CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Washington Tyrone Carvalhais

RECONHECIMENTO FACIAL APLICADO À SEGURANÇA UTILIZANDO TÉCNICAS DE HOG, SVM E CNN

Timóteo

2022

Washington Tyrone Carvalhais

RECONHECIMENTO FACIAL APLICADO À SEGURANÇA UTILIZANDO TÉCNICAS DE HOG, SVM E CNN

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Lucas Pantuza Amorim

Timóteo

2022

Washington Tyrone Carvalhais

RECONHECIMENTO FACIAL APLICADO A SEGURANCA UTILIZANDO TECNICAS DE HOG, SVM E CNN

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Trabalho aprovado, Timóteo, 15 de Junho de 2022.

_____

Prof. Dr. Lucas Pantuza Amorim
Orientador

_____

Prof. Dr. Bruno Rodrigues Silva
Professor Convidado

_____

Prof. Me. Odilon Correa da Silva
Professor Convidado

Emitido em 15/06/2022

ATA Nº 7/2022 - DCCTM (11.63.05)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 15/06/2022 21:29)
BRUNO RODRIGUES SILVA
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 1675935

(Assinado digitalmente em 15/06/2022 21:28)
LUCAS PANTUZA AMORIM
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 2897411

(Assinado digitalmente em 17/06/2022 14:25)
ODILON CORREA DA SILVA
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 2794495

(Assinado digitalmente em 15/06/2022 21:31)
Washington Tyrone Carvalhais
DISCENTE
Matrícula: 201517060109

Para verificar a autenticidade deste documento entre em https://sig.cefetmg.br/documentos/ informando seu número: **7**, ano: **2022**, tipo: **ATA**, data de emissão: **15/06/2022** e o código de verificação: **ae23dcbd91**

Agradecimentos

Primeiramente, agradeço à minha família, em especial a meus pais Weverton e Deborah, e ao meu irmão Weverton Duty, por sempre estarem ao meu lado servindo de alicerce para meus sonhos e conquistas. Agradeço ao meu professor orientador, Lucas Pantuza, pelo imenso apoio durante a Graduação e durante a realização deste trabalho, pelas pontuações sempre assertivas e pelas sugestões sempre enriquecedoras.

A todos os docentes do curso de Engenharia de Computação do CEFET-MG.

Por último, a todos os meus familiares e amigos que me auxiliaram nessa jornada e que sempre torceram por mim. E a Deus, que sempre se fez presente e que me colocou junto a todas estas pessoas muito especiais.

“Daria tudo que sei pela metade do que ignoro.”
René Descartes

Resumo

A detecção facial é uma técnica de visão computacional que, por meio da identificação de pontos de interesse, ângulos de captura, contingenciamento espacial e posicionamento, identifica em uma imagem a existência de uma face humana. A solução é capaz de realizar a identificação mesmo em cenas abarrotadas de informações, com relativa independência de fatores externos como ângulo de captura, iluminação e resolução de imagem. O objetivo deste trabalho é detectar e extrair características de face humana em imagens produzidas por câmeras de segurança, comparando-a com outras faces previamente conhecidas a fim de se identificar a pessoa. As imagens utilizadas são extraídas de vídeos de câmeras de segurança que são submetidas para detecção de faces usando o algoritmo HOG *(Histogram of Oriented Gradientes)* e SVM *(Support Vector Machine)*, que identificam regiões de interesse na face como queixo, lábio externo, lábio interno, sobrancelhas, olhos e nariz. Na etapa de identificação da pessoa, é usado redes neurais convolucionais (CNN), que comparam as faces identificadas com faces conhecidas de um grupo restrito de pessoas, resultando em uma ordenação por similaridade entre as imagens. Os resultados preliminares obtidos processando um vídeo de teste simulando câmera de segurança com resolução de 1080p foram acima de 90% na etapa de detecção das faces, e de acima de 95% na etapa de identificação da pessoa.

Palavras-chave: HOG, CNN, SVM, detecção facial.

Abstract

Face detection is a computer vision technique that, through the identification of points of interest, capture angles, spatial contingency and positioning, identifies the existence of a human face in an image. The solution is capable of performing identification even in scenes full of information, with relative independence from external factors such as capture angle, lighting and image resolution. The objective of this work is to detect and extract human face features in images produced by security cameras, comparing it with other previously known faces in order to identify the person. The images used are extracted from security camera videos that are submitted for face detection using the HOG *(Histogram of Oriented Gradients)* and SVM *(Support Vector Machine)* algorithm, which identify regions of interest in the face such as chin, outer lip, inner lip, eyebrows, eyes and nose. In the person identification step, convolutional neural networks (CNN) are used, which compare the identified faces with known faces of a restricted group of people, resulting in an ordering by similarity between the images. The preliminary results obtained by processing a test video simulating a security camera with a resolution of 1080p were above 90% in the face detection stage, and above 95% in the person identification stage.

Keywords: HOG, CNN, SVM, face detection.

Lista de ilustrações

Figura 1 – (a) Captura da região de interesse; (b) Correlação das linhas facias aplicadas à imagem. 13
Figura 2 – Etapas de um sistema de visão computacional. 15
Figura 3 – Componentes da automatização de um sistema de verificação facial. 15
Figura 4 – Resultados de detecção de amostra de nosso sistema de detecção humana em diferentes ambientes. 16
Figura 5 – (a) Amostras de treinamento positivo; (b) Amostras de treinamento negativo. 16
Figura 6 – Cadeia geral de extração de recursos. 17
Figura 7 – Representação de uma vizinhança 3×3 ao redor de um ponto de coordenadas (x,y) em uma imagem. 20
Figura 8 – Convolução de máscara de pixels sobre região da imagem. 20
Figura 9 – Máscaras 3×3 para o cálculo do filtro da média. 21
Figura 10 – (a) Imagem original; (b)-(f) resultados da aplicação do filtro da média com máscara de dimensões n x n, n = 3, 5, 7, 17, 31. 22
Figura 11 – Representação 3D da distribuição normal Gaussiana. 23
Figura 12 – (a) Imagem original; (b) Conversão para escala de cinza; (c) Resultados da aplicação do filtro Gaussiano com $\sigma = 3$. 23
Figura 13 – Representação da aplicação do filtro da Mediana. 24
Figura 14 – (a) Imagem original; (b) Imagem contaminada por ruído Gaussiano; (c) Resultados da aplicação do filtro da mediana com máscara de dimensão 3×3; (d) Resultados da aplicação do filtro da média com máscara de dimensão 3×3. 25
Figura 15 – (a) Imagem original; (b) Conversão para escala de cinza; (c) Resultados da aplicação do filtro de Laplace com máscara de dimensão 3 x 3; (d) Resultados da aplicação do filtro de Laplace com máscara de dimensão 5 x 5. 26
Figura 16 – Máscara de *pixels* 3×3. 27
Figura 17 – (a) Imagem original; (b) Conversão para escala de cinza; (c) Aplicação do filtro de Sobel em x; (d) Aplicação do filtro de Sobel em y. 27
Figura 18 – (a) Imagem original; (b) Conversão para escala de cinza; (c) Resultados da aplicação do filtro de Canny com valor da imagem alta = 0.5 e valor de imagem baixa = 0.05. 29
Figura 19 – *Kernels* de cálculo dos gradientes. 30
Figura 20 – *Kernels* de cálculo dos gradientes. 30
Figura 21 – Representação do funcionamento de uma Rede Neural Convolucional. 32
Figura 22 – Camadas de convolução de uma Rede Neural. 33
Figura 23 – Representação do posicionamento das câmeras 36
Figura 24 – Exemplo de imagem utilizada 38
Figura 25 – Reconhecimento facial por meio do HOG e Segmentação em Pontos Faciais 39
Figura 26 – Imagem da ROI original e com filtro Gaussiano 40

Figura 27 – Alinhamento facial 41
Figura 28 – Análise de performance em *multi-thread* da aplicação 42
Figura 29 – Separacão de grupos com nova inserção de dados 44
Figura 30 – Separação de semelhanças entre faces no banco e reconhecimento da ROI 44
Figura 31 – Câmeras posicionadas respectivamente à 4,5 metros e 2,5 metros dos pontos de referência para coleta 46
Figura 32 – Visão frontal e traseira das portarias da instituição 47
Figura 33 – Acurácia HOG 48
Figura 34 – Acurácia CNN 49
Figura 35 – Reconhecimento facial de aluno 49

Sumário

1 INTRODUÇÃO 12
1.1 Motivação 12
1.2 Justificativa 13
1.3 Objetivo Geral 14

2 TRABALHOS RELACIONADOS 15

3 FUNDAMENTAÇÃO TEÓRICA 18
3.1 Processamento Digital de Imagens 18
3.1.1 Pré-Processamento 18
3.1.2 Filtragem no Domínio Espacial 19
3.1.3 Suavização de Imagens no Domínio Espacial 20
3.1.4 Filtro da Média 21
3.1.5 Filtro Gaussiano 21
3.1.6 Filtro de Mediana 22
3.2 Segmentação 23
3.2.1 Operadores de Gradiente 24
3.2.2 Operador Laplaciano 25
3.2.3 Operador Sobel 26
3.2.4 Detector de Bordas de Canny 28
3.3 Representação e Descrição 28
3.3.1 Algoritmo *Histogram of Oriented Gradients* (HOG) 28
3.3.1.1 Pré-Processamento 29
3.3.1.2 Cálculo do Gradiente 30
3.3.1.3 Peso Ponderado de Cada Célula 31
3.3.1.4 Normalização de Blocos 31
3.3.2 Máquina de Vetores de Suporte (*Support Vector Machine* — SVM) 31
3.3.3 Redes Neurais Convolucionais — CNN 32
3.3.4 Algoritmo k-*Nearest Neighbors Algorithm* — KNN 33
3.3.5 Aprendizagem Supervisionada 34

4 PROCEDIMENTOS METODOLÓGICOS 35

5 DESENVOLVIMENTO 37
5.1 Detecção de faces 37
5.1.1 Aquisição das Imagens 37
5.2 Detecção de Pontos Facias 38
5.3 Povoamento do vetor de faces conhecidas por meio de uma CNN 38
5.4 Algoritmo proposto para identificação de faces 39

5.4.1 Pré-processamento 39
5.4.2 Aplicação do filtro Gaussiano 39
5.4.3 Aplicação do filtro de mediana 40
5.4.4 Rede Neural Convolucional 40
5.4.5 Alinhamento Facial 41
5.4.6 Paralelismo das CPUs 41
5.4.6.1 O Python no cenário *multi-thread* 41
5.4.7 Aprendizagem baseada em Instâncias 42
5.4.8 Avaliação preliminar do Algoritmo 43

6 RESULTADOS EXPERIMENTAIS 46
6.1 Validação - Portaria Residencial 46
6.2 Portaria CEFET-MG 46

7 CONCLUSÃO 50

Apêndices 50

Apêndice A ALGORITMOS UTILIZADOS NA IMPLEMENTAÇÃO DO TRABALHO 51

REFERÊNCIAS 54

1 Introdução

A habilidade de reproduzir características físicas e comportamentais humanas em sistemas computacionais tem sido um desafio na atualidade. Uma das habilidades humanas mais cotidianas é a capacidade de reconhecer outras pessoas em um meio. A compreensão de quais artifícios o cérebro humano utiliza para o reconhecimento facial ainda é alvo de estudo de psicólogos e neurocientistas em todo o mundo (NETO et al., 1997), sendo que estudos atuais mostram que em geral as pessoas são capazes de reconhecer cerca de 5.000 faces em média, apontando grandes diferenças individuais (JENKINS; DOWSETT; BURTON, 2018).

Uma das formas de um computador se assemelhar à metodologia na qual o cérebro humano realiza o processo de reconhecimento é por meio da detecção facial, que é uma técnica de visão computacional que objetiva facilitar a identificação de faces em imagens e vídeos. Além disso, essa detecção deve ser capaz de realizar a extração de características relevantes em uma região de interesse dentro de um cenário abarrotado de dados e informações heterogêneas, para assim, conseguir destacar objetos e/ou padrões em uma imagem com relativa independência de fatores externos (ângulos de captura, contingenciamento espacial, posicionamento e etc) (ARAÚJO, 2009).

Com o objetivo de aprender a reconhecer e distinguir faces de forma coesa, além de extrair informações como gêneros, expressões faciais e tons de pele, os sistemas precisam criar uma *Base de Conhecimento* (BC) para assim se tornarem especialistas na identificação e captura de características, por meio do treinamento na análise de imagens diversas (LAMBERT-TORRES; SILVA, 1994).

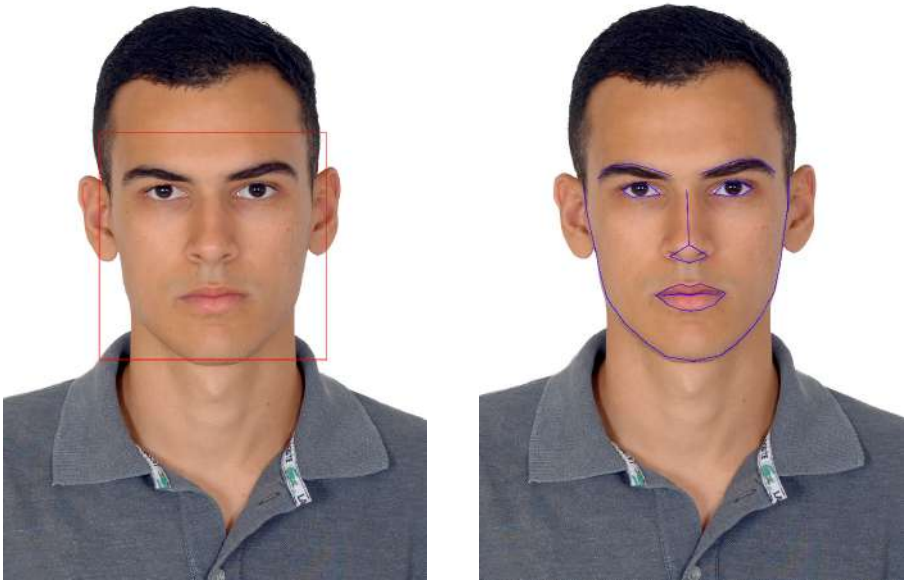
Tentando imitar o comportamento do cérebro humano, utilizamos como componente fundamental a estrutura de um neurônio artificial que, por meio de Redes de Neurônios Convolucionais (CNN), simulam a estrutura microscópica do cérebro em busca das características mais importantes (LAMBERT-TORRES; SILVA, 1994).

As faces humanas de forma geral possuem diversos padrões intrínsecos em sua composição de face, o que inclui a presença da linha do queixo, sobrancelha, ponte nasal, nariz inferior, olhos, e lábios. A implementação atual de sistemas computacionais busca determinar o padrão espacial da face na imagem por meio da identificação de pontos faciais de análise. A Figura 1 demonstra a captura da área de interesse e a subdivisão das linhas faciais em uma imagem.

1.1 Motivação

O Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) Campus Timóteo é uma instituição pública de ensino fundada e em 1998 em crescente ascensão, que oferece atualmente dois cursos superiores e sete cursos de ensino médio, e que em 2020 comportava 541 alunos ativos nos cursos técnicos e 265 alunos ativos nas engenharias, com

Figura 1 – (a) Captura da região de interesse; (b) Correlação das linhas facias aplicadas à imagem.



(a) (b)

Fonte: Elaborada pelo autor.

um total aproximado de 716 alunos. Uma das questões discutidas no campus é relacionada à segurança, uma vez que existe um grande fluxo de alunos e servidores, falhas de restrição de acesso podem ocorrer.

Atualmente a identificação é feita por meio do reconhecimento pelos próprios funcionários, o que é sujeito à falhas. Uma vez que a Visão Computacional é versátil, torna-se possível a identificação e reconhecimento facial dos indivíduos de forma automatizada. Isto implica em ganho de tempo real e qualidade na identificação do estudante e redução de recursos humanos. Consequentemente, a chance de entrada indevida é reduzida e há uma melhora na segurança dos internos e controle do fluxo.

1.2 Justificativa

Atualmente, a análise de entrada de pessoas nas dependências do campus do CEFET-MG em Timóteo é executada de forma não automatizada sendo realizada pelos funcionários, levando em conta apenas a identificação por meio de conhecimento prévio das pessoas. Desta forma, a autorização de entrada é dependente de valores subjetivos, como memória e comportamento humano, podendo variar de um colaborador de segurança para outro, sem padronização da metodologia para o controle do fluxo de entrada.

Mediante ao contexto apresentado, este projeto propõe uma solução de uma automação na identificação de faces de alunos e/ou servidores do CEFET-MG Campus Timóteo para sua entrada no Campus, realizando a captura da face e comparação com fotos já cadastradas.

A aplicação consiste em realizar a validação por meio de comparações em relação a

um banco de dados, para determinar se a pessoa analisada consta no banco de alunos e/ou servidores da instituição. E uma vez que a Lei Geral de Proteção de Dados - Lei 13.709/2018 endossa: a não aplicação da Lei em caso de tratamento de dados pessoais realizado para fins exclusivos de segurança pública e atividades de investigação e repressão de infrações penais (Artigo 4º, III, “a” e “d”), esse projeto pode ser utilizado com uma incremento do atual sistema.

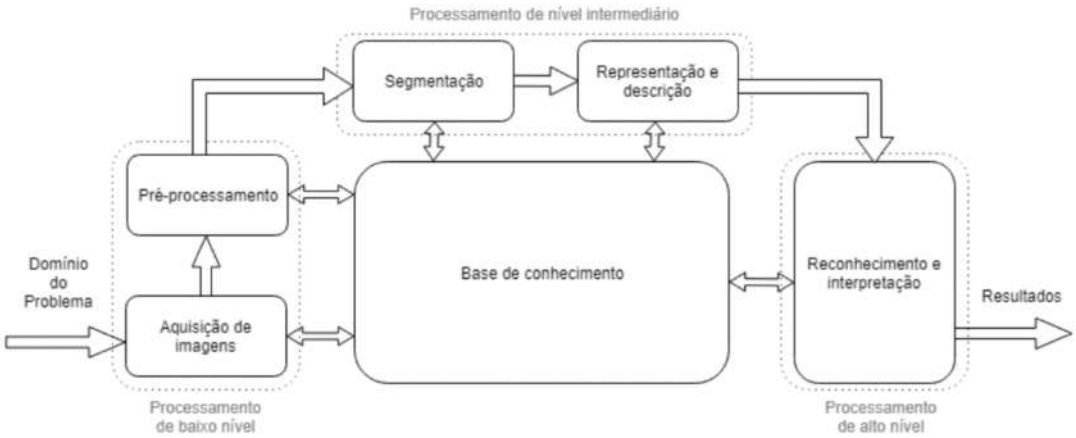
1.3 Objetivo Geral

Tendo em vista o problema apresentado, o objetivo deste trabalho é implementar técnicas para realizar a detecção e extração de várias características da face humana em imagens produzidas por câmeras de segurança instaladas no CEFET-MG, comparando-a com outras faces previamente conhecidas a fim de detectar e identificar as pessoas. O trabalho usa técnicas incrementadas de visão computacional com os recursos de *Deep Learning* do Dlib (KING, 2009), para assim realizar a construção de um detector, reconhecedor e identificador de faces.

2 Trabalhos Relacionados

Na literatura, são vários os exemplos de trabalhos que buscam extrair, em meio a um cenário abarrotado de informações áreas de interesse e atribuir ao computador a tarefa de reconhecer e interpretar informações em imagens, o que é denominado Visão Computacional (ARAÚJO, 2009). As etapas de um sistema de visão computacional podem ser exemplificadas pela Figura 2, conforme a definição de GONZALEZ; WOODS.

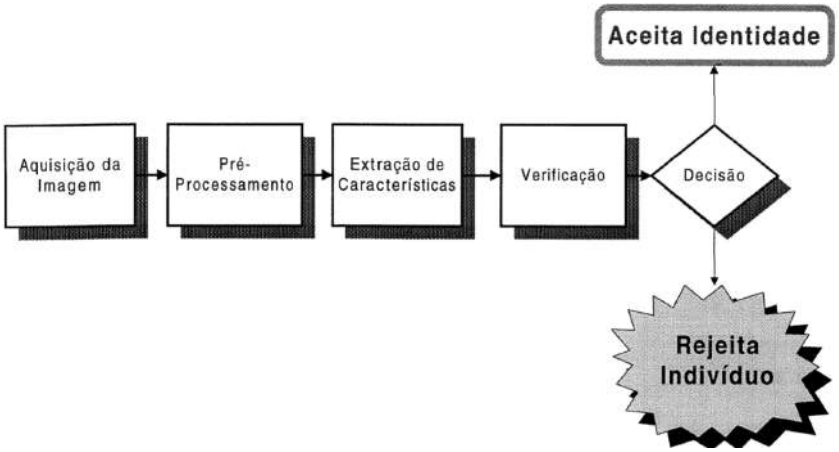
Figura 2 – Etapas de um sistema de visão computacional.



Fonte: (GONZALEZ; WOODS, 2000).

CONCI; AZEVEDO; LETA também define a subdivisão do processo de um sistema de visão computacional em etapas, começando pela digitalização, restauração/realce, segmentação, extração de características, classificação/reconhecimento e, por fim, em meio de um grupo de dados, a tomada de decisão. NETO et al. ilustra os componentes da automatização de um sistema de verificação facial segundo a Figura 3.

Figura 3 – Componentes da automatização de um sistema de verificação facial.



Fonte: (NETO et al., 1997).

Kelvin Lee et al. propõe identificar e classificar pessoas envolvendo o pré-processamento

e segmentação da imagem, classificando-as integralmente como humana ou não humana usando a estimativa da proporção do corpo, o método da janela deslizante da região local e o descritor humano de um histograma de gradientes orientados (HOG- *Histogram of oriented gradients*). Neste trabalho, as fotos são tiradas por câmeras de segurança e processadas em tempo real, além de realizar a captura dos frames, o autor também precisa corrigi-las, pois são afetadas por algumas variáveis externas (como luz e brilho). O processo desenvolvido para o treinamento e captura utilizado neste trabalho é ilustrado nas Figuras 4 e 5.

Figura 4 – Resultados de detecção de amostra de nosso sistema de detecção humana em diferentes ambientes.

Fonte: Adaptado (Kelvin Lee et al., 2010).

Figura 5 – (a) Amostras de treinamento positivo; (b) Amostras de treinamento negativo.

(a)

(b)

Fonte: (Kelvin Lee et al., 2010).

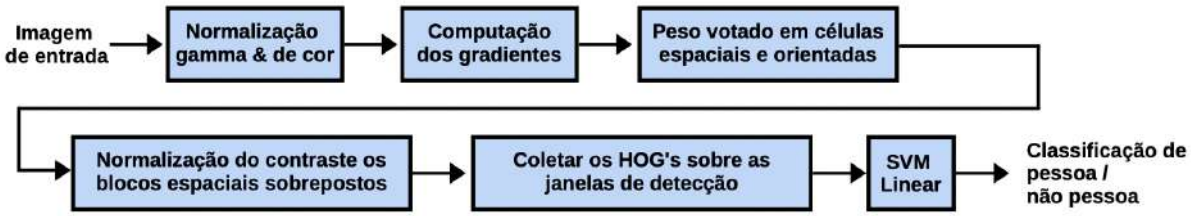
Zhang; Wang utiliza-se de histogramas aprimorados de recursos de gradientes orientados (HOG'S) para representar as informações das bordas de imagens para detecção de pessoas e rastreamento de objetos com acompanhamento em tempo real.

Em relação a quantidade de exemplos para o treinamento com o algoritmo do k-ésimo vizinho mais próximo (KNN — k-*nearest neighbors algorithm*) FERRERO propõe um método de armazenamento de amostras, onde apenas as mais representativas de cada classe são

armazenadas, e as informações mais importantes são agregadas em um conjunto menor de objetos de aprendizagem.

DALAL; TRIGGS adota o método de detecção humana baseada em uma máquina de vetores de suporte (SVM — *Support vector machine*) linear como um caso de teste, e em seu trabalho demonstra experimentalmente que os descritores de Histogramas de Gradiente Orientado (HOG) superam significativamente os conjuntos de recursos existentes para detecção humana. A Figura 6 demonstra a visão geral da cadeia de extração de recursos e detecção de objetos.

Figura 6 – Cadeia geral de extração de recursos.



Fonte: Adaptado (DALAL; TRIGGS, 2005).

Percebe-se, pelos trabalhos citados que todos precisam extrair áreas de interesse das imagens digitais e, posteriormente realizar um classificação/interpretação para atingir o objetivo proposto. Neste trabalho utilizamos técnicas semelhantes para recolher das imagens as regiões de interesse, também fazendo uso de um descritor de formas, máquinas de vetores de suporte e redes neurais convolucionais.

3 Fundamentação Teórica

3.1 Processamento Digital de Imagens

A definição de uma imagem digital se refere a qualquer representação bidimensional utilizando-se da quantificação binária codificada, de modo que sua construção é realizada através da leitura por meios eletrônicos. Em sua constituição, o seu elemento formador mínimo é denominado *pixel* (*picture element*). Quanto maior a quantidade de pixels na formação de uma imagem, mais verossímil a imagem se torna para representar elementos na realidade (SILVA, 2017).

O processamento digital de imagens consiste em empregar diversas técnicas que tornem possível a análise, aprimoramento, mutação e a transposição de imagens em geral, com o auxílio de dispositivos computacionais (PEDRINI; SCHWARTZ, 2008). A função primordial do processamento digital de imagens é fornecer ferramentas que facilitem a identificação e extração de dados de interesse, afim de posteriormente realizar sua interpretação e utilização (CRÓSTA, 1999). De acordo com com GONZALEZ; WOODS o processamento das imagens digitais pode ser dividido em etapas processamento de baixo nível, de nível intermediário e alto nível.

Uma imagem digital pode ser definida como uma função $f(x, y)$, onde seus parâmetros estão relacionados em x como o eixo das abscissas em um plano cartesiano e o parâmetro y com o eixo das ordenadas (ambos a partir de um referencial $f(0, 0)$), sendo f a intensidade visual dos *pixels* em uma escala de cinza ou referente ao seu brilho tricromático *Red Green Blue* (RGB) . Além disso, a imagem digital pode ser considerada como uma matriz de *pixels*, onde os índices referentes as colunas e linhas assinalam a localização de um ponto especifico da imagem e o elemento correspondente ao par ordenado na matriz à tonalidade de cinza (GONZALEZ; WOODS, 2000).

Consequentemente, uma imagem digital pode ser representada como sendo uma matriz enumerada $Z \times N$ tal como demonstrada na Equação 3.1 :

$$f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & f(0,2) & f(0,3) & \cdots & f(0,Z-1) \\ f(1,0) & f(1,1) & f(1,2) & f(1,3) & \cdots & f(1,Z-1)) \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ f(N-1,0) & f(N-1,1) & \vdots & \vdots & \cdots & f(N-1,Z-1) \end{pmatrix} \tag{3.1}$$

3.1.1 Pré-Processamento

O pré-processamento objetiva o aperfeiçoamento da qualidade dos atributos de uma imagem, preparando-a para as etapas subsequentes (QUEIROZ; HIRAKAWA, 2013). Nesta etapa é feito o realce de características como contraste, textura e brilho além da remoção de

ruídos e redimensionamento (GONZALEZ; WOODS, 2000) a fim de se gerar uma imagem mais apropriada para seu a aplicação.

A primeira etapa do pré-processamento consiste na conversão do padrão de cores RGB da imagem em uma correspondência de tons de cinza. Após essa etapa, são aplicados filtros espaciais para realizar a suavização e diminuição de ruídos, como os filtros de média, Gaussiano e mediana. Após a suavização é a realizada a segmentação por meio de operadores de gradiente afim de se obter as bordas.

Além da procura da face humana padronizada em um ambiente heterogêneo de informações, os algoritmos devem estar aptos a contornar problemas de dimensionalidade de dados, colocar em destaque os contornos, controlar nitidez e corrigir distorções geométricas.

3.1.2 Filtragem no Domínio Espacial

Segundo FILHO; NETO, todas as técnicas empregadas diretamente sobre a matriz de *pixels* onde a imagem é digitalizada consistem em filtragem no domínio espacial, que consequentemente produz uma nova imagem em relação a original. A função que descreve a dependência entre a intensidade de tons de cinza em um determinado ponto da imagem e seus pontos vizinhos após a transformação pode ser descrita como:

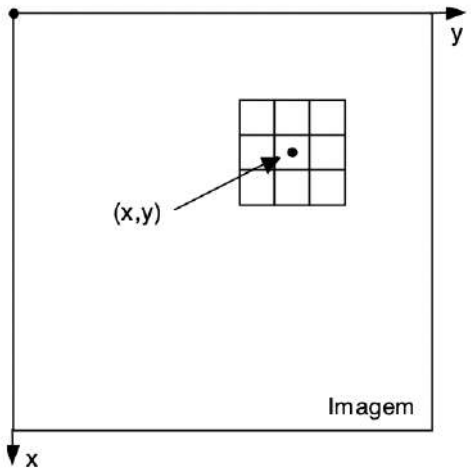
$$g(x, y) = T[f(x, y)], \tag{3.2}$$

onde $f(x, y)$ representa a imagem original, $g(x, y)$, representa a imagem após seu processamento, T representa um operador na função f que define pontos de vizinhança de (x, y).

Normalmente, a definição de uma máscara de vizinhança é feita levando em conta a simetria, utilizando matrizes quadradas de 8 *pixels* de referência (FILHO; NETO, 1999a), o que sugere a criação de uma matriz 3×3 onde o *pixel* central é o parâmetro. A região central da subimagem é transpassada *pixel* a *pixel*, iniciando-se no centro superior esquerdo.

A Figura 7 ilustra o funcionamento da filtragem espacial e a definição da região de parâmetro em uma matriz 3×3.

Figura 7 – Representação de uma vizinhança 3×3 ao redor de um ponto de coordenadas (x,y) em uma imagem.



Fonte: (FILHO; NETO, 1999b).

3.1.3 Suavização de Imagens no Domínio Espacial

A aplicação do uso de máscaras espaciais consiste em uma etapa de suavização, onde todos os filtros da etapa de pré processamento são aplicados nas imagens.

Segundo GONZALEZ; WOODS a suavização de imagens baseia-se na aplicação máscaras de convolução e correlação. As máscaras de convolução são específicas para um objetivo em questão, eliminando elementos que não serão utilizados nas próximas etapas de processamento (FILHO; NETO, 1999b). JESUS; JR define a convolução como o processo de realizar sobre a imagem o deslocamento de uma máscara, *pixel*-a-*pixel*, onde a máscara é definida com um número ímpar de colunas e linhas. A Figura 8 demonstra a demarcação de uma região da imagem onde ser aplicada a máscara de convolução.

Figura 8 – Convolução de máscara de pixels sobre região da imagem.

A_{00}	A_{01}	A_{02}	A_{03}	A_{04}	A_{05}	.....	A_{0n}
A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	.....	A_{1n}
A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	.....	.....
.....	A_{31}	A_{32}	A_{33}	A_{34}	A_{35}	.....	.....
.....	A_{41}	A_{42}	A_{43}	A_{44}	A_{45}	.....	.....
.....	A_{51}	A_{52}	A_{53}	A_{54}	A_{55}	.....	.....
.....	.....	.....	.....	.....	.....	.....	.....
A_{n0}	A_{n1}	A_{n2}	.....	.....	.....	.....	A_{nn}

(a) A imagem

M_{00}	M_{01}	M_{02}
M_{10}	M_{11}	M_{12}
M_{20}	M_{21}	M_{22}

(b) Máscara

Fonte: (JESUS; JR, 2015).

Segundo FILHO; NETO podemos subdividir os três principais filtros existentes em filtros passas-baixas, filtros passas-altas, e filtros passas-faixa. Os filtros passas-baixas são responsáveis pela dedução de componentes como regiões de bordas e detalhes de alta frequência na imagem, o que geralmente provoca um ligeiro efeito de borramento. Os filtros passas-altas estão relacionados à dedução de áreas de alto contraste e de baixa frequência já os

passa-faixas são filtros de frequência mediana, que na prática tem utilização bem reduzida ou nula.

Em meio a diversas técnicas de suavização de imagens podemos destacar os filtros da média, o filtro Gaussiano e o filtro da mediana, detalhados adiante.

3.1.4 Filtro da Média

O filtro da média é um filtro classificado como passa-baixas, ou seja, que atenua ou remove componentes de alta frequência. Ele realiza a substituição do valor da célula de *pixel* atual pelo valor médio das células ao seu redor. A ordem da máscara está diretamente ligado ao número de *pixels* considerados na vizinhança, onde quanto maior a ordem da máscara maior o efeito de suavização e consequentemente maior o nível de borramento da imagem após a aplicação. A Figura 9 demonstra a implementação de um filtro da média com uma máscara 3×3, com os coeficientes equivalentes a 1 e a divisão do resultado da convolução por um fator de normalização, neste caso referente ao número de *pixels*. A Figura 10 mostra exemplos de máscaras de filtragem pela média de diferentes dimensões aplicadas a uma mesma imagem.

Figura 9 – Máscaras 3×3 para o cálculo do filtro da média.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Fonte: Elaborada pelo autor.

3.1.5 Filtro Gaussiano

O filtro Gaussiano também é classificado como um filtro de passa-baixas, ou seja, realiza a atenuação dos componentes da imagem de alta frequência. O filtro Gaussiano segue o comportamento da função da curva de Gauss, que pode ser descrita por seus parâmetros de desvio padrão e média. O grau de suavização do filtro é definido pelo parâmetro σ que nada mais é do que o desvio padrão da função de curva Gaussiana. A função que descreve o comportamento do filtro Gaussiano em um conjunto x é definida pela Equação 3.3:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{3.3}$$

onde:

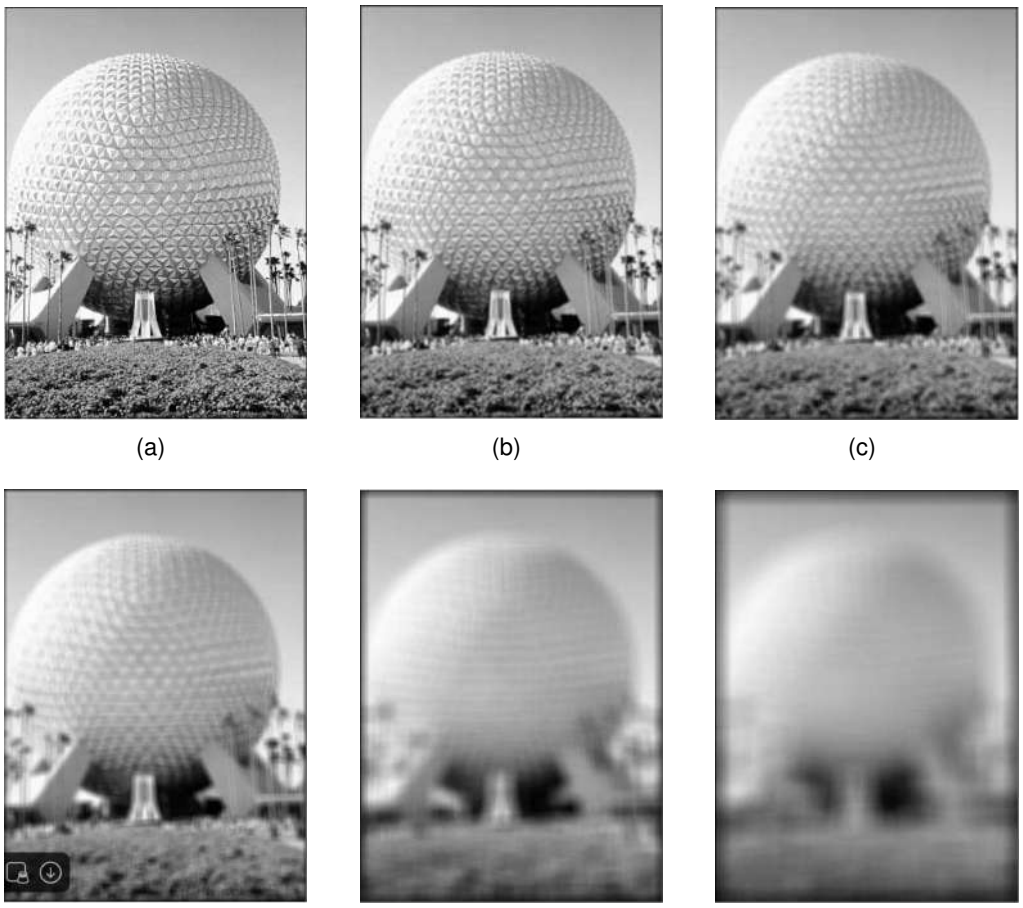
x — conjunto com n valores, tal que $-\infty < x < \infty$;

P — distribuição gaussiana dos valores de x;

σ — desvio padrão dos valores de x, tal que > 0;

μ — média dos valores de x.

Figura 10 – (a) Imagem original; (b)-(f) resultados da aplicação do filtro da média com máscara de dimensões n x n, n = 3, 5, 7, 17, 31.



(a) (b) (c)

(d) (e) (f)

Fonte: (FILHO; NETO, 1999a)

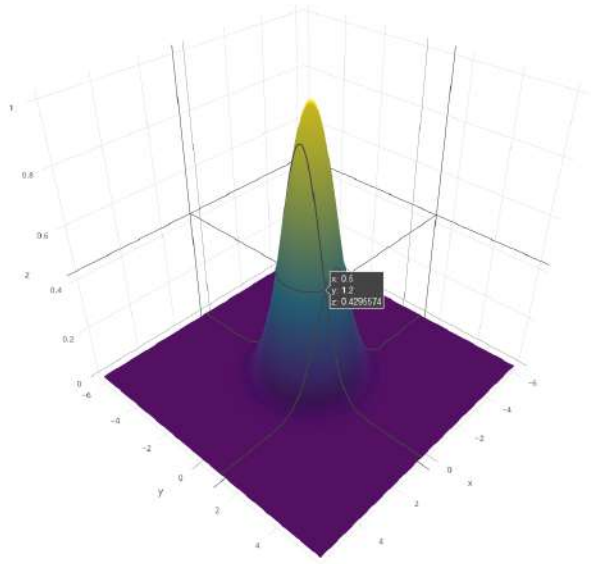
Considerando a aplicação do filtro Gaussiano em duas dimensões, a simetria em relação ao eixo de rotação para a suavização é idêntico em todas as direções (PEDRINI; SCHWARTZ, 2008). Além disso, cada *pixel* tem seu valor alterado pela média ponderada dos valores dos *pixels* em sua máscara de vizinhança, onde quanto maior a proximidade do *pixel* central, maior seu peso no cálculo (PEDRINI; SCHWARTZ, 2008).

Quanto maior o valor do desvio padrão σ, maior a taxa de suavização na imagem e maior o raio de atuação do filtro Gaussiano. A Figura 11 demonstra a representação da distribuição normal Gaussiana, e podemos observar o funcionamento do filtro Gaussiano na Figura 12.

3.1.6 Filtro de Mediana

O filtro da mediana tem como objetivo preservar as bordas e detalhes finos da imagem que possam ser ocasionalmente removidos pelo filtro da média (FILHO; NETO, 1999b), preservando dessa forma as regiões de interesse de contorno. É um filtro não linear que realiza a modificação do nível de cinza do *pixel* central pela mediana dos *pixels* que compõe sua vizinhança em situações onde a imagem é acometida de ruído impulsivo, tal como o sal-e-pimenta

Figura 11 – Representação 3D da distribuição normal Gaussiana.



Fonte: *Stack Exchange*.

Figura 12 – (a) Imagem original; (b) Conversão para escala de cinza; (c) Resultados da aplicação do filtro Gaussiano com $\sigma = 3$.

(a) (b) (c)

Fonte: Elaborada pelo autor.

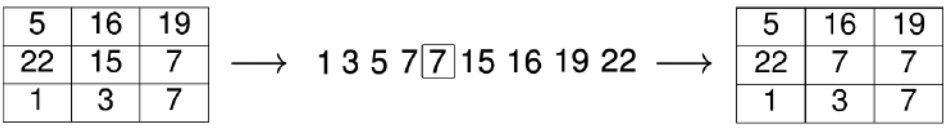
sua performance se mostra maior em relação ao filtro da média em tempo e qualidade.

3.2 Segmentação

A fase de realizar a separação da imagem em suas unidades significativas tais como localização, forma e regiões de borda é definida como segmentação. Subdividir uma imagem em suas partes ou objetos constituintes consiste em uma tarefa extremamente árdua e depende da correta extração de características de objetos (PEDRINI; SCHWARTZ, 2008).

Os algoritmos de segmentação de imagens baseiam-se em maioria em duas propriedades de valores de intensidade das escalas de cinza, sendo elas a descontinuidade e similaridade. A figura 13 mostra a representação da aplicação do filtro. A descontinuidade consiste em particionar a imagem tendo como referência suas mudanças abruptas dos níveis de cinza,

Figura 13 – Representação da aplicação do filtro da Mediana.



Fonte: Elaborada pelo autor.

onde são encontrados mais facilmente na detecção de linhas, detecção de bordas e detecção de pontos isolados. Já a similaridade baseia-se na divisão da imagem em relação a um conjunto predefinido de critérios baseados em limiares (GONZALEZ; WOODS, 2000). Destacam-se a junção e separação (*Split Merge*), aglomeração (*Clustering*) e crescimento de regiões (*Region Growing*). A Figura 14 demonstra o resultado visual após a aplicação dos filtros.

3.2.1 Operadores de Gradiente

Segundo FILHO; NETO, para realizar a segmentação e, consequentemente, a detecção de bordas em uma imagem, faz-se necessária a aplicação de filtros espaciais lineares baseados no gradiente da função de luminosidade (mudanças no tons de cinza) e laplaciano. As mudanças significativas nas tonalidades de cinza podem ser descritas pelo conceito de derivadas (PEDRINI; SCHWARTZ, 2008): o vetor gradiente aponta na direção da máxima taxa de variação de f em (x, y) por meio da análise vetorial o que justifica sua utilização para determinar a mudança dos tons de cor. O vetor gradiente pode ser $\nabla f(x, y)$ de uma imagem pode ser determinado a partir das derivadas parciais como demonstrado na Equação 3.4:

$$\nabla f(x, y) = \frac{\partial f(x, y)}{\partial x}\mathbf{i} + \frac{\partial f(x, y)}{\partial y}\mathbf{j} \tag{3.4}$$

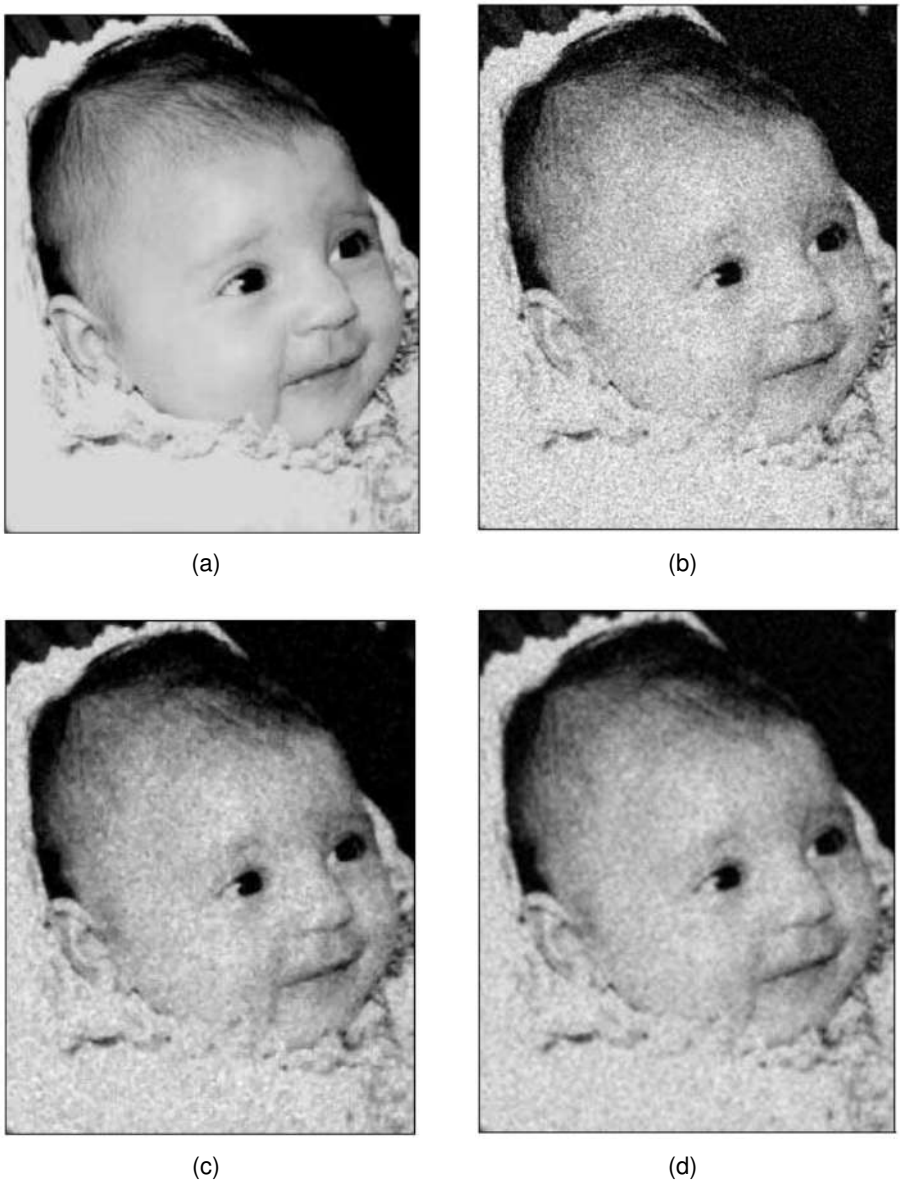
Os vetores unitários nas direções x e y são representados por **i** e **j**. A característica mais importante para a detecção de uma borda em uma imagem é referente ao módulo do vetor gradiente representado por ∇f onde:

$$\nabla f(x, y) = \sqrt{\left(\frac{\partial f(x, y)}{\partial x}\right)^2 + \left(\frac{\partial f(x, y)}{\partial y}\right)^2}, \tag{3.5}$$

que se refere a taxa de crescimento máxima em $f(x, y)$ por unidade de distância na direção de ∇f. O resultado mostra a direção do vetor gradiente e possivelmente um ponto de borda da imagem.

O Algoritmo 11 mostra uma possível implementação para a detecção de bordas por meio do cálculo do gradiente:

Figura 14 – (a) Imagem original; (b) Imagem contaminada por ruído Gaussiano; (c) Resultados da aplicação do filtro da mediana com máscara de dimensão 3×3; (d) Resultados da aplicação do filtro da média com máscara de dimensão 3×3.



(a) (b)

(c) (d)

Fonte: (FILHO; NETO, 1999a).

3.2.2 Operador Laplaciano

O operador laplaciano é um filtro espacial utilizado para detecção e realce de bordas (FILHO; NETO, 1999a). O laplaciano é um operador caracterizado por uma derivada de segunda ordem definida como:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y} \tag{3.6}$$

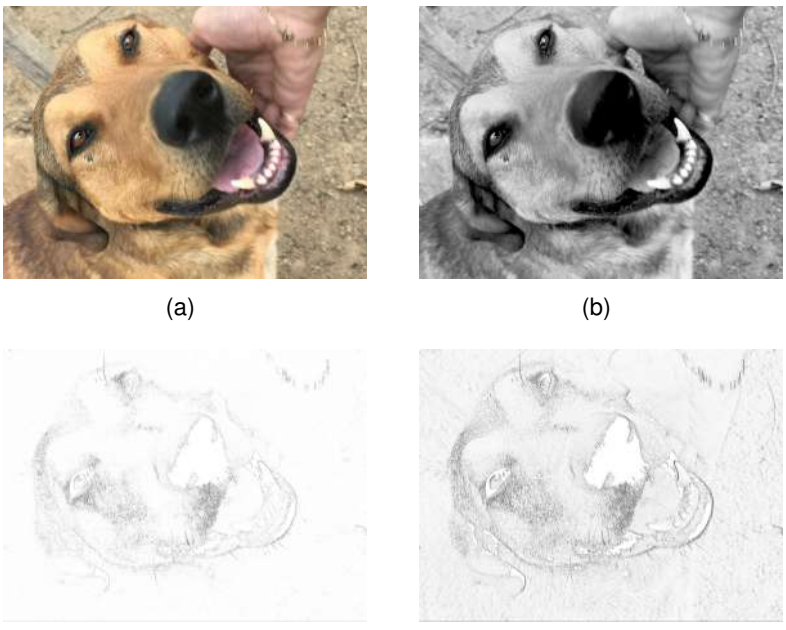
O laplaciano pode ser aproximado por meio de máscaras matriciais de ordem 3, onde ao percorrer a imagem é realizado a substituição do valor do *pixel* de referência pela média ponderada dos *pixels* vizinhos, após isso eleva-se ao quadrado o valor calculado. O fato de

Algoritmo 1: Método para determinação dos pontos de borda de uma imagem
Fonte: (PEDRINI; SCHWARTZ, 2008)

```
entrada: uma imagem f com dimensões M x N pixels e um limiar T
for x = 0 até M - 1 do
    for y = 0 até N - 1 do
        // calcular a magnitude do gradiente ∇f(x, y)
        ∇f(x, y) = sqrt( (∂f(x,y)/∂x)^2 + (∂f(x,y)/∂y)^2 )
        // efetuar a limiarização
        if ∇f(x, y) > T then
            (x, y) é um ponto da borda
        else
    end for
end for
```

o filtro laplaciano possuir insensibilidade a rotação o torna capaz de detectar bordas em todo aspecto vetorial, mas seu uso é restrito devido à sua disposição de ser acometido por ruídos(FILHO; NETO, 1999a). Para evitar a contaminação da imagem com ruídos, um filtro de passas baixas pode ser aplicado antes da aplicação no laplaciano. A Figura 15 demonstra a aplicação do filtro laplaciano em diferentes dimensões de máscaras.

Figura 15 – (a) Imagem original; (b) Conversão para escala de cinza; (c) Resultados da aplicação do filtro de Laplace com máscara de dimensão 3 x 3; (d) Resultados da aplicação do filtro de Laplace com máscara de dimensão 5 x 5.



(a) (b)

(c) (d)

Fonte: Elaborada pelo autor.

3.2.3 Operador Sobel

Segundo MARENGONI; STRINGHINI o operador de Sobel é um detector de bordas baseado na primeira derivada, onde esse detecta e realça bordas na imagem por meio do

processo de definição de máscaras que caracterizam variações de intensidade de cor dos *pixel* e realiza a convolução da imagem pela máscara definida.

A expressão matemática aplicada em um máscara representada pela Figura 16 que define o operador de Sobel é determinada pelas Equações 3.7 e 3.8 :

Figura 16 – Máscara de *pixels* 3×3.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

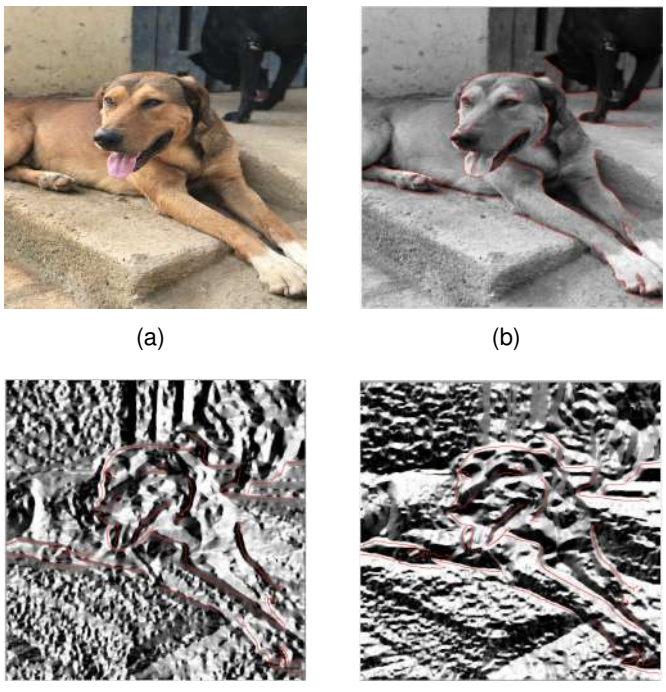
Fonte: Elaborada pelo autor

$$g(x) = \frac{\partial f(x,y)}{\partial x} = (w_7 + 2w_8 + w_9) - (w_1 + 2w_2 + w_3) \tag{3.7}$$

$$g(y) = \frac{\partial f(x,y)}{\partial y} = (w_3 + 2w_6 + w_9) - (w_1 + 2 * w_4 + w_7) \tag{3.8}$$

Linhas com orientação vertical ou horizontal são realçadas com a aplicação deste filtro. A Figura 17 demonstra a aplicação do filtro de Sobel em uma imagem.

Figura 17 – (a) Imagem original; (b) Conversão para escala de cinza; (c) Aplicação do filtro de Sobel em x; (d) Aplicação do filtro de Sobel em y.



(a) (b)

(c) (d)

Fonte: Elaborada pelo autor.

3.2.4 Detector de Bordas de Canny

O detector de bordas Canny é um algoritmo com uma das melhores taxas de eficiência na detecção de bordas (MARENGONI; STRINGHINI, 2010). Este método foi implementado por John Canny (CANNY, 1986) e parte de três heurísticas básicas:

- O método aplicado deve ser capaz de identificar todas as bordas existentes;
- Os pontos de bordas devem estar localizados próximos às bordas reais da imagem;
- Deve-se haver uma resposta única para cada borda, ou seja não pode-se haver a criação de bordas falsas.

A partir da primeira derivada Gaussiana Canny estabeleceu-se um detector de borda, que com auxílio da função Gaussiana, convolui uma imagem, calcula sua magnitude e sua direção (MARENGONI; STRINGHINI, 2010), através das Equações 3.9, 3.10 e 3.11:

$$g(x,y) = f(x,y)^{**}G(x,y) \tag{3.9}$$

$$m(x,y) = \sqrt{{g_x}^2 + {g_y}^2} \tag{3.10}$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{g_y}{g_x}\right) \tag{3.11}$$

onde:

$g(x,y)$ — corresponde aos valores obtidos após a aplicação de um operador tal como o filtro de Sobel;

$m(x,y)$ — corresponde a uma imagem que contém a informação da magnitude em cada pixel;

$\alpha(x,y)$ — corresponde a uma imagem que contém a direção da normal à borda para cada pixel.

Após os cálculos é realizada a operação de corte para reduzir pontos de borda falsos e diminuir a presença de ruído ou textura fina, nesse processo é feita a utilização de um limiar. O operador de corte possui dois valores: um denominado alto e outro denominado baixo (MARENGONI; STRINGHINI, 2010). Com eles, são obtidas duas imagens e a imagem final é gerada a partir da operação de subtração entre a elas. A Figura 18 ilustra a aplicação do detector de bordas em uma imagem.

3.3 Representação e Descrição

3.3.1 Algoritmo *Histogram of Oriented Gradients* (HOG)

Um descritor de formas é uma representação simplificada de uma imagem ou grupo de imagens que busca evidenciar informações úteis e recorrentes e rejeitar informações sem interesse (DALAL, 2006). Neste trabalho o detector fornece uma saída binária indicando se

Figura 18 – (a) Imagem original; (b) Conversão para escala de cinza; (c) Resultados da aplicação do filtro de Canny com valor da imagem alta = 0.5 e valor de imagem baixa = 0.05.

(a) (b) (c)

Fonte: Elaborada pelo autor.

uma região específica de uma imagem contém uma instância do objeto desejado (no nosso caso, um rosto humano).

Cada imagem passada para o detector é primeiro convertida em um conjunto de gradientes que são espacialmente discretizados. Uma sub-imagem é extraída deste histograma de gradiente e convertido em um vetor de recursos (HOG) (Rybski et al., 2010).

De maneira geral, um descritor de formas converte uma imagem com um tamanho de $largura \times altura$ para um vetor de característica / matriz de recursos de comprimento N. Para o descritor de recursos HOG, o tamanho da imagem de entrada é $64 \times 128 \times 3$, e o comprimento do vetor de recursos de saída é 3780 (SATYA, 2016).

No descritor de recursos HOG, a distribuição (histograma) da direção do gradiente (gradiente direcional) é usada como um recurso. O gradiente da imagem (derivativas de x e y) é útil porque nas bordas e cantos (regiões onde a intensidade muda repentinamente), o tamanho do gradiente é maior uma vez que as regiões de borda e extremidades da imagem contêm mais informações sobre a forma do objeto retratado na imagem.

3.3.1.1 Pré-Processamento

Como mencionado anteriormente, a primeira etapa da geração do vetor de recursos HOG é realizar a conversão da imagem, e para isso um corte de uma parte da imagem definida por uma característica particular é realizado.

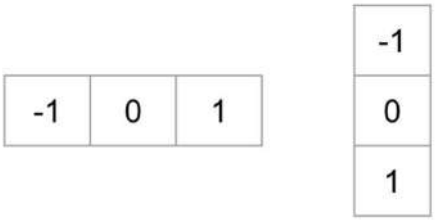
Por exemplo, ao inserir uma imagem com resolução de 1920×1080 *pixels*, com o objetivo de realizar o cálculo do descritor HOG, realizamos um corte de tamanho de 300×600 pixels e, posteriormente, este recorte é redimensionado para a dimensão do descritor (64×128).

Segundo (DALAL; TRIGGS; SCHMID, 2006), também pode ser utilizada uma operação não linear para codificar e decodificar a luminância conhecida como Correção Gamma, afim de se obter algum ganho de performance geral no corte da imagem.

3.3.1.2 Cálculo do Gradiente

O próximo passo envolve o cálculo dos gradientes horizontais e verticais da imagem. Para isso, utilizamos o operador de Sobel (MARENGONI; STRINGHINI, 2010) ou pelos seguintes *kernels* de cálculo (Figura 19):

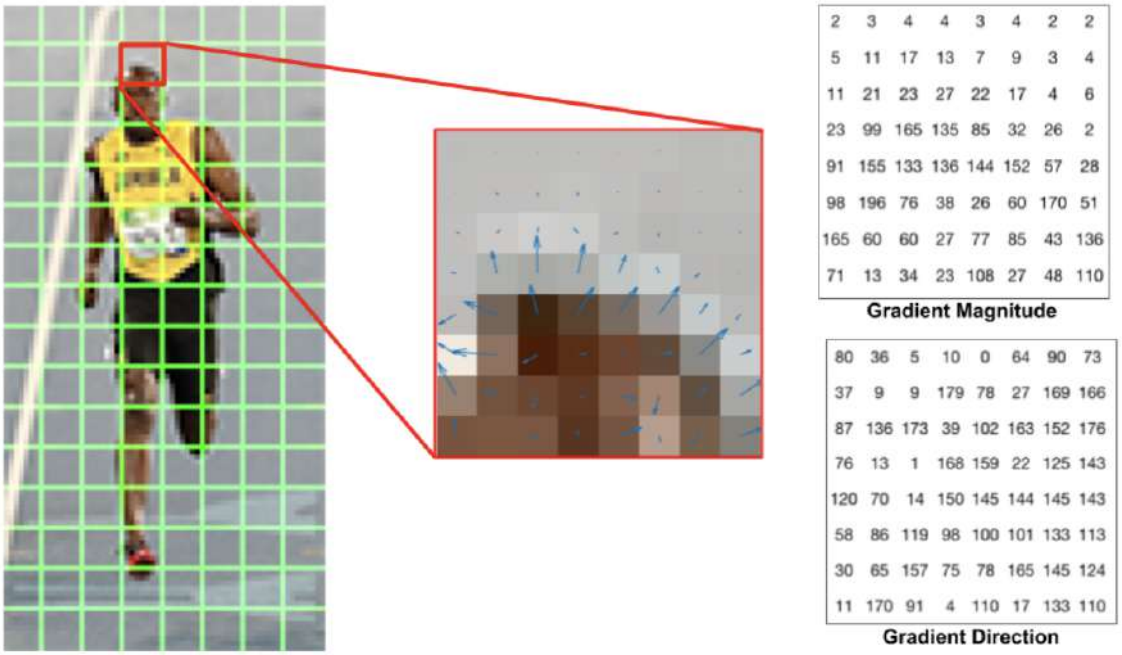
Figura 19 – *Kernels* de cálculo dos gradientes.



Fonte: (SATYA, 2016).

A magnitude e a direção do gradiente podem ser obtidos pela auxílio da função Gaussiana, representada pelas Equações 3.9, 3.10 e 3.11. O gradiente x corresponde às linhas verticais, e o gradiente y às linhas horizontais. O cálculo da magnitude do gradiente ajuda a identificar quaisquer mudanças bruscas de intensidade. Além de deixar o contorno mais óbvio, a imagem gerada não contém informações desnecessárias, como por exemplo fundo colorido. O tamanho do gradiente no *pixel* é o valor máximo do tamanho do gradiente dos três canais de cor, e o ângulo é o ângulo correspondente ao gradiente máximo (SATYA, 2016). A Figura 20 demonstra a obtenção dos gradientes e suas direções nas células em uma imagem.

Figura 20 – *Kernels* de cálculo dos gradientes.



Fonte: (SATYA, 2016).

3.3.1.3 Peso Ponderado de Cada Célula

Após calcular a magnitude e direção do gradiente, é necessário dividi-los em blocos de unidades. Em uma imagem de 1080×1080 *pixels* se definirmos a região da célula como sendo uma *pixel*/célula 8×8 teríamos 135×135 células, ou seja, 18.225.

Depois de definir a quantidade de células, o tamanho do gradiente calculado e a orientação serão usados para construir o HOG. Para isso, precisamos definir o número de orientações que controlam o número de células no histograma gerado. O ângulo de inclinação está dentro da faixa de [0,180] (sem sinal) ou [0,360] (com sinal).

Cada *pixel* contribui com um voto ponderado para o histograma, cujo peso é igual à magnitude do gradiente do determinado pixel (DALAL; TRIGGS; SCHMID, 2006).

3.3.1.4 Normalização de Blocos

Na etapa anterior, criamos um histograma com base no gradiente da imagem. Sabendo que a gradação da imagem é muito sensível à iluminação geral podemos normalizar o valor da gradação localmente para evitar mudanças na iluminação e no contraste. Isso requer o agrupamento das unidades em blocos conectados.

As nossas unidades para normalização não são os *pixels* e sim as próprias células. Segundo DALAL; TRIGGS; SCHMID o uso de células 2×2 ou 3×3 apresentam uma precisão maior na maioria dos casos. Para cada uma das células no bloco atual, concatenamos seus histogramas de gradiente correspondentes, normalizando todo o vetor de recursos concatenados. Esse tipo de normalização implica que cada uma das células será representada no vetor de recurso final várias vezes, mas normalizada por um valor diferente. Depois que todos os blocos são normalizados, pegamos os histogramas resultantes, os concatenamos e os tratamos como nosso vetor de recurso final (SATYA, 2016).

3.3.2 Máquina de Vetores de Suporte (*Support Vector Machine* — SVM)

As Máquinas de Vetores de Suporte (*Support Vector Machine* — SVM) são uma das mais eficientes aplicações de aprendizagem de máquina utilizadas atualmente (Hearst et al., 1998), obtendo muitas vezes resultados superiores às Redes Neurais Artificiais (RNAs) (BRAGA; CARVALHO; LUDERMIR, 2000).

O SVM padrão pega um conjunto de dados como entrada e prevê qual das duas categorias possíveis é a entrada para cada entrada fornecida, o que caracteriza o SVM como um classificador linear binário não probabilístico (LORENA; CARVALHO, 2007). O objetivo do SVM é encontrar uma linha divisória, que geralmente é chamada de hiperplano entre dois tipos de dados. Dessa forma, o SVM busca encontrar o melhor hiperplano para um determinado conjunto de dados que seja linearmente separável.

A dimensão do hiperplano é proporcional à dimensão n do conjunto de dados sendo

dada pela equação 3.12:

$$h_{dim} = n - 1 \tag{3.12}$$

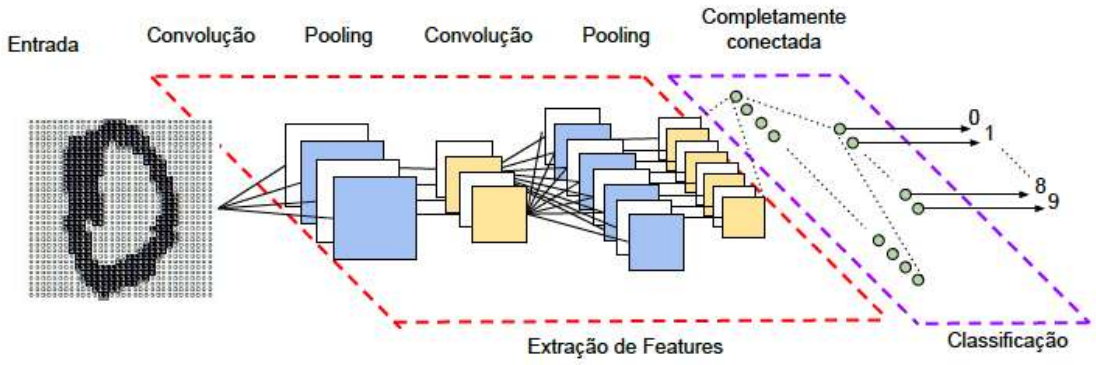
Em um conjunto de dados bidimensional, o hiperplano é uma linha. No conjunto de dados 3D, o hiperplano é representado por um plano. O hiperplano no ponto médio entre as duas categorias tem certo grau de simetria na classificação, onde a distância entre o ponto mais próximo de cada categoria e o hiperplano minimiza o erro de classificação do problema.

As origens mais próximas do hiperplano são denominados como vetores de suporte, e é por meio desses vetores que o modelo pode ser matematicamente desenvolvido, treinado e otimizado (LORENA; CARVALHO, 2007).

3.3.3 Redes Neurais Convolucionais — CNN

As Redes Neurais Convolucionais são parte de uma variante das Redes de multicamadas de Perceptrons, onde se é possível realizar a classificação e generalização de funções, simulando o aprendizado biológico (BITTENCOURT; OSÓRIO, 2001). Para realizar a assimilação e o aprendizado é necessário que haja uma base de modelos com um conjunto de dados a serem absorvidos pela Rede Neural Artificial (MATSUGU et al., 2003). A Figura 21 demonstra o funcionamento simplificado de uma CNN.

Figura 21 – Representação do funcionamento de uma Rede Neural Convolucional.

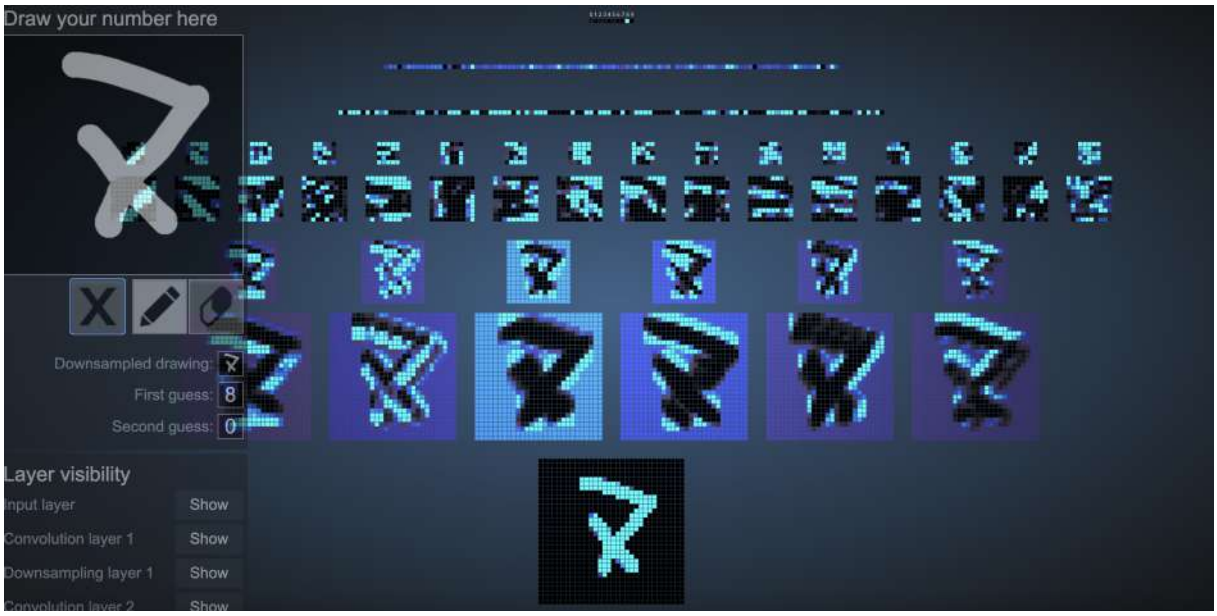


Fonte: (VARGAS; PAES; VASCONCELOS, 2016).

A rede neural realiza o aprendizado a partir de um processo de iterações entre os dados, onde ocorrem ajustes dos pesos sinápticos que são utilizadas para armazenar o conhecimento adquirido. (BITTENCOURT; OSÓRIO, 2001).

Na Figura 22 temos a demonstração de um exemplo prático da conversão da camadas de entrada em camadas ocultas (intermediárias) e sua saída ao final.

Figura 22 – Camadas de convolução de uma Rede Neural.



Fonte: Elaborada pelo autor.

3.3.4 Algoritmo k-*Nearest Neighbors Algorithm* — KNN

O algoritmo k-*Nearest Neighbor* — KNN é um algoritmo de aprendizado supervisionado (AHA; KIBLER; ALBERT, 1991) que visa encontrar os k elementos do conjunto que estão mais próximos de elementos ainda não classificados e associá-los aos rótulos desses conjuntos (FERRERO, 2009).

Este algoritmo utiliza a distância entre os vetores de recursos da imagem. Neste algoritmo, cada imagem está associada a uma categoria (rótulo), ou seja, o algoritmo KNN classifica os pontos de dados desconhecidos para encontrar entre os k-neighbors (vizinhos) uma classe mais comum entre os exemplos próximos. Segundo FERRERO a seleção desses vizinhos abrange duas questões centrais, a cardinalidade do conjuntos de vizinhos próximos e a seleção pela similaridade.

Para aplicar a classificação de vizinhança k mais próxima, precisamos definir uma métrica de distância. As escolhas mais comuns incluem a distância euclidiana Equação (5.1) e a Geometria do Táxi (*Taxicab Geometry*) Equação (5.2):

$$d\left(p,q\right) = \sqrt{\sum_{i=1}^{n}\left(q_i - p_i\right)^2} \tag{3.13}$$

$$d\left(p,q\right) = \sum_{i-1}^{n}\left|q_i - p_i\right| \tag{3.14}$$

Por meio de indicadores de aprendizagem (supervisão), o algoritmo ainda pode ser significativamente melhorado.

3.3.5 Aprendizagem Supervisionada

Na abordagem realizada neste trabalho, o classificador e reconhecedor de faces é aperfeiçoado em reconhecimento de padrões de forma a reter um conjunto de características específicas por meio do treino. Assim ele torna-se capaz de identificar padrões comuns nas imagens. O fato de haver a necessidade de um conjunto padrão já pré-definido para comparação é o que torna a aprendizagem do algoritmo supervisionada.

4 Procedimentos metodológicos

Este trabalho apresenta uma alternativa para realizar o reconhecimento facial de indivíduos que entram no CEFET-MG campus Timóteo, para se obter a liberação ou não de sua entrada nas dependências da instituição. As referências acerca dos diversos temas tratados, tais como *machine learning*, visão computacional, processamento digital de imagens, redes neurais convolucionais, além da utilização de algoritmos e implementações de bibliotecas foram obtidos por meio da leitura e estudo de artigos acadêmicos, livros, cursos de plataformas digitais relacionados aos temas tratados, ademais a monografias, teses de doutorados e dissertações de mestrado.

Foram utilizadas bibliotecas para o desenvolvimento dos algoritmos de reconhecimento, extração de características, *machine learning* e *deep learning*, a Dlib (versão 19.18) e o OpenCV (versão 4.2.0). A implementação dos algoritmos foi realizada na linguagem de programação Python (versão 3.7.6). Além disso, o gerenciamento e implantação de pacotes do Dlib e OpenCV, ficou a cargo da plataforma Anaconda versão 5.1.

A máquina utilizada para implementação foi um *MacBook Air* (13-*inch*, 2017) com as seguintes especificações técnicas:

- Processador Intel Core i5 de 1,8 GHz *Dual Core* com cache L3 compartilhado de 3 MB;
- Memória Ram de 8 GB 1600 MHz DDR3;
- Sistema Operacional macOS Catalina Versão 10.15.5 (19F96);
- Disco de Memória 126 GB SSD;
- Placa Gráfica Intel HD *Graphics* 6000 1536 MB.

Referente às imagens utilizadas no experimento, foram todas coletadas à partir de fontes públicas seguindo as seguintes propriedades:

- Resolução mínima de 640x480 pixels;
- Visão frontal do rosto, focado;
- Sem filtro de cores artificiais.

Referente às câmeras, foram utilizadas 2 câmeras sendo uma GoPro HERO 5 Black e uma Speed Dome Câmera Ip 1080p Full HD, ambas com resolução de 1920x1080. A altura de captura foi de 1,70 metros em um estande de apoio como demonstrado na Figura 23. O tempo de coleta das imagens foi segmentado em 5 minutos em diferentes partes do dia, sendo preferencialmente em horários com maior fluxo de indivíduos nas dependências de coleta.

As etapas fundamentais para alcançar o objetivo do trabalho foram:

Figura 23 – Representação do posicionamento das câmeras

Fonte: Elaborada pelo autor.

1. Realização de coleta e armazenamento de imagens com aparição da face humana em diferentes ambientes e cenários, nuances de iluminação e expressões faciais;
2. A detecção de faces por meio da implementação do algoritmo HOG *(Histogram of Oriented Gradients)* que, por meio de *machine learning*, delimita a região de interesse (ROI);
3. Definição dos pontos de referência na região de interesse (face) por meio do HOG juntamente com o SVM a fim de se obter a classificação distinta de pontos de dados;
4. Utilização do preditor de formas da biblioteca do Dlib (trata-se de um preditor já treinado com o HOG + SVM) a fim de se extrair os 68 pontos faciais de análise;
5. Treinamento de uma CNN a fim de se realizar a identificação das semelhanças de características entre as fotos;
6. Elaboração de comparação entre as faces cadastradas com vídeos em tempo real com o propósito de realizar a validação do algoritmo e identificação dos indivíduos.

5 Desenvolvimento

5.1 Detecção de faces

Para a aplicação do algoritmo HOG foi utilizada a biblioteca do Dlib e o gerenciamento de pacotes ficou a cargo da plataforma Anaconda, capaz de realizar a detecção dos pontos de referência e, consequentemente, as faces em uma imagem. Para realizar o reconhecimento de faces foi feita a coleta e armazenamento de imagens diversas com a presença da face humana em diferentes ambientes. As fotografias contemplavam a visão frontal da face, lateralizada à direita, lateralizada à esquerda, com inclinações verticais superiores e inferiores dos indivíduos.

A biblioteca Dlib é a biblioteca responsável por realizar a detecção de faces neste projeto. Foi originalmente desenvolvida em C++ e posteriormente fora criada uma API para Python. A principal função da biblioteca Dlib é que ela pode reconhecer os pontos de referência, podendo assim reconhecer faces por identificação de padrões. O preditor de formas da biblioteca do Dlib já é treinado com o HOG + SVM, e é capaz de extrair 68 pontos faciais de análise.

A platforma Anaconda é uma distribuição de código aberto das linguagens de programação Python e R para computação científica que gerencia pacotes de *software*, informações e seu uso. Ela pode ser executada em Windows, MacOS e Linux. A Anaconda irá instalar, executar e atualizar pacotes e suas dependências para integrar aplicativos locais com as ferramentas usadas. É desenvolvida e mantida pela Anaconda Inc (KADIYALA; KUMAR, 2017), uma empresa cofundada por Peter Wang e Travis Oliphant em 2012.

Nesta primeira fase, foram submetidas 1992 fotografias para identificação, com a definição de um limiar para a reconhecimento da região de interesse (limiar de tolerância), exata escala das imagens e a importação do componente de detecção frontal de faces do Dlib.

5.1.1 Aquisição das Imagens

As imagens utilizadas foram um junção entre bancos públicos de imagens tais como o *Yale Face Database*, *Flickr-Faces-HQ Dataset (FFHQ)*, *Tufts-Face-Database* além de um banco de fotos pessoais, uma exemplificação do formato das imagens pode ser demonstrado pela Figura 24. As imagens utilizadas foram subdividas em 4 classes conforme a Tabela 1.

Tabela 1 – Subdivisão dos grupos de imagens

Banco	**Imagens Utilizadas**	**Resolução Média**
Yale Face Database	688	460×528
Flickr-Faces-HQ Dataset (FFHQ)	1000	1024×1024
Tufts-Face-Database	196	1024×1024
Banco de dados pessoal	108	1280×960

Figura 24 – Exemplo de imagem utilizada

Fonte: *Yale Face Database*.

5.2 Detecção de Pontos Facias

A próxima etapa consiste na identificação dos 68 pontos de refências facias, onde a imagem é analisada pelo preditor do Dlib já previamente construído, o algoritmo implementado pode ser visualizado no Apêndice A. O preditor de formas da biblioteca do Dlib já é treinado com o HOG + SVM capaz de extrair 68 pontos faciais de análise que se subdividem conforme a Tabela 2. À partir da identificação da ROI, o reconhecedor faz o tracejamento dos pontos em busca do mapeamento facial, como demonstrado na Figura 25.

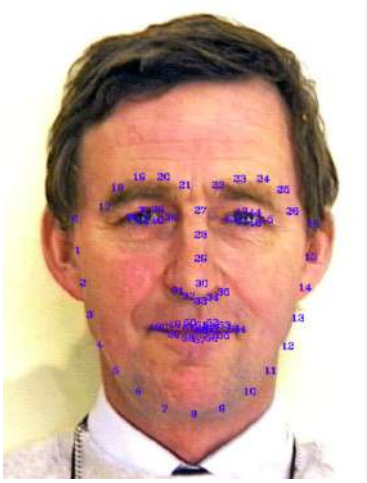
Tabela 2 – Correlação HOG + SVM

Número Facial	Representação
0, 16	Linha do Queixo
17, 21	Sobrancelha Direita
22, 26	Sobrancelha Esquerda
27, 30	Ponte Nazal
30, 35	Nariz Inferior
36, 41	Olho Esquerdo
42, 47	Olho Direito
48, 59	Lábio Externo
60, 67	Lábio Interno

5.3 Povoamento do vetor de faces conhecidas por meio de uma CNN

Nesta etapa é necessário que as imagens que compõem a base de dados sejam classificadas. Para cada indivíduo, foram utilizadas pelo menos 4 imagens a serem incorporadas à CNN. É importante salientar que, dentre os tratamentos de erros do algoritmo, foram levados

Figura 25 – Reconhecimento facial por meio do HOG e Segmentação em Pontos Faciais



Fonte: *Elaborada pelo autor*.

em consideração a quantidade de faces identificadas, descartando qualquer outra que não identificasse a ROI. As informações coletadas são utilizadas para a criação de um descritor facial por meio de uma Rede Neural Convolucional. Esta é quem seleciona em um vetor de 128 posições as características mais pertinentes da imagem, após a coleta da *Bounding Box* (face) e seus 68 pontos faciais.

A rede neural convolucional busca em cada uma dessas imagens 128 pontos estratégicos para posterior comparação. É necessário realizar a conversão do descritor de faces no formato do Dlib, os pontos adquiridos são convertidos em um vetor no formato numpy (pacote do Python que suporta *arrays* e matrizes multidimensionais) (OLIPHANT, 2006). As imagens serão armazenadas em uma matriz bidimensional, que respectivamente representam a quantidade de faces armazenadas por suas 128 informações adquiridas do descritor.

5.4 Algoritmo proposto para identificação de faces

5.4.1 Pré-processamento

Na etapa de pré-processamento, a imagem resulta do reconhecimento da região interesse pelo HOG, que posteriormente é cortada apenas com a região delimitada e colorida em tons de cinza. Portanto, o primeiro passo foi redimensionar a imagem recebida para uma escala menor, neste caso para a dimensão do descritor (64×128), após sua suavização e a detecção de bordas.

5.4.2 Aplicação do filtro Gaussiano

A aplicação do filtro Gaussiano, como descrito em detalhes no referencial teórico deste trabalho na seção 3.2.1, o grau de suavização do filtro Gaussiano está relacionado com o parâmetro σ. Segundo (GONZALEZ; WOODS, 2000) a ordem da matriz utilizada deve ser de parâmetro ímpar, a própria biblioteca define σ como sendo igual à 3.

A remoção de ruído de uma imagem por meio desse filtro é realizada pela supressão de detalhes finos e bordas da imagem como demonstrado na Figura 26. Como dito anteriormente, o tamanho da matriz é geralmente ímpar o que implica em que o resultado geral seja calculado no pixel central. Além disso o *kernel* é simétrico, tendo igualdade em número de linhas e colunas, o valor dentro do *kernel* é calculado pela função Gaussiana.

Figura 26 – Imagem da ROI original e com filtro Gaussiano

Fonte: *Elaborada pelo autor*.

Na implementação da função Gaussiana em Python foi utilizada uma função PIL *(Python Imaging Library)* chamada *filter()* para passar a imagem inteira para um kernel gaussiano predefinido, o que gera um desfoque radial na imagem inteira. Em resumo a sintaxe é feita pelo filtro *(kernel)* e através de uma convolução do mesmo é passado o parâmetro e, como retorno temos a imagem.

5.4.3 Aplicação do filtro de mediana

O filtro da mediana fornece redução de ruído e menos desfoque em comparação com um filtro de suavização linear de tamanho semelhante. O filtro da mediana é eficaz na presença de ruído de impulso que consiste de impulsos elétricos que sobrepõem a amplitude do sinal de informação. Tal como na aplicação do filtro Gaussiano, os limites definidos para o *kernel* do filtro da mediana, ou seja, a ordem da matriz é definida como sendo igual a 5.

Dadas as imagens que compõem o banco de dados, o método da etapa de pré-processamento visa suprimir ruídos e delinear linhas e contornos para supressão de características não interessantes para identificação das faces.

5.4.4 Rede Neural Convolucional

Durante a fase de treinamento, um conjunto de dados contendo imagens de pessoas a serem reconhecidas foram organizadas em pastas com seus nomes. Como parte do sistema

em questão, dez pessoas eram elegíveis para identificação, e cada pessoa tinha pelo menos quatro imagens no banco. Cada conjunto de imagens recebe uma estrutura de repetição e dentro dessa estrutura, é feito um *loop* para que o algoritmo de treinamento passe por cada imagem. Para reduzir a taxa de erro resultante, um limiar mais rigoroso foi definido.

5.4.5 Alinhamento Facial

Devido à grande quantidade de dados processados foi pensada uma forma de minimizar o custo computacional na identificação facial uma vez que, caso utilizado em larga escala, pode haver perda de dados submetidos para análise. A solução encontrada é a realização de alinhamento facial nas imagens onde são coletados apenas 5 pontos faciais, como demonstrado na Figura 27.

O próximo passo é a rotação da imagem com o propósito de se endireitar os pontos de angulação facial, centralizando a mesma para coleta de características.

Figura 27 – Alinhamento facial

Fonte: *Elaborada pelo autor*.

5.4.6 Paralelismo das CPUs

O processamento assíncrono nas CPUs foi realizado visando um processamento em tempo real e de várias faces simultaneamente, distribuindo a carga de processamento da aplicação. Uma vez que a identificação das faces não possui dependência entre si, a utilização deste método quebra o processo em diferentes *threads* e não deixa com que nenhuma face tenha seu processamento eventualmente ignorado.

O processamento paralelo utiliza a análise de dados no banco convolucional de cada face explorando com mais eficiência o uso da memória e a capacidade da CPU.

5.4.6.1 O Python no cenário *multi-thread*

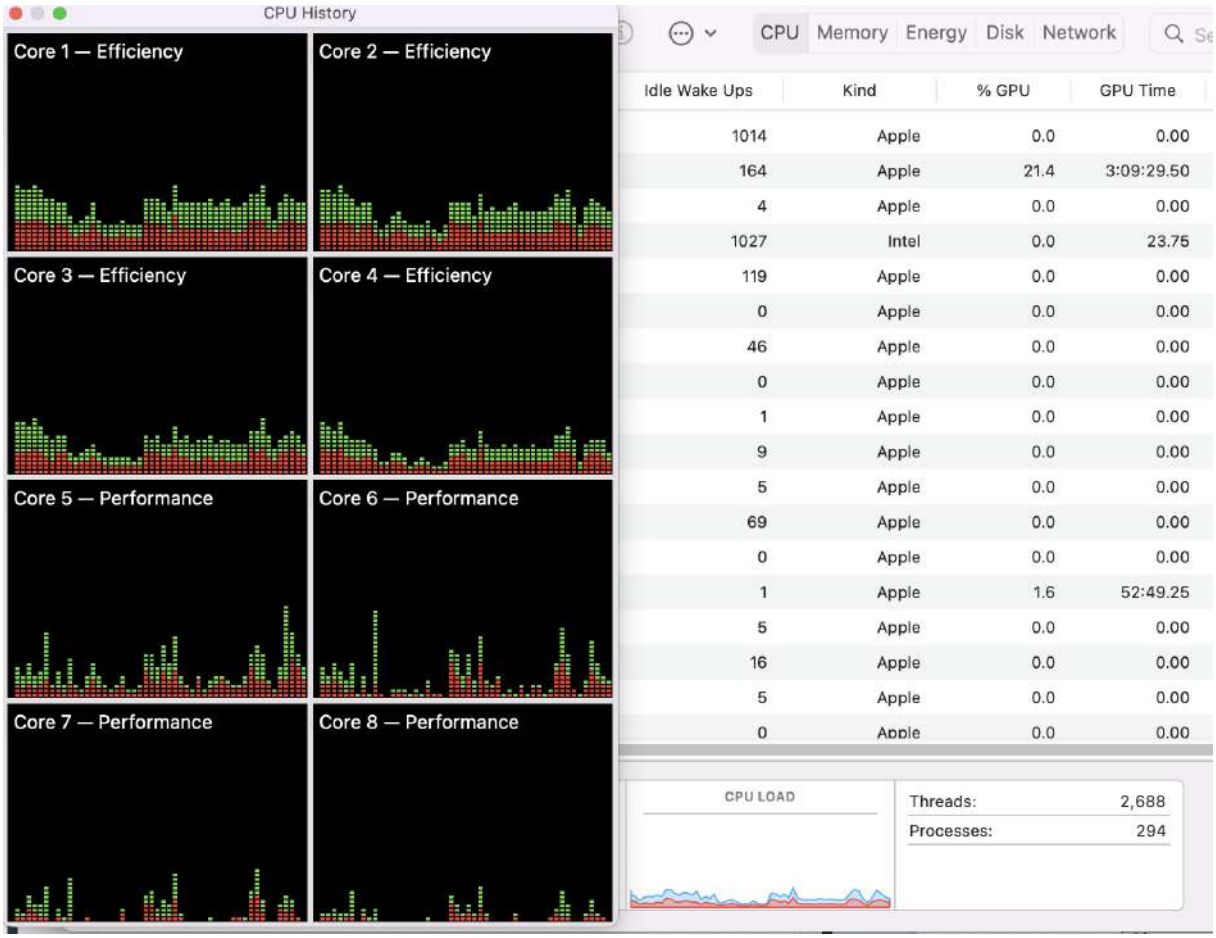
O Python possui um módulo nativo chamado *thread*, que é interpretado por sistemas que suportam o *posix threads (pthreads)*, que é uma API padrão para criar e manipular *threads*,

otimizando as tarefas de processamento.

Para utilizar o ambiente *multi-threading* das bibliotecas DLIB, foi realizada a implementação de substituição para a interface de bloqueios do sistema, e a implementação do código que trata da criação de *threads*, destruição de *threads* e métodos de acesso TLS para DLIB.

O suporte para threads em DLIB usa dois tipos de *threads*, uma na sequência principal do código, criado e inicializado automaticamente pela sequência de inicialização do programa e destruído automaticamente pela sequência de destruição do programa. E nas *threads* secundárias que devem ser criadas e inicializadas manualmente além de serem destruídas manualmente. A Figura 28 demonstra o comportamento de carga das CPUs durante a execução do código.

Figura 28 – Análise de performance em *multi-thread* da aplicação



Fonte: *Elaborada pelo autor*.

5.4.7 Aprendizagem baseada em Instâncias

A aprendizagem baseada em instâncias tem como objetivo realizar a separação em grupos mediante a similaridade dos valores apresentados. O algoritmo KNN recebe um vetor não classificado referente aos pontos faciais, e a partir disso será capaz de medir a distância

euclidiana por meio da Equação (5.1):

$$d\left(p,q\right)=\sqrt{\sum_{i=1}^{n}\left(q_i-p_i\right)^2} \tag{5.1}$$

$$d\left(p,q\right)=\sum_{i-1}^{n}\left|q_i-p_i\right| \tag{5.2}$$

Para a definição das classes no qual o ponto de dados pertencerá, é realizada a comparação dos valores por meio da raiz do somatório das diferenças ao quadrado entre as posições do vetor. Não existe uma forma específica de determinar o melhor valor para “K”, que representa a quantidade de "vizinhos"mais próximos do vetor de comparação, na literatura se vê o constante uso do K=5. Valores muito baixos de K (como K=1 ou K=2) podem criar ruído e causar valores atípicos no modelo. No modelo proposto fora utilizado K=5.

Após realizado o cálculo verifica-se a classe de cada da um dos dados além da quantidade de indivíduos em cada classe. A classificação do novo dado é feita de acordo com resultado da comparação, como mostrado da Figura 29.

Com o objetivo de ajustar o classificador K-NN aos dados de treinamento realizamos a importação de uma classe específica que é a *Neighbor Classifier* pertencente à biblioteca *Sklearn Neighbors*. Após importar a classe, criaremos o objeto classificador da classe. Os parâmetros desta classe serão:

- *nneighbors*: Que é a definição dos vizinhos necessários do algoritmo, no caso foram 5.
- *metric='minkowski'*: Que é o parâmetro padrão e decide a distância entre os pontos.
- P=2: Que é equivalente à métrica euclidiana padrão.

O objetivo final desse cálculo é aproximar o novo dado inserido ao resultado mais próximo da base de dados, podendo reconhecer a face aos seus itens "semelhantes".

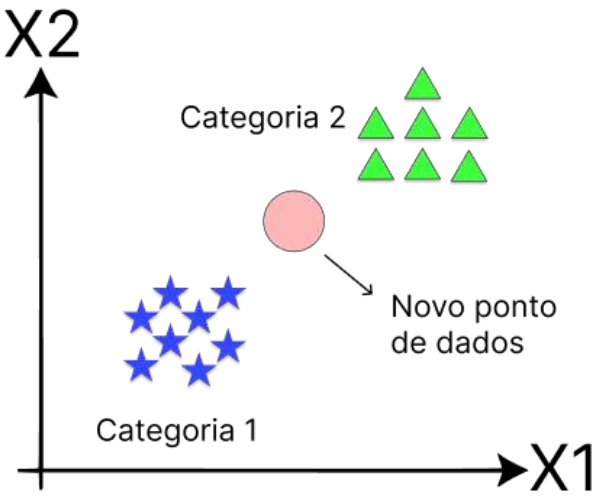
A Figura 30 mostra os dados de saída, em relação à identificação da ROI e as faces já inseridas do banco. Como podemos ver na Figura 30, o resultado previsto é relativamente favorável, pois a maioria dos pontos vermelhos está na área vermelha e a maioria dos pontos verdes está na área verde. No entanto, existem alguns pontos verdes na área vermelha e alguns pontos vermelhos na área verde. Portanto, este é um equívoco que vimos na matriz de confusão. o algoritmo implementado para a captura em tempo real pode ser visualizado no Apêndice A.

5.4.8 Avaliação preliminar do Algoritmo

A primeira fase de avaliação sobre a acurácia do algoritmo foi pela análise de um vídeo de 5 minutos conforme a Tabela 3.

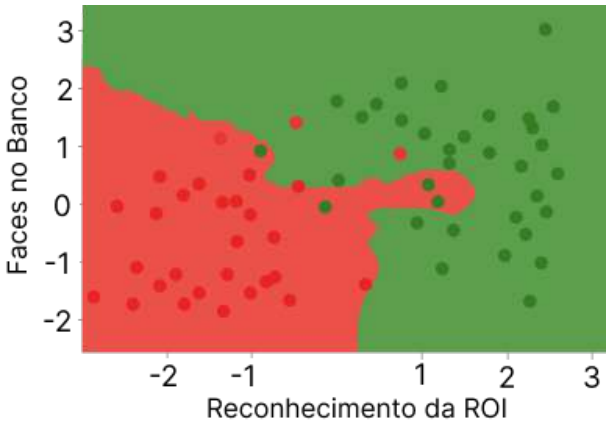
O método desenvolvido foi aplicado a uma sequência de imagens, com as mesmas características das imagens do banco. O vídeo contém originalmente 30 *frames*/segundo e

Figura 29 – Separacão de grupos com nova inserção de dados



Fonte: *Elaborada pelo autor*.

Figura 30 – Separação de semelhanças entre faces no banco e reconhecimento da ROI



Fonte: *Elaborada pelo autor*.

Tabela 3 – Acurácia do algortimo HOG + CNN

Câmera	**N 1**	**N 2**
Taxa de *frames* / Segundos processados	0.1	0.1
Frames Processados	300	300
Erro delimitação da ROI	57	78
Percentual de acerto global	81%	74%
Percentual de acerto do algoritmo em reconhecimento facial / Frames	95%	88%

resolução em FULL HD (1920x1080 pixels) de 5 minutos. Como mostra a Tabela 3, 9000 frames foram processados no total, resultando em uma assertividade global de 81% na primeira câmera e 74% na segunda.

Em análise sobre a falsos positivos sendo contabilizados dentro do banco de dados a

Tabela 4 – Resultados para a detecção com algoritmo HOG + CNN

Falsos positivos	2
Total de indivíduos coletadas para a base de dados	31
Percentual de erro	1,55%
Percentual de acerto	98.45

Tabela 4 demonstra o percentual de erro ignorado. Geralmente é ocasionado pela não detecção dos 64 pontos faciais.

6 Resultados Experimentais

6.1 Validação - Portaria Residencial

Os primeiros testes realizados na validação deste trabalho foram realizados em vídeos obtidos em uma portaria residencial, em períodos de acesso normal à entrada do mesmo. As capturas de vídeos foram subdivididas em 3 classes distintas referentes às distâncias das câmera aos pontos de análise, sendo respectivamente: 4,5 metros, 2,5 metros e 1 metro. As imagens capturadas na fase de experimentação não foram utilizadas para alimentação dos dados da fase de treinamento, objetivando-se aumentar a acurácia e confiabilidade dos resultados.

O vídeo capturado tem duração aproximada de 120 minutos e foi feito durante o período vespertino como mostrado na Figura 31.

Figura 31 – Câmeras posicionadas respectivamente à 4,5 metros e 2,5 metros dos pontos de referência para coleta

Fonte: *Elaborada pelo autor*.

6.2 Portaria CEFET-MG

Posteriormente foram utilizados vídeos obtidos na portaria escolar do CEFET-MG, em períodos de acesso normal à entrada do mesmo. A aplicação foi realizada em um ambiente real, que contempla mudanças repentinas de luminosidade, obstrução de visão e tremulação.

As capturas de vídeos foram subdivididas em 2 classes distintas referentes as distâncias das câmeras aos pontos de análise, sendo respectivamente: Portaria-Entrada 1 e Portaria-Entrada 2 como demonstrado na Figura 32. Novamente as imagens capturadas na fase de experimentação não foram utilizadas para alimentação dos dados da fase de treinamento.

Os vídeos capturados tem duração aproximada de 5 minutos e foram feitos durante todos os períodos do dia.

Figura 32 – Visão frontal e traseira das portarias da instituição

Fonte: *Elaborada pelo autor*.

A quantidade total de indivíduos que passaram pela câmera nesta análise foi de 53 na Portaria-Entrada 1 e 24 na Portaria-Entrada 2, a análise de performance do contador de faces era realizado instantaneamente pela máquina e quantificado em tempo real. O tempo médio de passagem em movimento foi de 5 segundos e a análise feita em 30 quadros por segundo, totalizando 150 capturas lançadas para análise. Quando mais de uma face era identificada a comparação era feita de forma paralela. A Tabela 5 e 6 mostram os dados referentes aos percentuais de erros dos algoritmos implementados em cada etapa.

Tabela 5 – Percentual de erros e acertos do algoritmo HOG

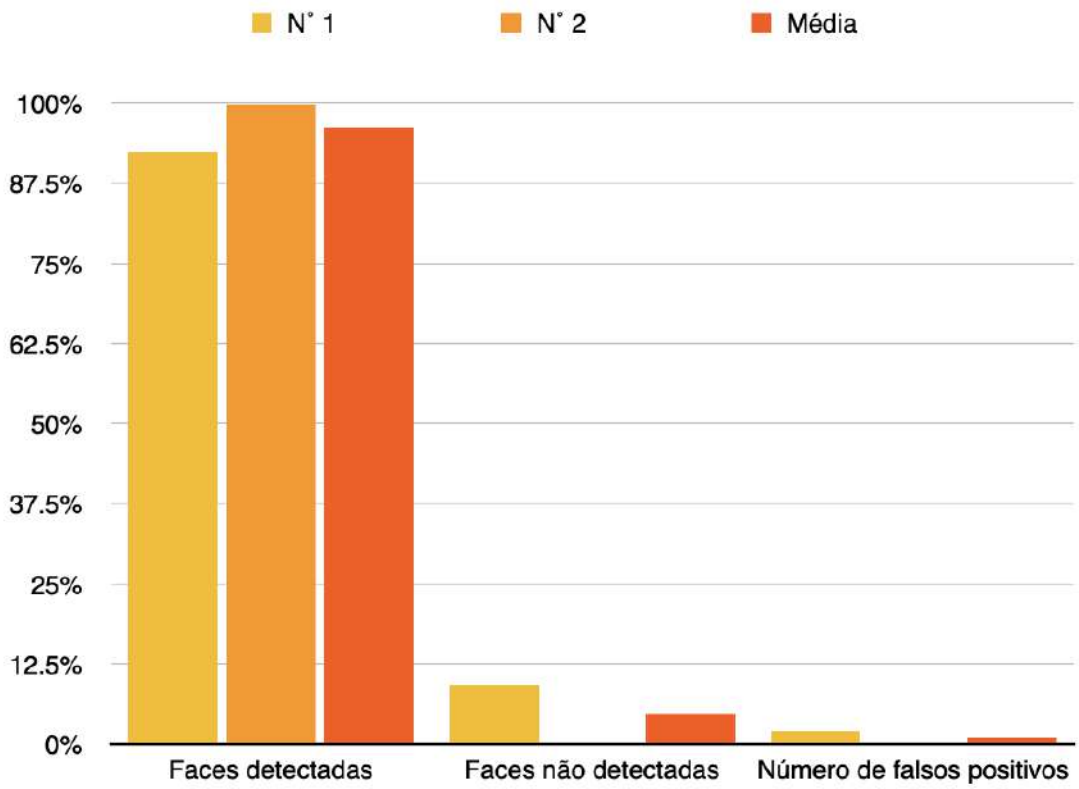
Câmera	**N° 1**	**N° 2**	**Total**
Indivíduos que passaram pela Câmera	53	24	77
Faces detectadas	49	24	73
Faces não detectadas	5	0	5
Número de falsos positivos (Objetos que possivelmente o algoritmo confundiu como face).	1	0	1

Tabela 6 – Percentual de erros e acertos do algoritmo HOG + CNN

Câmera	**N° 1**	**N° 2**
Pessoas identificadas	10	9
Relação Pessoas Identificadas/Faces Detectadas	20.41%	37.5%
Número de falsos positivos	6	2
Número de pessoas não reconhecidas (por não constarem no BD)	43	15
Número de pessoas corretamente identificadas dentre as constadas no BD	9	10
Relação Pessoas Identificadas/Pessoas totais no banco de dados	5.29%	4.76%
Relação pessoas no Banco de Bacos passíveis de identificação / Pessoas Identificadas	100%	90%

Podemos concluir que o reconhecedor se comportou de forma esperada, fazendo a correta identificação em 95% dos casos. Esse número só não foi maior porque nem todas as pessoas que passaram pelas câmeras estavam listadas no Banco de Dados com suas respectivas fotos.

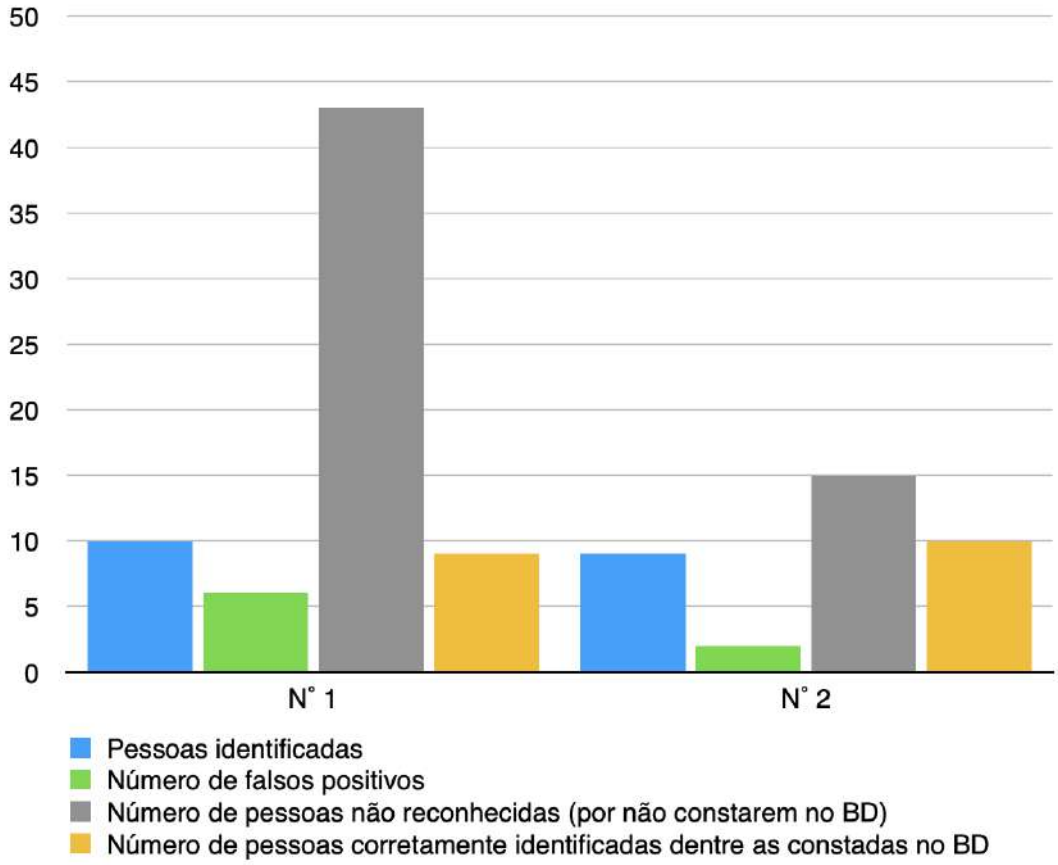
Figura 33 – Acurácia HOG



Fonte: Elaborada pelo autor.

As Figuras 33 e 34 demonstram a acurácia dos algoritmos nas capturas realizadas por cada câmera. A Figura 35 monstra um aluno adentrando ao campus e sendo reconhecido.

Figura 34 – Acurácia CNN



Fonte: Elaborada pelo autor.

Figura 35 – Reconhecimento facial de aluno

Fonte: Elaborada pelo autor.

7 Conclusão

Tendo em vista o montante pesquisado e apresentado no referencial teórico deste trabalho, juntamente com as pesquisas utilizadas para implementação e desenvolvimento, é proposto um método para o reconhecimento e identificação de faces na portaria da instituição de ensino CEFET-MG. Por meio dele é apresentada uma forma otimizada para o uso computacional na identificação, com mudanças de parâmetros visando aumentar a acurácia e a efetividade do reconhecimento, alcançando assim o objetivo geral.

O método desenvolvido é capaz de identificar 64 pontos faciais, distinguir a região de interesse em um cenário abarrotado de outras informações, juntamente com um banco de dados com respectivos dados de coleta de cada face que seja capturada pela câmera, apontar a similaridade de um indivíduo e reconhecê-lo em tempo real. O maior problema no desenvolvimento foi adquirir uma base de dados consistente e grande o suficiente para o submetimento em teste, uma vez que as pessoas no banco necessariamente deveriam passar pela câmera. Porém, como demonstrado na seção de resultados, as ROIs foram devidamente identificadas e, se o indivíduo constava no (BD) a assertividade foi maior que 90% , poderia ser ainda maior com mais inserções ao banco.

Como trabalhos futuros para a continuidade deste projeto podemos citar:

- A construção de um banco de dados onde a alimentação se desse (100%) pela coleta de fotos pela câmera;
- Automatização da chamada presencial de aulas;
- Melhora no sistema para identificação de indivíduos com máscaras;
- Implementação de reconhecimento de objetos juntamente com faces para traçar perfis comportamentais;

Com o método implementado, podemos inferir que a identificação automática de pessoas é possível, uma vez que o algoritmo apresentou assertividade de 93% na etapa de definição da ROI como face e 95% na identificação do indivíduo no banco. Toda a análise e processamento foi feito em tempo real, a efetividade dos filtros de suavização e detecção de bordas em destacar a área de interesse para análise é demonstrada pelo nível de assertividade total do algoritmo. Ressalta-se que os falsos positivos são gerados pela captura incompleta da face ou pontos de dados muito semelhantes.

A Algoritmos utilizados na implementação do trabalho

```
import dlib
import cv2
import numpy as np

fonte = cv2.FONT_HERSHEY_COMPLEX_SMALL  # mostra os resultados
imagem = cv2.imread("Fotos/washington.0.3.jpg") #carregamento da foto

detectorFace = dlib.get_frontal_face_detector() # importa o detecctor de faces da biblioteca Dlib

detectorPontos = dlib.shape_predictor("recursos/shape_predictor_68_face_landmarks.dat")
# importa o detecctor de pontos da biblioteca Dlib (predotor de formas), arquivo j  treinado HOG + SVM

facesDetectadas = detectorFace(imagem, 1) # Passa a imagem como par metro para o detecctor e realiza a contagem de numero

print("Faces totais reconhecidas: ", len(facesDetectadas))
x = 0

def imprimePontos(imagem, pontosFaciais):
    for p in pontosFaciais.parts():
        cv2.circle(imagem, (p.x, p.y), 6, (0, 255, 0), 2)

def imprimeNumeros(imagem, pontosFaciais):
    for i, p in enumerate(pontosFaciais.parts()):
        cv2.putText(imagem, str(i), (p.x, p.y), fonte, .55, (0, 0, 255), 1)

def imprimeLinhas(imagem, pontosFaciais):
    p68 = [[0, 16, False], # linha do queixo
           [17, 21, False], # sombrancelha direita
           [22, 26, False], # sombancelha esquerda
           [27, 30, False], # ponte nasal
           [30, 35, True], # nariz inferior
           [36, 41, True], # olho esquerdo
           [42, 47, True], # olho direito
           [48, 59, True], # labio externo
           [60, 67, True]] # labio interno
    for k in range(0, len(p68)):
        pontos = []
        for i in range(p68[k][0], p68[k][1] + 1):
            ponto = [pontosFaciais.part(i).x, pontosFaciais.part(i).y]
            pontos.append(ponto)
        pontos = np.array(pontos, dtype=np.int32)
```

```
42          cv2.polylines(imagem, [pontos], p68[k][2], (255, 0, 0), 2)
43
44  for rosto in facesDetectadas:
45      pontos = detectorPontos(imagem, rosto) #identifica dentro do bouding box os
        68 pontos
46      print("Localiza  o dos Pontos: ", pontos.parts()) # locais X, Y de
        localiza  o dos pontos
47      print("N mero de pontos do Dlib: ", len(pontos.parts())) # Quantidade de
        pontos detectados
48      #imprimePontos(imagem, pontos)
49      #imprimeNumeros(imagem, pontos)
50      imprimeLinhas(imagem, pontos)
51
52  cv2.imshow("Pontos faciais", imagem)  #mostra imagem ao final
53  cv2.waitKey(0)
54  cv2.destroyAllWindows()
```

Listing A.1 – Fonte: Elaborado pelo autor

```
1   import sys
2   import dlib
3   import cv2
4   import glob
5   import _pickle as cPickle
6   import dlib
7   import cv2
8   import numpy as np
9   import os
10
11  pula_quadros = 30
12  captura = cv2.VideoCapture(0)
13  contadorQuadros = 0
14  detectorFace = dlib.get_frontal_face_detector()
15  detectorPontos = dlib.shape_predictor("recursos/
        shape_predictor_68_face_landmarks.dat")
16  reconhecimentoFacial = dlib.face_recognition_model_v1("recursos/
        dlib_face_recognition_resnet_model_v1.dat")
17  indices = cPickle.load(open('recursos/indices_indices.pickle', 'rb'))
18  descritoresFaciais = np.load("recursos/descritores_128.npy")
19  limiar = 0.5
20
21  while captura.isOpened():
22      conectado, frame = captura.read()
23      contadorQuadros += 1
24      if contadorQuadros % pula_quadros == 0:
25          FacesDetectadas = detectorFace(frame, 1)
26          print("Faces totais reconhecidas: ", len(FacesDetectadas))
27          for face in FacesDetectadas:
28              e, t, d, b = (int(face.left()), int(face.top()), int(face.right()),
        int(face.bottom()))
29              pontosFaciais = detectorPontos(frame, face)
30              descritorFacial = reconhecimentoFacial.compute_face_descriptor(frame
        , pontosFaciais)
```

```
31              listaDescritorFacial = [fd for fd in descritorFacial]
32              npArrayDescritorFacial = np.asarray(listaDescritorFacial, dtype=np.
    float64)
33              npArrayDescritorFacial = npArrayDescritorFacial[np.newaxis, :]
34
35              distancias = np.linalg.norm(npArrayDescritorFacial -
    descritoresFaciais, axis=1)
36              print("Dis: {}".format(distancias))
37              minimo = np.argmin(distancias)
38              print(minimo)
39              distanciaMinima = distancias[minimo]
40              print(distanciaMinima)
41
42              if distanciaMinima <= limiar:
43                  nome = os.path.split(indices[minimo])[1].split(".")[0]
44              else:
45                  nome = 'N o reconhecido no banco de dados'
46
47              cv2.rectangle(frame, (e, t), (d, b), (222, 255, 255), 2)
48              texto = "{} {:.4f}".format(nome, distanciaMinima)
49              cv2.putText(frame, texto, (d, t), cv2.FONT_HERSHEY_COMPLEX_SMALL, 2,
     (255, 255, 0))
50
51          cv2.imshow("Detector hog", frame)
52
53          if cv2.waitKey(1) & 0xFF == 27:
54              break
55
56 captura.release()
57 cv2.waitKey(0)
58 cv2.destroyAllWindows()
59 sys.exit(0)
```

Listing A.2 – Fonte: Elaborado pelo autor

Referências

AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-based learning algorithms. *Machine learning*, Springer, v. 6, n. 1, p. 37–66, 1991. Citado na página 33.

ARAÚJO, S. A. d. *Casamento de padrões em imagens digitais livre de segmentação e invariante sob transformações de similaridade.* 2009. Tese (Doutorado) — Universidade de São Paulo, 2009. Citado nas páginas 12 e 15.

BITTENCOURT, J. R.; OSÓRIO, F. Annef–artificial neural networks framework: Uma solução software livre para o desenvolvimento, ensino e pesquisa de aplicações de inteligência artificial multiplataforma. In: SN. *Anais do II Workshop sobre Software Livre. Porto Alegre: SBC*. [S.l.], 2001. p. 13–16. Citado na página 32.

BRAGA, A.; CARVALHO, A.; LUDERMIR, T. Redes neurais artificiais: teoria e aplicaçoes, editora ltc. *Rio de janeiro, Brasil*, 2000. Citado na página 31.

CANNY, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, Ieee, n. 6, p. 679–698, 1986. Citado na página 28.

CONCI, A.; AZEVEDO, E.; LETA, F. R. Computação gráfica–teoria e prática,[v. 2]. *Rio de Janeiro*, 2008. Citado na página 15.

CRÓSTA, A. P. *Processamento digital de imagens de sensoriamento remoto*. [S.l.]: UNICAMP/Instituto de Geociências, 1999. Citado na página 18.

DALAL, N. *Finding people in images and videos*. july 2006. Tese (Doutorado) — Institut National Polytechnique de Grenoble, july 2006. Disponível em: <http://lear.inrialpes.fr/pubs/2006/Dal06>. Citado na página 28.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. [S.l.], 2005. v. 1, p. 886–893. Citado na página 17.

DALAL, N.; TRIGGS, B.; SCHMID, C. Human Detection Using Oriented Histograms of Flow and Appearance. In: LEONARDIS, A.; BISCHOF, H.; PINZ, A. (Ed.). *Computer Vision – ECCV 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 428–441. ISBN 978-3-540-33835-2. Citado nas páginas 29 e 31.

FERRERO, C. A. *Algoritmo kNN para previsão de dados temporais: funções de previsão e critérios de seleção de vizinhos próximos aplicados a variáveis ambientais em limnologia*. 2009. Tese (Doutorado) — Universidade de São Paulo, 2009. Citado nas páginas 16 e 33.

FILHO, O. M.; NETO, H. V. *Processamento digital de imagens*. [S.l.]: Brasport, 1999. Citado nas páginas 19, 22, 24, 25 e 26.

FILHO, O. M.; NETO, H. V. *Processamento digital de imagens*. BRASPORT, 1999. (Cuadernos de extensión universitaria (Santa Fe, Argentina).: Serie académica). ISBN 9788574520094. Disponível em: <https://books.google.com.br/books?id=MAEOtwAACAAJ>. Citado nas páginas 20 e 22.

GONZALEZ, R. C.; WOODS, R. E. *Processamento de imagens digitais*. Edgard Blucher, 2000. ISBN 9788521202646. Disponível em: <https://books.google.com.br/books?id=d3MnAgAACAAJ>. Citado nas páginas 15, 18, 19, 20, 24 e 39.

Hearst, M. A. et al. Support vector machines. *IEEE Intelligent Systems and their Applications*, v. 13, n. 4, p. 18–28, 1998. Citado na página 31.

JENKINS, R.; DOWSETT, A.; BURTON, A. How many faces do people know? *Proceedings of the Royal Society B*, The Royal Society, v. 285, n. 1888, p. 20181319, 2018. Citado na página 12.

JESUS, E. O.; JR, R. C. A utilização de filtros gaussianos na análise de imagens digitais. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 3, n. 1, 2015. Citado na página 20.

KADIYALA, A.; KUMAR, A. Applications of python to evaluate environmental data science problems. *Environmental Progress & Sustainable Energy*, Wiley Online Library, v. 36, n. 6, p. 1580–1586, 2017. Citado na página 37.

Kelvin Lee et al. Human detection using histogram of oriented gradients and human body ratio estimation. In: *2010 3rd International Conference on Computer Science and Information Technology*. [S.l.: s.n.], 2010. v. 4, p. 18–22. Citado nas páginas 15 e 16.

KING, D. E. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, JMLR. org, v. 10, p. 1755–1758, 2009. Citado na página 14.

LAMBERT-TORRES, G.; SILVA, P. Aplicação de sistemas inteligentes em engenharia de potência. *Sba Controle Automação Sociedade Brasileira de Automatica*, v. 5, p. 144–152, 10 1994. Citado na página 12.

LORENA, A. C.; CARVALHO, A. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007. ISSN 21752745. Citado nas páginas 31 e 32.

MARENGONI, M.; STRINGHINI, S. Tutorial: Introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, v. 16, n. 1, p. 125–160, 2010. ISSN 21752745. Citado nas páginas 26, 28 e 30.

MATSUGU, M. et al. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, Elsevier, v. 16, n. 5-6, p. 555–559, 2003. Citado na página 32.

NETO, E. L. A. et al. Sistemas de identificação pessoal utilizando tecnicas de reconhecimento e verificação facial automaticas. [sn], 1997. Citado nas páginas 12 e 15.

OLIPHANT, T. E. *A guide to NumPy*. [S.l.]: Trelgol Publishing USA, 2006. v. 1. Citado na página 39.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações*. Editora Thomson Learning Edições Ltda., 2008. 528 p. Disponível em: <https://www.ic.unicamp.br/{~}helio/book{_}aid/index>. Citado nas páginas 18, 22, 23, 24 e 26.

QUEIROZ, L. C. L.; HIRAKAWA, A. R. Classificação das amostras do ensaio baumann através do processamento digital de imagens. *Tecnologia em Metalurgia, Materiais e Mineração*, ABM-Associação Brasileira de Metalurgia, Materiais e Mineração, v. 10, n. 2, p. 183–189, 2013. Citado na página 18.

Rybski, P. E. et al. Visual classification of coarse vehicle orientation using histogram of oriented gradients features. In: *2010 IEEE Intelligent Vehicles Symposium*. [S.l.: s.n.], 2010. p. 921–928. Citado na página 29.

SATYA, M. *Histogram of Oriented Gradients explained using OpenCV*. 2016. 2 p. Disponível em: <https://learnopencv.com/histogram-of-oriented-gradients/>. Citado nas páginas 29, 30 e 31.

SILVA, J. A. da. *Detecção de Imagens Manipuladas utilizando Descritores Locais*. 2017. Tese (Doutorado) — Universidade Federal de Pernambuco, 2017. Citado na página 18.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: *Proceedings of the xxix conference on graphics, patterns and images*. [S.l.: s.n.], 2016. v. 1, n. 4. Citado na página 32.

Zhang, S.; Wang, X. Human detection and object tracking based on histograms of oriented gradients. In: *2013 Ninth International Conference on Natural Computation (ICNC)*. [S.l.: s.n.], 2013. p. 1349–1353. Citado na página 16.