Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LUIZ MARCELO DE OLIVEIRA JUNIOR

TREINANDO UM *HAAR CASCADE* PARA A DETECÇÃO DA PARTE FRONTAL DO ROSTO HUMANO

Assis/SP
2021

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LUIZ MARCELO DE OLIVEIRA JUNIOR

TREINANDO UM *HAAR CASCADE* PARA A DETECÇÃO DA PARTE FRONTAL DO ROSTO HUMANO

Projeto de pesquisa apresentado ao curso de Bacharelado em Ciência da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Luiz Marcelo de Oliveira Junior
Orientador(a): Prof. Dr. Luiz Carlos Begosso

Assis/SP
2021

FICHA CATALOGRÁFICA

Oliveira, Luiz Marcelo.

Treinando um Haar Cascade para a Detecção da Parte Frontal do Rosto Humano / Luiz Marcelo de Oliveira Junior. Fundação Educacional do Município de Assis –FEMA – Assis, 2021.

35p.

1. Reconhecimento de padrões. 2. OpenCV. 3.Haar Cascade. 4. Visão Computacional. 5. Machine Learning.

CDD:
Biblioteca da FEMA

TREINANDO UM *HAAR CASCADE* PARA A DETECÇÃO DA PARTE FRONTAL DO ROSTO HUMANO

LUIZ MARCELO DE OLIVEIRA JUNIOR

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: _____
Luiz Carlos Begosso

Examinador: _____
Fábio Éder Cardoso

Assis/SP
2021

DEDICATÓRIA

Dedico este trabalho ao meu pai que se foi, a toda a minha família, em especial a minha mãe e esposa por estarem presentes na minha vida, me apoiando a cada conquista e me provendo força para seguir em frente.

AGRADECIMENTOS

Primeiramente agradeço ao grande **Luiz Carlos Begosso**, por aceitar esse desafio de me orientar nesse Trabalho de Conclusão de Curso.

Agradeço a minha mãe **Vera Lucia Nascimento da Silva** e meu pai **Luiz Marcelo de Oliveira** por sempre me ajudarem e estarem ao meu lado.

A minha esposa **Ana Carolini Rodrigues de Oliveira** por alegrar a minha vida e enfrentar as dificuldades sempre com um sorriso no rosto.

A todos os matemáticos, físicos, teóricos e acadêmicos que desenvolvem, ensinam e aplicam seu conhecimento para que outras pessoas possam aprender e posteriormente compartilharem seu conhecimento pois seu trabalho é de suma importância para o desenvolvimento de todos.

A minha família e a todos os professores que me ajudaram diretamente ou indiretamente.

RESUMO

A Visão Computacional tal como as áreas relacionadas a Inteligência Artificial, a cada dia vem ganhando espaço, particularmente cada campo de estudo procura simular algum sentido humano afim de automatizar alguma tarefa.

Independente de cada área há um estudo prático e teórico que resulta na criação de técnicas e algoritmos onde os mais populares e eficientes acabam se tornando métodos clássicos com o passar dos anos e posteriormente são estudados, analisados gerando uma nova técnica de acordo com o estado da arte.

Não existe técnica, algoritmo melhor ou pior existem áreas de aplicação um exemplo são os detectores baseados em *Haar Cascades* que foram criados em 2001 e são utilizados até hoje para o reconhecimento de padrões e vem inspirando a criação de novos métodos de detecção baseados em *Machine Learning*. A proposta desde trabalho consiste em estudar o algoritmo Viola-Jones, compreendê-lo e demonstrar o passo a passo desde a criação, treinamento e teste para a detecção de objetos com a biblioteca OpenCV.

Palavras-chave: Reconhecimento de padrões; OpenCV; Haar Cascade;Visão Computacional; Machine Learning;

ABSTRACT

Computer Vision, as well as the areas related to Artificial Intelligence, has been gaining ground every day, particularly each field of study seeks to simulate some human sense in order to automate some tasks.

Regardless of each area, there is a practical and theoretical study that results in the creation of techniques and algorithms where the most popular and efficient ones end up becoming classic methods over the years and are later studied, analyzed, generating a new technique according to the state of art.

There is no technique, better or worse algorithm, there are application areas an example are the detectors based on Haar Cascades that were created in 2001 and are still used today for pattern recognition and have been inspiring the creation of new detection methods based on Machine Learning. The purpose of this work is to study the Viola-Jones algorithm, understand it and demonstrate the step by step from creation, training and testing for object detection with the OpenCV library.

Keywords: Pattern Recognition; OpenCV;Haar Cascade;Computer Vision; Machine Learning;

Lista de Ilustrações

Sumário

1. INTRODUÇÃO

No fim dos anos 60 começaram os estudos sobre Visão Computacional e seu objetivo era imitar a visão biológica para dotar robôs inteligentes, apenas na década de 70 foram criados os primeiros algoritmos que atualmente servem de base. Com o avanço da eletrônica os computadores podiam processar grandes quantidades de dados tal como imagens e com o amadurecimento da internet foram disponibilizados banco de imagens para análises.

A Visão Computacional é um campo da Inteligência Artificial responsável pelo treinamento de computadores em busca da compreensão e interpretação do mundo visual, isso ocorre através do *Machine Learning* com o uso de imagens que permitem as máquinas a reagirem de acordo com o que é processado. Ela desenvolve a teoria e tecnologia para a construção de sistemas artificiais que estão presentes no nosso dia a dia tais como veículos autônomos, robôs industriais, detecção de eventos, biometria facial e até mesmo ajudando um míssil militar a acertar seu alvo com precisão.

O mercado da Visão Computacional deve crescer a uma taxa de 7.6% de 2020 a 2027 e arrecadando o valor de USD 19.04 bilhões a cada ano de acordo com a *Grand View Research* (2020).

Sistemas de Visão Computacional são usados para melhorar a segurança, um exemplo popular é o reconhecimento de faces usado na segurança dos smartphones estes sistemas são capazes de reconhecer diferentes padrões na retina e íris humana, em casos avançados pode ser usado como segurança residencial devido ao seu reconhecimento de faces. Análises de digitais são usadas no setor bancário e em laboratórios de pesquisas para acessos a diferentes áreas de acordo com o nível de privilégio do portador da digital.

Os avanços no campo de Visão Computacional são surpreendentes e de acordo com SaS (2021) a taxa de precisão de identificação de objetos passou de 50 a 90% em menos de uma década e os sistemas de hoje são mais precisos e rápidos no quesito de detecção e reação comparados a um ser humano.

Atualmente, existem bons algoritmos que abordam o tema do presente trabalho. De acordo com Rosebrock (2021), o *Haar Cascade* é um destes algoritmos utilizados para detecção e classificação de objetos. Ele faz parte da biblioteca de visão computacional OpenCV e

apresenta bons resultados nesta área e, por este motivo ele foi escolhido como objeto de estudo para este Trabalho de Conclusão de Curso.

1.1 OBJETIVOS

O objetivo deste trabalho consiste na exploração das etapas de construção de um sistema de Visão Computacional e no treinamento de um *Haar Cascade* para a detecção da parte frontal do rosto de humanos. Para atingir o objetivo, serão utilizados métodos e algoritmos de visão computacional e ferramentas da biblioteca OpenCV.

1.2 JUSTIFICATIVAS

O desenvolvimento desse trabalho deu-se pelo fato de que a Visão Computacional está em ascensão. A junção das tecnologias contidas no campo de Inteligência Artificial é capaz de gerar grande demanda para o mercado e sua aplicação está em toda parte, ou seja, o aprendizado pode ser aplicado em várias áreas.

A biblioteca OpenCV possui algoritmos de processamento, o que de acordo com Marengoni e Stringhini (2013), possibilita ao desenvolvedor uma vasta opção de métodos de manipulações e processamento de imagens digitais, assim como acesso simplificado para a realização de tarefas necessárias com as imagens a serem reconhecidas.

1.3 MOTIVAÇÃO

A principal motivação para o desenvolvimento desse trabalho é contribuir com o universo da visão computacional e introduzir novatos de maneira simples na área. De forma secundária, pretende-se adquirir mais conhecimento e experiências sobre a área de estudo.

1.4 ESTRUTURA DO TRABALHO

A estrutura do trabalho foi dividida em seis capítulos. O capítulo 1 é esta introdução no qual se estabeleceu os objetivos, justificativas e as motivações para a execução do trabalho. O capítulo 2 aborda os conceitos, características da grande área onde está situado o presente trabalho, a Visão Computacional. O algoritmo Viola-Jones é discutido na sessão 2.1, por conta de sua importância para o atingimento dos objetivos propostos. O capítulo 3 é exclusivamente para introduzir as tecnologias utilizadas no trabalho. No capítulo 4 é exemplificado o uso de um *Haar Cascade* treinado. No capítulo 5 são abordadas todas as etapas referentes ao treino de um *Haar Cascade* e na sessão 5.1 é aplicado o algoritmo treinado e por fim, no capítulo 6, apresenta-se a conclusão do trabalho.

2. VISÃO COMPUTACIONAL

A visão computacional é a ciência e tecnologia das máquinas que enxergam. Ela é responsável pelo desenvolvimento teórico e tecnológico extraindo informações significativas do ambiente a sua volta para a construção de sistemas de visão computacional.

As imagens são capturadas a partir de diversos meios tais como câmeras de vídeo, sensores, scanners entre outros dispositivos, estas informações permitem reconhecer, manipular e processar dados sobre objetos que compõem a imagem capturada.

Como muitas tecnologias distintas a visão computacional busca imitar a natureza humana e pode ser vista como complemento da visão biológica outra definição é de que visão computacional estuda e implementa sistemas capazes de enxergar por meio de processos artificiais, implementado por hardwares e softwares (BARELLI, 2018).

Essa tecnologia, alinhada a técnicas de aprendizado de máquinas, permite ao computador aprender e aperfeiçoar seu desempenho em diversas tarefas desde a encontrar anomalias em exames médicos a detectar defeitos em uma linha de montagem de carros (MILANO;HONORATO, 2021).

A Visão Computacional é complementada por inúmeras áreas onde destaca-se a Inteligência Artificial que é responsável por simular a inteligência humana e Aprendizado de Máquina ou *Machine Learning* que explora a construção de algoritmos que compreendem de maneira autônoma (PAIXAO, 2018).

Um sistema de visão computacional é empregado em diversas áreas e é implementado com diferentes tecnologias e apresentam um fluxo comum que deve ser capaz de adquirir, processar e interpretar imagens correspondentes a cenas reais (MARQUES FILHO; VIEIRA NETO, 1999). A figura 1 ilustra as etapas que compõem o processo para a efetivação de um sistema de visão computacional.

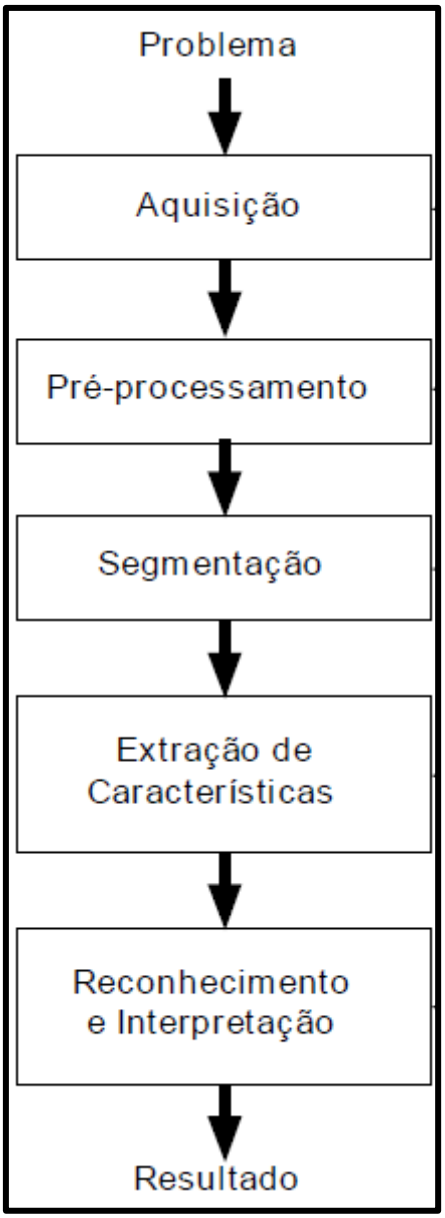

Figura 1 - Etapas do sistema de Visão Computacional (Fonte: MARQUES FILHO; VIEIRA NETO, 1999, p. 9)

Antes de começar a desenvolver um sistema de Visão Computacional deve-se dominar o problema e definir o objetivo da solução. A seguir são detalhadas as etapas do sistema de visão computacional de acordo com Marques Filho e Vieira Neto (1999, p.9).

Aquisição da imagem: Dispositivos são utilizados para obter e digitalizar a imagem com qual será trabalhada podendo variar entre uma imagem bidimensional, tridimensional ou um vídeo que nada mais é uma sequência de imagens.

Pré-processamento: Após obter a imagem digitalizada é necessário aprimorar sua qualidade pois a imagem pode apresentar diversas imperfeições tais como presença de pixels ruidosos, contraste, brilho inadequado.

Segmentação: Uma das tarefas mais difíceis presentes nos sistemas de visão computacional, consiste em definir os objetos ou região de interesse afim de facilitar a extração de características desses objetos.

Extração de características: A partir de figuras geométricas ou padrões naturais presentes no objeto de interesse tal como os olhos, boca e nariz em seres humanos em imagens são uteis para diferenciar dos demais objetos no mesmo plano, e devem ser representados por uma estrutura de dados adequada ao algoritmo de reconhecimento. Nesta etapa a entrada é uma imagem e a saída é um conjunto de dados correspondentes a imagem processada.

Reconhecimento de padrões: Por fim ocorre o processamento de alto nível com o objetivo de reconhecer o objeto através das suas características e defini-lo em uma determinada classe. Esta etapa é responsável por validar os resultados e definir se eles são satisfatórios ou não.

2.1. ALGORITMO VIOLA-JONES

O algoritmo Viola-Jones é um detector de objetos a partir de padrões que foi proposto em 2001 por Paul Viola e Michael Jones que também é popularmente conhecido por *Haar Cascade*. Os autores foram motivados pela dificuldade na detecção de faces. O algoritmo proposto, o Viola-Jones, pode ser usado para detectar uma variedade de objetos.

Sua abordagem é baseada em *Machine Learning*, são usadas imagens positivas e negativas para o treinamento onde imagens positivas contém o objeto de interesse e negativas não contém o objeto de interesse. Este algoritmo consiste em encontrar padrões onde determinada região da imagem possui diferença de contraste, em conjunto com o algoritmo AdaBoost utilizado para aumentar a performance de outros algoritmos de *Machine Learning* (VIOLA; JONES, 2001).

Para encontrar diferenças de contraste são utilizadas máscaras conhecidas como *Haar Features* que funcionam como filtros. Existem diversos tipos de máscaras que usadas em

conjunto provêm maior confiabilidade na detecção do objeto. A Figura 2 ilustra *Haar Features* existentes.

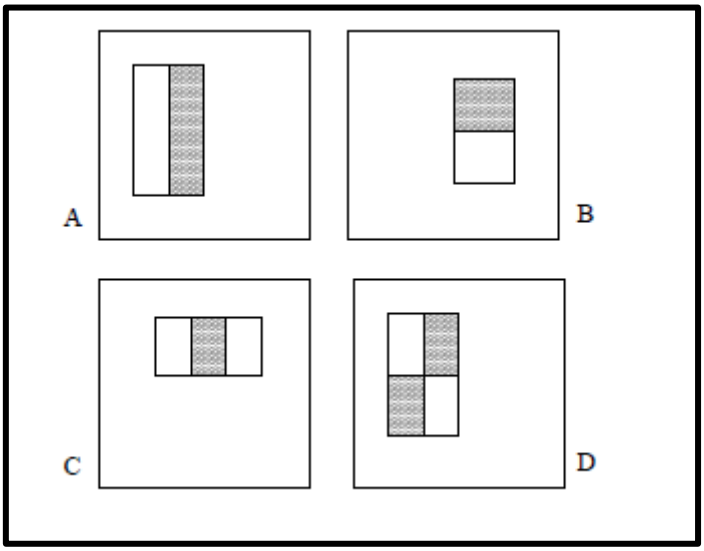

Figura 2 - *Haar Features* (Fonte: VIOLA; JONES, 2001, p. 2)

A variação da luz difere nos resultados sendo assim as fotos devem ser capturadas em um ambiente com iluminação controlada e caso necessário a imagem é convertida em tons de cinza, diferentes máscaras sobrepõem o objeto de interesse (BARELLI, 2018). A Figura 3 ilustra a aplicação das *Haar Features*.

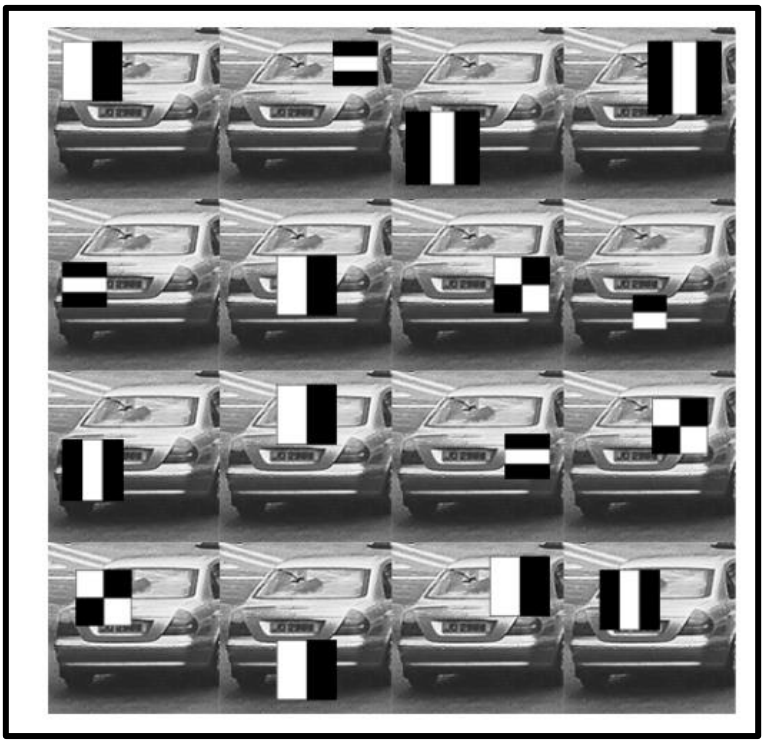

Figura 3 - Aplicação do Filtro (Fonte: BARELLI, 2018, p. 282)

A soma dos pixels que se encontram dentro dos retângulos brancos são subtraídos da soma dos pixels nos retângulos cinzas, o resultado dessa operação é a imagem integral que contém a soma dos pixels acima e a esquerda (x,y), que podem ser reconhecidas rapidamente ao percorrer a imagem (VIOLA;JONES, 2001). A Figura 4 ilustra o funcionamento da imagem integral.

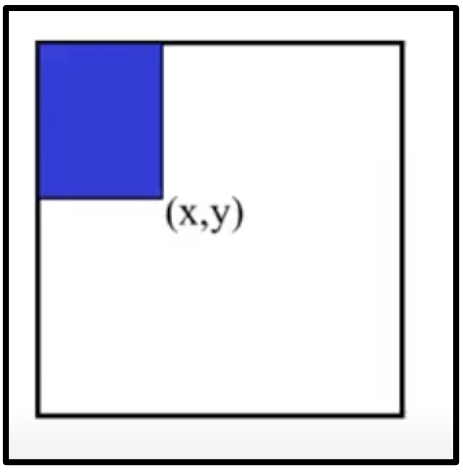

Figura 4 – Imagem Integral (Fonte: o autor)

3. OPENCV E PYTHON

De acordo com a empresa mantenedora OpenCV (2021), a OpenCV é um acrônimo de *open source computer vision library* ou biblioteca de Visão Computacional de uso livre, originalmente desenvolvida pela Intel nos anos 2000 para criação de aplicativos na área de Visão Computacional. Esta biblioteca foi desenvolvida utilizando a linguagem de programação C++ e suporta linguagens tais como Java, Python e Visual Basic.

A biblioteca possui módulos de processamento de imagens, entrada e saída de vídeo, estrutura de dados, álgebra linear e interface gráfica. Contando com mais de 350 algoritmos de visão computacional como filtro de imagens, calibração de câmera, reconhecimento de objetos, análise estrutural e seu processamento é em tempo real.

Suas áreas de aplicação são as mais diversas desde reconhecimento facial, identificação de objetivos, robôs moveis e veículos autônomos. É compatível com a maioria das plataformas disponíveis desde Android, Linux até Windows.

É uma das mais completas bibliotecas referentes a Visão Computacional e seus principais módulos são (OpenCV, 2021):

- core – Principais funcionalidades
- imgproc – Processamento de imagens
- imgcodes – Leitura e escrita em imagens e vídeos
- videoio – Entrada e saída de vídeo
- highgui – Interface de usuário de alto nível
- vídeo – Análise de vídeo
- calib3d – Calibração de câmera e reconstrução em 3D
- features2d – Recursos de framework 2D
- objdetect – Detecção de objetos
- ml – Machine learning
- flann – Clustering e pesquisa em espaços multidimensional
- photo – Fotografia computacional
- stitching – Costura de imagens
- cudaarithm – Operações em matrizes

- cudabgsegm – Segmentação em background
- cudacodec – Codificação e decodificação em vídeos
- cudafeatures2d – Recursos de detecção e descrição
- cudafilters – Filtro de imagens
- cudaimgproc – Processamento de imagens
- cudalegacy – Suporte aos recursos legados
- cudaobjtdetect – Detecção de objetos
- cudaopticalflow – Fluxo ótico
- cudastereo – Correspondência estérea
- cudawarping – Distorção de imagens
- cudev – Camada do dispositivo
- shape – Distância e reconhecimento em imagens
- superres – Alta resolução
- videostab – estabilização em vídeos
- viz – reprodutor 3D

Python é a linguagem escolhida para a criação e execução dos códigos, pois é uma linguagem de programação de fácil aprendizado e a maioria dos conteúdos referente a Visão Computacional são escritos nesta linguagem. Lançada em 1991 por Guido Van Rossun é de programação de alto nível interpretada de script, imperativa, funcional e orientada a objetos, amplamente utilizada em Ciência de Dados, Aprendizado de Máquina, Desenvolvimento Web e Automatização de Tarefas.

4. DETECTANDO ROSTOS COM UM *HAAR CASCADE* TREINADO

A biblioteca OpenCV provê diversos recursos para trabalhar com detecção de padrões e objetos em imagens, um deles é que a biblioteca contém alguns *haar cascades* treinados que permitem a detecção de rostos, olhos e emoções em imagens ou vídeos.

O objetivo deste capítulo é utilizar recursos de OpenCV para detectar os rostos na Figura 5, que ilustra dois jogadores de futebol.

Figura 5 - Imagem Teste (Fonte: o autor)

A função detectMultiscale ilustrada na Figura 6 necessita de três parâmetros para ser executada onde o primeiro é a imagem em tons de cinza, o segundo é a scaleFactor que é o fator de escala para a redução da imagem que deve ser maior que 1.0 e o terceiro é a quantidade mínima de vizinhos que que a imagem será comparada, o resultado de saída é descrito pela seguinte imagem.

face_detection.py > ...

```python
#importação da biblioteca de visão computacional
import cv2 as cv

#carrega os arquivos de caracteristicas dos rostos
cascadeFace = cv.CascadeClassifier('haar_face.xml')

#importação da imagem que será analisada
originalImage = cv.imread('photos\messiandronaldo.jpeg')

#Imagem convertida em cinza
grayImage = cv.cvtColor(originalImage, cv.COLOR_BGR2GRAY)

#faces detectadas
faces = cascadeFace.detectMultiScale(grayImage,scaleFactor=1.3, minNeighbors=5)

#desenha um retangulo nas faces detectadas
for(x,y,w,h) in faces:
    cv.rectangle(originalImage, (x,y), (x+w, y+h), (000, 255, 000), 2)

#mostra a imagem com as devidas faces localizadas
cv.imshow("Results", originalImage)

#espera uma tecla para fechar a imagem
cv.waitKey(0)

#destroi todas as janelas
cv.destroyAllWindows()

```

Figura 6 - Código de Detecção (Fonte: o autor)

A Figura 7 ilustra o resultado da execução do código detectando os dois rostos contidos na imagem de teste.

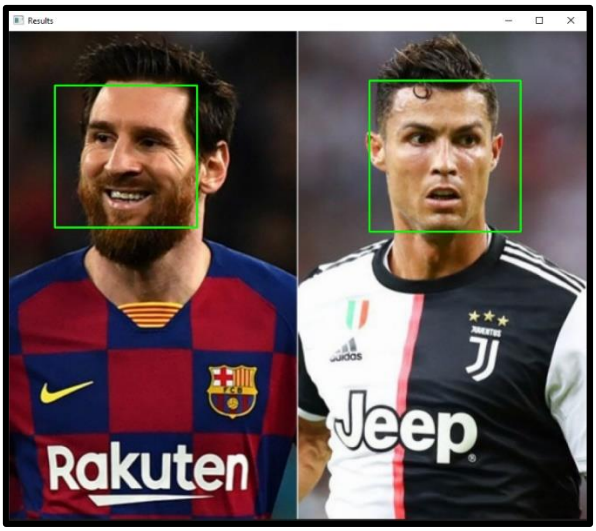

Figura 7 - Resultado Final (Fonte: o autor)

Caso haja falsos positivos deve-se alterar os parâmetros referentes a escala da imagem ou número mínimo de vizinhos para ter uma melhor detecção.

5. TREINANDO O PRÓPRIO *HAAR CASCADE*

O treino de um *Haar Cascade* é composto por quatro etapas onde a primeira é a coleta das imagens positivas e negativas do objeto de interesse, em que neste trabalho optou-se por identificar a parte frontal do rosto humano (REZAEI, 2017).

Com o intuito de explicar a mecânica para o treinamento do próprio *Haar Cascade*, optou-se por apresentar o trabalho de Rezael (2017). Em seu trabalho, o autor utilizou 200 imagens positivas para o treino. As pessoas, em sua maioria, são nativos de Auckland na Nova Zelândia. A Figura 8 ilustra alguns dos rostos contidos dentre as 200 imagens positivas usadas no treinamento do *Haar Cascade*.

Figura 8 - Imagens Positivas (Fonte: REZAEI, 2017)

A Figura 9 ilustra as imagens negativas que não possuem o objeto de interesse que em sua maioria são fotos de paisagens, prédios, salas de estar e ruas. As imagens foram convertidas em tons de cinza e foram utilizadas 200 imagens.

Estas etapas caracterizam a aquisição e pré-processamento em um sistema de Visão Computacional, pois as imagens capturadas, chamadas de imagens negativas, foram processadas em tons cinzas para facilitar o processo de identificação.

Figura 9 - Imagens Negativas em Tons Cinzas (Fonte: REZAEI, 2017)

Treinar um *Haar Cascade* é algo trabalhoso e demorado, mas a biblioteca OpenCV facilita a vida do programador com ferramentas próprias para agilizar o trabalho duro. Foi utilizado a versão 3.4.11, que contém as ferramentas de treino, criação de amostras e anotação,

versões acima desta utilizam outra abordagem quando a questão é detecção de objetos. A estrutura do presente projeto está ilustrada na Figura 10.

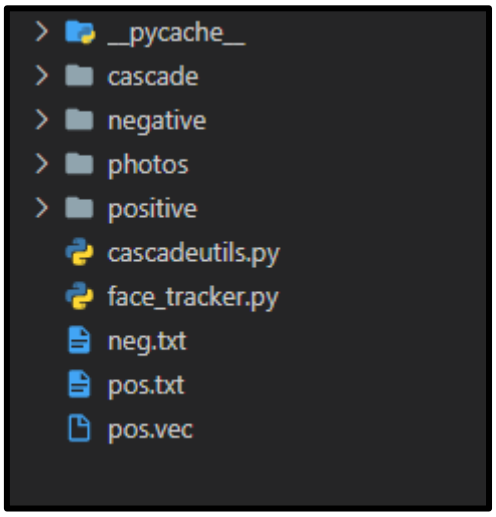

Figura 10 - Estrutura do Projeto (Fonte: o autor)

A pasta *cascade* é onde ficarão armazenados as etapas de treinamento e o *Haar Cascade* treinado. As pastas *negative* e *positive* contêm as imagens positivas e negativas, o arquivo face_tracker contém o mesmo código utilizado no Capítulo 4 para a detecção de objetos e, por fim, a pasta *photos* armazena as imagens para testar o *Haar Cascade*.

O segundo passo é criar arquivos de texto com o caminho de imagens negativas e marcar as áreas de interesse nas imagens positivas. O arquivo *cascadeutils* com extenção.py é usado para criar um arquivo de texto com o caminho das imagens negativas que será chamado neg. A Figura 11 ilustra o código utilizado para listar o caminho das imagens.

```
1  import os
2
3  def generate_negative_descripton_file():
4      with open('neg.txt', 'w') as f:
5          for filename in os.listdir('negative'):
6              f.write('negative/' + filename + '\n')
```

Figura 11 - Código para Listar Caminhos (Fonte: o autor)

Para executar o código acima deve-se acessar o compilador de códigos com o comando python no terminal e depois executar as instruções ilustradas na Figura 12.:

```
>>> from cascadeutils import generate_negative_descripton_file
>>> generate_negative_descripton_file()
>>> exit()
```

Figura 12 - Instruções de Execução de Código (Fonte: o autor)

Assim, o arquivo de texto neg é gerado com todos os caminhos referente as imagens negativas do projeto. A Figura 13 ilustra o resultado das instruções.

Figura 13 - Caminhos Gerados (Fonte: o autor)

A parte mais trabalhosa é desenhar retângulos nas áreas de interesse de todas as imagens positivas. Os retângulos serão desenhados a partir da ferramenta *annotation*, que gera um arquivo com o caminho e coordenadas das imagens positivas. A ferramenta é acessada pelo terminal na pasta raiz com o caminho onde a biblioteca OpenCV está instalada com os seguintes parâmetros --annotations=pos.txt indicando qual será o nome do arquivo que

contém o caminho das imagens positivas e --images=positive/ sinalizando o caminho das imagens. A Figura 14 ilustra a instrução a ser executada.

```
C:\Users\luizm\OneDrive\Documentos\opencv\build\x64\vc15\bin\opencv_annotation.exe --annotations=pos.txt --images=positive/
```

Figura 14 - Execução da Ferramenta *Annotation* (Fonte: o autor)

Executando o comando acima a ferramenta é aberta, assim inicia-se o desenho dos retângulos, pressiona-se a tecla c para confirmar a área selecionada, d para cancelar a área selecionada, n para a próxima imagem após confirmar a área e esc caso seja necessário fechar a ferramenta. A Figura 15 ilustra a utilização da ferramenta *annotation*.

Esta etapa é caracterizada pela segmentação, pois é definida a área de interesse e é realizada sua demarcação.

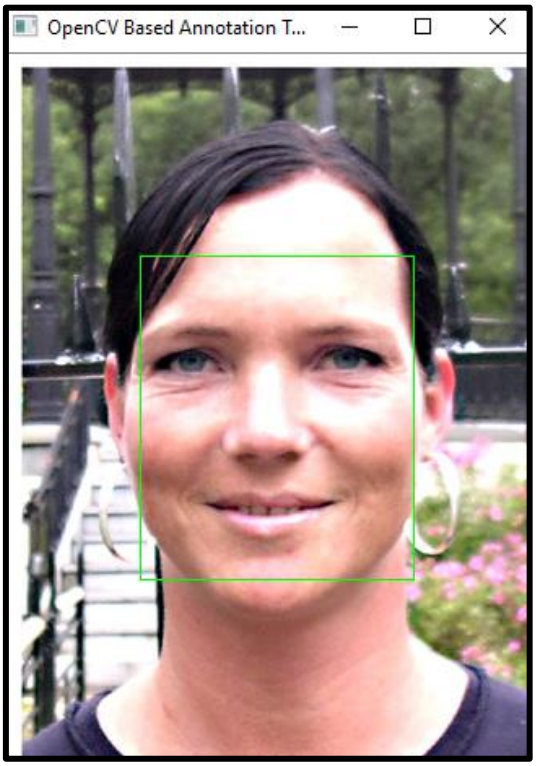

Figura 15 - Area de Interesse (Fonte: o autor)

Por fim desta etapa é gerado um arquivo de texto chamado pos que contém o caminho e coordenadas dos retângulos em cada imagem, semelhante a imagem da Figura 16:

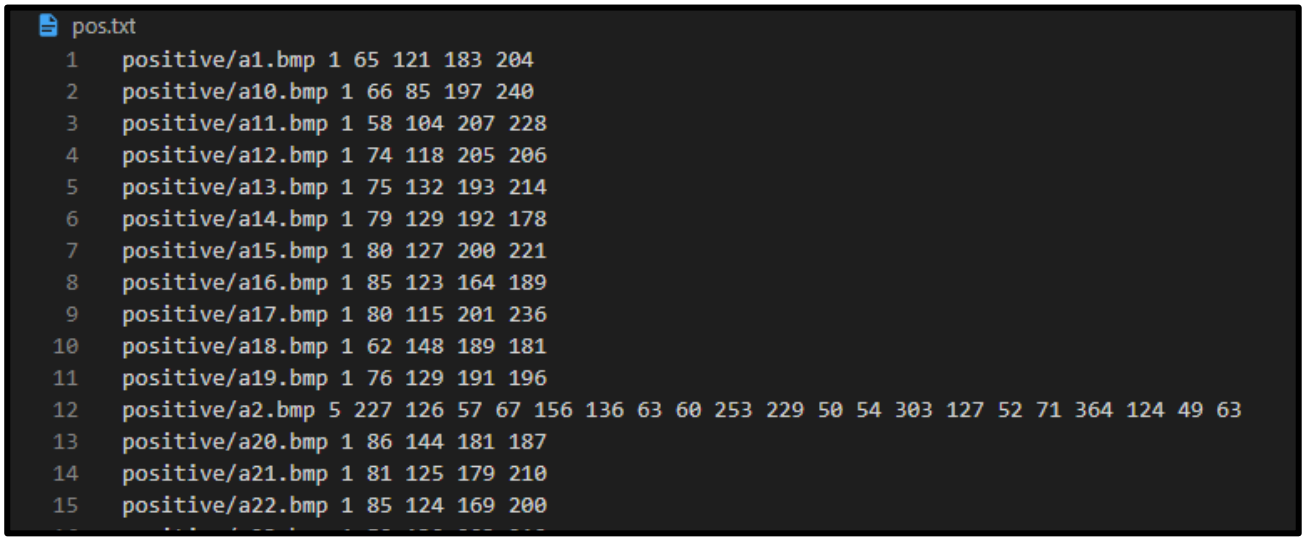

Figura 16 - Coordenadas de Áreas de Interesse (Fonte: o autor)

O terceiro passo é gerar o vetor de imagens baseado nas coordenadas na etapa anterior, semelhante ao penúltimo passo deve-se acessar a ferramenta *create samples* no terminal com o caminho onde a biblioteca OpenCV está instalada com os seguintes parâmetros:

-info com o caminho do arquivo pos.txt

-w com a largura do objeto

-h com a altura do objeto

-num com o número de imagens positivas

-vec com o nome do arquivo vetorial que será gerado

A Figura 17 ilustra a instrução ao ser executada.

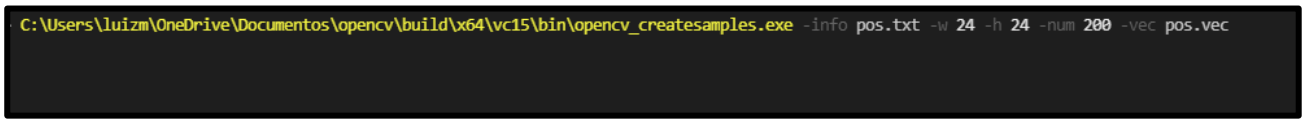

Figura 17 - Execução da Ferramenta Create Samples (Fonte: o autor)

No terceiro passo é realizado implicitamente a etapa de extração de características e com o arquivo vetorial gerado inicia-se o quarto e último passo que consiste em treinar o *Haar Cascade,* novamente é realizado implicitamente a etapa de reconhecimento e interpretação e ocorre o processamento em alto nível. Para isso, deve-se acessar a ferramenta *train cascade* pelo terminal a partir do caminho onde a biblioteca OpenCV está instalada com os seguintes parâmetros:

-data com o caminho onde serão armazenados os arquivos gerados no treino

-vec com o caminho do vetor de imagens

-bg com o caminho do arquivo que contém o caminho das imagens negativas

-w com a largura do objeto

-h com a altura do objeto

-numPos com o número de imagens positivas

-numNeg com o número de imagens negativas

-numStages com o número de estágios que o treino será executado.

A Figura 18 ilustra a instrução ao ser executada.

Figura 18 - Execução da Ferramenta Train Cascade (Fonte: o autor)

Após o término de execução do comando acima, é gerado o relatório de cada estágio do treinamento, *POS count* indica a contagem do número de imagens consumidas, *NEG count* contém a taxa de aceitação das imagens negativas, N indica o número de *Haar Features* utilizadas no treino, HR representando a taxa de acertos, FA com o número de alarmes falsos e por fim o tempo que o treino demorou gerando o arquivo *cascade* de cada estágio. A Figura 19 ilustra o relatório do último estágio do treino.

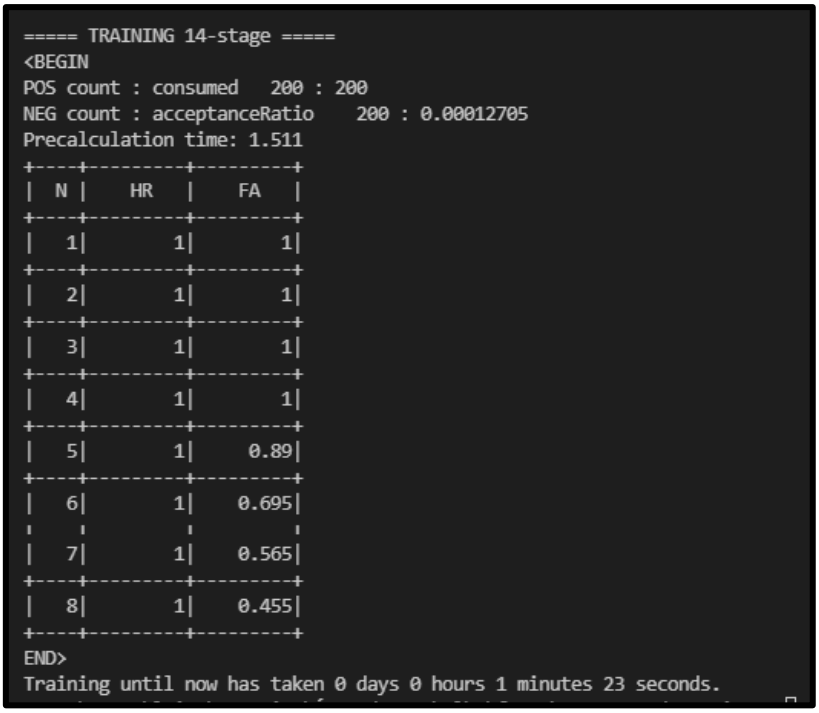

Figura 19 - Resultado do Último Estágio de Treino (Fonte: o autor)

5.1. APLICANDO O ALGORITMO TREINADO

Nos seguintes testes ilustrados na Figura 20 foi utilizado o mesmo código do capítulo 4 com alguns ajustes na escala e vizinhos. Os primeiros testes foram com alguns professores e os resultados não foram satisfatórios em uma imagem foi detectado apenas a parte central do rosto e em outro alguns professores não foram detectados.

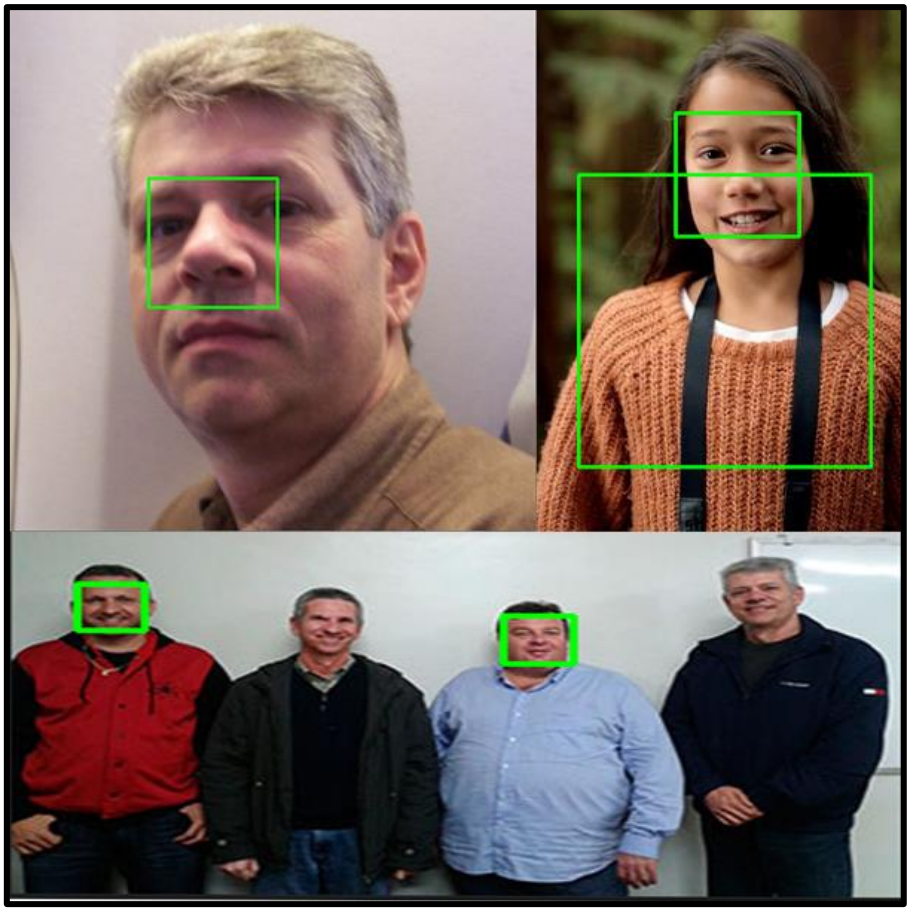

Figura 20 - Teste de Detecção (Fonte: o autor)

Assim, testou-se em nativos de Auckland que não foram usados no treinamento e o resultado foi o satisfatório como ilustra a Figura 21 detectando ambos os rostos.

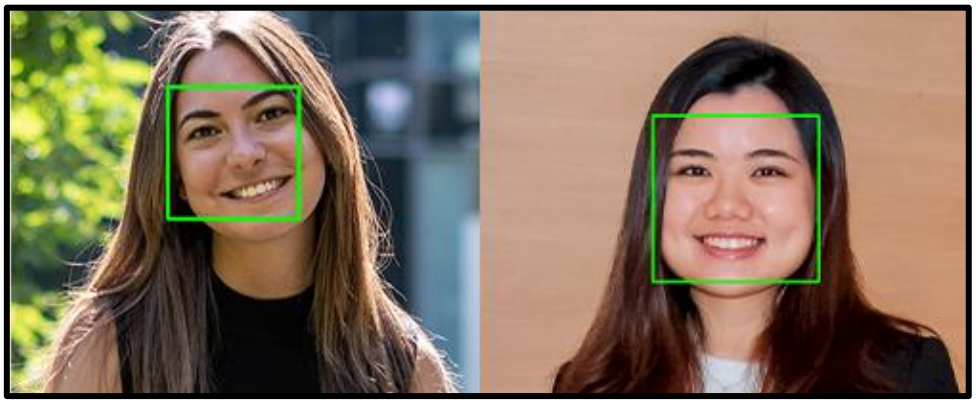

Figura 21 - Teste de Detecção em Nativos (Fonte: o autor)

Tamanho do rosto e etnia tem grande influência no detector. Para o aperfeiçoamento é necessário incluir uma variedade de rostos de diferentes pessoas ao redor do mundo e dobrar o número de imagens negativas. A métrica comum usada para avaliar é *precision* e *recall,* onde *recall* faz referência a quantos objetos de interesse foram detectados e *precision* a quantas detecções do objeto de interesse foram verdadeiras, a partir dessas duas métricas pode-se aplicá-las a fórmula *F-Measure* onde F = 2 * ((*precision* * *recall*) / (*precision* * *recall*)), resultados acima de 95% tornam o detector aceitável.

Para criar um detector melhor é aconselhável ao menos 2500 imagens positivas e 5000 imagens negativas, áreas de interesse de 24x24 são perfeitas para o treinamento podendo variar entre 50x50 e 80x80. Quanto maior a área de interesse maior será o tempo de treinamento e possivelmente o resultado será melhor. Deve-se treinar ao menos 15 estágios e sempre verificar os resultados de cada estágio com a *F-Measure* e quando encontrar um equilíbrio entre as duas métricas e o resultado *F-Measure* utiliza-se o *cascade* referente àquele estágio.

6. CONCLUSÃO

Atualmente os sistemas de visão computacional vem adotando técnicas de *Deep Learning* e provando que são mais eficazes que os métodos clássicos tais como os de *Cascade,* isto não é um ponto negativo, pois estes métodos foram os que inspiraram os métodos mais avançados e entender seu funcionamento é necessário para a criação de novos métodos.

O treinamento de um *Haar Cascade* é algo que demanda muito tempo e trabalho devido as etapas de aquisição de imagens, demarcação da área de interesse em cada imagem positiva e a carga de trabalho realizada implicitamente pela ferramenta de treino da biblioteca OpenCV.

O presente Trabalho de Conclusão de Curso objetivou explorar as etapas para a construção de um sistema de Visão Computacional e no treinamento de um Haar Cascade. Especificamente, utilizou-se a linguagem Python e a biblioteca OpenCV para a detecção da parte frontal do rosto humano.

Os resultados obtidos sobre as imagens dos nativos da Nova Zelândia, foram positivos, ao passo que, quando foram utilizadas imagens dos professores da Fema, o sistema não conseguiu reconhecer todos os rostos nas imagens.

Futuramente, para melhorar a acurácia do sistema, um maior número de imagem positivas e negativas poderiam fazer parte do processo de treinamento.

7. REFERÊNCIAS

ALBUQUERQUE, Márcio Portes de; ALBUQUERQUE, Marcelo Portes de. **Processamento de Imagens: Métodos e Análises.** Rio de Janeiro, 2021.

ANTONELLO, Ricardo. **Introdução a Visão Computacional com Python e OpenCV.** Santa Catarina, 2016.

BARELLI, Felipe da Costa. **Introdução A Visão Computacional – Uma Abordagem pratica com Python e OpenCV.** Espírito Santo: Casa do Código, 2018. 5 p.

Cascade Cassifier Training. Disponível em:< https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html >. Acesso em: 10 julho 2021.

Computer Vision Market Size, Share & Trends Analysis Report by component (Hardware Software) By Product Type (Smart Camera-Based, PC based) By Application, By Vertical, By Region and Segment Forecasts 2020 – 2027. Disponível em: https://www.grandviewresearch.com/industry-analysis/computer-vision-market>. Acesso em: 9 julho 2021.

FERRUGEM, Anderson Priebe. **Indexação Automatizada de Imagens.** Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Rio Grande do Sul, 2004.

LIENHART, Rainer; MAYDT, Jochen. **An Extended Set of Haar-Like Features for Rapid Object Detection.** Santa Clara, 2002. 2 p.

MARENGONI, Maurício; STRINGHINI, Denise. Tutorial: Introdução à Visão Computacional usando OpenCV. **Revista de Informática Teórica e Aplicada**: RITA, Cidade, v. 1, p.1-36, 01 fev. 2009.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens.** Rio de Janeiro: Brasport, 1999. 9 p.

MILANO, Danilo de; HONORATO, Luciano Barrozo. **Visão Computacional.** Limeira, 2021.

MÜLLER, Daniel Nehme; DARONCO, Everaldo Luis. **Operações Aritméticas em Imagens.** Porto Alegre, 2000.

ROSEBROCK, Adrian. Pyimagesearch,2021. **OpenCV Haar Cascades.** Disponível em: < https://www.pyimagesearch.com/2021/04/12/opencv-haar-cascades/>. Acesso em: 26 julho 2021.

OpenCV Modules. Disponível em:< https://docs.opencv.org/3.4.11/>. Acesso em: 10 julho 2021.

PAIXAO, E P. **Aplicação do Algoritmo Viola-Jones na Detecção de Objetos**. 2018. Trabalho de Conclusão de Curso. Curso de Engenharia Elétrica, Universidade Regional do Noroeste do Estado do Rio Grande do Sul - UNIJUÍ, Ijuí, 2018.

QUEIROZ, José Eustáquio Rangel de; Gomes, Herman Martins. **Introdução ao Processamento Digital de Imagens.** Campina Grande, 2021.

REZAEI, Mahdi. **Creating A Cascade of Haar-Like Classifiers: Step by Step.** Auckland, 2017. 1 – 8 p.

ROCHA, Carlos Diego Franco da. **Aplicação do algoritmo Haar Cascade em um Sistema Embarcado para detecção de ovos do mosquito Aedes Aegypti em Palhetas de Ovitrampas.** Trabalho de Conclusão de Curso Pau dos Ferros: Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – IFRN, 2018.

ROCHA, Patrick Francis Gomes. **Abordagem teórica e aplicabilidade de Detecção facial em Aplicativos Móveis com *OpenCV* e *Google Android*.** Trabalho de Conclusão de Curso Assis: Fundação Educacional do Município de Assis - FEMA, 2013.

RODRIGUES, Matheus Bezerra Estrela. **Estudo da aplicação do Algoritmo Viola-Jones à Detecção de Pneus com Vistas ao Reconhecimento de Automóveis.** Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina, Centro de Engenharia Elétrica e Informática. Campina Grande, 2012.

VIOLA, Paul; JONES, Michael. **Rapid Object Detection Using a Boosted Cascade of Simple Features.** Cambridge, 2001. 2 p.

Visão Computacional: o que é e qual sua importância. Disponível em:< https://www.sas.com/pt_br/insights/analytics/computer-vision.html/>. Acesso em: 9 julho 2021.

Visão Computacional: porque você precisa mirar nessa área de IA. Disponível em:< https://www.supero.com.br/blog/visao-computacional-por-que-voce-precisa-mirar-nessa-area-de-ia/>. Acesso em: 9 julho 2021.