

**FACULDADE ANHANGUERA DE SÃO JOSÉ DOS CAMPOS**

**6º CIÊNCIA DA COMPUTAÇÃO**

**INTELIGÊNCIA ARTIFICIAL E COMPUTACIONAL**

**PROF. NEWTON S. JUAN**

Reconhecimento facial e de objetos com OpenCV

Breno Renan da Cunha **RA:231910411919**

Gustavo da Silva Gouvêa **RA:227486011919**

|  |  |
| --- | --- |
| **Proposta de projeto de IAC** | |
| **Proposta:** | Localizar e destacar faces e objetos em imagens ou vídeos. |

Criação de um programa para detecção de faces, Será utilizada a linguagem python juntamente com os classificadores **haarcascade** da biblioteca OpenCV para localizar a área onde a face se encontra e destacá-la através de uma marcação. Inicialmente encontrar faces genericamente, se possível implementarmos um banco de dados para guardar e identificar pessoas específicas.

Componentes a serem utilizados:

● **OpenCV:** Biblioteca voltada ao desenvolvimento de aplicativos na área de Visão computacional.

● **haarcascade:** Grupo de classificadores disponíveis no Opencv, juntamente com ferramentas para treinar o seu próprio.

● **Numpy:** Trabalha arrays e matrizes multidimensionais.

Funcionamento das haarcascades na detecção:

Uma das grandes habilidades dos seres humanos é a capacidade de rapidamente identificar padrões em imagens. Isso sem dúvida foi crucial para a sobrevivência da humanidade até os dias de hoje. Busca-se desenvolver a mesma habilidade para os computadores através da visão computacional e várias técnicas foram criadas nos últimos anos visando este objetivo. O que há de mais moderno (estado da arte) atualmente são as técnicas de “deep learning” ou em uma tradução livre “aprendizado profundo” que envolvem algoritmos de inteligência artificial e redes neurais para treinar identificadores. Outra técnica bastante utilizada e muito importante são os “**haar-like cascades features**” que traduzindo seria algo como “características em cascata do tipo haar” já que a palavra “haar” não possui tradução já que o nome deriva dos “wavelets Haar” que foram usados no primeiro detector de rosto em tempo real.

Historicamente os algoritmos sempre trabalharam apenas com a intensidades dos pixels da imagem. Contudo, uma publicação de Oren Papageorgio "A general framework for object detection" publicada em 1998 mostrou um recurso alternativo baseado em Haar wavelets *(A Transformada de Haar é um* [*transformada*](https://pt.wikipedia.org/w/index.php?title=Transformada&action=edit&redlink=1) *matemática* [*discreta*](https://pt.wikipedia.org/wiki/Discreta) *usada no* [*processamento*](https://pt.wikipedia.org/wiki/Processamento_de_sinais) *e* [*análise de sinais*](https://pt.wikipedia.org/w/index.php?title=An%C3%A1lise_de_sinais&action=edit&redlink=1)*, na* [*compressão de dados*](https://pt.wikipedia.org/wiki/Compress%C3%A3o_de_dados) *e em outras aplicações de engenharia e ciência da computação tendo como característica um pulso quadrado)* em vez das intensidades de imagem, a principal vantagem da técnica é a baixa necessidade de processamento para realizar a identificação dos objetos, o que se traduz em alta velocidade de detecção. Seus criadores então adaptaram a idéia de usar ondas Haar e desenvolveram as chamadas Haar-like features ou características Haar.

Uma característica Haar-like considera as regiões retangulares adjacentes num local específico (janela de detecção) da imagem, então se processa a intensidades dos pixel em cada região e se calcula a diferença entre estas somas. Esta diferença é então usada para categorizar subseções de uma imagem. Neste trabalho por exemplo utilizamos faces humanas. É uma característica comum que entre todas as faces a região dos olhos é mais escura do que a região das bochechas. Portanto, uma característica Haar comum para a detecção de face é um conjunto de dois retângulos adjacentes que ficam na região dos olhos e acima da região das bochechas. A posição desses retângulos é definida em relação a uma janela de detecção que age como uma caixa delimitadora para o objeto alvo (a face, neste caso).

Na fase de detecção da estrutura de detecção de objetos, uma janela do tamanho do alvo é movida sobre a imagem de entrada, e para cada subseção da imagem é calculada a característica do tipo Haar. Essa diferença é então comparada a um limiar aprendido que separa não-objetos de objetos. Como essa característica Haar é apenas um classificador fraco (sua qualidade de detecção é ligeiramente melhor que a suposição aleatória), um grande número de características semelhantes a Haar são necessárias para descrever um objeto com suficiente precisão. Na estrutura de detecção de objetos Viola-Jones, as características de tipo Haar são, portanto, organizadas em algo chamado cascata de classificadores para formar classificador forte. A principal vantagem de um recurso semelhante ao Haar sobre a maioria dos outros recursos é a velocidade de cálculo. Devido ao uso de imagens integrais, um recurso semelhante a Haar de qualquer tamanho pode ser calculado em tempo constante (aproximadamente 60 instruções de microprocessador para um recurso de 2 retângulos).

Outra característica Haar-like pode ser definida como a diferença da soma de pixels de áreas dentro do retângulo, que pode ser em qualquer posição e escala dentro da imagem original. Esse conjunto de características modificadas é chamado de características de 2 retângulos. Viola e Jones também definiram características de 3 retângulos e características de 4 retângulos. Cada tipo de recurso pode indicar a existência (ou ausência) de certos padrões na imagem, como bordas ou alterações na textura. Por exemplo, um recurso de 2 retângulos pode indicar onde a borda está entre uma região escura e uma região clara. A OpenCV já possui o algoritmo pronto para detecção de Haar-like features, contudo, precisamos dos arquivo XML que é a fonte dos padrões para identificação dos objetos. A OpenCV já oferece arquivos prontos que identificam padrões como faces e olhos.

O código:

Nosso código começa em um menu, cujas opções são:

1. Identificar rostos em webcam/ vídeo.

a. O programa primeiramente abre um feed de vídeo, então ele roda cada frame pelos classificadores e tem como output aquele que obteve melhor resultado, então é desenhado um quadrado ao redor da área detectada e mandada na janela do programa.

2. Identificar rostos em imagens contidas na pasta padrão.

a. O programa lê as imagens em formato .jpg e as carrega em um vetor, então as roda em loop pelos classificadores que tem como saída o que encontrou mais rostos, então os quadrados são desenhados ao redor da área detectada e temos então o output.

3. Opção que mostra o treinamento de cascades:

O programa lê os diretórios de imagens positivas e negativas, e realiza o tratamento:

I- Para positivas, passa para escala de cinza, redimensiona para 50x50px (para agilidade no treinamento). Lembrando que estas imagens devem conter somente o objeto que se quer identificar em si.

II- para as negativas, passa para escala de cinza, redimensiona para 100x100px, pois as positivas serão passadas de acordo com a forma “janela deslizante”.

III- Cria-se os chamados descritores, arquivos com nome, caminho, e informação dos pixels onde foi localizada a imagem solicitada (exceto nas negativas, as quais logicamente, não o contém), os descritores do fundo das imagens negativas e por fim o arquivo de vetores com todas as identificações positivas juntadas.

IV- Em seguida, criar os modelos (samples), os vetores e realizar de fato, o treinamento.

*opencv\_createsamples -img dado5050.jpg -bg bg.txt -info Saida/info.lst -pngoutput Saida -maxxangle 0.5 -maxyangle -0.5 -maxzangle 0.5 -num 1950*

*opencv\_createsamples -info Saida/info.lst -num 1950 -w 20 -h 20 -vec positives.vec*

4. Identificar objetos específicos, de acordo com características

a. O programa foi treinado com base em imagens ‘positivas’, ou seja, que contém o objeto desejado, colocando-as contra imagens ‘negativas’ (que não contêm), assim criamos um classificador melhorado para detecção de uma objeto específico. O processo de detecção é o mesmo, porém utilizamos o novo classificador **haarcascade** gerado neste processo.

import numpy as np

from scipy.stats import mode

import cv2

import os

import csv

import time

import pathlib

class Controle:

*#Pega o caminho das pastas automaticamente*

path = os.path.dirname(os.path.abspath(\_\_file\_\_)) +'\\Entrada'

path1 = os.path.dirname(os.path.abspath(\_\_file\_\_)) +'\\positive'

path2 = os.path.dirname(os.path.abspath(\_\_file\_\_)) +'\\negative'

path3 = os.path.dirname(os.path.abspath(\_\_file\_\_)) +'\\Saida'

os.chdir(path)

def menu(self):

txt = """

Opções:

1-Reconhecimento por câmera;

2-Reconhecimento por foto;

3-Treinamento de cascade;

4-Reconhecimento com o casacade treinado;

5-Sair;

Digite a sua opção:

"""

opcao = int(input(txt))

return opcao

def main(self):

opcoes = {1:self.opcao1, 2:self.opcao2, 3:self.opcao3, 4:self.opcao4}

while True:

opcao = self.menu()

if opcao in opcoes:

opcoes[opcao]()

else:

if opcao == 5:

break

else:

print("Opção inválida")

def opcao1(self):

*#carrega um classificador de um arquivo*

aux = 0

aux1= 1

*#Para cada arquivo na lista faça:*

*#carrega um vídeo*

print("[INFO] Inicializando webcam... ")

cap = cv2.VideoCapture(0)

cap.set(3, 800) # Set largura

cap.set(4, 600) # Set altura

while(not cv2.waitKey(1) & 0xFF == ord('q')):

aux = 0

aux1= 1

*#carrega o frame de vídeo*

frameExiste, frame = cap.read()

*#chegou ao último frame ou houve erro? então sair!*

if(frameExiste == False):

print("Erro na câmera!")

cap.release()

return

*#somente funciona com tons de cinza*

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

*#Faz as classificações*

faces = face\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(50,50))

faces2 = face\_alt\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(50,50))

faces3 = face\_alt2\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(50,50))

faces4 = face\_alt\_tree\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(50,50))

*#Organiza numa as classificações numa lista para loop*

classifiers = [faces, faces2, faces3, faces4]

for (classifier) in classifiers:

test = format(len(classifier))

aux1 = int(test)

print("Faces:",test)

if aux1 > aux:

aux = aux1

print ("Melhor Resultado:",aux)

maior = (classifier)

*#para cada face detectada*

for (x, y, w, h) in maior:

#desenhe um retângulo (imagem, posição inicial, final, cor, espessura)

frame = cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0), 2)

#time.sleep(0.3)

#visualizar o detectado:

cv2.imshow("deteccao", frame)

*#Desliga webcam*

cap.release()

*#Destroi janela*

cv2.destroyAllWindows()

def opcao2(self):

*#=== caminhos ===*

*#path ../Entrada*

*#path1 ../positive*

*#path2 ../negative*

*#path3 ../Saida*

file\_list = []

file\_list = busca\_diretorio(controle.path,".jpg")

if len(file\_list) != 0:

for file in file\_list:

print("[INFO] Buscando as imagens na pasta padrão... ")

aux = 0

aux1= 1

*#Para cada arquivo na lista faça:*

*#Lê a imagem e converte para escala de cinza*

img = cv2.imread(file)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

#Faz as classificações

faces = face\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(30,30))

faces2 = face\_alt\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(30,30))

faces3 = face\_alt2\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(30,30))

faces4 = face\_alt\_tree\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(30,30))

faces5 = face\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(20,20))

faces6 = face\_alt\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(20,20))

faces7 = face\_alt2\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(20,20))

faces8 = face\_alt\_tree\_cascade.detectMultiScale(gray, 1.1, 5, minSize=(20,20))

*#Organiza numa as classificações numa lista para loop*

classifiers = [faces, faces2, faces3, faces4,faces5, faces6, faces7, faces8]

for (classifier) in classifiers:

test = format(len(classifier))

aux1 = int(test)

print("Faces:",test)

if aux1 > aux:

aux = aux1

print ("Melhor Resultado:",aux)

maior = (classifier)

*# Coloca os quadrados nas faces*

for (x,y,w,h) in maior:

cv2.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]

*#Exibe as imagens com retangulos, salva imagem com retangulos se ela não existe, so mostra se ela existe*

os.chdir(controle.path3) *#Diretorio ativo no momento*

existe = pathlib.Path(file)

if existe.exists() is True:

cv2.imshow('img',img)

print("Para a imagem "+file+", foram encontradas {0} faces!".format(len(maior)))

else:

cv2.imshow('img',img)

print("Para a imagem "+file+", foram encontradas {0} faces!".format(len(maior)))

cv2.imwrite(file,img)

cv2.waitKey(0)

cv2.destroyAllWindows()

os.chdir(controle.path)

cv2.destroyAllWindows()

*#controle.saida.close()*

else:

print("[INFO] Pasta Vazia ")

def opcao3(self):

*#=== caminhos ===*

*#path ../Entrada*

*#path1 ../positive*

*#path2 ../negative*

*#path3 ../Saida*

file\_list = []

file\_list\_positive = []

file\_list\_negative = []

file\_list = busca\_diretorio(controle.path,".jpg")

file\_list\_positive = busca\_diretorio(controle.path1,".jpg")

file\_list\_negative = busca\_diretorio(controle.path2,".jpg")

tratamentoIMG(controle.path1,file\_list\_positive,"positive",50)

tratamentoIMG(controle.path2,file\_list\_negative,"negative",100)

criarDescritoresPos()

criarDescritoresNeg()

*#Criar modelo (sample):*

mycmd="opencv\_createsamples -img dado5050.jpg -bg bg.txt -info Saida/info.lst -pngoutput Saida -maxxangle 0.5 -maxyangle -0.5 -maxzangle 0.5 -num 1950"

*#Criar arquivo de vetores das detecções positivas:*

mycmd1 = "opencv\_createsamples -info Saida/info.lst -num 1950 -w 20 -h 20 -vec positives.vec"

*#Treinamento de fato:*

mycmd2 = "opencv\_traincascade -data Saida -vec positives.vec -bg bg.txt -numPos 1800 -numNeg 900 -numStages 10 -w 20 -h 20"

os.system(mycmd)

os.system(mycmd1)

os.system(mycmd2)

def opcao4(self):

file\_list = []

file\_list = busca\_diretorio(controle.path,".jpg")

if len(file\_list) != 0:

for file in file\_list:

print("[INFO] Buscando as imagens na pasta padrão... ")

aux = 0

aux1= 1

*#Para cada arquivo na lista faça:*

*#Lê a imagem e converte para escala de cinza*

img = cv2.imread(file)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

*#Faz as classificaçõe*s

cascade = cascade\_treinado.detectMultiScale(gray, 1.1, 5, minSize=(30,30))

*#Organiza numa as classificações numa lista para loop*

classifiers = [cascade]

for (classifier) in classifiers:

test = format(len(classifies))

aux1 = int(test)

print("Faces:",test)

*# Coloca os quadrados nas faces*

for (x,y,w,h) in classifiers:

cv2.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]

*#Exibe as imagens com retangulos, salva imagem com retangulos se ela não existe, so mostra se ela existe*

os.chdir(controle.path3) *#Diretorio ativo no momento*

existe = pathlib.Path(file)

if existe.exists() is True:

cv2.imshow('img',img)

print("Para a imagem "+file+", foram encontradas {0} faces!".format(len(maior)))

else:

cv2.imshow('img',img)

print("Para a imagem "+file+", foram encontradas {0} faces!".format(len(maior)))

cv2.imwrite(file,img)

cv2.waitKey(0)

cv2.destroyAllWindows()

os.chdir(controle.path)

cv2.destroyAllWindows()

else:

print("[INFO] Pasta Vazia ")

*#=================================================*

*#======== VARIÁVEIS E MÉTODOS GLOBAIS ============*

*#=================================================*

*#Estabelece os classificadores de face*

cascade\_treinado = cv2.CascadeClassifier('../haarcascades/casacade.xml')

face\_cascade = cv2.CascadeClassifier('../haarcascades/haarcascade\_frontalface\_default.xml')

face\_alt\_cascade = cv2.CascadeClassifier('../haarcascades/haarcascade\_frontalface\_alt.xml')

face\_alt2\_cascade =cv2.CascadeClassifier('../haarcascades/haarcascade\_frontalface\_alt2.xml')

face\_alt\_tree\_cascade=cv2.CascadeClassifier('../haarcascades/haarcascade\_frontalface\_alt\_tree.xml')

def tratamentoIMG(path,file\_list,tipo,tamanho):

pic\_num=0

os.chdir(path)

#Tratamento imagens negativas:

if len(file\_list) != 0:

print("[INFO] Buscando na pasta",tipo)

for file in file\_list:

img = cv2.imread(file)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

resized\_img = cv2.resize(gray, (tamanho, tamanho))

cv2.imwrite(tipo+str(pic\_num)+ ".jpg", resized\_img)

#print("Echo!")

pic\_num+=1

else:

print("[INFO] Pasta",tipo,"Vazia ")

*#funcao de busca de arquivos no diretório*

def busca\_diretorio(diretorio,extensao):

file\_list = []

for file in os.listdir(diretorio):

if file.endswith(extensao):

file\_list.append(file)

#print(file)

return file\_list

def criarDescritoresNeg():

print("[INFO] Criando descritores Negativos")

for file\_type in [controle.path2]:

for img in os.listdir(file\_type):

if file\_type == controle.path2:

line = file\_type+'/'+img+'\n'

os.chdir(controle.path)

with open('bg.txt','a') as f:

f.write(line)

def criarDescritoresPos():

print("[INFO] Criando descritores Positivos")

for file\_type in [controle.path1]:

for img in os.listdir(file\_type):

if file\_type == controle.path1:

line = file\_type+'/'+img+' 1 0 0 50 50\n'

os.chdir(controle.path)

with open('info.dat','a') as f:

f.write(line)

controle = Controle()

controle.main()

Referências:

CREPALDE – Neylson – Reconhecimento facial com Python e OpenCV - Machine Learning – 2017 –<http://neylsoncrepalde.github.io/2017-02-21-reconhecimento-facial/>– acesso em:12/11/2019.

SILVA - Regis - Gerenciando banco de dados SQLite3 com Python - Parte 1 - 2014 -<http://pythonclub.com.br/gerenciando-banco-dados-sqlite3-python-parte1.html> - acesso em: 12//11/2019.

CARLOS - Juan Ferreira - Reconhecimento\_facial - 2018 - [https://github.com/juanFCA/Reconhecimento-Facial](https://github.com/juanFCA/Reconhecimento-Facial/blob/testes/ReconhecimentoBasico/03_ReconhecimentoFacialCAM.py) - acesso em: 12/11/2019.

ROSEBROCK - Adrian - OpenCV Face Recognition - 2018 -<https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/> - acesso em: 12/11/2019.

ANTONELLO - Ricardo - Introdução a Visão Computacional com Python e OpenCV - 2017 -<http://professor.luzerna.ifc.edu.br/ricardo-antonello/wp-content/uploads/sites/8/2017/02/Livro-Introdu%C3%A7%C3%A3o-a-Vis%C3%A3o-Computacional-com-Python-e-OpenCV-3.pdf> - acesso em: 12/11/2019.

Python Programming - Creating your own Haar Cascade OpenCV Python Tutorial -

<https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/> - acesso em: 12/11/2019.