INSTITUTO MAUÁ DE TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

14.00556-5 LUCAS DE MOURA RODRIGUES

Laboratório Autônomo – Reconhecimento Facial

Trabalho realizado no Instituto Mauá de Tecnologia sob a orientação de **Marcelo Gomes** da

SÃO CAETANO DO SUL

2018

Introdução e Objetivos

Esse é o projeto final de processamento de imagens que possui o objetivo de reconhecer a face e os olhos de uma foto frontal do rosto de uma pessoa.

A seguir estão as instruções seguidas para o projeto.

- ESTA ATIVIDADE É DIFERENTE DAQUELAS EXECUTADAS AO LONGO DO SEMESTRE. ATENÇÃO ÀS REGRAS:
  - a atividade deverá ser executada sem a interferência do professor;
  - cada equipe deverá entregar a solução ao professor por escrito até às 13h de hoje, contendo a descrição dos passos das soluções, devidamente justificados, os códigos utilizados e os resultados obtidos devidamente comentados, em um documento formato .DOCX. Entregas realizadas fora do prazo serão desconsideradas;
  - a entrega deverá ser feita pelo e-mail marcelo.gomes@maua.br;
  - a avaliação irá se basear nos seguintes critérios:
    - dificuldade da implementação;
    - resolução correta dos problemas propostos;
    - utilização (justificada) de recursos apresentados no curso;
    - originalidade da solução;
    - apresentação da solução.
  - soluções consideradas de **mesma origem** receberão nota zero!
- Esta atividade será avaliada e formará uma das notas do semestre.
- Toda consulta a materiais é permitida.
- Boa atividade!

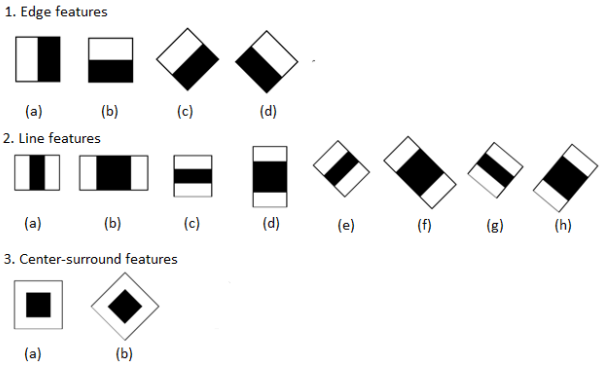
Escolher uma imagem de seu interesse, de preferência uma fotografia. Escolher e aplicar as técnicas desenvolvidas ao longo do curso de modo a ***representar/descrever*** pelo menos dois objetos distintos presentes na imagem. Comparar essa representação/descrição entre os objetos e tirar conclusões.

Tecnologias e Metodologia

Para a realização desse projeto foi utilizada a linguagem Python na versão 3.6.6, a biblioteca OpenCV na versão 3.4.4 para a manipulação da imagem e a biblioteca Matplotlib para visualização da imagem. Todo código desenvolvido foi realizado na IDE Pycharm.

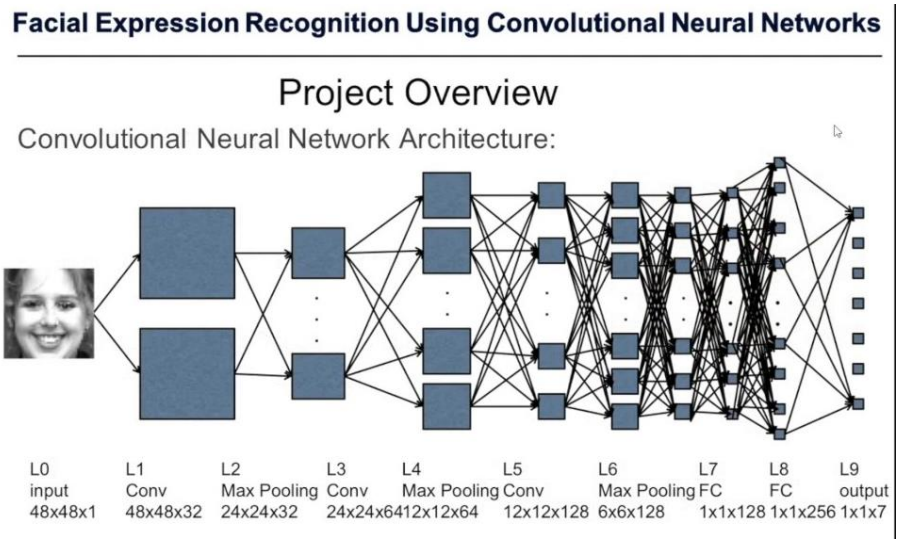
Para obter as fotos foi utilizada uma webcam Logitech C270 HD que gera uma imagem 1280x720x24 BPP do formato JPG.

O algoritmo utilizado para reconhecer as imagens é conhecido como **Haar-Cascade.** Ele utiliza um classificador que a partir de dados de entrada ele gera diversas classificações, para um objeto ser reconhecido, várias condições têm que ser atingidas. A seguir uma imagem com as máscaras disponíveis.



E como pode ser visto na imagem a seguir, essas mascaras são testadas para uma região da imagem que considera a vizinhança de 8 vizinhos a partir de um pixel central.

Vale lembrar que também seria possível utilizar redes neurais para o mesmo fim, no entanto o projeto perderia o foco de processamento de imagens. Além de também ser necessário maior poder de processamento e quantidade de dados.



Desenvolvimento

Foram tiradas duas fotos, uma sem óculos e outra com óculos. Posteriormente, foram movidas para a pasta do projeto do Pycharm.

Com isso, criei um arquivo em Python **main.py** e adicionei as dependências do programa.

```
import cv2
from matplotlib import pyplot as plt
```

Em seguida li a imagem sem óculos (img) e a imagem com óculos (img_oculos).

```
# le a imagem
img = cv2.imread('img.jpg')
img_oculos = cv2.imread('img_oculos.jpg')
```

Logo criei um classificador utilizando os dados da própria biblioteca do OpenCV, um para a face e outro para os olhos.

```
# cria os classificadores
classificador_face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
classificador_olhos = cv2.CascadeClassifier('haarcascade_eye_tree_eyeglasses.xml')
```

Depois criei duas listas (similar à um vetor/array) com as imagens e as suas descrições, para em seguida iterar sobre elas.

```
imagens = [img, img_oculos]
imagens_descricao = ['Foto', 'Foto com óculos']
for (img, img_descricao) in zip(imagens, imagens_descricao):
```

Para realizar o algoritmo de reconhecimento transformei a imagem para que ficasse somente em tons de cinza.

```
img_cinza = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Treina a imagem para reconhecer o rosto.

```
faces = classificador_face.detectMultiScale(img_cinza, 1.3, 5)
```

Testa os classificadores nas regiões de interesse para o reconhecimento do rosto.

```
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
    regiao_interesse_cinza = img_cinza[y:y + h, x:x + w]
    regiao_interesse_colorido = img[y:y + h, x:x + w]
```

E dentro do rosto, é feito o mesmo procedimento para o reconhecimento dos olhos.

```
olhos = classificador_olhos.detectMultiScale(regiao_interesse_cinza)
for (ex, ey, ew, eh) in olhos:
    cv2.rectangle(regiao_interesse_colorido, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)
```

O OpenCV usa o formato de cor BGR (Blue, Green, Red), enquanto o Matplotlib precisa da imagem em RGB, logo foi feito a conversão necessária.

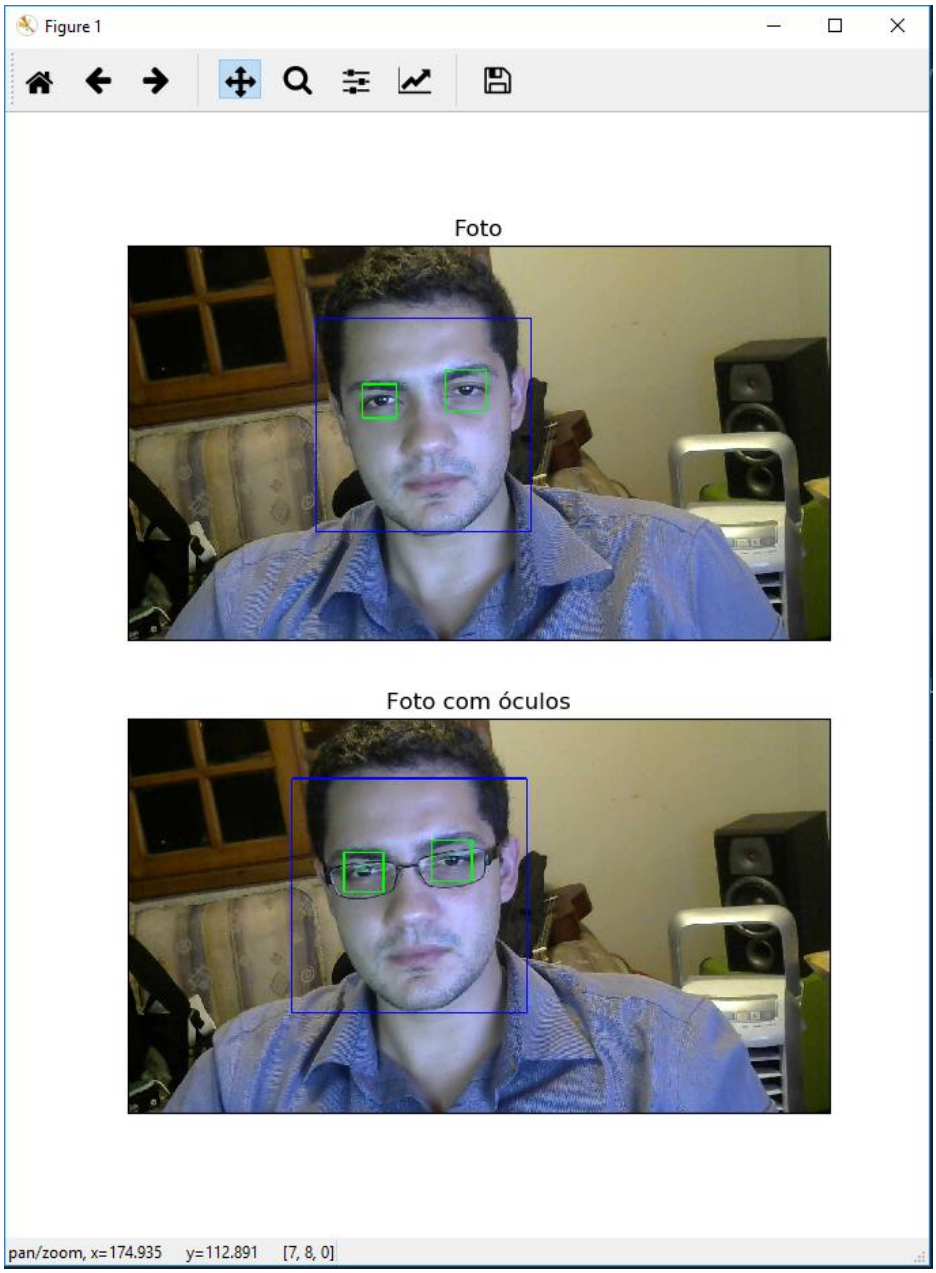
```
RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Por fim, foram exibidas as duas imagens utilizando o Pyplot do Matplotlib. As imagens resultantes podem ser vistas no capítulo a seguir.

```
    plt.subplot(2, 1, i), plt.imshow(RGB_img), plt.title(img_descricao), plt.xticks([]), plt.yticks([])
    i += 1
plt.show()
```

Resultados e Conclusão

Foi possível chegar no objetivo e identificar a face (retângulo azul) e também os dois olhos (retângulos verdes), com e sem óculos. Como pode ser visto na figura abaixo, os retângulos delimitam as regiões identificadas.



Para ver o código completo, arquivos utilizados e outros projetos só acessar o repositório do GitHub que está nas referências.

Referências

https://github.com/lucasrodrigues10/processamento_imagens

https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

https://pt.stackoverflow.com/questions/262437/o-que-s%C3%A3o-os-termos-cascade-e-classifier-em-rela%C3%A7%C3%A3o-a-vis%C3%A3o-computacional

https://memoria.ifrn.edu.br/bitstream/handle/1044/1535/TCC-Diego.pdf?sequence=1&isAllowed=y