmos otimizados que incluem um conjunto abrangente tanto de algoritmos de visão computacional quanto algoritmos de inteligência artificial (ITSEEZ, 2014). Ela está dividida em cinco grupos de funções: Processamento de imagens; Análise estrutural; Análise de movimento e rastreamento de objetos; Reconhecimento de padrões

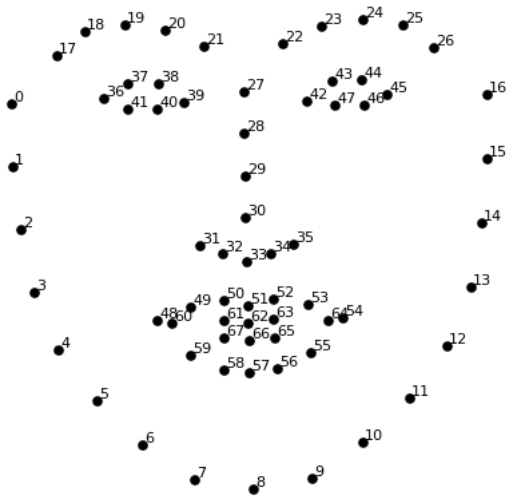
e Calibração de câmera e reconstrução 3D. Esses algoritmos podem ser utilizados para detectar e reconhecer faces, manipular SFM (*Struct From Motion*), identificar e rastrear movimento de objetos, aplicar filtros, costurar mapas, dentre muitas outras funções. É possível desenvolver com OpenCV em diversas linguagens como: C, C++, Python e Java.

2.3.5 Dlib

Dlib é uma biblioteca desenvolvida em C++ que contém vários algoritmos, com foco maior em aprendizagem de máquina e visão computacional (KING, 2009). Também possui API em Python o que possibilita a utilização dos seus algoritmos nessa linguagem. Dlib disponibiliza um conjunto de ferramentas para detecção de objetos em imagens incluindo detecção e estimação de pose de faces.

Seu detector de *landmarks* em faces é capaz de encontrar até 68 pontos (x, y) em faces frontais, a Figura 8 mostra a distribuição desses pontos. Sobre a detecção com 5 pontos, King (2017) comenta, "*Added a 5 point face landmarking model that is over 10x smaller than the 68 point model, runs faster, and works with both HOG and CNN generated face detections.*". Esses 5 pontos são as duas pontas dos dois olhos e a ponta do nariz.

Figura 8 – Pontos detectados pela Dlib.



Fonte: Adotada de Korshunov (2018).

2.3.6 Keras e TensorFlow

Keras (CHOLLET et al., 2015) é uma biblioteca de Python que pode ser usada como uma camada de abstração para bibliotecas como TensorFLow. Sua principal vantagem é permitir a escolha do *framework* que rodará no *backend* e facilitar o acesso as APIs. No momento da escrita deste trabalho, também é possível rodar Keras em Apache MXNet, Core ML da Apple, Javascript ou Typescript (para rodar Keras no browser), ou PlaidML (extendendo para outros tipos de GPU, não somente Nvidia).

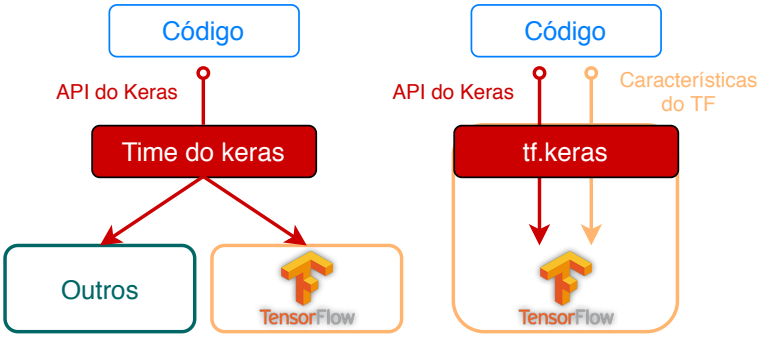
TensorFlow (ABADI et al., 2015) é uma biblioteca de Python desenvolvida por pesquisadores do Google que trabalhavam no projeto Google Brain. TensorFlow usa biblioteca de *Data Flow Graph* para fazer suas computações numéricas. Ele funciona através de *tensors* [4] e, portanto, seu nome decorre disto. Sua principal aplicação é em projetos de aprendizagem de máquina, contudo, ele também apresenta boa utilidade em projetos que exigem processamento pesado. Ele se tornou *software* livre em novembro de 2015 e é atualmente a biblioteca mais popular de *deep learning* [5] (GÉRON, 2019). Uma vez que somente ele foi usado nessa dissertação, apenas ele será abordado nesta Seção.

Este *framework* possui diversas vantagens em relação a outros similares incluindo, mas não limitadas a (GÉRON, 2019):

- Seu núcleo é similar ao NumPy, mas com suporte a GPU;
- Suporta computação distribuída (através de diversos dispositivos e servidores);
- Seus gráficos podem ser exportados para um formato portável, permitindo treinar um modelo em um ambiente e executá-lo em outro.
- Dispõe diversos modelos de *autoenconders* e redes neurais.

A partir do TensorFlow 2.0, ele vem com sua própria implementação do Keras chamada *tf.keras*, ilustrado na Figura 9. Ele somente suporta TensorFlow como *backend*, mas sua principal vantagem é fornecer suporte para algumas das principais características avançadas do TensorFlow, por exemplo, tornar mais fácil carregar e processar as APIs de dados do TensorFlow.

Figura 9 – Implementações do Keras - À esquerda, time do Keras, e à direita, tf.keras.



Fonte: Elaborada pelo autor.

2.4 Aprendizagem profunda

Aprendizagem profunda (AP) é uma subárea de aprendizagem de máquina. Modelos desse tipo têm se tornado cada vez mais populares e conseguido resultados satisfatórios em

[4] Um tensor é basicamente um array multidimensional, mas é também capaz de manter um escalar.
[5] Considerando citações em artigos, adoção em empresas, estrelas no github, etc.

diversos problemas de visão computacional (VOULODIMOS et al., 2018). Muitos algoritmos de aprendizagem profunda têm sido propostos e utilizados em várias aplicações incluindo verificação de parentesco (WU et al., 2019). O propósito de AP é aprender características não lineares hierárquicas ao criar redes com múltiplas camadas.

Redes neurais convolucionais (do inglês: *convolutional neural network* - CNN) (Lecun et al., 1998) são métodos discriminativos de redes neurais profundas. Elas são um exemplo de redes neurais do tipo *feedforward* que provaram ter um desempenho notavelmente bom em várias tarefas ao longo dos anos (IOANNIDOU et al., 2017). Esse tipo de rede neural tira vantagem da sua estrutura hierárquica (GÉRON, 2019):

- Camadas mais baixas (do inglês: *lower layers*) descrevem características de baixo nível (do inglês *low-level features*), por exemplo, linhas em diferentes formatos e orientações;
- Camadas intermediárias (do inglês: *intermediate layers*) descrevem estruturas de nível intermediário (do inglês: *intermediate-level structures*), por exemplo, formatos circulares ou quadrangulares;
- Camadas mais altas (do inglês: *highest layers*) descrevem estruturas de nível intermediário (do inglês: *high-level structures*), por exemplo, rostos.

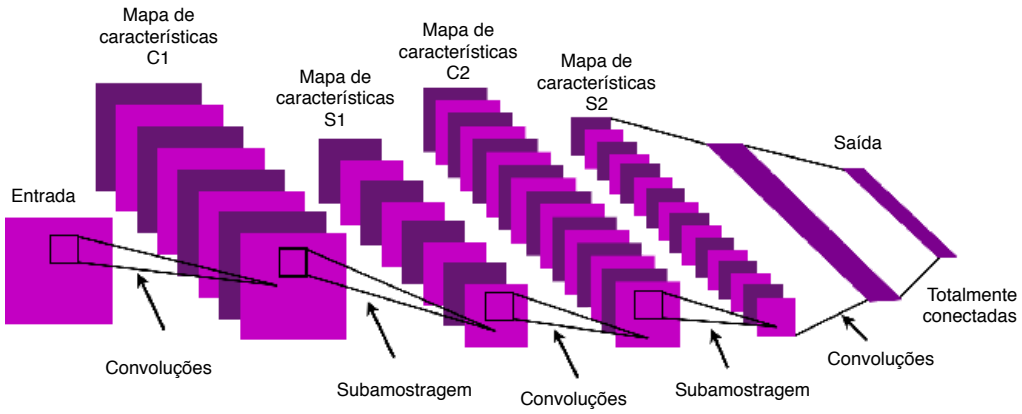
Como Jeon e Rhee (2017) explicam, CNNs geralmente consistem em várias camadas convolucionais e de *poolings* (ou subamostragem), seguidas por camadas totalmente conectadas (do inglês: *fully conected layers* - FC), como ilustrado na Figura 10. Em tarefas de classificação, normalmente a última camada FC é conectada à um classificador (por exemplo, *softmax*) que fornece a resposta da rede dada uma certa entrada. Suas camadas convolucionais e FC estão relacionadas à parâmetro com pesos e *biases* que precisam ser adaptados ao problema. Em uma CNN para imagens, o dado de entrada x no índice l tem uma estrutura espacial do tipo:

$$x_l \in R^{H_l \times W_l \times C_l},$$

onde H_l se refere a altura, W_l se refere a largura e a terceira dimensão C_l se refere ao número de canais da imagem. Ainda há uma outra dimensão nomeada *batch*, para executar um processamento a partir de subconjuntos.

Vários filtros convoluem sobre as imagens de entrada e o resultado dessa operação é a geração de mapas de características (do inglês: *feature maps*). Como Ioannidou et al. (2017) explicam, um ponto importante sobre as CNNs é que, ao contrário de redes neurais tradicionais, seus filtros usados nas camadas convolucionais são aplicados na imagem inteira, com o mesmo peso. Se o passo (do inglês: *stride*) que esses filtros executam for configurado com um valor largo, o filtro será aplicado menos vezes sobre a entrada na camada. Outro parâmetro importante é o *padding*, ele é o responsável por fornecer controle do tamanho espacial das saídas após as

Figura 10 – Rede convolucional típica - 2 estágios de extração de características são utilizados.



Fonte: Adotada de Lecun, Kavukcuoglu e Farabet (2010).

convoluções. Ao final do processo de convolução, a largura e altura da imagem é dado por

$$\frac{W - F_w + 2P}{S} + 1$$

e

$$\frac{H - F_h + 2P}{S} + 1,$$

respectivamente, onde:

- W e H são a largura e altura da imagem de entrada;
- F_W e F_H são a altura e largura do filtro;
- P é o *padding*;
- S é o *stride*.

Nas CNNs, cada *pixel* é tratado por um neurônio (GÉRON, 2019). Por sua vez, cada neurônio localizado na linha i, coluna j do mapa de característica k em uma camada convolucional l é conectado à saída dos neurônios na camada prévia $l-1$, localizado da linha $i \times s_h$ até $i \times s_h + f_h - 1$ e da coluna $j \times s_w$ até $j \times s_w + f_w - 1$, através de todos os mapas de características (na camada $l-1$). Na equação a seguir é ilustrado o resultado da saída do neurônio na camada convolucional, $z_{i,j,k}$, localizado na linha i, coluna j no mapa de características k da camada l:

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} . w_{u,v,k',k} \text{ com } \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases},$$

onde:

- s_h e s_w são os *strides* horizontal e vertical, respectivamente. f_h e f_w são a altura e largura da área receptiva, e $f_{n'}$ é o número de mapas de características na camada prévia (camada $l-1$);
- $x_{i',j',k'}$ é a saída do neurônio localizado na camada $l-1$, linha i', coluna j' e mapa de características k';
- b_k é o *bias* para o mapa de características k (na camada l);
- $w_{u,v,k',k}$ é o peso da conexão entre qualquer neurônio no mapa de características k da camada l e sua entrada localizada na linha u, coluna v e mapa de característica k'.

As próximas Subseções explanam sobre os principais conceitos utilizados para ajustar os hiperparâmetros das CNNs. Como explica Sun (2019), esses ajustes são úteis para: melhorar e acelerar o aprendizado, diminuir a perda, aumentar a acurácias, evitar *overfitting*, entre outros. Por fim, algumas técnicas usadas para visualização das representações aprendidas por tipos de redes como convolucionais.

2.4.1 Camadas escondidas

Em algumas situações, apenas uma camada escondida permite conseguir resultados razoáveis. Contudo, redes neurais profundas possuem uma eficiência de parâmetros muito maior do que redes rasas: elas podem modelar funções complexas com menos neurônios exponencialmente do que redes rasas, permitindo que elas alcancem resultados mais satisfatórios com a mesma bade de dados (GÉRON, 2019).

A adição de mais camadas não somente permite a rede convergir mais rápido para uma solução boa como também melhora a capacidade de generalização. Isso é muito útil em transferência de aprendizado (do inglês: *transfer learning*), por exemplo, se uma CNN é treinada em uma base de dados para reconhecer faces e se deseja treinar uma rede para reconhecer parentesco, então, uma boa prática é fazer o treinamento reutilizando as camadas inferiores da primeira rede neural (DORNAIKA; ARGANDA-CARRERAS; SERRADILLA, 2019). Isso pode ser uma abordagem melhor do que inicializar aleatoriamente os pesos e *biases*, ou seja, fazer com que a rede não precise aprender do zero características de baixo nível.

2.4.2 Neurônios por camada

O número de neurônios na camada de entrada e saída é determinado pelo problema que se pretende resolver, por exemplo, em uma verificação de parentesco, ao utilizar imagens de entrada em escala de cinza de tamanho 64x64 resultaria em 4096 neurônios de entrada e 2 neurônios na saída, para parentes e não parentes.

Como Géron (2019) explica, normalmente, as camadas escondidas são configuradas com quantidades de neurônios em ordem crescente ou decrescente, por exemplo, para 3 camadas

escondidas, a primeira poderia ter 300 neurônios, a segunda com 200 e a terceira com 100. Contudo, geralmente, essa forma de diminuir a quantidade de neurônios ao longo da rede não tem apresentado bons resultados e é preferível manter ou ir aumentando a quantidade deles. Uma abordagem mais simples é usar uma rede com mais camadas e mais neurônios do que se espera e diminuir o *overfitting* ao usar métodos de parada antecipada (do inglês: *early stopping*) e/ou *dropout*.

2.4.3 Funções de ativação

As funções de ativação definem a saída do neurônio com base em um determinado conjunto de entradas. A soma ponderada do valor de entrada linear é passada através de uma função de ativação por uma transformação não linear. Uma escolha ruim dessas funções pode causar problemas como dissipação ou explosão do gradiente (do inglês: *vanishing or exploding gradient*), como explica (GLOROT; BENGIO, 2010). Para uma entrada x, as funções de ativação mais utilizadas em redes neurais são (GÉRON, 2019):

- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$, com $f'(x) = f(x)(1 - f(x))$;
- Tanh: $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$, com $f'(x) = 1 - f(x)^2$;
- ReLU: $f(x) = \begin{cases} 0, & se\ x < 0 \\ x, & se\ x \geq 0 \end{cases}$, com $f'(x) = \begin{cases} 0, & se\ x < 0 \\ 1, & se\ x \geq 0 \end{cases}$;
- Leaky ReLU: $f(x) = \begin{cases} 0.01x, & se\ x < 0 \\ x, & se\ x \geq 0 \end{cases}$, com $f'(x) = \begin{cases} 0.01, & se\ x < 0 \\ 1, & se\ x \geq 0 \end{cases}$;
- Softmax: $f(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{d} e^{x_k}}$, com $\frac{\partial f(x_j)}{\partial x_i} = f(x_j)(\delta_{ij} - f(x_i))$ e $\delta_{ij} = \frac{\partial}{\partial x_i}$.

2.4.4 Outros hiperparâmetros

A fim de eficientemente treinar uma rede (IOANNIDOU et al., 2017), não somente o número de camadas e a quantidade de neurônios são configurados, outros hiperparâmetros também são considerados:

- A taxa de aprendizagem. Segundo Géron (2019), uma boa prática é começar com um valor largo de modo que o algoritmo de treinamento diverge inicialmente, então a medida que esse valor diminui o modelo vai parando de divergir e se aproximará da taxa de aprendizagem ótima;
- Tipo de *batch*. O *batch* pode ser dado pelo conjunto todo de treinamento, por uma amostra aleatória a cada passo do processo ou uma quantidade de amostras aleatórias, para cada um desses, eles normalmente são conhecidos por *full batch*, *stochastic batch* ou *mini-batch*, respectivamente;

- Número de épocas. O número de épocas pode levar um modelo a se tornar sobreajustado. Dessa maneira, Géron (2019) recomenda que esse parâmetro seja configurado com métodos de *early stopping* para que isso seja evitado.

2.4.5 Visualização do que as CNNs aprendem

Como Chollet (2017) explica, normalmente modelos de aprendizagem profunda são encarados como "caixas pretas", isso porque representações de aprendizado podem ser difíceis de extrair e representar de uma forma natural para seres humanos. Contudo, com relação à redes do tipo convolucionais suas representações de aprendizado são mais acessíveis e legíveis, isso acontece em grande parte porque são representações de conceitos visuais. Uma das principais formas de visualização é dada por mapas de calor (do inglês: *heatmap*) da ativação da classe em uma imagem e eles são úteis para o entendimento de que partes de uma imagem foram identificadas como importantes para qual classe.

Visualização de mapas de ativação de classe (do inglês: *class activation map* - CAM) consiste em produzir mapas de calor das ativações das classes sobre as imagens de entrada (CHOLLET, 2017). Um mapa de calor da ativação de uma classe é uma imagem 2D associada com uma saída de uma classe específica, obtida para cada localização em qualquer imagem de entrada, isso indica quanto cada localização pode ser importante para a classificação da classe. Por exemplo, ao alimentar uma rede com uma imagem com face, a rede pode permitir gerar mapas de calor que indicam diferentes partes do rosto que são mais importantes ou não para a determinação daquela classe, como ilustrado na Figura 11.

Figura 11 – *Heatmaps* de faces.

Fonte: Adotada de Castanon e Byrne (2018).

Um forma de obter os mapas de calor é através da técnica *Gradient weighted Class Activation Map* (Grad - CAM) proposta por Selvaraju et al. (2019). Basicamente, é considerado o *feature map* da camada convolucional e então é ponderado todos os canais naquela *feature* com o gradiente da classe em relação ao canal. Sendo assim, é uma maneira de visualizar um mapa espacial de "quão intensivo a imagem de entrada ativa diferentes canais" por "quão importante cada canal é em relação à classe", resultando em um mapa espacial de "quão importante a

imagem de entrada ativa a classe". Isso é obtido sem necessidade de retreinamento da rede ou mudança na arquitetura. Esse mapa espacial é como representado pela equação

$$S^c = \frac{1}{z} \sum_i \sum_j \sum_k \frac{\delta y^c}{\delta A_{ij}^k} A_{ij}^k$$

Na qual é obtida pelo *pooling* médio global sobre as dimensões i e j para o gradiente da saída da classe respectiva y^c com relação ao mapa de características A_{ij}^k, com z sendo o número de *pixels* nesse mapa de características e k o eixo do canal.

3 TRABALHOS RELACIONADOS

Neste capítulo, é feita uma revisão da literatura com os principais trabalhos que se relacionam com essa dissertação.

3.1 Bases de dados

Qin, Liu e Wang (2019) apresentaram o *survey* mais recente até o momento da escrita dessa dissertação. Eles fizeram o levantamento das bases de dados de rostos que forneciam alguma informação de parentesco. A tabela 1 fornece em ordem cronológica algumas dessas bases. Esses autores notaram que ao longo do tempo, o desenvolvimento profundo do estudo de verificação de parentesco não apenas aumentou a escala dos dados, mas também levou alguns pesquisadores a considerarem relações de parentesco mais diversas. Também, todas as bases estão em formato RGB e além de fornecerem imagens frontais dos rostos, algumas dessas bases consideram imagens de perfil e vídeos.

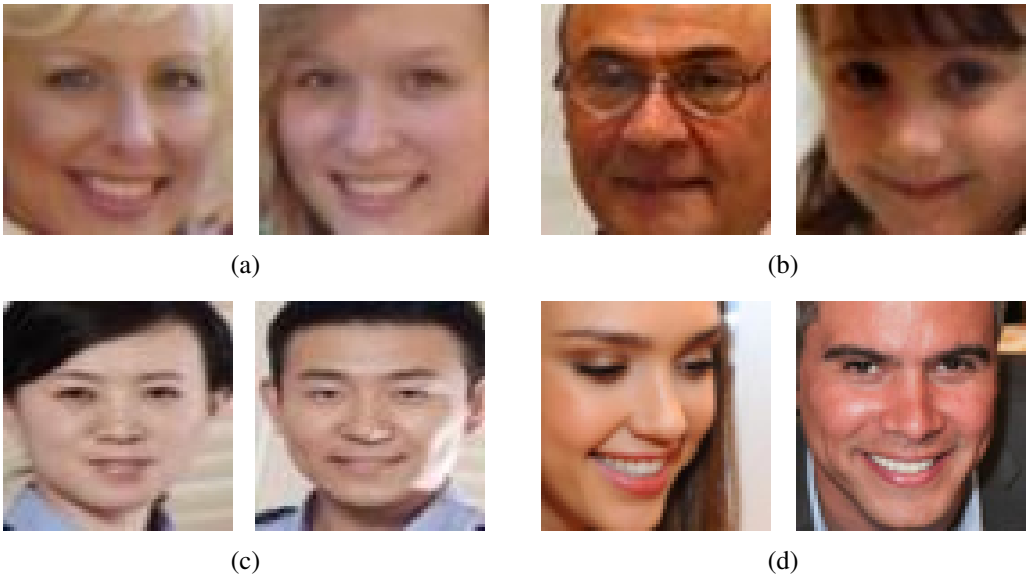
Tabela 1 – Bases de dados de faces com informações de parentesco - "Tamanho" se refere ao número de relacionamentos de parentesco ou grupos (família de relação sanguínea), "Estrutura familiar" se refere à existência de relacionamento familiar na base e "Múltiplas imagens" se refere à se cada pessoa na base tem múltiplas imagens suas.

Base de dados	Tamanho	Est. familiar	Múlti. imagens
CornellKin (FANG et al., 2010b)	150 pares	Não	Não
UB KinFace 1.0 (XIA; SHAO; FU, 2011)	90 grupos	Não	Sim
UB KinFace 2.0 (Shao; Xia; Fu, 2011)	200 grupos	Não	Sim
HQfaces (BOTTINO et al., 2012)	227 pares	Não	Sim
LQfaces (BOTTINO et al., 2012)	98 pares	Não	Não
Family101 (FANG et al., 2013)	>607 pares	Sim	Sim
KinFaceW-I (LU et al., 2014)	533 pares	Não	Não
KinFaceW-II (LU et al., 2014)	1000 pares	Não	Não
TSKinFace (Qin; Tan; Chen, 2015)	1015	Sim	Sim
WVU (KOHLI et al., 2016)	113 pares	Não	Sim
FIW (ROBINSON et al., 2016)	>418k pares	Sim	Sim
KFVW (YAN; HU, 2017)	418 pares	Não	Sim
KIVI (Kohli et al., 2019)	355 pares	Sim	Sim

Fonte: Adotada de Qin, Liu e Wang (2019).

Segundo Georgopoulos, Panagakis e Pantic (2018) e Qin, Liu e Wang (2019), em trabalhos de visão computacional, as bases de dados com informação de parentesco mais utilizadas na literatura são: KinFaceW-I (LU et al., 2014), KinFaceW-II (LU et al., 2014), TSKinFace (Qin; Tan; Chen, 2015) e Families in the Wild (ROBINSON et al., 2016). As imagens faciais nesses bancos de dados são coletadas principalmente da Internet por meio de mecanismos de pesquisa baseados em palavras-chave. A figura 12 mostra algumas amostras positivas dessas 4 bases.

Figura 12 – Amostras de pares positivos de diferentes bases de dados com informação de parentesco - As bases são de (a) KinFaceW-I, (b) KinFaceW-II, (c) TSKinFace e (d) Families in the Wild, respectivamente.



Fontes: Adotadas de Lu et al. (2014), Qin, Tan e Chen (2015) e Robinson et al. (2016).

Existem 4 relações de parentesco nos dois datasets de Kinship Face in the Wild ou KinFaceW (KFW-I e KFW-II): Pai-Filha (PFa), Pai-Filho (PFo), Mãe-Filha (MFa) e Mãe-Filho (MFo). Para o KinFaceW-I as imagens são obtidas de fotos diferentes e contêm para essas relações 134, 156, 127 e 116 pares, respectivamente. Para o KinFaceW-II as imagens são obtidas da mesma foto e todas as relações contêm 250 pares. Como forma de padronizar a avaliação, os *folds* são fornecidos para validação cruzada, onde ambos os pares positivos e negativos são especificados para cada um dos cinco *folds*, como mostrado na Tabela 2.

Tabela 2 – Índices de faces dos 5 *folds* para validação cruzada nas bases de dados KinFaceW-I e KinFaceW-II.

Fold	**KFW-I**				**KFW-II**
	PFa	**PFo**	**MFa**	**MFo**	**Todas**
1	[1, 27]	[1, 31]	[1, 25]	[1, 23]	[1, 50]
2	[28, 54]	[32, 64]	[26, 50]	[24, 46]	[51, 100]
3	[55, 81]	[65, 96]	[51, 75]	[47, 69]	[101, 150]
4	[82, 108]	[97, 124]	[76, 101]	[70, 92]	[151, 200]
5	[109, 134]	[125, 156]	[102, 127]	[93, 116]	[201, 250]

Fonte: Adotada de Lu et al. (2014).

A base de dados TSKinFace (Qin; Tan; Chen, 2015) foi introduzida como uma alternativa às bases de parentesco projetadas para avaliação em pares. Ela contêm imagens de pais e mães com seus filhos, ambos com as faces recortadas como também todos juntos em uma mesma foto. Essa base contêm 274, 285 e 228 fotos de famílias para Pai-Mãe-Filha (PM-Fa), Pai-Mãe-Filho (PM-Fo) e Pai-Mãe-Filha-Filho (PM-FaFo), respectivamente. Para os relacionamentos em pares,

existem 513 Pai-Filho, 502 Pai-Filha, 513 Mãe-Filho e 502 Mãe-Filha. Os *folds*, Tabela 3, são fornecidos para validação cruzada.

Tabela 3 – Índices de faces dos 5 *folds* para validação cruzada na base de dados TSKinFace - Neste contexto, Pai é expressado como P, mãe é expressa como M, filho é expressado como Fo e filha é expressada como Fa.

Fold	**TSKinFace**	
	PM-Fa	**PM-Fo**
1	[1, 100]	[1, 102]
2	[101, 200]	[103, 204]
3	[201, 300]	[205, 306]
4	[301, 400]	[307, 408]
5	[401, 502]	[409, 513]

Fonte: Adotada de Qin, Tan e Chen (2015).

Até a escrita desta dissertação, Families In the Wild (FIW) (ROBINSON et al., 2016) é a maior base de dados com imagens e anotações de parentesco. Ela possui 11 relações: Pai-filho, Pai-Filha, Mãe-Filho, Mãe-Filho, Avô-Neto, Avô-Neta, Avó-Neto, Avó-Neta, Irmão-Irmã, Irmão-Irmão e Irmã-Irmã. Ela inclui 11163 fotos de famílias, para um total de 1000 famílias, cada família contém pelo menos 3 membros. Todos esses pares de relações resultam em 418060 imagens. Seus membros são compostos por jogadores, atores, políticos, a família real, entre outros. Essa base se tornou disponível como parte da competição de verificação de parentesco *Recognising Families in the Wild* (RFIW) em 2018.

3.2 Faces para verificação de parentesco

Mesmo antes e depois das primeiras pesquisas, em visão computacional, sobre verificação de parentesco a partir de imagens de rostos usando modelos de aprendizagem de máquina (FANG et al., 2010b), várias pesquisas em psicologia tentaram entender como o rosto pode fornecer informações úteis na sua verificação e consequentemente de parentesco (BRESSAN; MARTELLO, 2002; MARTELLO; MALONEY, 2006; DEBRUINE et al., 2009; ALVERGNE et al., 2014), encontrando características importantes para medir similaridades principalmente entre crianças e adultos. Estes trabalhos geralmente são executados ao pedir para avaliadores julgarem se duas pessoas, a partir de duas imagens 2D dos rostos delas, possuem uma relação de parentesco ou não.

Alguns pesquisadores avaliaram quais partes do rosto influenciam mais na verificação de parentesco, por exemplo, Martello e Maloney (2006) perceberam que a parte superior do rosto carrega mais informações úteis no julgamento do que a parte inferior, além do mais, o fornecimento de características ligadas aos formatos do rosto (como o formato do nariz) se mostraram bastante informativas. Martello, DeBruine e Maloney (2015) perceberam que a rotação ou inversão não afetam a avaliação de parentesco. Também, que características relacionadas ao

formato do rosto são muito importantes no julgamento de parentesco. Não só o formato da face contribui bastante, mas como Djordjevic et al. (2016) adicionam: altura, largura e proeminência do nariz; distância interocular; e proeminência labial. Apesar de faces poderem sofrer influencias por certo fatores, como ambientais, Fasolt et al. (2019) apontam que normalmente famílias vivem em regiões próximas e portanto isso contribui com a similaridade facial, impulsionando ainda mais o uso de faces para esse tipo de problema.

Outras características que as imagens de faces carregam são: a cor da pele e dos olhos. Suas contribuições também têm sido examinadas por pesquisadores na área de reconhecimento facial e de parenteso. Wu et al. (2016) mostraram que os resultados em bases de dados de parentesco podem ser aprimorados usando imagens coloridas ao invés de imagens em escala de cinza. Martello e Maloney (2006) perceberam que no julgamento de parentesco, a acurácia diminuiu cerca de 20% quando a região dos olhos foi obscurecida. Contudo, principalmente a coloração da pele pode sofrer muita influência por fatores internos (por exemplo: alimentação e hormônios) e externos (por exemplo: a exposição à luz solar), a iluminação e sombreamento do ambiente também influencia na captura da imagem (LOPEZ et al., 2018).

Dadas as pesquisas sobre as caraterísticas relacionadas as formas e textura/tonalidade 2D em faces e parentesco, os pesquisadores Fasolt et al. (2019) investigaram a contribuição direta de formatos faciais e informações de refletância[1] de superfície para detecção de parentesco em 3D, ao pedir para alguns participantes julgarem esses objetos tridimensionais. Sua base de dados foi projetada em três versões: (i) uma versão que combinava as informações individuais de refletância e informações de superfície; (ii) uma versão que matinha as informações de refletância mas com a superfície padronizada entre as amostras; (iii) e outra que mostrava os formatos de forma individual mas sem a informação de refletância. Eles observaram que tanto a forma quanto a refletância do rosto contribuem positivamente para a avaliação de parentesco, embora este último não tenha atingido significância estatística.

3.3 Classificação de parentesco

Ao longo dos últimos anos, muitos pesquisadores têm feito progresso na verificação de parentesco (GEORGOPOULOS; PANAGAKIS; PANTIC, 2018). Segundo Zhang et al. (2015), as primeiras tentativas de resolver o problema de verificação de parentesco foram baseadas em características faciais locais. Fang et al. (2010a) foram os primeiros pesquisadores a lidar com verificação de parentesco, empregando descritores HOG com características de cores locais e características de distâncias faciais ao dividir a face em partes para representar a face inteira, focando em cor da pele, cor do cabelo, cor do olho, distância entre os olhos e a boca, etc. O classificador K vizinhos mais próximos (do inglês: *K nearest neighboors* – KNN) foi usado para classificação desses pares de imagens.

[1] Esse termo se refere ao mapa de textura das imagens 3D, tais como tonalidade da pele, textura e cor dos olhos.

Hu et al. (2014), Hu, Lu e Tan (2014), Lu et al. (2014), Zhou et al. (2012), Zhou et al. (2011) propuseram vários métodos para verificação de parentesco ao usar, por exemplo, características métricas e pirâmide de imagens. Em suas pesquisas também podem ser encontrados classificadores como máquina de vetores de suporte (do inglês: *support vector machine* - SVM). Eles também são os repensáveis por disponibilizarem uma das bases de dados mais utilizadas para essa linha de pesquisa, KinFaceW-I e KinFaceW-II. Lu et al. (2014) lidaram em tentar também tornar amostras interclasses menos similares e amostras intraclasses mais similares.

Como Zhang et al. (2015) citam, embora pesquisas com aprendizagem métrica tenham avançado ao longo do tempo, elas são baseadas em características de baixo nível (do inglês: *low-level features*) e métodos rasos, cujos resultados não eram suficientemente satisfatórios. Considerando faces, isso se torna mais complicado com as diferentes variações na sua superfície, transformações não lineares, deformações devido a expressões, pose e idade. Zhang et al. (2015) foram os primeiros a propor métodos de aprendizagem profunda com redes neurais convolucionais (do inglês: *convolutional neural networks - CNN (Lecun et al., 1998)*) para o problema de verificação de parentesco. Além disso, eles também avaliaram o uso de partes do rosto para a verificação de parentesco. Suas abordagens aumentaram em mais de 5% o estado da arte. Ainda segundo a pesquisa mais recente, até o momento da escrita deste trabalho, levantada por Qin, Liu e Wang (2019), o método utilizado por Zhang et al. (2015) é um dos que atinge as maiores acurácias na literatura.

CNN vêm sendo usado em diversas aplicações de faces e parentesco (GEORGOPOULOS; PANAGAKIS; PANTIC, 2018). Hu et al. (2017) usaram CNN para estimar idade ao treinar com uma arquitetura que utiliza dados fracamente rotulados. Xing et al. (2017) elaboraram uma outra arquitetura de CNN profunda para predizer idade, raça e gênero. A rede neural VGG16 foi usada para explorar largas bases de dados como Families In the Wild (ROBINSON et al., 2018), como explicado em (GEORGOPOULOS; PANAGAKIS; PANTIC, 2018). Outros pesquisadores como Rehman et al. (2019) e Wu et al. (2019) utilizaram redes neurais profundas para verificação de parentesco e conseguiram superar o estado da arte em base de dados como KFW-I e KFW-II.

4 MATERIAIS E MÉTODOS

Seguindo o fluxograma ilustrado na Fundamentação, Figura 3, este capítulo abordará os métodos utilizados para o desenvolvimento desse trabalho. O processo de desenvolvimento começará abordando a criação da base de dados. As próximas seções são referentes ao que chamamos de Método I e Método II. O primeiro está relacionado com a validação da base Kin3D ao verificar parentesco entre pais e filhos. O último está relacionado a verificação de parentesco entre parentes de primeiro grau. Além disso, durante o desenvolvimento foram utilizadas as seguintes ferramentas: OpenCV 4.1.2, CloudCompare 2.10.2, MeshLab v2016.12, MATLAB 2018a, Dlib 19.18 e TensorFlow 2.0.

4.1 Construção da base de dados

Essa seção explica as etapas seguidas para a construção da base de dados, Kin3D. Começando com a coleta dos rostos dos participantes da pesquisa, em seguida a extração da informação de profundidade e geração de nuvens de pontos, registro e, por fim, obtenção de *features*.

4.1.1 Coleta das amostras

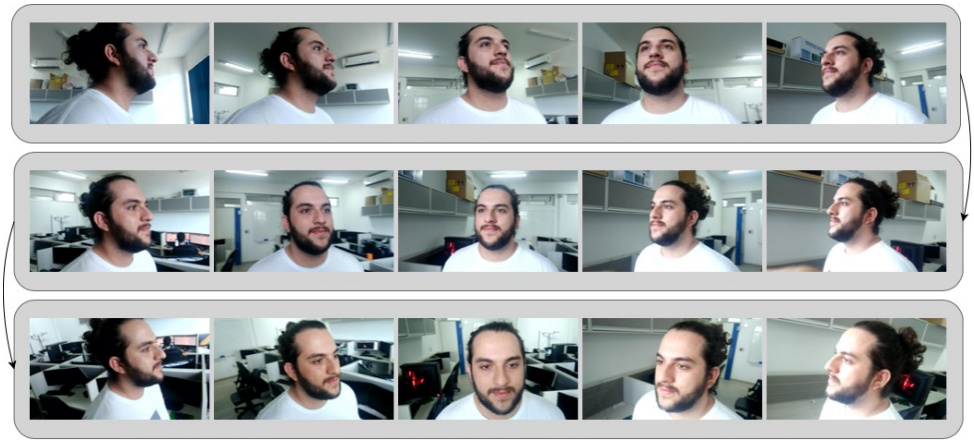
Primeiramente tentou-se obter reconstruções 3D, das bases de dados de parentes 2D, ao usar técnicas propostas na literatura (TRAN et al., 2018) para reconstruir malhas a partir de uma única imagem do rosto.

Dado o resultado não satisfatório, um formulário *online* foi criado para coletar imagens rotuladas dos usuários, juntamente com seus direitos de imagem para fins apenas acadêmicos. Através desse formulário, os usuários puderam enviar os vídeos do rosto deles e de seus parentes. Cada vídeo deveria incluir as regiões inferior, mediana e superior da face, de um perfil ao outro, como mostrado na Figura 13. Assim, detalhes que seriam sobrepostos em uma imagem estática frontal também seriam capturados.

Além dos vídeos, os usuários deviam informar os relacionamentos parentais dele com cada pessoa que ele também enviava. Não há restrições sobre as classes de relacionamento, sendo portanto permitido pessoas com mais de um grau na relação de ascendentes e descendentes. Visando um baixo custo e escalabilidade do nosso banco de dados e uma vez que também seria inviável as pessoas e seus parentes se deslocarem para o nosso laboratório, os vídeos são gravados através das câmeras de *smartphones* das pessoas nos seus próprios lares. No formulário é fornecido um vídeo demonstrativo sobre a gravação da face e alguns requerimentos são instruídos a fim de evitar falhas na reconstrução das faces:

- Não usar óculos, pois ele pode prejudicar a reconstrução 3D;

Figura 13 – Sequência de *frames* - Obtidos de uma lateral até a outra lateral da face e em 3 ângulos: superior, médio e inferior.



Fonte: Elaborada pelo autor.

- Ambiente iluminado, uma vez que quanto mais escuro o ambiente, menos visíveis as características do rosto se tornam;
- Deixar a resolução da câmera em HD ou Full HD.

4.1.2 Reconstrução

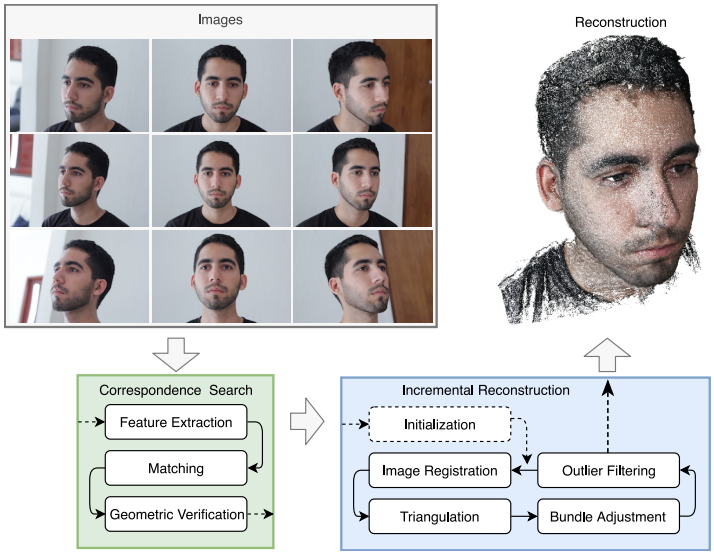
A partir de cada vídeo obtido no processo explicado na seção anterior, um número reduzido de *frames*, aproximadamente 70, foram extraídos de modo que eles sequencialmente cobrem todo o movimento, algumas dessas imagens são ilustradas na Figura 14. Cada imagem gerada tem sua largura redimensionada para 1080 *pixels* e sua altura é ajustada de acordo com a largura. Esses *frames* são utilizados na aplicação da técnica de *SFM* para que sejam geradas as imagens de profundidade e a nuvem de pontos.

O processo de reconstrução 3D, como ilustrado na figura 14, ocorreu de forma automática através do programa COLMAP (SCHöNBERGER et al., 2016; SCHöNBERGER; FRAHM, 2016) e esse processo foi executado em 3 partes:

- Extração e detecção de *features*;
- Verificação geométrica e associação de *features*;
- Reconstrução do movimento e da estrutura.

Alguns parâmetros do programa foram ajustados baseados no tamanho da imagem de entrada e no objetivo que é reconstruir faces. Portanto, são feitos ajustes como aumento da resolução de imagem no programa, aumento do raio da janela de correção e redução do *threshold* de filtragem.

Figura 14 – Fluxograma da técnica de estrutura a partir do movimento.



Fonte: Elaborada pelo autor.

No momento que a nuvem de pontos do rosto é gerada, também são gerados os mapas de profundidades. Ressaltasse que ao utilizar todas as imagens que contornam o rosto, é possível saber a parte do rosto que está mais perto ou distante da câmera. Portanto, imagens de profundidade geradas possuem *pixels* mais azuis ou mais vermelhos, dependendo da distância da parte do rosto com relação à câmera.

Ao fim desta etapa, as nuvens de pontos resultantes do processo apresentam ruídos que estão relacionados à iluminação, qualidade da câmera e distância entre a pessoa gravada e a câmera que altera conforme o usuário grava o rosto.

4.1.3 Limpeza e registro

Etapas de limpeza e registro são executadas nas imagens 2D -RGB e profundidade- e nas nuvens de pontos 3D. As próximas subseções explicam como são realizados esses processos.

4.1.3.1 RGB e profundidade

O alinhamento de faces consiste na identificação da estrutura geométrica das faces nas imagens digitais e a obtenção de um alinhamento canônico da face com base na translação, escala e rotação. No processo explicado por Rosebrock (2017), usando *affine transformation*, as faces se tornam centradas na imagem, os olhos residem em uma linha horizontal e elas são escaladas de modo que seus tamanhos são aproximadamente iguais.

Usando os *frames* em RGB mais frontais de cada pessoa, Dlib (KING, 2009) é capaz de encontrar a face na imagem e 68 *landmarks* presentes sobre ela. Uma vez que é conhecido a parte do rosto na qual esses pontos estão localizados, as coordenadas (x, y) do centro dos olhos são encontradas.

A diferença nas coordenadas (x, y), do centro de massa dos olhos, é calculada juntamente com a arcotangente para obter o ângulo de rotação entre os olhos, como mostrado no Anexo A, que permitirá corrigir a rotação da face. Ao ser convertido para graus, o ângulo entre os centroides dos olhos é retornado.

A coordenada do olho direito é encontrada utilizando o olho esquerdo como referência. Logo em seguida, é encontrada a escala da imagem desejada ao considerar a razão entre os olhos na imagem original e os olhos da imagem desejada.

Esses passos permitem conseguir as constantes para criar a matriz de rotação, MR, usando OpenCV: centro de rotação (centro dos olhos), ângulo de rotação (garantindo que os olhos residam em uma linha horizontal) e a escala (porcentagem que aumenta ou diminui a imagem). Essa matriz de rotação obtida usando as imagens RGB é reaproveitada para normalizar as imagens de profundidades equivalentes.

4.1.3.2 Nuvem de pontos

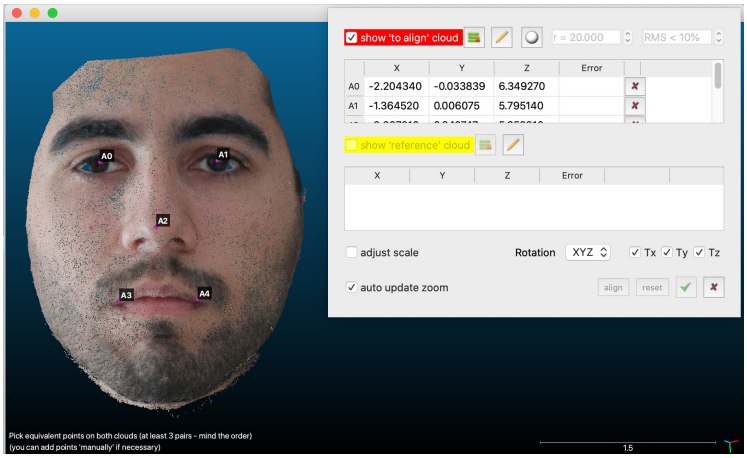
O primeiro passo no registro das faces é limpar os pontos ruidosos que surgem separados e mesclados à ela. Esses ruídos são às vezes causados por conta da iluminação que reflete sobre a face. Com ajuda do programa MeshLab (CIGNONI; CORSINI; RANZUGLIA, 2008), o primeiro tipo de ruído, separado da face, é selecionado no espaço e excluído, enquanto que o outro tipo de ruído, mesclado à face, é selecionado usando *color picker* e em seguida excluído.

Com as faces livres de pontos indesejados, o próximo passo é rotacionar, transladar e escalar. Primeiramente uma face é escolhida como referência e através do programa CloudCompare, 5 pontos ordenados são escolhidos na face. Como ilustrado na Figura 15, esses pontos correspondem ao: olho direito e esquerdo, nariz, canto esquerdo e direito da boca. A partir desses pontos na face de referência, outros 5 pontos são selecionados em outras faces, na mesma ordem, e o programa é capaz de aproximar, com uma tolerância, os 5 pontos da face nova com os 5 pontos da face de referência.

Nesta etapa do processo as nuvens de pontos estão aproximadas, contudo, elas ainda não estão alinhadas em um mesmo eixo e a quantidade de pontos é diferente. A fim de resolver isso, foi criado um algoritmo no MATLAB para executar os seguintes passos:

- Uma função de *affine3d* é utilizada nas faces, com a matriz: $mat = [roty(-30), [5.3, 0, 0]'; zeros(1, 3), 1]$, de modo a ficarem aproximadamente paralelas ao eixo-z;
- Uma função de redução de ruído (*pcdenoise*), com número de vizinhos igual à 40, é aplicada;
- Um *meshgrid* de tamanho 200×200 é criado e uma função de interpolação do MATLAB ($griddata$), com método de interpolação do tipo natural, é utilizada para aproximar os pontos da nuvem original nesse *mesh*;

Figura 15 – Seleção de 5 pontos ordenados em uma nuvem de pontos.



Fonte: Elaborada pelo autor.

- O mesmo processo de interpolação é usado para aproximar as cores RGB.

No fim da execução desses métodos, os pontos das nuvens estão alinhados no plano (x, y) e livre de espaços como buracos nas nuvens. A textura também se torna normalizada graças à interpolação.

4.1.4 Extração de *features* 3D

Uma vez que as nuvens de pontos são geradas e normalizadas, 16 características são extraídas em imagens 2D para compor o banco Kin3D, como ilustrado na Figura 16.

Para a etapa desta seção, uma biblioteca de Python chamada PyntCloud, para trabalhar com nuvens de pontos 3D, é utilizada e o número de k vizinhos mais próximos configurado foi igual à 10. Algumas dessas características são explicadas na Seção 2.2, as outras correspondem à:

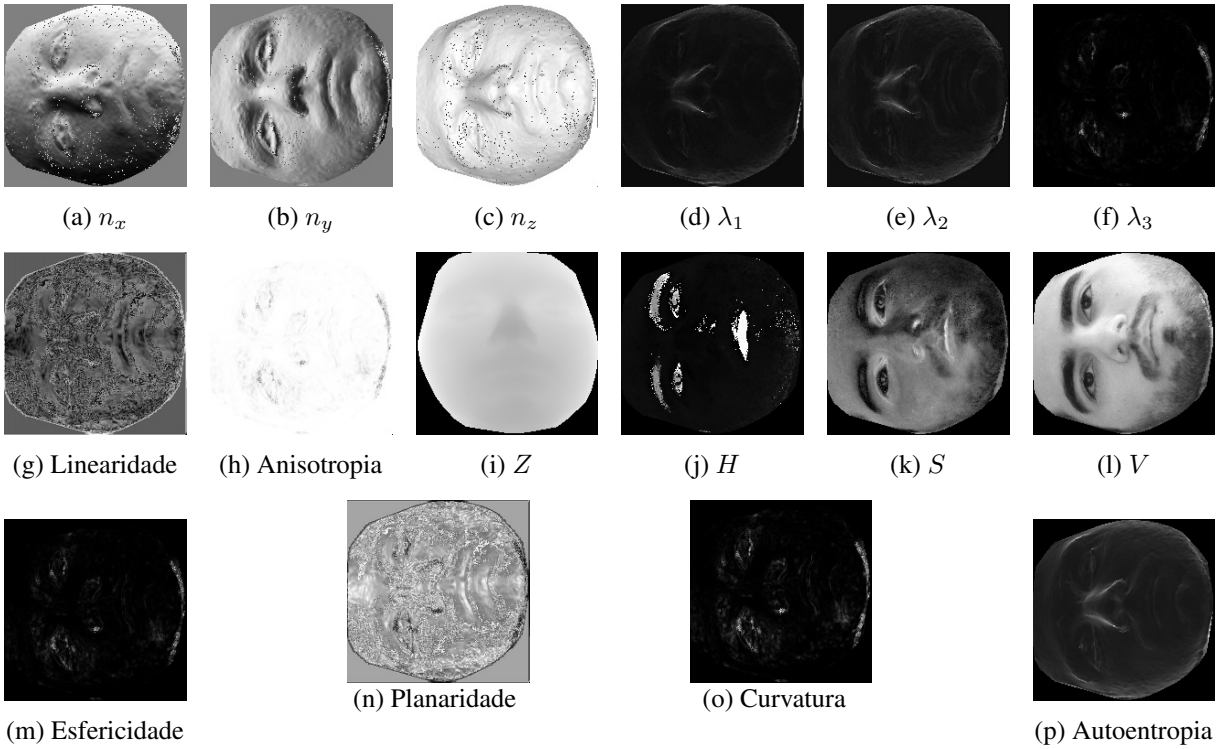
- n_x, n_y e n_z são os vetores normais associados aos pontos em (x, y, z);
- H (matiz), S (saturação) e V (valor) são os valores obtidos a partir do RGB de cada ponto;
- Z é a profundidade de cada ponto na nuvem;

4.2 Método I: Verificação de pais e filhos

A fim de avaliar os modelos de classificação consistentemente, alguns passos foram executados:

1. Foram reproduzidos os resultados de Zhang et al. (2015) a fim de avaliar a aplicabilidade de CNN em um dos bancos de dados mais utilizados (GUO; MA; LAN, 2018) para

Figura 16 – Características extraídas de uma nuvem de pontos.



(a) n_x (b) n_y (c) n_z (d) λ_1 (e) λ_2 (f) λ_3

(g) Linearidade (h) Anisotropia (i) Z (j) H (k) S (l) V

(m) Esfericidade (n) Planaridade (o) Curvatura (p) Autoentropia

Fonte: Elaborada pelo autor.

reconhecimento de parentesco em 2D, KinFaceW-I e KinFaceW-II (LU et al., 2014). Além disso, a fim de evitar *overfitting* do modelo por conta da quantidade baixa de amostras, aumento de dados (do inglês: *data augmentation*) foi utilizado como Zhang et al. (2015) propuseram.

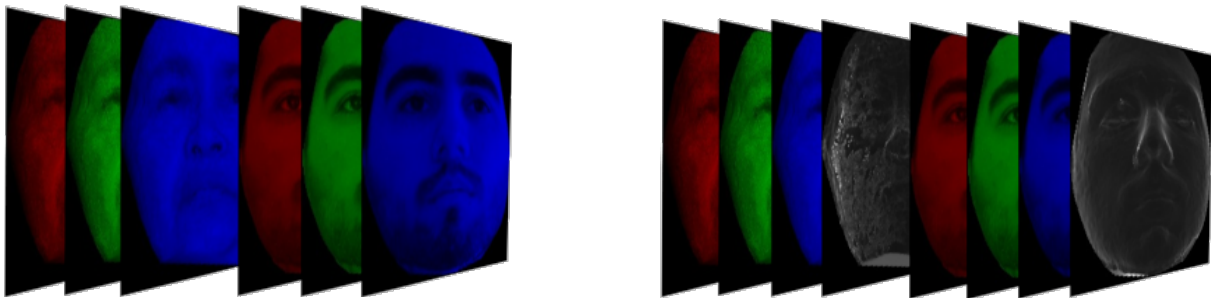
2. Uma vez que o Kin3D é o primeiro banco de dados desse tipo, foi testado um método de classificação bem conhecido para banco de dados não muito grandes - *Support Vector Machine* - nas amostras de imagens de face 2D do nosso banco de dados Kin3D, gerando um *baseline*. Para tanto, foi seguido o procedimento proposto recentemente por Thilaga et al. (2018);

3. Por fim, aplicou-se o modelo ajustado desenvolvido no item 1 para classificar o banco de dados Kin3D e avaliar se a profundidade influencia positivamente na acurácia do método. Vale ressaltar, que neste Método I, foram utilizadas as imagens RGB e RGBD, as *features* das nuvens de pontos foram utilizadas na metodologia da Seção 4.3.

Com relação as amostras na base, para a tarefa de verificação de parentesco, as imagens de um pai e um filho são concatenadas em série (ZHANG et al., 2015) aumentando sua profundidade, como ilustrado na Figura 17. Além disso, as imagens RGB e profundidade foram passadas para o classificador de duas formas: (i) $R_1G_1B_1R_2G_2B_2$ e $D_{R_1}D_{G_2}D_{B_3}D_{R_1}D_{G_2}D_{B_3}$ separados; ou (ii) $R_1G_1B_1D_1R_2G_2B_2D_2$ todos juntos. Da Seção 4.1.2, as imagens de profundidade são geradas

em RGB ou convertidas para escala de cinza. A Figura 17b é a representação de canais RGB mais um canal da imagem de profundidade em escala de cinza, resultando 8 canais. Quando as imagens de profundidade também estão em RGB, 12 canais são concatenados.

Figura 17 – Concatenação de canais - Em (a) 6 canais RGB são concatenados e em (b) 8 canais com RGB e alguma característica de profundidade D, em escala de cinza, são concatenados.



(a) $R_1G_1B_1R_2G_2B_2$ (b) $R_1G_1B_1D_1R_2G_2B_2D_2$

Fonte: Elaborada pelo autor.

Como retratado por Elmahmudi e Ugail (2019), Zhang et al. (2015), partes dos rostos podem ser trabalhadas separadamente a fim de combinar o processamento delas no final e conseguir um resultado ainda melhor. Portanto, como próximo passo, as imagens das faces inteiras são decompostas em 9 partes individuais, totalizando 10 imagens com a face inteira, como ilustrado na Figura 18. Cada uma das 10 partes são passadas para uma CNN diferente. Analisando de cima para baixo, elas correspondem à: face inteira, centro dos dois olhos, cantos laterais da boca, nariz e 4 cantos do rosto (mais abaixo à esquerda e à direita, mais acima à esquerda e à direita).

A estrutura da Rede Neural Convolucional (CNN) usada neste projeto consiste, inicialmente, em várias redes neurais menores concatenadas em paralelo. Primeiramente, a rede mais simples contém a camada de entrada que recebe as partes dos rostos do par de parentes redimensionada para 39×39. Em seguida, duas camadas convolucionais, com 32 neurônios, ambas seguidas por uma camada de *max-pooling*. Logo após a segunda camada de *pooling*, são adicionadas duas camadas totalmente conectadas com 128 neurônios cada uma, conforme mostrado na Tabela 4.

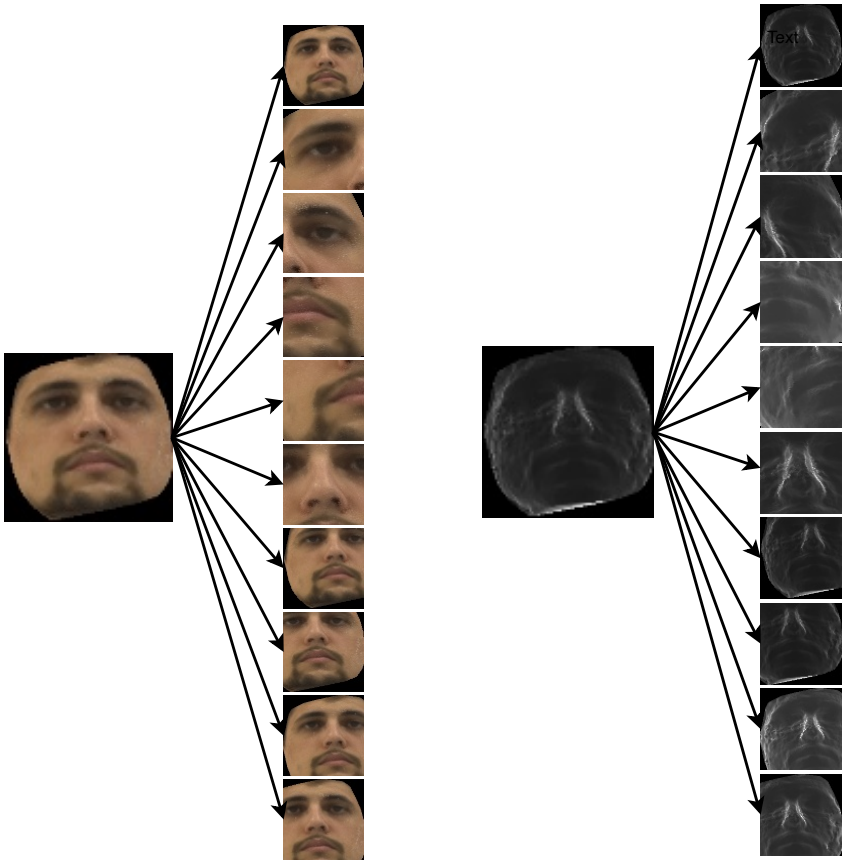
Tabela 4 – Rede básica antes da concatenação.

Camadas				
Conv1	Pool1	Conv2	Pool2	FC
conv-32	max-2	conv-32	max-2	FC1-128 FC1-128

Fonte: Elaborada pelo autor.

As camadas convolucionais são basicamente parametrizadas pelo o número de mapas, o tamanho dos mapas e o tamanho do filtro. Considerando o caso da Figura 17a, a primeira camada

Figura 18 – 10 regiões da face para alimentação de 10 CNNs.



(a) RGB (b) Curvatura

Fonte: Elaborada pelo autor.

convolucional recebe os canais RGB e profundidade, ambos com tamanho 39×39, e processa eles com 32 filtros de tamanho $5 \times 5 \times 6$. Esses filtros deslizam ou convoluem sobre a imagem de entrada com um *stride* de 1.

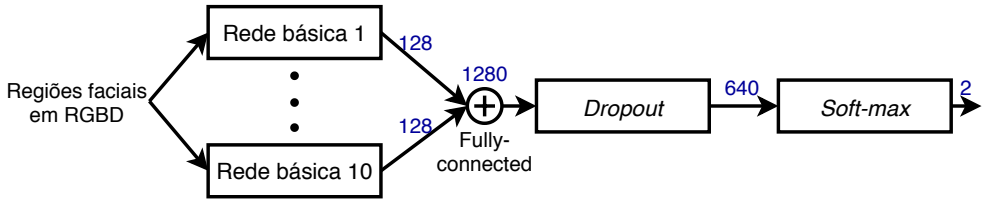
A fim de preservar o tamanho das imagens de entrada e o tamanho das imagens de saída na camada convolucional, um *padding* P foi configurado como "*same*" fazendo com que a camada use *zero padding* se necessário durante seu processo de convolução. Para inicializar os pesos dessas camadas convolucionais, utilizou-se uma distribuição normal com média zero e desvio padrão de 0.05. Uma função de Unidade Linear Retificadora (ReLU) foi usada como função de ativação.

As camadas de *pooling* são baseadas em operações como *max* e *average*. Foi escolhido o *max-pooling* porque ele impactou em melhores acurácias. Essas camadas têm filtros de tamanho 2×2 aplicados com um *stride* de 2, isso reduz a imagem de entrada ao longo da largura e da altura pela metade.

A concatenação dos canais das imagens em RGB e profunidade é avaliada em 2 topologias, ou seja, dependendo de como eles são apresentados para as redes, que foram da seguinte forma:

- Em série (Topologia A): Nesta abordagem, os canais das imagens em RGB e profundidade dos dois parentes são passados em série para 10 redes, divididos em 10 regiões faciais representadas na Figura 18. Cada região é passada para uma rede básica diferente no formato da Tabela 4. Portanto, existem 10 redes em paralelo e suas camadas totalmente conectadas são concatenadas, como ilustrado na Figura 19.

Figura 19 – Passagem das imagens RGB e profundidade (D) em série.

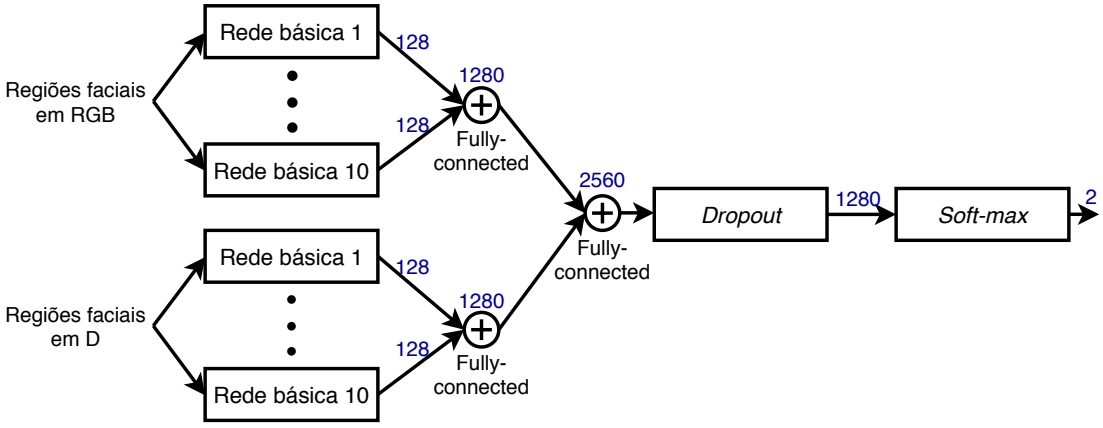


Fonte: Elaborada pelo autor.

- Em paralelo (Topologia B): Nesta abordagem, os canais das imagens em RGB e profundidade dos dois parentes são passados em paralelo para as redes, ambos divididos nas mesmas 10 regiões faciais. Além disso, uma vez que RGB e profundidade estão separados, existem 10 redes básicas responsáveis por RGB e mais 10 redes básicas responsáveis por profundidade. As camadas totoalmente conectadas das redes com RGB são concatenadas e o mesmo é feito para as redes com profundidade. A concatenação dessas redes também é concatenada, como ilustrado na Figura 20.

Após a concatenação final dessas topologias, uma camada de *dropout* é adicionada, reduzindo pela metade o número de neurônios, seguida por uma camada de saída. Para essa última camada, ela apresenta um neurônio por classe totalizando dois neurônios para parentes e não parentes. Uma função de ativação *softmax* é aplicada.

Figura 20 – Passagem das imagens em RGB e profundidade (D) em paralelo.



Fonte: Elaborada pelo autor.

O modelo é compilado com o otimizador de descida de gradiente estocástica (do inglês: *stochastic gradient descent* - SGD), configurado com *momentum* de 0.9 e taxa de decaímento de 0.005. Além disso, *binary crossentropy* foi utilizada como a função de perda.

Diante das classes de parentesco possíveis, foi adotada uma estratégia *one-versus-one* (OvO). Além dessa estratégia ser mais rápida, também é mais apropriada para bases pequenas. Adicionalmente, as quatro principais categorias foram utilizadas, ou seja, a CNN foi treinada para validar a relação entre um pai (ou mãe) e seu filho (ou filha). As amostras positivas e negativas tiveram a mesma quantidade de pares de parentes. Ademais, as amostras negativas foram geradas seguindo duas condições: (i) os membros das famílias não eram usados para gerar pares negativos com outros membros da mesma família; e (ii) todos os membros dos pares positivos eram usados para gerar os pares negativos.

4.3 Método II: Verificação de parentesco com parentes de primeiro grau

Na Seção 4.2 foi explicada a metodologia utilizada para classificar parentesco passando os dados de entrada na ordem, mãe-filho(a) e pai-filho(a) como muitos outros pesquisadores nessa área vêm procedendo (QIN; LIU; WANG, 2019). Contudo, nossa base Kin3D não possui tantas amostras quanto outras bases com informação de parentesco em 2D. Além das técnicas de aumento de dados, uma metodologia utilizada, nesse Método II, para aumentar ainda mais a quantidade de pares na nossa base foi reunir todas as categorias de parentes estabelecidas seguindo apenas uma restrição: cada pessoa possui uma relação de no máximo um grau de distância com seu parente. Sendo assim, um indivíduo foi apresentado para a rede neural com seu pai/mãe, irmão/irmã e filho/filha. Isso também permitiu lidar com o problema de verificação de parentesco de uma forma mais genérica. A rede continuou com a classificação binária, porém, com maiores relações de pessoas de sexo e idades diferentes.

Nesse Método II, utilizou-se as imagens das características extraídas das nuvens de pontos e não as imagens de profundidade. Uma vez que levaria muito tempo para testar todas as combinações possíveis dessas características, algumas delas como a curvatura foram selecionadas por conta da contribuição na literatura, como demonstra Zhou e Xiao (2018), e outras a partir de uma análise mais intuitiva. Essas combinações de características e a quantidade de canais, para cada face, são listados a seguir:

- RGB, $\lambda_1\lambda_2\lambda_3$ e $n_x n_y z_y$ possuem 3 canais. No caso da autoentropia que é 1 canal ela é convertida em uma mapa de cores de 3 canais;
- $RGB\_\lambda_1\lambda_2\lambda_3$ e $RGB\_n_x n_y n_z$ possuem 6 canais;
- $RGB\_curvatura$ e $RGB\_autoentropia$ possuem 4 canais, uma vez que curvatura e autoentropia foram mantidas em escala de cinza;

- RGB+$\lambda_1\lambda_2\lambda_3$, RGB+$n_x n_y n_z$, RGB+$curvatura$ e RGB+$autoentropia$ possuem 3 canais, RGB, e em paralelo 6 canais, características 3D. Curvatura e autoentropia foram convertidas em mapas de cores de 3 canais.

Sendo assim, ambas as Topologias A e B também foram avaliadas nessa nova metodologia.

4.3.1 Otimização

Para testar alguns processos de otimização como explicados na Seção 2.4, usando a CNN com a topologia do Método I, um conjunto de abordagens foram executadas na rede a fim de obter resultados ainda melhores na verificação de parentesco explicada na Seção 4.3. A seguintes subseções abordam essas otimizações que foram feitas sob restrições de *early stopping* para evitar o problemas de *overfitting*.

4.3.1.1 Topologia

Primeiramente, a rede foi treinada com apenas uma camada convolucional seguida por uma camada de *max-pooling*. As quantidades de filtros dessa camada convolucional variaram de 16, 32, 64 e 128 com tamanho 5×5. A inicialização dos pesos das camadas convolucionais permaneceu a mesma, ou seja, uma distribuição normal com média zero e um desvio padrão de 0.05. A mesma metodologia se repetiu para a adição da segunda camada convolucional e *max-pooling*. A melhor acurácia foi atingida quando utilizando 2 camadas convolucionais. Além disso, também foi avaliada a adição de 1 e 2 camadas totalmente conectadas após as camadas convolucionais, o melhor resultado foi obtido com apenas 1.

Além de aumentar a quantidade de dados no Kin3D, como mais uma maneira de evitar o *overfitting*, alguns métodos de regularização foram introduzidos como, por exemplo, *dropout* ou *batch normalization* entre as camadas convolucionais e regularização do tipo L_2. Não houveram mudanças significativas na acurácia ao menos quando foi adicionado *dropout* logo após a concatenação das redes.

4.3.1.2 Hiperparâmetros

Um dos hiperparâmetros testados foi a taxa de aprendizagem do modelo. No Keras, por padrão, o SGD apresenta a taxa de aprendizagem que diminui seguindo a fórmula

$$lr = lr_0 * \frac{1.0}{1.0 + dec * epoc},$$

onde:

- lr e lr_0 são a taxa de aprendizagem na época atual e anterior;

- dec é o decaimento;
- $epoc$ é o número de épocas.

O otimizador Adam (KINGMA; BA, 2017) substituiu o SGD utilizado previamente. Nesta nova metodologia na qual todos os parentes são apresentados à rede, SGD não obtinha bons resultados para ambas as topologias. Além disso, Adam é uma versão aprimorada do algoritmo de gradiente descendente ao incorporar tempo e taxa de aprendizado adaptativos considerando os parâmetros atuais. Sua taxa de aprendizagem padrão utilizada pelo TensorFlow (ABADI et al., 2015) é de 0.001. Como analisado por Wilson et al. (2018), a taxa de aprendizagem pode causar mudanças críticas no aprendizado da rede e embora convencionalmente seja sugerido que o otimizador Adam não requer ajustes, a mudança em sua taxa de aprendizagem inicial pode produzir melhorias significativas sobre suas configurações padrão. Portanto, os valores testados assumiram pequenas variações como: 0.05, 0.01, 0.005 e 0.0005.

Outro hiperparâmetro avaliado foi o tamanho do *batch*. Como retratado por Ioannidou et al. (2017), o tamanho do batch é normalmente configurado sendo potência de dois . Portanto, os valores escolhidos foram 16, 32, 64, 128 e 256.

5 RESULTADOS E DISCUSSÃO

Neste capítulo será abordado os resultados obtidos no desenvolvimento deste trabalho e a discussão sobre os pontos positivos e negativos encontrados.

5.1 Resultados

Nesta seção serão apresentados os resultados obtidos para a criação da base de dados Kin3D e os dois métodos utilizados para avaliar ela.

5.1.1 Base de dados

Os primeiros resultados na construção da base de dados podem ser vistos na Figura 21. Ao replicar o artigo de Tran et al. (2018), foi possível obter algumas reconstruções a partir de uma imagem como o artigo propõe. Contudo, as estruturas resultantes não possuem textura e o *shape* base que é usado por padrão torna as reconstruções bastante similares, principalmente de uma vista de perfil. Além disso, a técnica utilizada neste artigo é muito sensível a iluminação e sombreamento, tornando as reconstruções mais inchadas e fundas, respectivamente.

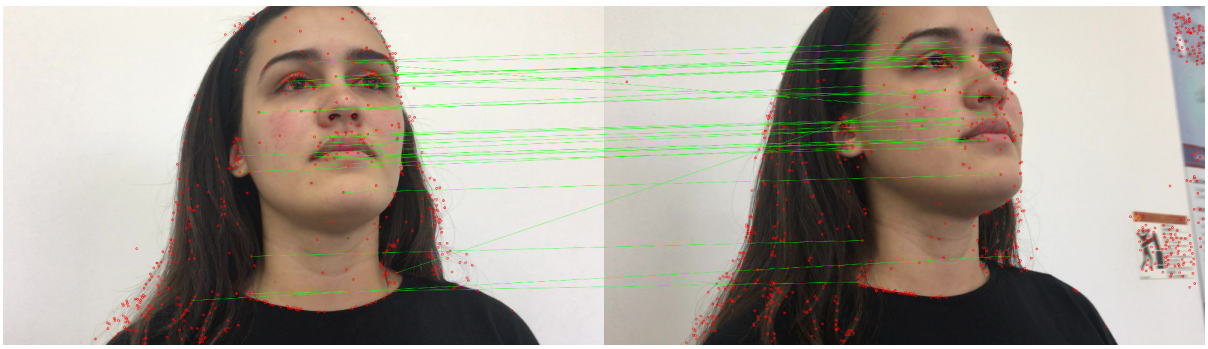
Figura 21 – Reconstrução 3D de uma única imagem 2D.

Fonte: Elaborada pelo autor.

Uma vez que as técnicas de reconstrução a partir de uma imagem não apresentaram resultados satisfatórios, o programa COLMAP (SCHöNBERGER et al., 2016; SCHöNBERGER;

FRAHM, 2016) foi utilizado para se obter a reconstrução 3D a partir dos *frames* extraídos dos vídeos enviados no formulário. O programa encontra as características das faces nas imagens e faz a associação delas entre os *frames*, como ilustrado na Figura 22.

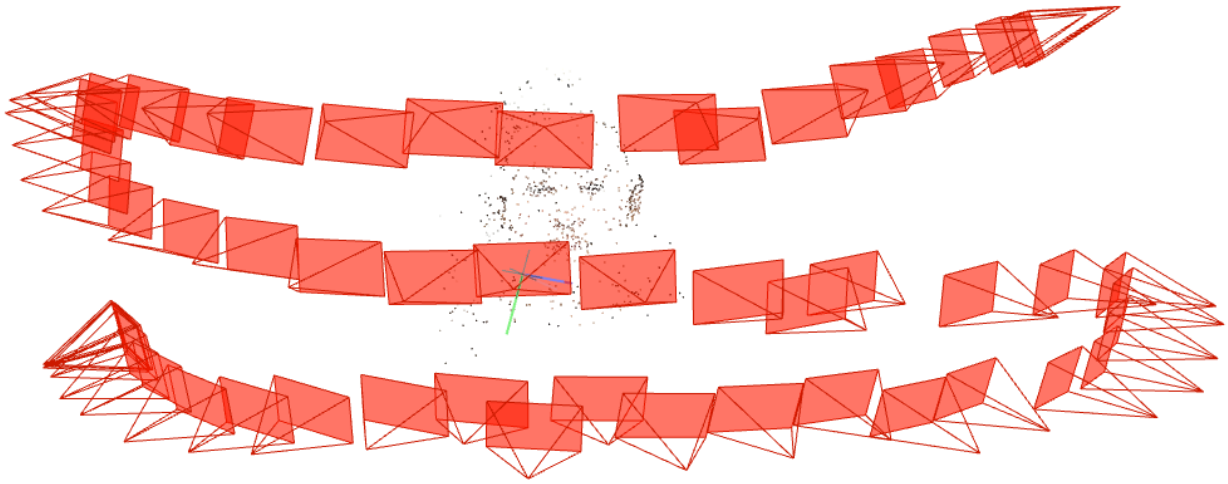
Figura 22 – Associação de características encontradas em *frames*.



Fonte: Elaborada pelo autor.

Após a associação de características, o programa executa uma reconstrução espaça e consegue obter os movimentos das câmeras dos vídeos gravados, como mostrado na Figura 23.

Figura 23 – Movimento da câmera - Obtido no processo de obtenção da estrutura a partir do movimento.

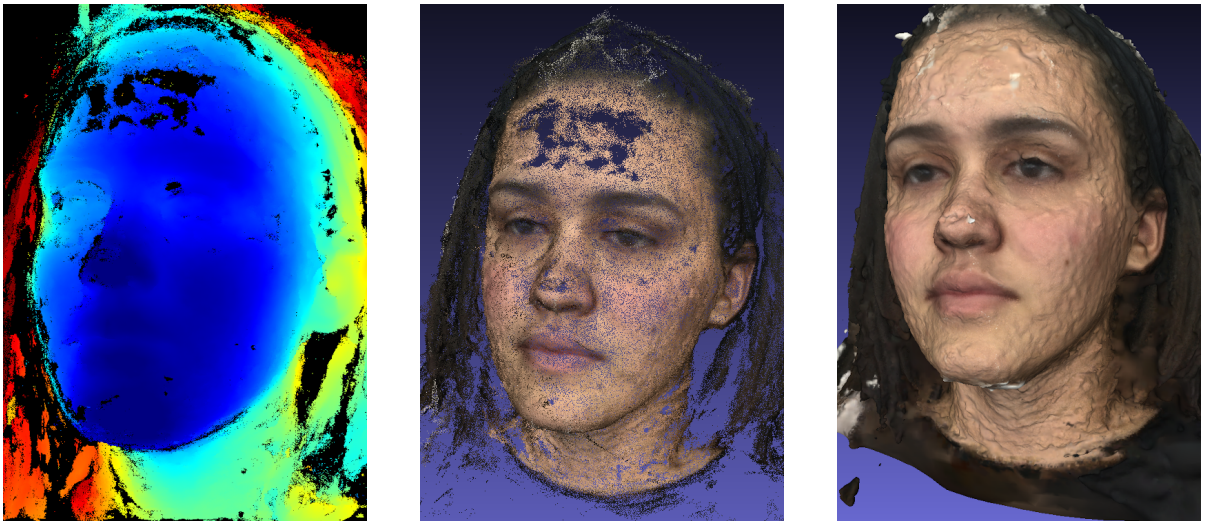


Fonte: Elaborada pelo autor.

O próximo passo foi a reconstrução densa. A partir dessa reconstrução, foi possível obtermos as imagens de profundidade, as nuvens de pontos e as malhas. Como ilustrado na Figura 24, o programa geralmente não consegue encontrar características suficientes em partes como a testa e o cabelo para a reconstrução da nuvem de pontos nessas regiões. Contudo, esses buracos puderam ser fechados durante a reconstrução da malha e no processo de normalização que foi executado no Matlab (MATLAB, 2019).

Com relação a nuvem de pontos, o pós processamento do COLMAP (SCHöNBERGER et al., 2016; SCHöNBERGER; FRAHM, 2016) gerou nuvens ruidosas que estão relacionadas principalmente ao ambiente e a qualidade da câmera. A limpeza das nuvens foi realizada de forma

Figura 24 – Resultados do pós processamento do programa COLMAP.



(a) Profundidade (b) Nuvem de pontos (c) Malha

Fonte: Elaborada pelo autor.

manual no MeshLab (CIGNONI; CORSINI; RANZUGLIA, 2008) e muitos pontos indesejados foram removidos, como ilustra a Figura 25.

Figura 25 – Limpeza da nuvem de pontos - Em (a) estão alguns pontos indesejados e em (b) a nuvem sem esses pontos.

(a) (b)

Fonte: Elaborada pelo autor.

No Matlab (MATLAB, 2019), a interporlação executada resultou em nuvens com pontos alinhados no plano (x, y), ou seja, sem espaçamentos irregulares ou buracos. Também, as nuvens apresentaram a mesma quantidade de pontos (200×200), como pode ser visto na Figura 26.

No entanto, mesmo depois desses passos algumas nuvens de pontos e suas texturas não obtiveram resultados satisfatórios, a Figura 27 ilustra alguns desses casos. Em (a), praticamente não é possível identificar o nariz e a região dos olhos, da mesma forma, a boca parece ter

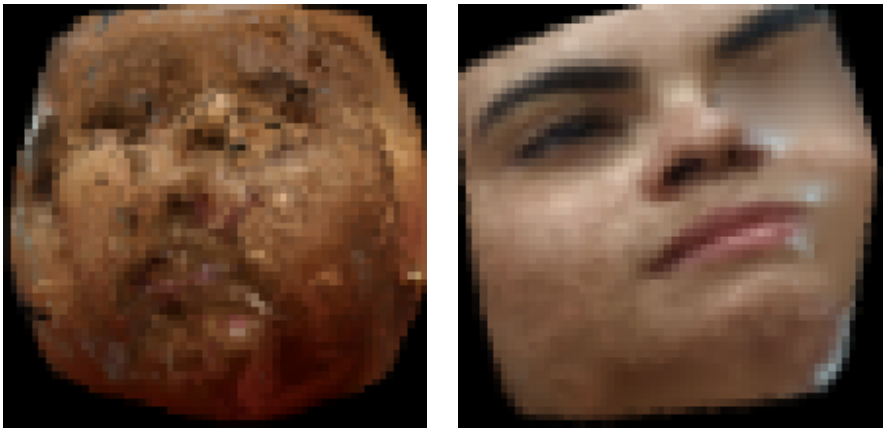
Figura 26 – Interpolação da nuvem de pontos - Em (a) antes da interpolação e (b) depois da interpolação.

(a) (b)

Fonte: Elaborada pelo autor.

se misturado um pouco com a região ao redor. Em (b), o olho esquerdo da mulher não foi reconstruído e portanto o processo de interpolação gerou pontos e texturas nessa região que possuem características próximas ao redor do olho.

Figura 27 – Nuvens malformadas.



(a) (b)

Fonte: Elaborada pelo autor.

Como resultado da participação das pessoas na pesquisa, até o momento da escrita dessa dissertação foram coletados e reconstruídos um total de 140 indivíduos com e sem parentes. Quando organizados por categorias, o número de pares foram como os descritos na Tabela 5. Até a ocasião da implementação do Métdodo I, havia um total de 61 pares de parentes incluindo as principais categorias de pais e filhos. Os experimentos executados com o Método II contaram com um total de 105 pares de parentes ao ser consideras as relações de até um grau entre eles.

Aumento de base também foi realizado como sugerido por Zhang et al. (2015), portanto, a quantidade de pares de parentes chegou a ser quadruplicada.

Tabela 5 – Número de pares de parentes para cada categoria de parentesco - Obtidos da base de dados Kin3D.

Categoria	Método I	Método II
Pai-Filho	14	16
Pai-Filha	9	9
Mãe-Filho	27	31
Mãe-Filha	11	13
Irmão-Irmão	-	18
Irmã-Irmã	-	6
Irmão-Irmã	-	12

Fonte: Elaborada pelo autor.

5.1.2 Método I: Verificação de pais e filhos

Primeiramente, foi replicada a rede neural de Zhang et al. (2015) a partir das bases de dados de KinFaceW (LU et al., 2014). A fim de obter a mesma faixa de acurácias que o referido trabalho, algumas mudanças foram feitas na rede: (i) com relação às camadas convolucionais, foram utilizadas somente duas; (ii) na inicialização dos pesos, a rede foi incapaz de aprender com os valores fornecidos pelo artigo de referência, portanto, foi usada uma distribuição Gaussiana com média 0 e desvio padrão 0.5. Como mostra a Tabela 6, os resultados obtidos foram próximos aos do artigo (ZHANG et al., 2015) nas categorias: pai-filho (PFo), pai-filha (PFa), mãe-filho (MFo) e mãe-filha (MFa).

Tabela 6 – Acurácias (%) das CNNS - Resultados obtidos com a base de dados KinFaceW (LU et al., 2014).

Métodos	KinFaceW-I				KinFaceW-II			
	PFo	PFa	MFo	MFa	PFo	PFa	MFo	MFa
Zhang et al. (2015)	71.8	76.1	78.0	84.1	89.4	81.9	89.9	92.4
Nosso método	69.2	77.8	72.6	83.8	86.2	83.3	84.1	88.8

Fonte: Elaborada pelo autor.

Como forma de desenvolver um *baseline* no Kin3D, foi usado um modelo de *support vector machine* (SVM). Esse SVM foi aplicado em vetores de faces de 128 dimensões, conforme descrito por Schroff, Kalenichenko e Philbin (2015), ao ser extraídas as características fornecidas pelo Dlib (KING, 2009). Para os pares das categorias Pai-Filho, Pai-Filha, Mãe-Filho e Mãe-Filha, um vetor foi formado ao encontrar a distância Euclidiana a partir deles. O resultado, com a médias dos 5 *folds*, podem serem encontrados na Tabela 7.

Tendo replicado e adaptado a CNN de Zhang et al. (2015), o próximo passo foi validar o nosso banco de dados Kin3D com essa CNN. As imagens são passadas e particionadas como na Figura 18. Os seguintes passos foram executados com a base Kin3D e as categorias Pai-Filho, Pai-Filha, Mãe-Filho e Mãe-Filha:

1. Primeiramente, foram utilizadas somente as imagens RGB e a Topologia A da Seção 4.2. Portanto, 6 canais foram passados (RGB dos pais e RGB dos filhos);

2. Em seguida, foram utilizadas somente as imagens de profundidade, Figura 24a, também seguindo a Topologia A;

3. Por último, a fim de testar a influência das imagens de profundidade com as imagens em RGB, foi avaliado o comportamento nas duas Topologias A e B (Seção 4.2), ou seja, canais RGB e profundidade foram passados para as redes em uma mesma imagem (série) ou imagens separadas (paralelo). Usando 5 *folds*, o modelo aprendeu pobremente com a Topologia A, contudo, após alimentar as redes com RGB e profundidade separadas, a rede conseguiu acurácias maiores.

A Tabela a seguir ilustra os principais resultados que são as médias dos *folds* para cada abordagem. O desvio padrão variou entre 2% e 5% nas quatro categorias.

Tabela 7 – Acurácias (%) do *baseline* e das CNNs - Obtidos com a base de dados Kin3D.

Métodos	**Kin3D**			
	PFo	**PFa**	**MFo**	**MFa**
SVM	69.9	56.6	66.6	73.8
CNN (RGB)	71.5	75.0	86.4	90.5
CNN (Profundidade)	62.4	59.6	63.2	66.7
CNN (RGB-D)	76.4	76.9	88.4	94.3

Fonte: elaborada pelo autor.

Os resultados da Tabela 7 indicam que ao incluir as imagens de profundidade a CNN conseguiu aumentar sua acurácia, ou seja, ao passar imagens em RGB e de profundidade em redes separadas, foi possível melhorar a verificação de parentesco. Um artigo relatando os resultados desse método foi submetido e aceito na conferência *Advanced Concepts for Intelligent Vision Systems* (CRISPIM; VIEIRA; LIMA, 2020). Nesse ponto, Kin3D é a primeira base de dados de parentesco com informações de profundidade avaliada em métodos de visão computacional.

5.1.3 Método II: Verificação de parentesco com parentes de primeiro grau

A partir do Método II, foi avaliado como a CNN utilizada no Método I se comportava diante de parentes de primeiro grau sendo passados para ela, ou seja, a junção de todas as categorias da Tabela 5. Após a interpolação, no Matlab (MATLAB, 2019), das cores RGB e dos pontos em xyz das nuvens de pontos, 16 imagens de características em escala de cinza foram extraídas das nuvens de pontos, Figura 16. A partir de uma análise visual dessas características, algumas combinações foram avaliadas. A Tabela 8 fornece as combinações de pares de parentes e os resultados obtidos ordenados pela média.

Da Tabela 8, pode-se notar que ao usar somente as características de profundidade, a rede neural não conseguiu extrair informações suficientes para uma boa verificação de parentesco.

Tabela 8 – Acurácias (%) das combinações de características das nuvens de pontos - Ambas as Topologias A e B da Seção 4.2 foram utilizadas.

Topologia	Combinações	Fold1	Fold2	Fold3	Fold4	Fold5	Média	Std
A	$RGB\_curvatura$	90.55	80.95	88.80	87.20	86.40	86.78	3.24
A	$RGB\_\lambda_1\lambda_2\lambda_3$	89.76	78.57	88.80	81.60	92.00	86.14	5.14
A	$RGB\_autoentropia$	85.83	80.16	88.00	86.40	89.60	85.99	3.20
B	RGB+$autoentropia$	84.25	81.20	82.61	86.46	88.01	84.50	2.47
B	RGB+$n_xn_yn_z$	87.40	78.57	84.80	84.00	86.40	84.23	3.07
B	RGB+$curvatura$	81.20	83.49	81.84	84.92	84.15	83.23	1.39
A	$RGB\_n_xn_yn_z$	80.45	77.30	81.96	88.78	83.71	82.57	4.61
B	RGB+$\lambda_1\lambda_2\lambda_3$	83.46	74.60	80.80	84.80	84.00	81.53	3.71
A	RGB	81.88	76.19	80.80	82.40	80.00	80.25	2.19
A	$\lambda_1\lambda_2\lambda_3$	55.11	59.52	62.40	56.80	57.60	58.28	2.49
A	$n_xn_yn_z$	71.65	53.17	53.60	53.60	58.40	58.08	7.05
A	autoentropia	53.54	53.17	53.60	53.60	53.60	53.50	0.16

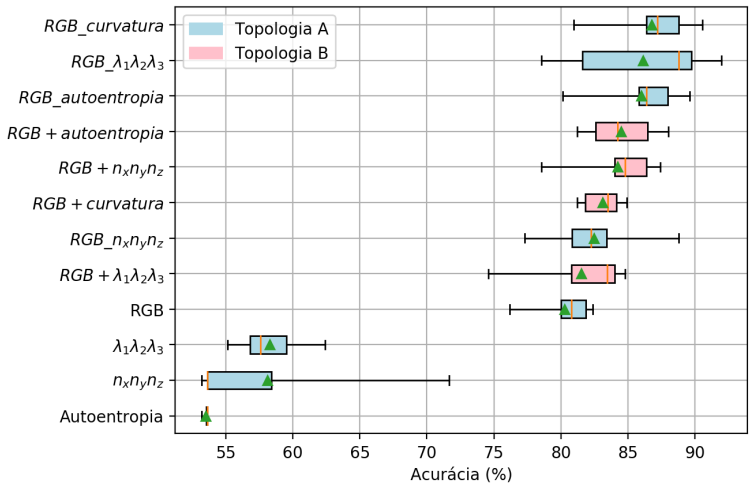
Fonte: Elaborada pelo autor.

Ao usar somente os canais RGB, a rede conseguiu atingir uma acurácia de 80.25%. Além dessa acurácia para RGB, também estão destacados em verde as maiores acurácias obtidas a partir da topologia A e B. Além do mais, quando foram testadas as combinações de cor (RGB) com informações de profundidade (D), todos os casos forneceram acurácias melhores do que quando avaliado somente com RGB. O que pôde-se notar também foi que ao contrário do Método I que obteve melhores resultados ao passar as informações RGB e informação de profundidade em redes neurais separadas, neste Método II foi notado que as melhores acurácias eram obtidas ao concatenar todos os canais em uma única imagem e passando todos na mesma rede, ou seja, usando a topologia A. A melhor acurácia obtida foi na concatenação de RGB e curvatura, 86.78%. Também estão destacados em vermelho as combinações da topologia A e B com maiores devios padrão e a combinação $n_xn_yn_x$ que foi a que apresentou o maior desvio padrão da tabela.

A Figura 28 ilustra o gráfico de caixas (do inglês: *box plot*) gerado a partir dos *folds* da Tabela 8. Esse gráfico deixa mais claro como o modelo se comportou nos *folds* através das combinações de RGB e as características 3D. Uma vez que as relações de parentes de primeiro grau não apresentavam quantidades de pares iguais quando combinadas, por exemplo, mãe-filho estavam mais presentes na base do que irmã-irmã, isso ocasionou que alguns *folds* tinham mais relações de um tipo do que de outros, embora as categorias de parentes e não parentes fossem praticamente iguais. Como resultado, a CNN encontrou um pouco mais dificuldade para aprender através de alguns *folds* em algumas combinações de características, essa situação pode ser vista através dos *outliers* presentes no gráfico de caixas. As combinações com dispersões mais consideráveis no intervalo interquartil foram apresentadas por $n_xn_yn_z$ e $RGB\_\lambda_1\lambda_2\lambda_3$. Adicionalmente, as medianas e as médias na maioria das combinações ficaram próximas, exceto para $n_xn_yn_z$ e quando $\lambda_1\lambda_2\lambda_3$ foi combinado com RGB em série ou paralelo.

A Figura 29 demonstra como a CNN avaliou os pares de parentes ao longo das épocas com RGB e curvatura. Pode-se notar que, a partir da época 40 a rede não consegue mais aumentar

Figura 28 – *Box plot* das combinações da Tabela 8.



Fonte: Elaborada pelo autor.

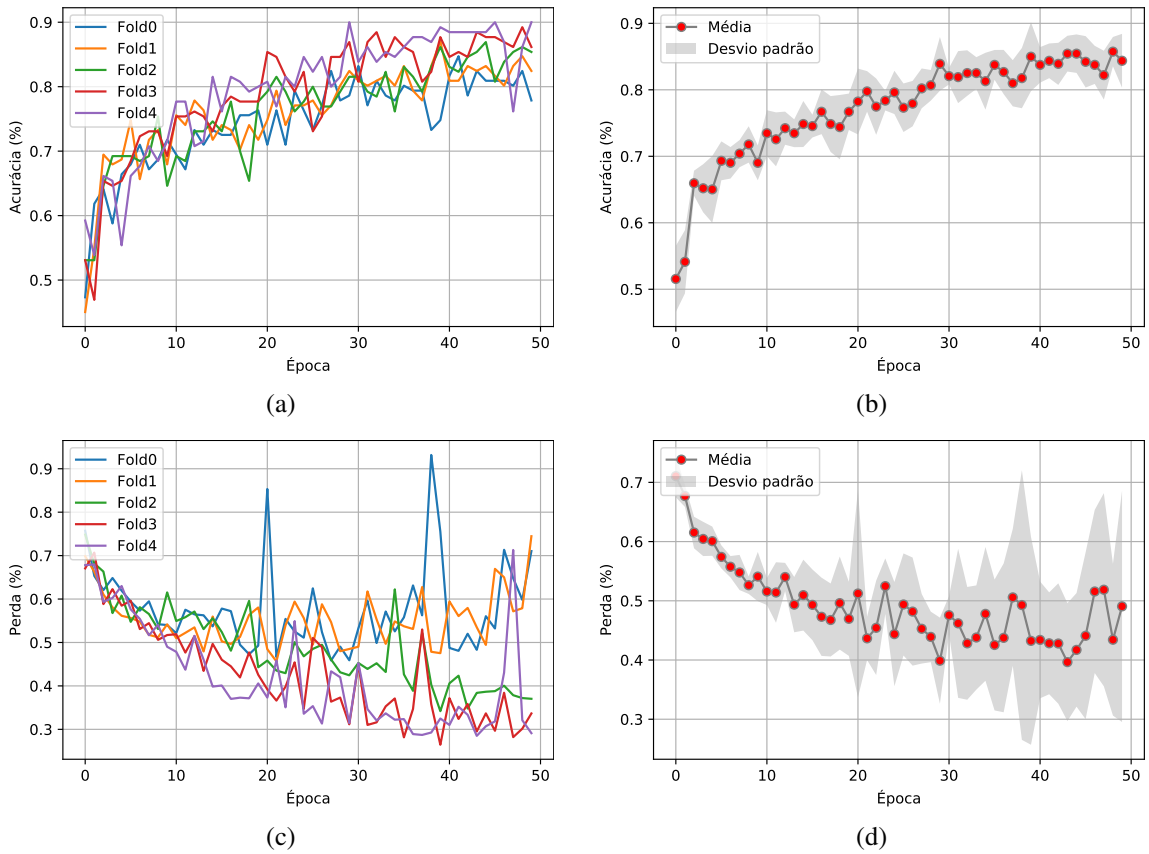
a acurácia adequadamente e percebe-se também que as dispersões entre as curvas se tornaram cada vez maiores sugerindo o surgimento de *overfitting*. O mesmo acontece nos gráficos da perda após época 35. Embora a base de dados tenha sido bem normalizada, foi esperado que a rede apresentasse algumas instabilidades no conjunto de validação, uma vez que a base não é tão grande e o problema é complexo. Ainda assim, as imagens indicam que a rede conseguiu verificar positivamente a diferenças entre parentes e não parentes ao longo das épocas.

Com a matriz de confusão obtida, Tabela 9, foi possível ter um entendimento melhor do número de vezes que as instâncias das classes de parentes e não parentes foram corretamente ou erroneamente classificadas. As seguintes métricas podem ser destacadas:

- A precisão é igual é 0.90 e a sensibilidade[1] é igual à 0.84. Isso indica que a CNN estava mais frequentemente correta quando o modelo previu uma amostra positiva do que ao encontrar as amostras de parentes positivas da base;
- A taxa de falsos positivos é igual à 0.1 indicando quão frequente o modelo previu pessoas como parentes quando elas não eram;
- A média harmônica F_1 é igual à 0.87, o que indica um balanço de quase 0.9 entre a precisão e a sensibilidade;
- A especificidade é igual à 0.89, ela está relacionada ao acerto das previsões das amostras de não parentes, além de estar próxima da sensibilidade;
- O número de falsos negativos é quase o dobro de falsos positivos, um indício desse resultado pode ser por conta do ambiente não controlado na coleta da base, o que faz com que os familiares possam ter algumas diferenças durante a aquisição das imagens;

[1] Também conhecida por *recall* ou taxa de positivos verdadeiros.

Figura 29 – Aprendizado da CNN (RGB com curvatura) ao longo das épocas no conjunto de validação - (a) e (b) são referentes às acurácias e (c) e (d) são referentes as perdas.



Fonte: Elaborada pelo autor.

- A prevalência é igual 0.52 uma vez que a quantidade de amostras das duas classes é quase a mesma na distribuição dos *folds*.

Tabela 9 – Matriz de confusão - Obtida a partir da validação de RGB com curvatura.

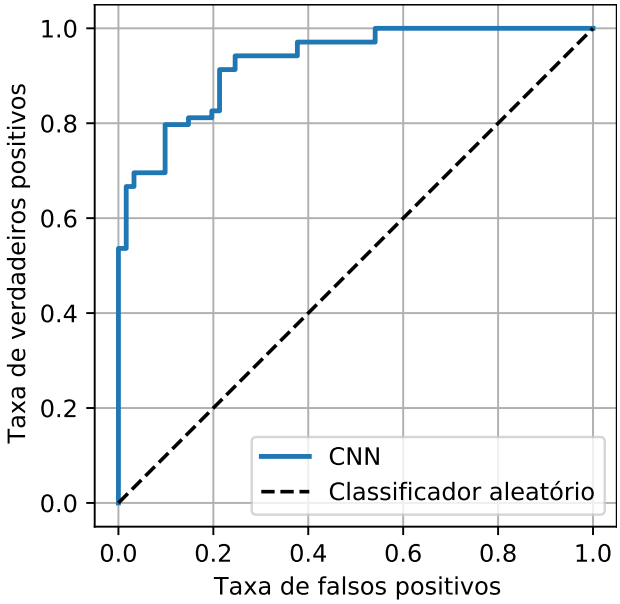
		Previsto	
		Não parentes	Parentes
Real	Não parentes	277	31
	Parentes	55	289

Fonte: elaborada pelo autor.

A curva ROC (*receiver operating characteristic*) também é um indicativo da performance de classificadores binários, ela plota a sensibilidade contra um menos a especificidade. Essa curva é mais utilizada quando os falsos negativos estão sendo levados mais em consideração do que os falsos positivos. Uma vez que o número de falsos negativos foi maior que o de falsos positivos, como informado na Tabela 9, uma curva ROC foi gerada dos resultados da CNN, como mostrado na Figura 30.

A partir da curva ROC, Figura 30, pode-se perceber que quanto maior a taxa de verdadeiros positivos mais falsos positivos o classificador produz. Através desse gráfico outra

Figura 30 – Curva ROC.



Fonte: Elaborada pelo autor.

medida como a área sob a curva (do inglês: *area under the curve* - AUC) pode ser extraída. Um classificador perfeito teria a área igual à 1, enquanto um aleatório teria 0.5. O gráfico da Figura 30 tem uma AUC igual à 0.93.

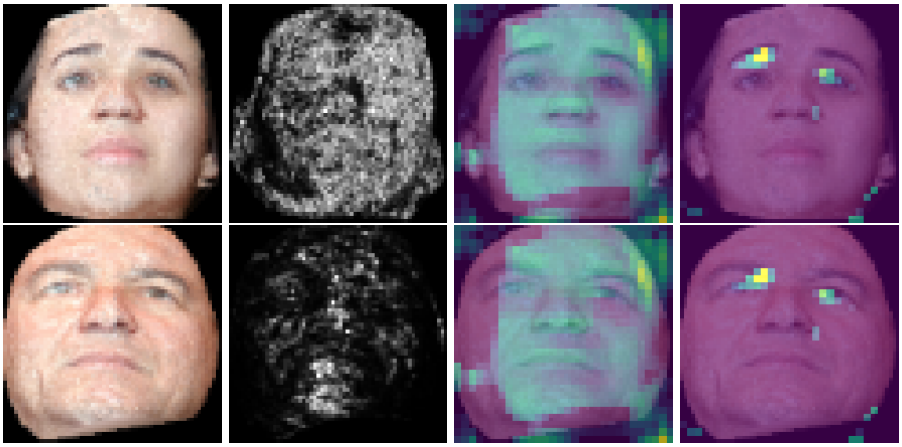
5.1.4 Influência ilustrativa da profundidade

Usando a técnica de Grad-CAM (SELVARAJU et al., 2019) para visualização dos mapas de calor foi possível obter uma visualização das características relacionadas à profundidade que afetaram a avaliação da CNN. As Figuras 31 e 32 mostram alguns casos na qual a CNN conseguiu mais ativações para o rosto inteiro e a imagem focada no nariz.

A Figura 31d mostra o caso quando foi usado somente RGB para a classificação. A rede não foi capaz de ser ativada tão fortemente como quando adicionada curvatura. Embora nas outras imagens que são focadas nas partes do rosto, desse par de parentes, houveram mais ativações utilizando somente RGB. Contudo, a curvatura permitiu a rede encontrar mais informações na face inteira. É possível notar também que a rede apresentou algumas ativações nos contornos do rosto e isso está relacionado ao fato que, normalmente, o contorno dos rostos apresentam altas curvaturas.

Geralmente os olhos nas imagens com os rostos completos não apresentaram ativações fortes independentemente da adição das características de profundidade. Além disso, para o caso da nossa base de dados, partes do rosto como sobrancelhas, nariz e boca, tiveram mais ativações na nossa rede, indicando que essas as informações são mais descritivas para análise de parentesco. Adicionalmente, pôde-se notar que ao usar somente RGB, pode tornar a rede sensível à iluminação. Em casos em que sombras estavam presentes nas faces, a rede não foi

Figura 31 – Ativações da rede nas faces - As figuras representam em (a) RGB, (b) curvatura, (c) mapas de calor ao usar RGB e curvatura e (d) mapas de calor ao usar apenas RGB.



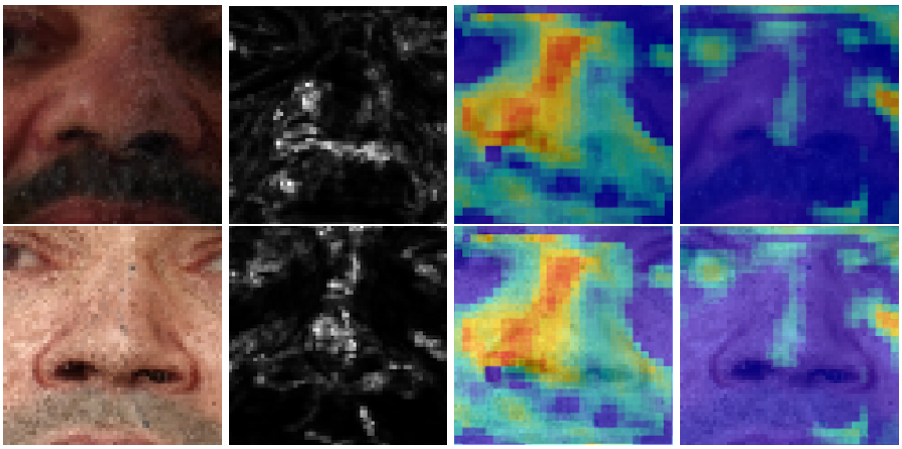
(a) (b) (c) (d)

Fonte: Elaborada pelo autor.

ativada nessas regiões.

As imagens dos rostos mostradas na Figura 31a e 31b são divididas em 9 partes que são alimentadas na rede. Dentre essas partes, a que mais teve fortes ativações ao adicionar curvatura foi a parte centrada no nariz, como ilustrada na Figura 32. Acredita-se que isso tenha sido ocasionado por conta que o nariz é uma das partes do rosto que possui maior curvatura. Isso também mostra como ao usar imagens de curvatura estimulou a rede à ativar mais em partes do rosto que somente com RGB não apresentava fortes contribuições.

Figura 32 – Ativações da rede nos narizes - As figuras representam em (a) RGB, (b) curvatura, (c) mapas de calor ao usar RGB e curvatura e (d) mapas de calor ao usar apenas RGB.



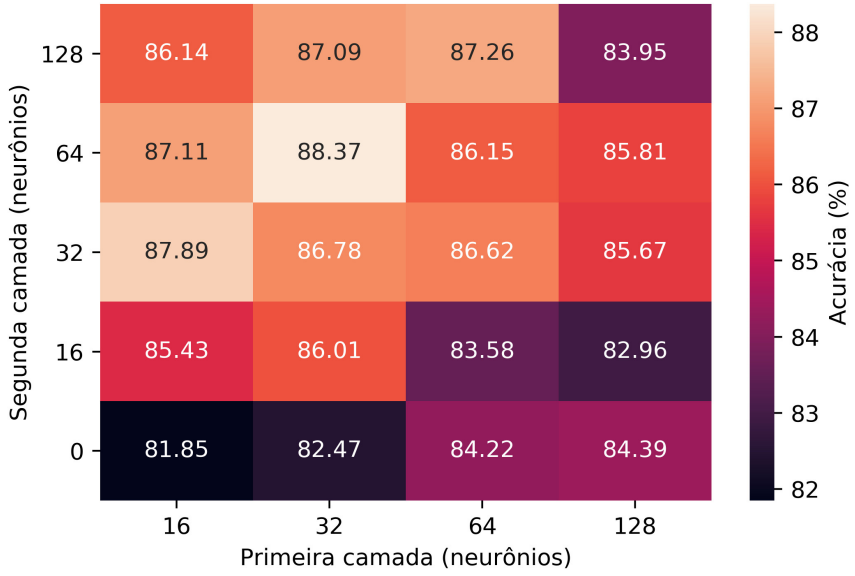
(a) (b) (c) (d)

Fonte: Elaborada pelo autor.

5.1.5 Otimização

Com o intuito de testar um processo de otimização, foi utilizado o melhor caso da Tabela 8 como base. Primeiramente foi avaliada a influência do número de neurônios na primeira e segunda camada seguindo Géron (2019). A rede anterior era composta por duas camadas convolucionais com 32 neurônios cada uma. Como ilustrado pela Figura 33, a melhor acurácia foi obtida com 32 neurônios na primeira camada e 64 na segunda camada. Pode-se notar através do gráfico que a adição de neurônios ou mais camadas não influenciam positivamente no aprendizado da rede neural.

Figura 33 – Relação entre o número de camadas e o número de neurônios - O desvio padrão varia entre 1.7% e 4.2%.



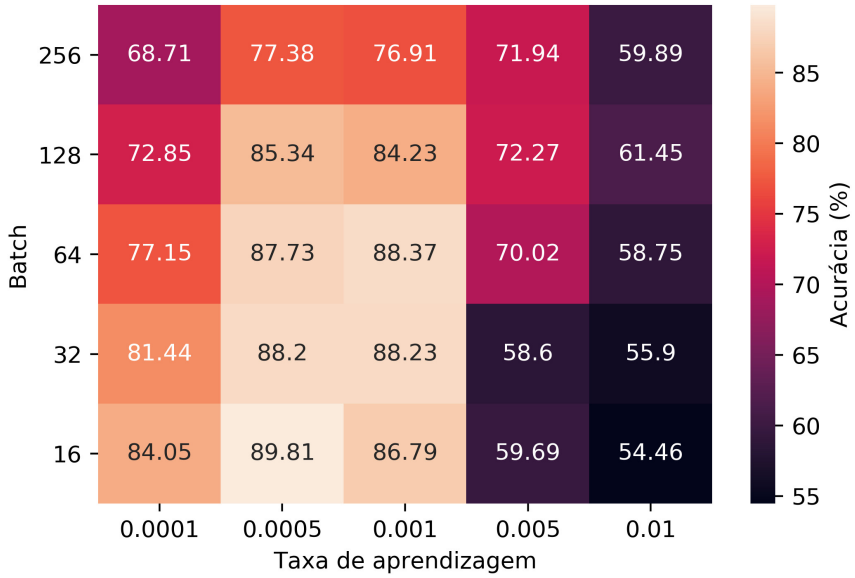
Fonte: Elaborada pelo autor.

Segundo Ioannidou et al. (2017), a taxa de aprendizagem e o tamanho do *batch* são alguns dos parâmetros que podem também impactar consideravelmente durante o treinamento da rede. Por padrão, o otimizador Adam do TensorFlow (ABADI et al., 2015) têm uma taxa de aprendizagem configurada para 0.001. Essa taxa foi alterada com passos de 0.0005. Com relação aos *batches*, foram utilizados tamanhos de 16, 32, 64, 128 e 256. A Figura 34 ilustra os resultados obtidos ao variar esses parâmetros. Para valores da taxa de aprendizagem mais altos como 0.01, a rede apresentou os resultados mais baixos. Para valores da taxa de aprendizagem mais baixos, a medida que o *batch* aumenta, a acurácia se torna pior. O melhor valor obtido foi com *batch* igual à 16 e taxa de aprendizagem igual à 0.0005.

5.2 Discussão

Nesta seção são apresentadas algumas discussões sobre os resultados obtidos nesta pesquisa que mostraram que ao adicionar informações relacionadas à profundidade em uma CNN, na tarefa de verificar parentesco, contribuem para o seu aprendizado.

Figura 34 – Relação entre a taxa de aprendizagem e o tamanho do *batch* - O desvio padrão varia entre 1.6% e 7.7%.



Fonte: Elaborada pelo autor.

5.2.1 Base de dados

Alguns métodos de visão computacional têm sido propostos na literatura a fim de gerar estruturas 3D a partir de uma única imagem 2D. Contudo, nota-se que seus resultados ainda apresentam muitos problemas a serem lidados. O artigo replicado nesta pesquisa lida com questões como oclusões e orientação. Contudo a falta de textura e o fato das malhas resultantes serem provenientes de um mesmo *shape*, não exibiram resultados satisfatórios que poderiam ser usados para VP. Ainda mais, a sensibilidade à luz da técnica ocasionou em depressões no rosto por conta de sombras na imagem.

Um método bastante utilizado é o de *strucutre from motion* (SFM). Mas quando lidando com parentes, não foi possível ir na casa dos participantes ou gerar um evento para chamar o familiares. Desse modo, a base de dados Kin3D foi coletada com ajuda dos participantes e seus próprios *smartphones*.

Dentre os desafios encontrados durante o treinamento da nossa rede neural, um deles esteve associado à como ocorreu a coleta da base de dados em ambientes não controlados. Esse tipo de situação acaba gerando bases mais difíceis de serem normalizadas e serem tratadas por métodos de aprendizagem de máquina. Se as amostras não forem processadas de forma mais adequada possível ao longo dos passos para registro, ruídos surgirão e dificultarão a validação do problema. Ainda assim, apesar da coleta ter sido feita por *smartphones* e por pessoas não especializadas, foi possível produzir uma base de dados com imagens em RGB e de profundidade, nuvens de pontos e *features* extraídas a partir delas, suficientes para o treinamento de CNNs.

Vale destacar que, como a maioria das outras bases de dados de parentesco, a nossa

base ainda possui uma restrição na qual os participantes apresentam algumas características em comum por conta da etnia. Portanto, é sabido que ao adicionar amostras de pessoas de outras origens e regiões (por exemplo orientais) podem tornar a generalidade da base ainda maior. A partir do nosso formulário online, esperamos que outras pessoas e pesquisadores ao redor do mundo possam contribuir no Kin3D.

5.2.2 Métodos I e II

Basicamente, quando lidando com informações oriundas de formas tridimensionais são encontradas na literatura: profundidades, nuvens de pontos ou malhas e características extraídas delas.

O objetivo da execução do Método I foi de validar a base de dados usando informações de profundidade. De modo geral, essa etapa da pesquisa se deu seguindo os passos do artigo de (ZHANG et al., 2015). Apesar desse artigo não ter fornecido todas informações necessárias para sua replicação e, além disso, não ter sido possível conseguir os mesmos valores, os resultados obtidos se encontraram na mesma faixa. A categoria Mãe-Filha atingiu uma acurácia maior que a deles, enquanto as outras três categorias ficaram abaixo, ambos nos dois bancos de dados de KinFaceW (LU et al., 2014). Com essa CNN, foi possível atingir acurácias maiores ao usar as imagens RGB e profundidade do Kin3D.

Ao final da execução do Método I, as características das nuvens de pontos foram extraídas e normalizadas. A fim de se aprofundar mais no problema de verificação de parentesco, foram utilizadas essas informações das nuvens de pontos e o problema passou a ser abordado de uma outra forma. Foram formados pares de parentes com até 1 grau e utilizado a mesma CNN do Método I para validar essas *features* e a capacidade da rede de encontrar informações que poderiam estar presentes em pessoas de sexos diferentes e margens de idades maiores. Além do mais, isso também foi motivado pelo fato de que modelos de aprendizagem de máquina estão cada vez mais tendendo à lidar com várias categorias de uma só vez, como pode ser percebido através dos desafios de *ImageNet Large Scale Visual Recognition Challenge* (RUSSAKOVSKY et al., 2015). Por sua vez, esse problema mais complexo tornou o aprendizado da rede mais instável, contudo, ao usar algumas abordagens e métodos foi possível amenizar alguns desses problemas típicos de CNN.

Através da análise dos resultados, foi possível também perceber que o número de falsos negativos foi maior que o de falsos positivos. Uma vez que a verificação de parentesco tem diversas aplicações, o impacto de haver mais falsos negativos ou positivos se torna dependente do tipo de problema. Por exemplo, na busca de uma criança desaparecida dentre milhares, o número de falsos positivos maior do que o de falsos negativos se torna mais interessante. Dado que seria melhor indicar com uma confiança maior que uma criança é parente de alguém, embora ela não seja, do que ter chance maior de descartar uma criança buscada. Após a busca, um exame como análise do DNA pode ser aplicado para conclusão do resultado. Contudo, no cenário de

redes socais, por exemplo Facebook, apresentar maiores chances de falsos negativos pode ser melhor. Na tarefa de recomendar um usuário como parente, recomendar parentes erroneamente pode manchar a imagem da empresa sobre sua confiabilidade e precisão. Portanto nesse caso, um número maior de falsos negativos soa mais interessante.

Ademais, pôde-se perceber que essas novas informações das nuvens de pontos também contribuíram para a acurácia na verificação de parentesco. De fato, Fasolt et al. (2019), na área de psicologia, analisaram como as informações tridimensionais conseguem contribuir para a verificação de parentesco ao usar avaliadores para julgar essas formas. Nós, na área de ciências da computação, analisamos como informações tridimensionais podem contribuir nas avaliações de um modelo de aprendizagem máquina no mesmo tipo de problema.

6 CONCLUSÃO

O objetivo principal que levou ao desenvolvimento desta pesquisa foi a investigação do uso de características relacionadas à profundidade na verificação de parentesco. Dados os avanços tecnológicos nas câmeras dos *smartphones*, *hardware* de computador e *software* de processamento de imagem, a adição de informações 3D às tarefas de visão computacional e aprendizado de máquina parece ser a melhor maneira de tornar essas tarefas mais precisas. Além disso, a falta de estudos relacionados à esse assunto na verificação de parentesco nos trouxe a oportunidade de sermos os primeiros à avaliar e publicar seu desempenho em modelos de aprendizagem de máquina o que motivou ainda mais o emprego dessa metodologia.

Nesta dissertação, foi respondida a questão de pesquisa inicial. Para alcançar este resultado foi necessário construir nossa própria base de dados, Kin3D, que além das informações fornecidas sobre relações de parentesco também fornece informações como idade e etnia que podem ser usadas para estudos relacionados à idade, síntese de rostos de crianças com base nos pais, entre outros.

No geral, os experimentos mostraram que informações de profundidade e características extraídas de nuvens de pontos contribuem para o desempenho do modelo. É reconhecido que, para representar melhor um problema amplo, como o de parentesco, é recomendável coletar um conjunto de dados ainda maior. No entanto, é esperado que esta pesquisa juntamente com a nossa base de dados possam fornecer uma contribuição inicial para a comunidade de pesquisa focada na verificação de parentesco e interessada em investigar mais o uso de informações 3D para encarar esta tarefa desafiante.

6.1 Trabalhos futuros

Durante o desenvolvimento desta dissertação algumas questões e possibilidades não foram consideradas, contudo, elas também se mostram boas oportunidades de trabalhos futuros:

- Avaliar com detalhes a influência do ambiente e do modo de aquisição da coleta das imagens. Esta dissertação mostrou que mesmo usando aparelhos de baixo custo foi possível fazer a reconstrução das faces, embora algumas vezes com uma qualidade não muito boa, mas é preciso ainda realizar um estudo mais profundo, por exemplo, como a distância da obtenção das fotos pode influenciar na reconstrução 3D;
- Aumentar o tamanho da base de dados. Modelos complexos de aprendizagem de máquina normalmente requerem grandes quantidades de amostras, dado a dificuldade de obtenção de imagens de parentes para reconstrução 3D deles, essa se torna uma tarefa contínua;

- Investigar se o uso de informações geométricas podem superar o viés associado à etnia que bases de dados somente com textura apresentam, quando elas possuem amostras de regiões próximas;
- Investigar o uso de estruturas mais complexas. O uso de nuvens de pontos ou *voxels* podem fornecer ainda mais detalhes relevantes para identificação de parentesco em CNN3D;
- Gerar parentes em 3D. O uso de redes adversárias generativas (GAN) é uma ótima abordagem a ser aplicada, uma vez que ao usar características 3D de pais poderia ser possível gerar uma criança e envelhecê-la de maneira mais realista a partir delas.

REFERÊNCIAS

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <http://tensorflow.org/>. Citado 3 vezes nas páginas 25, 48 e 60.

ALVERGNE, A. et al. Identification of visual paternity cues in humans. *Biology letters*, v. 10, p. 20140063, 04 2014. Citado 2 vezes nas páginas 14 e 34.

BLOMLEY, R. et al. Shape distribution features for point cloud analysis - a geometric histogram approach on multiple scales. *ISPRS annals*, II-3, p. 9–16, 2014. ISSN 2194-9050. Citado na página 20.

BOTTINO, A. et al. A new problem in face image analysis: Finding kinship clues for siblings pairs. *ICPRAM 2012 - Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, v. 2, 02 2012. Citado na página 32.

BRESSAN, P.; MARTELLO, M. F. D. Talis pater, talis filius: Perceived resemblance and the belief in genetic relatedness. *Psychological Science*, v. 13, n. 3, p. 213–218, 2002. PMID: 12009040. Disponível em: <https://doi.org/10.1111/1467-9280.00440>. Citado na página 34.

Castanon, G.; Byrne, J. Visualizing and quantifying discriminative features for face recognition. In: *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*. [S.l.: s.n.], 2018. p. 16–23. Citado na página 30.

CHOLLET, F. *Deep Learning with Python*. Manning Publications Company, 2017. ISBN 9781617294433. Disponível em: <https://books.google.com.br/books?id=Yo3CAQAACAAJ>. Citado na página 30.

CHOLLET, F. et al. *Keras*. [S.l.]: GitHub, 2015. <https://github.com/fchollet/keras>. Citado na página 24.

CIGNONI, P.; CORSINI, M.; RANZUGLIA, G. Meshlab: an open-source 3d mesh processing system. *ERCIM News*, v. 2008, 01 2008. Citado 3 vezes nas páginas 22, 40 e 51.

CLOUDCOMPARE. *Distances Computation*. dxomark, 2015. Disponível em: <https://www.cloudcompare.org/doc/wiki/index.php?title=Distances_Computation>. Citado na página 22.

CRISPIM, F.; VIEIRA, T.; LIMA, B. *Verifying Kinship from RGB-D Face Data*. [S.l.: s.n.], 2020. 215-226 p. ISBN 978-3-030-40604-2. Citado 2 vezes nas páginas 17 e 54.

DEBRUINE, L. M. et al. Kin recognition signals in adult faces. *Vision Research*, v. 49, n. 1, p. 38 – 43, 2009. ISSN 0042-6989. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0042698908004707>. Citado na página 34.

DJORDJEVIC, J. et al. Genetic and environmental contributions to facial morphological variation: A 3d population-based twin study. *PLoS ONE*, v. 11, 09 2016. Citado na página 35.

DORNAIKA, F.; ARGANDA-CARRERAS, I.; SERRADILLA, O. Transfer learning and feature fusion for kinship verification. *Neural Computing and Applications*, 04 2019. Citado na página 28.

DUAN, Q.; ZHANG, L.; ZUO, W. From face recognition to kinship verification: An adaptation approach. In: . [S.l.: s.n.], 2017. p. 1590–1598. Citado 2 vezes nas páginas 14 e 16.

ELMAHMUDI, A.; UGAIL, H. Deep face recognition using imperfect facial data. *Future Generation Computer Systems*, v. 99, p. 213 – 225, 2019. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X18331133>. Citado na página 43.

ERDOGMUS, N.; MARCEL, S. Spoofing 2d face recognition systems with 3d masks. In: . [S.l.: s.n.], 2013. p. 1–8. Citado na página 16.

Ertugrul, I. O.; Dibeklioglu, H. What will your future child look like? modeling and synthesis of hereditary patterns of facial dynamics. In: *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*. [S.l.: s.n.], 2017. p. 33–40. ISSN null. Citado na página 14.

FANG, R. et al. Kinship classification by modeling facial feature heredity. In: . [S.l.: s.n.], 2013. p. 2983–2987. ISBN 978-1-4799-2341-0. Citado na página 32.

FANG, R. et al. Towards computational models of kinship verification. In: . [S.l.: s.n.], 2010. p. 1577–1580. Citado na página 35.

FANG, R. et al. Towards computational models of kinship verification. In: *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010. p. 1577–1580. ISBN 978-1-4244-7992-4. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5652590http://chenlab.ece.cornell.edu/projects/KinshipVerification/>. Citado 2 vezes nas páginas 32 e 34.

FASOLT, V. et al. Contribution of shape and surface reflectance information to kinship detection in 3d face images. *Journal of Vision*, v. 19, p. 9, 10 2019. Citado 2 vezes nas páginas 35 e 63.

GANGULY, S.; BHATTACHARJEE, D.; NASIPURI, M. 3d face recognition from range images based on curvature analysis. *ICTACT Journal on Image and Video Processing*, v. 4, p. 748–753, 02 2014. Citado 2 vezes nas páginas 16 e 18.

GEORGOPOULOS, M.; PANAGAKIS, Y.; PANTIC, M. Modelling of facial aging and kinship: A survey. *Image and Vision Computing*, v. 80, 2018. Citado 3 vezes nas páginas 32, 35 e 36.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019. ISBN 9781492032618. Disponível em: <https://books.google.com.br/books?id=HHetDwAAQBAJ>. Citado 7 vezes nas páginas 25, 26, 27, 28, 29, 30 e 60.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, v. 9, p. 249–256, 01 2010. Citado na página 29.

GUO, Q.; MA, B.; LAN, T. Ensemble learning based on convolutional kernel networks features for kinship verification. In: *2018 IEEE International Conference on Multimedia and Expo (ICME)*. [S.l.: s.n.], 2018. p. 1–6. ISSN 1945-7871. Citado 2 vezes nas páginas 15 e 41.

Hu, J.; Lu, J.; Tan, Y. Discriminative deep metric learning for face verification in the wild. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 1875–1882. ISSN 1063-6919. Citado na página 36.

HU, J. et al. Large margin multi-metric learning for face and kinship verification in the wild. In: . [S.l.: s.n.], 2014. p. 252–267. Citado na página 36.

HU, Z. et al. Facial age estimation with age difference. *IEEE Transactions on Image Processing*, PP, p. 1–1, 2017. Citado na página 36.

IOANNIDOU, A. et al. Deep learning advances in computer vision with 3d data: A survey. *ACM Computing Surveys*, v. 50, 06 2017. Citado 5 vezes nas páginas 15, 26, 29, 48 e 60.

ITSEEZ. *The OpenCV Reference Manual*. 2.4.9.0. ed. [S.l.], 2014. Citado na página 23.

ITSEEZ. *Open Source Computer Vision Library*. 2015. <https://github.com/itseez/opencv>. Citado na página 23.

JEON, W.-S.; RHEE, S.-Y. Plant leaf recognition using a convolution neural network. *The International Journal of Fuzzy Logic and Intelligent Systems*, v. 17, p. 26–34, 03 2017. Citado na página 26.

KABBAI, L.; ABDELLAOUI, M.; DOUIK, A. Image classification by combining local and global features. *The Visual Computer*, v. 35, 2018. Citado na página 20.

KING, D. E. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, v. 10, p. 1755–1758, 2009. Citado 3 vezes nas páginas 24, 39 e 53.

KING, D. E. *dlib C++ Library: 19.7 Released*. 2017. <https://sourceforge.net/p/dclib/news/2017/09/dlib-c-library-197-released/>. [Online; accessed 10-January-2020]. Citado na página 24.

KINGMA, D. P.; BA, J. *Best Practices for Deep Learning for Science*. 2017. Disponível em: <https://www.arxiv-vanity.com/papers/1412.6980/>. Citado na página 48.

KOHLI, N. et al. Hierarchical representation learning for kinship verification. *IEEE Transactions on Image Processing*, v. 26, p. 1–1, 09 2016. Citado na página 32.

Kohli, N. et al. Supervised mixed norm autoencoder for kinship verification in unconstrained videos. *IEEE Transactions on Image Processing*, v. 28, n. 3, p. 1329–1341, 2019. Citado na página 32.

KORSHUNOV, P. *The 68 landmarks detected by dlib*. 2018. <https://www.researchgate.net/figure/The-68-landmarks-detected-by-dlib-library-This-image-was-created-by-Brandon-Amos-of-CMU_fig2_329392737>. [Online; accessed 10-January-2020]. Citado na página 24.

KRISS, M. Digital imaging: An introduction to image processing. In: . [S.l.: s.n.], 2015. Citado na página 14.

Lecun, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, Nov 1998. ISSN 1558-2256. Citado 2 vezes nas páginas 26 e 36.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. Convolutional networks and applications in vision. In: . [S.l.: s.n.], 2010. p. 253–256. Citado na página 27.

LOPEZ, M. B. et al. Kinship verification from facial images and videos: human versus machine. *Machine Vision and Applications*, p. 1–18, 05 2018. Citado 2 vezes nas páginas 15 e 35.

LU, J. et al. Neighborhood repulsed metric learning for kinship verification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 36, n. 2, p. 331–345, Feb 2014. ISSN 0162-8828. Citado 7 vezes nas páginas 10, 32, 33, 36, 42, 53 e 62.

MARTELLO, M.; MALONEY, L. Where are kin recognition signals in the human face? *Journal of vision*, v. 6, p. 1356–66, 02 2006. Citado 2 vezes nas páginas 34 e 35.

MARTELLO, M. F. D.; DEBRUINE, L. M.; MALONEY, L. T. Allocentric kin recognition is not affected by facial inversion. *Journal of vision*, v. 15 13, p. 5, 2015. Citado na página 34.

MATHWORKS. 2019. <https://www.mathworks.com>. Matlab R2019a User's Guide. Citado na página 23.

MATLAB. *What is MATLAB?* MATLAB, 2019. Disponível em: <https://www.mathworks.com/discovery/what-is-matlab.html>. Citado 5 vezes nas páginas 19, 23, 50, 51 e 54.

OZYESIL, O. et al. *A Survey of Structure from Motion*. 2017. Citado na página 19.

PRIYA, S. et al. 3d reconstruction of a scene from multiple 2d images. *International Journal of Civil Engineering and Technology*, v. 8, p. 324–331, 12 2017. Citado na página 15.

QIN, X.; LIU, D.; WANG, D. A literature survey on kinship verification through facial images. *Neurocomputing*, v. 377, 10 2019. Citado 6 vezes nas páginas 16, 17, 21, 32, 36 e 46.

Qin, X.; Tan, X.; Chen, S. Tri-subjects kinship verification: Understanding the core of a family. In: *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*. [S.l.: s.n.], 2015. p. 580–583. ISSN null. Citado 3 vezes nas páginas 32, 33 e 34.

REHM, L. *Disruptive technologies in mobile imaging: Taking smartphone cameras to the next level*. dxomark, 2018. Disponível em: <https://www.dxomark.com/disruptive-technologies-mobile-imaging-taking-smartphone-cameras-next-level/>. Citado na página 14.

Rehman, A. et al. Kinship verification using deep neural network models. In: *2019 International Symposium on Recent Advances in Electrical Engineering (RAEE)*. [S.l.: s.n.], 2019. v. 4, p. 1–6. ISSN null. Citado na página 36.

ROBINSON, J. P. et al. Families in the wild (fiw). *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*, ACM Press, 2016. Disponível em: <http://dx.doi.org/10.1145/2964284.2967219>. Citado 3 vezes nas páginas 32, 33 e 34.

ROBINSON, J. P. et al. Visual Kinship Recognition of Families in the Wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1, 2018. ISSN 0162-8828. Citado na página 36.

ROSEBROCK, A. *Face Alignment with OpenCV and Python*. PyImageSearch, 2017. Disponível em: <https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/>. Citado 2 vezes nas páginas 39 e 72.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. Citado na página 62.

SCHöNBERGER, J. L.; FRAHM, J.-M. Structure-from-motion revisited. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Citado 3 vezes nas páginas 38, 49 e 50.

SCHöNBERGER, J. L. et al. Pixelwise view selection for unstructured multi-view stereo. In: *European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2016. Citado 3 vezes nas páginas 38, 49 e 50.

SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 815–823. Citado na página 53.

SELVARAJU, R. R. et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, Springer Science and Business Media LLC, v. 128, n. 2, p. 336–359, Oct 2019. ISSN 1573-1405. Citado 2 vezes nas páginas 30 e 58.

Shao, M.; Xia, S.; Fu, Y. Genealogical face recognition based on ub kinface database. In: *CVPR 2011 WORKSHOPS*. [S.l.: s.n.], 2011. p. 60–65. ISSN 2160-7516. Citado na página 32.

SHU, X. et al. Kinship-guided age progression. *Pattern Recognition*, v. 59, 01 2016. Citado na página 14.

SUN, R. *Optimization for deep learning: theory and algorithms*. 2019. Citado na página 28.

Thilaga, P. J. et al. Modern face recognition with deep learning. In: *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. [S.l.: s.n.], 2018. p. 1947–1951. Citado na página 42.

TIDJANI, A. et al. Deep learning features for robust facial kinship verification. *IET Image Processing*, Institution of Engineering and Technology, v. 12, p. 2336–2345(9), December 2018. ISSN 1751-9659. Disponível em: <https://digital-library.theiet.org/content/journals/10.1049/iet-ipr.2018.5552>. Citado na página 15.

TRAN, A. et al. Extreme 3d face reconstruction: Seeing through occlusions. In: . [S.l.: s.n.], 2018. p. 3935–3944. Citado 2 vezes nas páginas 37 e 49.

TRIGGS, B. et al. Bundle adjustment – a modern synthesis. In: *VISION ALGORITHMS: THEORY AND PRACTICE, LNCS*. [S.l.]: Springer Verlag, 2000. p. 298–375. Citado na página 19.

VIEIRA, T. F.; BOTTINO, A.; ISLAM, I. U. Automatic verification of parent-child pairs from face images. In: SPRINGER. *Iberoamerican Congress on Pattern Recognition*. [S.l.], 2013. p. 326–333. Citado na página 15.

VOULODIMOS, A. et al. Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, v. 2018, p. 1–13, 02 2018. Citado na página 26.

WEI, X. et al. *DeepSFM: Structure From Motion Via Deep Bundle Adjustment*. 2019. Citado na página 18.

WILSON, A. C. et al. *The Marginal Value of Adaptive Gradient Methods in Machine Learning*. 2018. Citado na página 48.

WU, X. et al. On the usefulness of color for kinship verification from face images. In: . [S.l.: s.n.], 2016. p. 1–6. Citado na página 35.

WU, X. et al. *Kinship Verification Based on Deep Learning*. [S.l.: s.n.], 2019. 113-132 p. ISBN 978-981-10-5151-7. Citado 2 vezes nas páginas 26 e 36.

XIA, S.; SHAO, M.; FU, Y. Kinship verification through transfer learning. In: . [S.l.: s.n.], 2011. p. 2539–2544. Citado na página 32.

XING, J. et al. Diagnosing deep learning models for high accuracy age estimation from a single image. *Pattern Recognition*, v. 66, 01 2017. Citado na página 36.

XU, M.; SHANG, Y. Kinship verification using facial images by robust similarity learning. *Mathematical Problems in Engineering*, v. 2016, p. 1–8, 01 2016. Citado na página 15.

YAN, H.; HU, J. Video-based kinship verification using distance metric learning. *Pattern Recognition*, v. 75, 03 2017. Citado na página 32.

YAN, H.; LU, J. *Facial Kinship Verification*: A machine learning approach. 1. ed. [S.l.]: Springer Singapore, 2017. Citado na página 14.

ZHANG, K. et al. Kinship verification with deep convolutional neural networks. In: XIE, X.; JONES, M. W.; TAM, G. K. L. (Ed.). *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2015. p. 148.1–148.12. ISBN 1-901725-53-7. Disponível em: <https://dx.doi.org/10.5244/C.29.148>. Citado 7 vezes nas páginas 35, 36, 41, 42, 43, 53 e 62.

ZHOU, S.; XIAO, S. 3d face recognition: a survey. *Human-centric Computing and Information Sciences*, v. 8, p. 2192–1962, 2018. Citado 3 vezes nas páginas 16, 21 e 46.

ZHOU, X. et al. Kinship verification from facial images under uncontrolled conditions. In: . [S.l.: s.n.], 2011. p. 953–956. Citado na página 36.

ZHOU, X. et al. Gabor-based gradient orientation pyramid for kinship verification under uncontrolled environments. In: . [S.l.: s.n.], 2012. p. 725–728. Citado na página 36.

A ANEXO: ALINHAMENTO DE FACES EM PYTHON

Este anexo exibe o código criado por Rosebrock (2017) para o alinhamento de faces. Ele foi adaptado para alinhar as faces 2D da base de dados Kin3D.

```
def align(self, image, gray, rect):
        # convert the landmark (x, y)-coordinates
        # to a NumPy array
        shape = self.predictor(gray, rect)
        shape = shape_to_np(shape)
        # extract the left and right eye (x, y)-coordinates
        (lStart, lEnd) = FACIAL_LANDMARKS_IDXS["left_eye"]
        (rStart, rEnd) = FACIAL_LANDMARKS_IDXS["right_eye"]
        leftEyePts = shape[lStart:lEnd]
        rightEyePts = shape[rStart:rEnd]

        # compute the center of mass for each eye
        leftEyeCenter = leftEyePts.mean(axis=0).astype("int")
        rightEyeCenter = rightEyePts.mean(axis=0).astype("int")
        # compute the angle between the eye centroids
        dY = rightEyeCenter[1] - leftEyeCenter[1]
        dX = rightEyeCenter[0] - leftEyeCenter[0]
        angle = np.degrees(np.arctan2(dY, dX)) - 18

        # compute the desired right eye x-coordinate based on the
        # desired x-coordinate of the left eye
        desiredRightEyeX = 1.0 - self.desiredLeftEye[0]
        # determine the scale of the new resulting image by taking
        # the ratio of the distance between eyes in the *current*
        # image to the ratio of distance between eyes in the
        # *desired* image
        dist = np.sqrt((dX ** 2) + (dY ** 2))
        desiredDist = (desiredRightEyeX - self.desiredLeftEye[0])
        desiredDist *= self.desiredFaceWidth
        scale = desiredDist / dist

        # compute center (x, y)-coordinates (i.e., the median point)
        # between the two eyes in the input image
        eyesCenter = ((leftEyeCenter[0] + rightEyeCenter[0]) // 2,
            (leftEyeCenter[1] + rightEyeCenter[1]) // 2)
        # grab the rotation matrix for rotating and scaling the face
        M = cv2.getRotationMatrix2D(eyesCenter, angle, scale)
        # update the translation component of the matrix
        tX = self.desiredFaceWidth * 0.5
        tY = self.desiredFaceHeight * self.desiredLeftEye[1]
        M[0, 2] += (tX - eyesCenter[0])
```

```
        M[1, 2] += (tY - eyesCenter[1])

        # apply the affine transformation
        (w, h) = (self.desiredFaceWidth, self.desiredFaceHeight)
        output = cv2.warpAffine(image, M, (w, h),
            flags=cv2.INTER_CUBIC)
        # return the aligned face
        return output
```