UNIBTA - CENTRO UNIVERSITÁRIO
SISTEMAS PARA INTERNET

Computação Gráfica

ANDRE MONTEIRO CORTE BRILHO
CLAUDIA HIKARY YAMADA
GUILHERME BEZERRA DOS SANTOS
HÉRICLES THOMAS DA SILVA
VICTOR AUGUSTO SOUZA GROSSI

ESTUDO DE DETECÇÃO E RECONHECIMENTO DE FACES E OBJETOS

Orientador: PROF. SERGIO R. M. PENEDO
Turma: 63SIZA

SÃO PAULO
8 de Junho de 2018

Trabalho do Projeto Modular
Apresentado ao Centro Universitário UNIBTA.
Curso Tecnólogo em Sistemas para Internet
Orientador: Prof. Sergio R. M. Penedo

SÃO PAULO
8 de Junho de 2018

O fator decisivo para vencer o maior obstáculo é, invariavelmente, ultrapassar o obstáculo anterior.

Henry Ford

RESUMO

Este artigo apresenta conceitos e técnicas sobre o uso de computação gráfica em função de detecção de faces e objetos. O objetivo principal desta pesquisa é mostrar como é a composição de um sistema que realiza detecção e reconhecimento facial, utilizando-se de algoritmos e bibliotecas de linguagem de programação e fazer uma comparação entre as abordagens utilizadas.

Palavras-chave: Visão Computacional. Computação Gráfica. Detecção. Reconhecimento. Facial. Objetos. OpenCV. Dlib. Haarcascade. CNN. HOG.

ABSTRACT

VersThis article presents concepts and techniques on the use of computer graphics in face and object detection function. The main objective of this research is to show how the composition of a system is which performs facial recognition and recognition using algorithms and programming language libraries and compares the approaches used.

Keywords: Computer Vision. Computer Graphics. Detection. Recognition. Facial. Objects. OpenCV. Dlib. Haarcascade. CNN. HOG.

Conteúdo

1 Introdução **3**
- 1.1 Origem da Computação Gráfica . . . 3
- 1.2 Áreas . . . 3
- 1.3 Segmentos . . . 4

2 Considerações Gerais **4**
- 2.1 Campos relacionados . . . 5
- 2.2 Identificação de bordas . . . 5
- 2.3 Reconhecimento de objeto . . . 5

3 Ambiente de Desenvolvimento **6**
- 3.1 Bibliotecas Utilizadas . . . 6

4 Abordagens **6**
- 4.1 Técnicas . . . 6
- 4.2 Algoritmos . . . 7

5 Introdução aos Algoritmos **7**
- 5.1 Haarcascade . . . 7
- 5.2 Histogram of Oriented Gradiente(HOG) . . . 7
- 5.3 Support Vector Machine(SVM) . . . 8
- 5.4 Convonulsed Neural Network(CNN) . . . 8
- 5.5 K Nearest Neighbor(KNN) . . . 9

6 Etapas para o Reconhecimento Facial **9**
- 6.1 Detecção de Faces . . . 9
- 6.2 Detecção de Pontos de Referência Facial . . . 9
- 6.3 Treinamento do Algoritmo . . . 9

7 Etapas para o Reconhecimento Objetos **9**
- 7.1 Imglab . . . 9
- 7.2 Gerar XML . . . 10
- 7.3 Preditores de Forma . . . 10

8 Codificação **10**
- 8.1 App.py que contém a interface gráfica . . . 10
- 8.2 Detecção Facial usando o Haarcascade . . . 12
- 8.3 Detecção Facial usando o HOG . . . 12
- 8.4 Detecção usando o HOG com pontuação de classificação . . . 13
- 8.5 Detecção Facial usando CNN . . . 13
- 8.6 Comparação entre o Haarcascade e CNN . . . 14
- 8.7 Comparação entre o Haarcascade, HOG e CNN . . . 14
- 8.8 Alinhamento de Faces . . . 15
- 8.9 Extração de Pontos Faciais . . . 16
- 8.10 Treinamento do Algoritmo para Reconhecimento Facial . . . 17
- 8.11 Implementação e Teste do Reconhecimento Facial com CNN + KNN . . . 18
- 8.12 Treinamento do Algoritmo para Detecção de Objetos . . . 19
- 8.13 Treinamento do Algoritmo para Predição de Forma . . . 19

8.14 Implementação e Teste do Reconhecimento de Objeto 20
8.15 Exibicação dos Pontos encontrados pelo algoritmo já treinado 20
8.16 Reconhecimento de Logotipo 21
8.17 Reconhecimento de Logotipo a partir da webcam 21

9 Referências **22**

1 Introdução

Computação Gráfica é o conjunto de técnicas que contempla a área de estudo de imagens em forma de representação de dados e informação ou em sua forma de recriação do mundo real.

Segundo a ISO – (*International Organization for Standardization*), a definição de computação gráfica é: "um conjunto de ferramentas e técnicas para converter dados para ou de um dispositivo gráfico através do computador".

1.1 Origem da Computação Gráfica

Parece existir um consenso entre os pesquisadores de que o primeiro computador a possuir recursos gráficos de visualização de dados numéricos foi o Whirlwind I, desenvolvido pelo MIT.

- Em 1959 surgiu o termo Computer Graphics criado por Verne Hudson;
- Em 1962, surgiu uma das mais importantes publicações da computação gráfica de todos os tempos, a tese de Ivan Sutherland (*Sketchpad – A Man-Machine Graphical Communication System*);
- Década 50: Começaram a surgir os primeiros computadores com recursos gráficos;
- Década 60: Surgimento do CAD;
- Década 70: Surgimento do Predecessor do Macintosh e algoritmos que são usados até hoje;
- Década 80: Surgimento de técnicas de iluminação global);
- Década 90: Em 1992 surge o OpenGL, Filme Jurassic Park em 1993, primeiras placas gráficas para PC da NVIDIA, em 1999.;
- Ano 2000: Em 2001, são lançados diversos sucessos de bilheteria, como Shrek (Dreamworks), com novos métodos de síntese e animação de personagens e Final Fantasy, o triunfo da modelagem de personagens 3D; também não poderíamos deixar de citar Matrix Reloaded, com personagens virtuais sendo usados, dentre outras coisas, para cenas de risco.

1.2 Áreas

A computação gráfica atualmente é uma área que engloba, para melhor descrição didática, pelo menos três grandes subáreas: a **Síntese de Imagens**, o **Processamento de Imagens** e a **Análise de Imagens**.(CONCI; AZEVEDO,cap 1, p.8, 2003).

A **Síntese de Imagens** são as representações visuais de objetos criados pelo computador a partir de formas geométricas. Dentre os três essa subárea é a principal quando se trata de computação gráfica.

O **Processamento de Imagens** se trata do processamento de imagens para o meio digital, um exemplo seria o Photoshop e seus softwares correlatos que realizam processamento de imagens com objetivo de se fazer modificações, tratamentos e realces em imagens.

Quanto a **Análise de Imagens** ela esta inserida no contexto das imagens digitais e as analisa para se obter características desejadas.

1.3 Segmentos

Quando falamos a palavra computação gráfica as pessoas tendem a associar que o tema é ligado somente a tecnologia, porém existe uma infinidade de aplicações e segmentos que não são exclusivos da área computacional. Veja na tabela abaixo:

Artes	Efeitos especiais, modelagens criativas, esculturas e pinturas
Medicina	Exames, diagnósticos, estudo, planejamento de procedimentos
Arquitetura	Perspectivas, projetos de interiores e paisagismo
Engenharia	Em todas as suas áreas(mecânica, civil, aeronáutica, etc)
Geografia	Cartografia, GIS, georreferenciamento, previsão de colheitas
Meteorologia	Previsão do tempo, reconhecimento de poluição
Astronomia	Tratamento de imagens, modelagem de superfícies
Marketing	Efeitos especiais, tratamento de imagens, projetos de criação
Segurança Pública	Definição de estratégias, treinamento, reconhecimento
Indústria	Treinamento, controle de qualidade, projetos
Turismo	Visitas virtuais, mapas, divulgação de reservas
Moda	Padronagem, estamparias, criação, modelagens, gradeamentos
Lazer	Jogos, efeitos em filmes, desenhos animados, propaganda
Processamento de Dados	Interface, projeto de sistemas, mineração de dados
Psicologia	Terapias de fobia e dor, reabilitação
Educação	Aprendizado, desenvolvimento motor, reabilitação

Tabela retirada do livro: CONCI, Aura ; AZEVEDO, Eduardo . Computação gráfica: teoria e prática. 2003. 384 cap.1, p.9 ,Rio de Janeiro, 2003. 1

2 Considerações Gerais

A ideia da introdução explanada anteriormente visa introduzir ao leitor a respeito do que é a computação gráfica e os benefícios de sua aplicação prática. Esta dissertação não tem como foco abordar todos os conceitos existentes dentro do mundo da computação gráfica e sim em um contexto específico, que é o estudo de como funciona o processo e quais técnicas são necessárias para a detecção/reconhecimento facial de pessoas e detecção/reconhecimento de objetos.
Sendo isto um dos sub-campos de pesquisa que está inserido na *Computer Vision*[1]

[1]Área de estudo computacional que visa programar um computador para reproduzir a capacidade de "enxergar"como a função biológica da visão dos seres humanos

2.1 Campos relacionados

Diversos campos estão relacionados à visão computacional. O mais notável é a própria visão biológica, no qual a visão computacional é baseado e que, logicamente, é diretamente relacionado. Na verdade, é justo dizer que essas são áreas análogas, uma para a computação, outra, para a biologia. Afinal, enquanto o último estuda os processos psicológicos envolvidos na formação e percepção de imagens pelos seres vivos, o primeiro estuda os processos e algoritmos usados por máquinas para enxergar. Estudos interdisciplinares nas áreas vêm se mostrando bastante proveitosos.

Além desse campo, o campo de inteligência artificial, principalmente as áreas de aprendizado de máquina e de reconhecimento de padrões, é muito empregado para o entendimento da informação gerada a partir da imagem. Outro campo importante é a própria física, que explica como radiação numa certa frequência sensibiliza sensores óticos e como sua trajetória se comporta ao atingir um anteparo. Outros campos também relacionados são o processamento de sinais, processamento de imagens, visão de máquina, entre outros.

2.2 Identificação de bordas

Uma das preocupações no reconhecimento de objetos em imagens é a detecção de bordas. Basicamente, consiste em detectar regiões da imagem nas quais ocorre uma mudança abrupta de brilho na imagem, que, geralmente, representam uma mudança das características do que está sendo visto.

É um problema importante porque mudanças abruptas no brilho da imagem, sob a percepção natural humana da visão, podem representar descontinuidade de profundidade – uma parede atrás de outra, por exemplo, geralmente é mais escura; descontinuidade da orientação da superfície – uma face da parede que está mais perpendicular à iluminação é mais clara que uma face que está paralela; mudanças nas propriedades do material – pedras pretas e brancas no chão são pedras diferentes – e variações na iluminação da cena – a pedra cinza do lado de fora da casa e a pedra preta do lado de dentro são, na verdade, da mesma cor, além de outras características como reflexão e refração, por exemplo.

Em relação a esses aspectos, uma representação por bordas é bem fiel às propriedades físicas do mundo. Além disso, a representação do mundo bidimensional por linhas unidimensionais tem a vantagem de ser compacta, pois leva em conta apenas os detalhes relevantes da imagem.

2.3 Reconhecimento de objeto

O reconhecimento de objetos é a tarefa de reconhecer um objeto predefinido na base de conhecimento ou um aprendido. Existem diversas formas de reconhecer objetos numa cena, mas, geralmente, os métodos empregados usam templates para gerar o conjunto de bordas do objeto requerido e, então, compara suas bordas com as bordas da imagem (métodos baseados em aparência). Outro conjunto de métodos bastante comum busca por semelhanças entre as características do modelo do objeto e as da imagem (métodos baseados em característica).

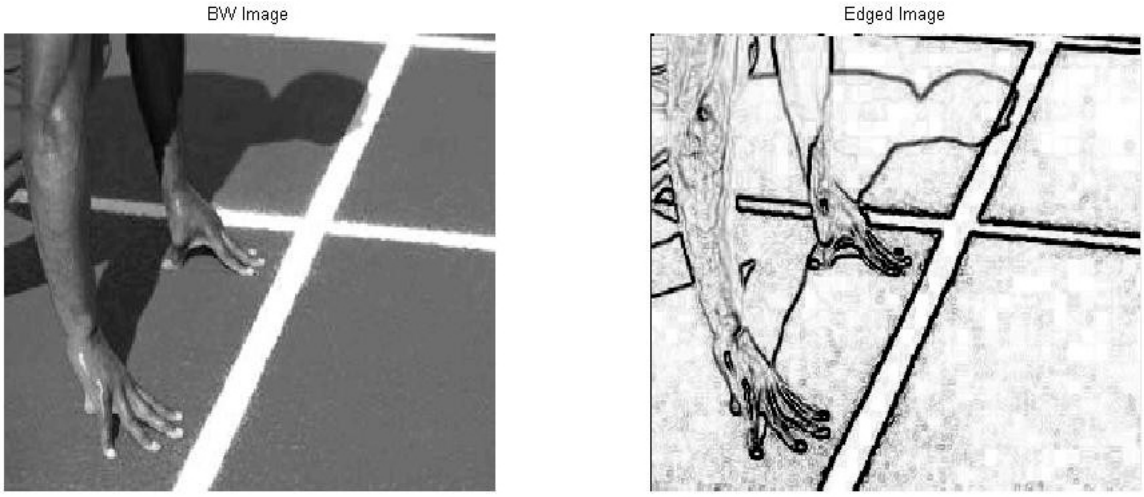

Figura 1: Exemplo de reconhecimento de borda: à esquerda, imagem original; à direita, as bordas da imagem

3 Ambiente de Desenvolvimento

- Pycharm Community Edition 2017.3(IDE para desenvolvimento em Python)
- Anaconda(Gerenciador de pacotes de Data Science e Machine Learning do Python)
- Python versão 3.6.5

3.1 Bibliotecas Utilizadas

- OpenCV
- Dlib
- Tkinter

4 Abordagens

Para o estudo foram utilizadas as técnicas e algoritmos abaixo.

4.1 Técnicas

- Detecção de Pontos Faciais
- Reconhecimento Facial com OpenCV e Dlib
- linhamento de Faces: Técnica utilizada para dispositivos com menos poder de processamento como os celulares.
- Imglab
- Preditores de Forma(definição de pontos de referência, é equivalente aos pontos faciais)

4.2 Algoritmos

- HAAR[2] 5.1
- HOG[3] 5.2
- CNN[4] 5.4
- SVM[5] 5.3
- KNN[6] 5.5

Mais detalhes quanto ao funcionamento dos algoritmos e como eles foram implementados serão dados nas seções posteriores.

5 Introdução aos Algoritmos

Será exposto de uma forma bem simplificada sobre o funcionamento e características de cada algoritmo.

5.1 Haarcascade

Componentes

Haar Cascades: Seleção das características pixel a pixel. Combinação de features[7] haar para formar um classificador.
AdaBoost: Remove as características não necessárias. Algoritmo que agrupa/combina vários classificadores fracos em um classificador forte.(o resultado final é um XML)

Cascade
Desliza pela imagem;
Computa a média dos valores dos pixels na área branca e preta;
Se a diferença entre as áreas é abaixo de um limiar, a característica coincide(match);
Apredizagem Supervisionada(Você já indica quais são as imagens de faces, como se tivesse ajudando ao algoritmo a encontrar padrões).

5.2 Histogram of Oriented Gradiente(HOG)

O HOG faz uso de descritores de caraterísticas[8] + textura(gradiente). Sendo assim podemos dizer que o HOG é um descritor de formas.

Ao analisar uma imagem e identificando a variação da cor de um pixel para seu pixel vizinho(gradiente) ele usa matrizes(Gradient Magnitude e Gradient Direction) para gerar

[2] *Haarcascade*
[3] do inglês,*Histogram Oriented Gradient*:Histograma de Gradientes Orientados
[4] do ingles, *Convolnused Neural Network*: Redes Neurais Convolucionais
[5] do inglês,*Support Vector Machine*:Máquinas de Vetores de Suporte
[6] do ingles, *K Nearest Neighbor*: K Vizinho mais próximo
[7] do ingles, características
[8] Forma que o computador usa para selecionar as partes principais de uma imagem.

um histograma.
Exemplos de descritores: Cor, textura, haar

1º Passo - Detecção da Forma(contorno)

- **Derivada:** É o cálculo matemático que permite medir a taxa de variação da imagem.
- **Vetor Gradiente :** Direção e sentido que os valores da variação aumentam.

2º Passo - Detecção de Textura
Com o resultado da etapa anterior temos como resultado o histograma(histogram of gradients) da imagem.

5.3 Support Vector Machine(SVM)

Em geral supera outros algoritmos de aprendizagem de máquina;
Muito utilizado em tarefas complexas: Reconhecimento de caracteres, voz, imagens;
Considerado por vários anos como o algoritmo mais eficiente;
Aprende hiperplanos de separação com margem máxima.

Em geral o SVM não consegue ser superior as Redeis Neurais quandos se utiliza técnicas de deep learning.

5.4 Convonulsed Neural Network(CNN)

Usado para visão computacional
É muito utilizado em carros autônomos, detecção de pedestres(umas das razões por deep learning funcionar bem);
Em geral, melhor do que SVM;
Redes Neurais Densas(uma rede neural tradicional)
Um neurônio está conectados com todos;
Redes Neurais Convolucionais
Não usa todas as entradas(pixels);
Usa uma rede neural tradicional, mas no começo transforma os dados na cama de entrada;
Vai escolher as características mais importantes e vai alimentar a entrada somente com essas informações/características mais importantes;
Usando as chamadas **funções de kernel** ou **convoluções**, vai ser descoberto as características mais importantes conseguindo diferenciar até uma face humana com a de um animal.

5.5 K Nearest Neighbor(KNN)

É um algoritmo de *machine learning*[9] que utiliza da distância euclidiana para classificar a agrupar elementos em um plano.
Abaixo segue a fórmula matemática que representa a Distância Euclidiana(distância métrica entre dois pontos).

$$DE_{\mathrm{xy}} = \sqrt{\sum_{i}^{p}(x_i - y_i)^2} \tag{1}$$

6 Etapas para o Reconhecimento Facial

Abaixo será introduzido as etapas que foram seguidas e explicações de como as técnicas e abordagens foram utilizadas para se chegar aos resultados esperados.

6.1 Detecção de Faces

Para o reconhecimento facial existem determinados processos que precisam ser seguidos cronologicamente por assim dizer. Para alcançar o resultado em que um dado sistema saiba que João é o João e não a Maria por exemplo.
E isso começa com a detecção de faces. Isto é, para que eu consiga identificar uma pessoa, primeiramente devemos detectar as faces em meio a uma imagem/video e diferenciar uma face de qualquer outra coisa(objetos) que possam estar presentes na imagem/video.

6.2 Detecção de Pontos de Referência Facial

Feito a detecção das faces, fazendo uso de algum algoritmo como o caso do Harcascade é realizada a extração dos pontos de uma face.
Como recurso foi utilizado a abordagem do HOG5.2 + SVM5.3.

6.3 Treinamento do Algoritmo

Nesta etapa devemos realizar o treinamento do algoritmo usando uma espécie de *database* de imagens do indivíduo ou vários indivíduos que queremos identificar. Aqui foi feito uso da CNN5.4 para a realização do treinamento.

7 Etapas para o Reconhecimento Objetos

7.1 Imglab

Realizar a instalação do Imglab na máquina.

[9]do inglês, Aprendizagem de Máquina

7.2 Gerar XML

Gerar XML com a área delimitada com o imglab. O XML é o arquivo de treinamento dos algoritmos(HOG 5.2 + SVM 5.3).

7.3 Preditores de Forma

Nessa etapa é onde é efetivamente realizado o treinamento. Usando o Imglab ele irá criar os preditores de forma com as imagens de treinamento disponibilizadas.

8 Codificação

Durante o estudo de detecção de objetos com python, criamos uma aplicação desktop (um protótipo de software) para executar os scripts utilizando a biblioteca nativa do Python Tkinter.

Esta seção contém todo o código produzido e utilizado para a pesquisa.
Todo o código e dependências necessárias estão disponiveis no repositório do Github:
https://github.com/gbdsantos/recognition-faces-and-objects

Figura 2: Imagem da interface gráfica criada em Python

A Aplicação dispões de 3 funções principais, sendo cada uma atribuída a um botão como demonstra a imagem acima. Suas funções tem como objetivo detectar respectivamente veículos, pessoas e faces utilizando a técnica HaarCascade 5.1.

Utilizamos um set de 5 imagens diferentes para demonstrar cada função, tanto em seus momentos de acerto como em momentos de falha pois, é relativamente fácil selecionar um pequeno dataset ou criar para um determinado grupo de imagens uma alta margem de acerto através dos treinos de reconhecimento. Todavia, é com proposito didático que expusemos as falhas nestas funções.

8.1 App.py que contém a interface gráfica

```
1 import tkinter as tk
  import cv2 as cv
3 import os

5 window = tk.Tk()
```

```
window.title('Projeto Modular')
window.geometry('550x250')
window.configure(background='#e1e1e1')

def car_detect():
    cascade_src = 'haarcascades/car.xml'
    car_cascade = cv.CascadeClassifier(cascade_src)
    for f in os.listdir('C:\\Users\\Gavic\\Desktop\\Projeto Modular\\img\\carro_jpg'):
        img = cv.imread('C:\\Users\\Gavic\\Desktop\\Projeto Modular\\img\\carro_jpg\\'+str(f))
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        cars = car_cascade.detectMultiScale(gray, 1.1, 1)
        for (x,y,w,h) in cars:
            cv.rectangle(img,(x,y),(x+w,y+h),(0,0,255),)
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = img[y:y+h, x:x+w]
        cv.imshow('img',img)
        cv.waitKey(0)
        cv.destroyAllWindows()

def people_detect():
    cascade_src = 'haarcascades/haarcascade_fullbody.xml'
    full_body_cascade = cv.CascadeClassifier(cascade_src)
    for f in os.listdir('C:\\Users\\Gavic\\Desktop\\Projeto Modular\\img\\pedestre_jpg'):
        img = cv.imread('C:\\Users\\Gavic\\Desktop\\Projeto Modular\\img\\pedestre_jpg\\'+str(f))
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        body = full_body_cascade.detectMultiScale(gray, 1.1, 1)
        for (x,y,w,h) in body:
            cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = img[y:y+h, x:x+w]
        cv.imshow('img',img)
        cv.waitKey(0)
        cv.destroyAllWindows()

def face_detect():
    #cascade_src = 'haarcascades/haarcascade_frontalface_default.xml'
    #cascade_src = 'haarcascades/haarcascade_frontalface_alt.xml'
    cascade_src = 'haarcascades/haarcascade_frontalface_alt_tree.xml'
    full_body_cascade = cv.CascadeClassifier(cascade_src)
    for f in os.listdir('C:\\Users\\Gavic\\Desktop\\Projeto Modular\\img\\rosto_jpg'):
        img = cv.imread('C:\\Users\\Gavic\\Desktop\\Projeto Modular\\img\\rosto_jpg\\'+str(f))
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        body = full_body_cascade.detectMultiScale(gray, 1.1, 1)
        for (x,y,w,h) in body:
            cv.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = img[y:y+h, x:x+w]
        cv.imshow('img',img)
        cv.waitKey(0)
        cv.destroyAllWindows()
```

```
w = tk.Label(window, text="Detectar", background = '#e1e1e1',fg='#173753',
    font=("Helvetica", 18)).pack(pady = 20)
btn1 = tk.Button(master=window, text="Veiculos", command = car_detect,
    width = 50, fg='#000000', background = '#6daedb' ).pack()
btn2 = tk.Button(master=window, text="Pessoas", command = people_detect,
    width = 50,fg='#000000', background = '#1d70a2' ).pack(pady = 20)
btn3 = tk.Button(master=window, text="Rostos", command = face_detect, width
     = 50,fg='#e1e1e1', background = '#173753' ).pack()

window.mainloop()
```

8.2 Detecção Facial usando o Haarcascade

```
import cv2 #Biblioteca OpenCV

image = cv2.imread("photos/group.0.jpg")

classificator = cv2.CascadeClassifier('assets/
    haarcascade_frontalface_default.xml')
imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faceDetected = classificator.detectMultiScale(imageGray, scaleFactor=1.2,
    minSize=(50,50))
print(faceDetected)
print("Faces Detected: ", len(faceDetected))
for (x, y, l, a) in faceDetected:
    cv2.rectangle(image,(x, y), (x + l, y + a), (0, 255, 0), 2)

cv2.imshow("Detector Haar", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.3 Detecção Facial usando o HOG

```
import cv2
import dlib

image = cv2.imread("photos/group.0.jpg")
detector = dlib.get_frontal_face_detector()
faceDetected = detector(image, 1)

print(faceDetected)
print("Faces Detected: ", len(faceDetected))

for face in faceDetected:
    #print(face)
    #print(face.left())
    #print(face.top())
    #print(face.right())
    #print(face.bottom())
    l, t, r, b = (int(face.left()), int(face.top()), int(face.right()), int
    (face.bottom()))
```

```
    cv2.rectangle(image, (l, t), (r, b), (0, 255, 0), 2)

cv2.imshow("Detector HOG", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.4 Detecção usando o HOG com pontuação de classificação

```
import cv2
import dlib

subdetector = ["Looking Forward", "Left view", "Right view",
        "The front turning the left", "The front turning right"]

image = cv2.imread("photos/group.0.jpg")
detector = dlib.get_frontal_face_detector()

faceDetected, score, idx = detector.run(image)

print(faceDetected)
print(score)
print(idx)

for i, face in enumerate(faceDetected):
    print(i)
    print(face)
    print("Detection: {}, score: {:.4f}, Sub-detector: {}".format(i, score[
    i], subdetector[int(idx[i])]))
    l, t, r, b = (int(face.left()), int(face.top()), int(face.right()), int
    (face.bottom()))
    cv2.rectangle(image, (l, t), (r, b), (0, 0, 255), 2)

cv2.imshow("Detector HOG", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.5 Detecção Facial usando CNN

```
import cv2
import dlib

image = cv2.imread("photos/group.0.jpg")
detector = dlib.cnn_face_detection_model_v1("assets/
    mmod_human_face_detector.dat")
facesDetected = detector(image, 2)
print(facesDetected)
print("Faces detected: ", len(facesDetected))
for face in facesDetected:
    l, t, r, b, c = (int(face.rect.left()), int(face.rect.top()), int(face.
    rect.right()), int(face.rect.bottom()), face.confidence)
print(c)
```

```
cv2.rectangle(image, (l, t), (r, b), (255, 255, 0), 2)

cv2.imshow("Detector CNN", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.6 Comparação entre o Haarcascade e CNN

```
import cv2
import dlib

#image = cv2.imread("photos/group.0.jpg")
#image = cv2.imread("photos/group.1.jpg")
#image = cv2.imread("photos/group.2.jpg")
#image = cv2.imread("photos/group.3.jpg")
#image = cv2.imread("photos/group.4.jpg")
#image = cv2.imread("photos/group.5.jpg")
#image = cv2.imread("photos/group.6.jpg")
image = cv2.imread("photos/group.7.jpg")

detectorHog = dlib.get_frontal_face_detector()
facesDetectedHog, score, idx = detectorHog.run(image, 2)

detectorCNN = dlib.cnn_face_detection_model_v1("assets/
    mmod_human_face_detector.dat")
facesDetectedCNN = detectorCNN(image, 2)

for i, d in enumerate(facesDetectedHog):
    print(score[i])
print("")
for face in facesDetectedCNN:
    print(face.confidence)
```

8.7 Comparação entre o Haarcascade, HOG e CNN

```
import cv2
import dlib

font = cv2.FONT_HERSHEY_COMPLEX_SMALL
#image = cv2.imread("photos/group.0.jpg")
#image = cv2.imread("photos/group.1.jpg")
#image = cv2.imread("photos/group.2.jpg")
#image = cv2.imread("photos/group.3.jpg")
#image = cv2.imread("photos/group.4.jpg")
#image = cv2.imread("photos/group.5.jpg")
image = cv2.imread("fotos/grupo.7.jpg")

# Haar
detectorHaar = cv2.CascadeClassifier("assets/
    haarcascade_frontalface_default.xml")
imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
facesDetectedHaar = detectorHaar.detectMultiScale(imageGray, scaleFactor
    =1.1, minSize=(10,10))

# Hog
detectorHog = dlib.get_frontal_face_detector()
facesDetectedHog = detectorHog(image, 2)

# CNN
detectorCNN = dlib.cnn_face_detection_model_v1("assets/
    mmod_human_face_detector.dat")
facesDetectedCNN = detectorCNN(image, 2)

for (x, y, l, a) in facesDetectedHaar:
    cv2.rectangle(image, (x, y), (x + l, y + a), (0, 255, 0), 2)
    cv2.putText(image, "Haar", (x, y - 5), font, 0.5, (0, 255, 0))

for face in facesDetectedHog:
    l, t, r, b = (int(face.left()), int(face.top()), int(face.right()), int
    (face.bottom()))
    cv2.rectangle(image, (l, t), (r, b), (0, 255, 255), 2)
    cv2.putText(image, "Hog", (l, t), font, 0.5, (0, 255, 255))

for face in facesDetectedCNN:
    l, t, r, b, c = (int(face.rect.left()), int(face.rect.top()), int(face.
    rect.right()), int(face.rect.bottom()), face.confidence)
    cv2.rectangle(image, (l, t), (r, b), (255, 255, 0), 2)
    cv2.putText(image, "CNN", (l, t), font, 0.5, (255, 255, 0))

cv2.imshow("Comparison  between detectors", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.8 Alinhamento de Faces

```
import dlib
import cv2
import numpy as np

def imprimePoints(image, facialPoints):
    for p in facialPoints.parts():
        cv2.circle(image, (p.x, p.y), 2, (0, 255, 0), 2)
detectorFace = dlib.get_frontal_face_detector()
detectorPoints = dlib.shape_predictor("assets/
    shape_predictor_5_face_landmarks.dat")
image = cv2.imread("photos/training/ronald.0.1.jpg")
imageRGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
facesDetected = detectorFace(imageRGB, 0)
pointsFaces = dlib.full_object_detections()
for face in facesDetected:
    points = detectorPoints(imageRGB, face)
    pointsFaces.append(points)
    imprimePoints(image, points)

images = dlib.get_face_chips(imageRGB, pointsFaces)
for img in images:
```

```
    imageBGR = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    cv2.imshow("Imagem original: ", image)
    cv2.waitKey(0)
    cv2.imshow("Imagem alinhada: ", imageBGR)
    cv2.waitKey(0)

#cv2.imshow("5 Pontos: ", image)
#cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.9 Extração de Pontos Faciais

```
import dlib
import cv2
import numpy as np

def imprimePontos(image, facePoints):
    for p in facePoints.parts():
        cv2.circle(image, (p.x, p.y), 2, (0, 255, 0), 2)

def imprimeNumeros(image, facePoints):
    for i, p in enumerate(facePoints.parts()):
        cv2.putText(image, str(i), (p.x, p.y), font, .55, (0, 0, 255), 1)

def imprimeLinhas(image, facePoints):
    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)):
        points = []
        for i in range(p68[k][0], p68[k][1] + 1):
            point = [facePoints.part(i).x, facePoints.part(i).y]
            points.append(point)
        point = np.array(points, dtype=np.int32)
        cv2.polylines(image, [point], p68[k][2], (255, 0, 0), 2)

font = cv2.FONT_HERSHEY_COMPLEX_SMALL
#image = cv2.imread("photos/training/ronald.0.1.jpg")
#image = cv2.imread("photos/group.0.jpg")
#image = cv2.imread("photos/group.1.jpg")
#image = cv2.imread("photos/group.2.jpg")
#image = cv2.imread("photos/group.3.jpg")
#image = cv2.imread("photos/group.4.jpg")
image = cv2.imread("photos/group.5.jpg")
#image = cv2.imread("fotos/grupo.6.jpg")
#image = cv2.imread("fotos/grupo.7.jpg")

detectorFace = dlib.get_frontal_face_detector()
```

```
detectorPoints = dlib.shape_predictor("assets/
    shape_predictor_68_face_landmarks.dat")
facesDetected = detectorFace(image, 2)
for face in facesDetected:
    points = detectorPoints(image, face)
    print(points.parts())
    print(len(points.parts()))
    #imprimePontos(image, points)
    #imprimeNumeros(image, points)
    imprimeLinhas(image, points)

cv2.imshow("Pontos faciais", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.10 Treinamento do Algoritmo para Reconhecimento Facial

```
import os
import glob
import _pickle as cPickle
import dlib
import cv2
import numpy as np

detectorFace = dlib.get_frontal_face_detector()
detectorPoints = dlib.shape_predictor("assets/
    shape_predictor_68_face_landmarks.dat")
recognitionFace = dlib.face_recognition_model_v1("assets/
    dlib_face_recognition_resnet_model_v1.dat")

indice = {}
idx = 0
descriptionFace = None

for arquive in glob.glob(os.path.join("photos/ronald/training","*.jpg")):
    image = cv2.imread(arquive)
    facesDetected = detectorFace(image, 1)
    numberFacesDetected = len(facesDetected)
    #print(numberFacesDetected)

    if numberFacesDetected > 1:
        print("There's more than one face in the image {}".format(arquive))
        exit()
    elif numberFacesDetected < 1:
        print("No face detected in archives {}".format(arquive))
        exit()

    for face in facesDetected:
        pointsFacial = detectorPoints(image, face)
        descritorFacial = recognitionFace.compute_face_descriptor(image,
    pointsFacial)
        #print(format(arquive))
        #print(len(descritorFacial))
        #print(descritorFacial)

```

```
        listDescritorFacial = [df for df in descritorFacial]
        #print(listDescritorFacial)

        npArrayDescritorFacial = np.asarray(listDescritorFacial, dtype=np.float64)
        #print(npArrayDescritorFacial)

        npArrayDescritorFacial = npArrayDescritorFacial[np.newaxis, :]
        #print(npArrayDescritorFacial)

        if descriptionFace is None:
            descriptionFace = npArrayDescritorFacial
        else:
            descriptionFace = np.concatenate((descriptionFace, npArrayDescritorFacial), axis=0)

        indice[idx] = arquive
        idx += 1

    #cv2.imshow("Training", image)
    #cv2.waitKey(0)

        print("Size: {} Format: {}".format(len(descriptionFace), descriptionFace.shape))
    #print(descritorFacial)
    #print(indice)

np.save("assets/descritors_rn.npy", descriptionFace)
with open("assets/indices_rn.pickle", 'wb') as f:
    cPickle.dump(indice, f)

#cv2.destroyAllWindows()
```

8.11 Implementação e Teste do Reconhecimento Facial com CNN + KNN

```
import os
import glob
import _pickle as cPickle
import dlib
import cv2
import numpy as np

detectorFace = dlib.get_frontal_face_detector()
detectorPoints = dlib.shape_predictor("assets/shape_predictor_68_face_landmarks.dat")
recognitionFacial = dlib.face_recognition_model_v1("assets/dlib_face_recognition_resnet_model_v1.dat")
indices = np.load("assets/indices_rn.pickle")
descritorsFaciais = np.load("assets/descritors_rn.npy")
limiar = 0.5

for arquive in glob.glob(os.path.join("photos", "*.jpg")):
    image = cv2.imread(arquive)
```

```
    facesDetected = detectorFace(image, 2)
    for face in facesDetected:
        l, t, r, b = (int(face.left()), int(face.top()), int(face.right()),
    int(face.bottom()))
        pointsFacial = detectorPoints(image, face)
        descritorFacial = recognitionFacial.compute_face_descriptor(image,
    pointsFacial)
        listDescritorFacial = [fd for fd in descritorFacial]
        npArrayDescritorFacial = np.asarray(listDescritorFacial, dtype=np.
    float64)
        npArrayDescritorFacial = npArrayDescritorFacial[np.newaxis, :]

        distances = np.linalg.norm(npArrayDescritorFacial -
    descritorsFaciais, axis=1)
        print("Distancias: {}".format(distances))
        minimum = np.argmin(distances)
        print(minimum)
        distanceMinimum = distances[minimum]
        print(distanceMinimum)

        if distanceMinimum <= limiar:
            name = os.path.split(indices[minimum])[1].split(".")[0]
        else:
            name = ' '

        cv2.rectangle(image, (l, t), (r, b), (0, 255, 255), 2)
        text = "{} {:.4f}".format(name, distanceMinimum)
        cv2.putText(image, text, (r, t), cv2.FONT_HERSHEY_COMPLEX_SMALL,
    0.5, (0, 255, 255))

    cv2.imshow("Detector hog", image)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

8.12 Treinamento do Algoritmo para Detecção de Objetos

```
import dlib

options = dlib.simple_object_detector_training_options()
options.add_left_right_image_flips = True
options.C = 5

dlib.train_simple_object_detector("assets/treinamento_relogios.xml", "
    assets/detector_relogios.svm", options)
```

8.13 Treinamento do Algoritmo para Predição de Forma

```
import dlib

options = dlib.shape_predictor_training_options()
```

```
dlib.train_shape_predictor("assets/treinamento_relogios_pontos.xml", "
    assets/detector_relogios_pontos.dat", options)
```

8.14 Implementação e Teste do Reconhecimento de Objeto

```
import os
import dlib
import cv2
import glob

print(dlib.test_simple_object_detector("assets/teste_relogios.xml", "assets
    /detector_relogios.svm"))

detectorClock = dlib.simple_object_detector("assets/detector_relogios.svm")
for image in glob.glob(os.path.join("relogios_teste", "*.jpg")):
    img = cv2.imread(image)
    objectDetected = detectorClock(img, 2)
    for d in objectDetected:
        l, t, r, b = (int(d.left()), int(d.top()), int(d.right()), int(d.
    bottom()))
        cv2.rectangle(img, (l, t), (r, b), (0, 0, 255), 2)
    cv2.imshow("Detector of Clocks: ", img)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

8.15 Exibicação dos Pontos encontrados pelo algoritmo já treinado

```
import dlib
import cv2
import glob
import os

detectorClock = dlib.simple_object_detector("assets/detector_relogios.svm")
clockPointsDetector = dlib.shape_predictor("assets/detector_relogios_pontos
    .dat")

print(dlib.test_shape_predictor("assets/teste_relogios_pontos.xml", "assets
    /detector_relogios_pontos.dat"))

def imprimirPoints(image, points):
    for p in points.parts():
        cv2.circle(image, (p.x, p.y), 2, (0, 255, 0))

for arquive in glob.glob(os.path.join("relogios_teste", "*.jpg")):
    image = cv2.imread(arquive)
    objectDetected = detectorClock(image, 2)
    for clock in objectDetected:
        l, t, r, b = (int(clock.left()), int(clock.top()), int(clock.right
    ()), int(clock.bottom()))
```

```
            points = clockPointsDetector(image, clock)
            imprimirPoints(image, points)
            cv2.rectangle(image, (l, t), (r, b), (0, 0, 255), 2)

        cv2.imshow("Points Detector :", image)
        cv2.waitKey(0)

cv2.destroyAllWindows()
```

8.16 Reconhecimento de Logotipo

```
import dlib
import glob
import cv2
import os

options = dlib.simple_object_detector_training_options()
options.add_left_right_image_flips = True
options.C = 5

#dlib.train_simple_object_detector("assets/treinamento_delirium.xml", "
    assets/detector_delirium.svm", options)

detector = dlib.simple_object_detector("assets/detector_delirium.svm")
for image in glob.glob(os.path.join("delirium", "*.jpg")):
    img = cv2.imread(image)
    objectDetected = detector(img, 2)
    for d in objectDetected:
        l, t, r, b = (int(d.left()), int(d.top()), int(d.right()), int(d.
    bottom()))
        cv2.rectangle(img, (l, t), (r, b), (0, 0, 255), 2)
    cv2.imshow("Detector of Logos: ", img)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

8.17 Reconhecimento de Logotipo a partir da webcam

```
import sys
import dlib
import cv2

skipCanva = 30
catch = cv2.VideoCapture(0)
countCanva = 0
detector = dlib.simple_object_detector("assets/detector_delirium.svm")

while catch.isOpened():
    conected, frame = catch.read()
    countCanva += 1
    if countCanva % skipCanva == 0:
```

```
        detectedObjects = detector(frame, 1)
        for o in detectedObjects:
            e, t, d, f = (int(o.left()), int(o.top()), int(o.right()), int(
    o.bottom()))
            cv2.rectangle(frame, (e, t), (d, f), (0, 0, 255), 2)
        cv2.imshow("Objects Predictor: ", frame)

        if cv2.waitKey(1) & 0xFF == 27:
            break

catch.release()
cv2.destroyAllWindows()
sys.exit(0)
```

9 Referências

OpenCV
Acessado em 17 Maio às 14:35

Dlib
Acessado em 17 Maio às 14:35

Histogram of Oriented Gradients | Learn OpenCV
Acessado em 17 Maio às 14:35

Curso Udemy Reconhecimento de Faces e de Objetos com Python e Dlib
Acessado em 19 Maio às 21:25

SCS
Acessado em 20 Maio às 13:50

pyumagesearh.com
Acessado em 21 Maio às 16:10

Computação Gráfica - Teoria e Prática - Eduardo Azevedo e Aura Conci.CONCI, Aura; AZEVEDO, Eduardo . Computação gráfica: teoria e prática. 2003. 384 p. ,Rio de Janeiro, 2003. 1.