Visão Artificial e Realidade Mista

Mestrado em Engenharia Informática e Multimédia

Deteção e Reconhecimento de Faces para Realidade Aumentada

Rúben Santos, 49063

Índice

1 Introdução

O desenvolvimento deste projeto teve como objetivo desenvolver uma aplicação de visão computacional que permitisse fazer a deteção e reconhecimento de faces (caras de pessoas) e que permitisse também incluir elementos virtuais do mundo real.

Como tal utilizou-se a biblioteca OpenCV (Open-Source Computer Vision) para desenvolver a aplicação em tempo real.

No início do desenvolvimento da aplicação, utilizaram-se os classificadores Haar Cascade e um modulo de Deep Neural Network (DNN) da biblioteca OpenCV (Open-Source Computer Vision) em tempo real para fazer a deteção de faces.

A segunda parte do projeto consistiu em explorar o reconhecimento das faces pelos que é necessário criar uma base de dados, normalizar as faces, classificá-las (com os classificadores EigenFaces e FisherFaces) e comprar os resultados dos classificadores de faces.

Na parte final do projeto aplicaram-se, em tempo real objetos da vida real no mundo virtual, ou seja, sempre que uma face seja detetada serão colocados objetos na face.

2 Face detection

No início do desenvolvimento da aplicação, utilizaram-se os classificadores Haar Cascade e um modulo de Deep Neural Network (DNN) da biblioteca OpenCV (Open-Source Computer Vision) em tempo real para fazer a deteção de faces.

2.1 Using Haar Cascade Classifier

Como primeira técnica de deteção facial utilizou-se o classificador **Haar Cascade** que foi baseado num algoritmo de deteção de objetos proposto por Paulo Viola e Michael Jones.

Para ser noção de como o classificador está a funcionar, detetou-se a cara da pessoa e optou-se por desenhar um retângulo na área onde a face foi detetada para se ver visualmente (como podemos ver nas seguintes imagens).

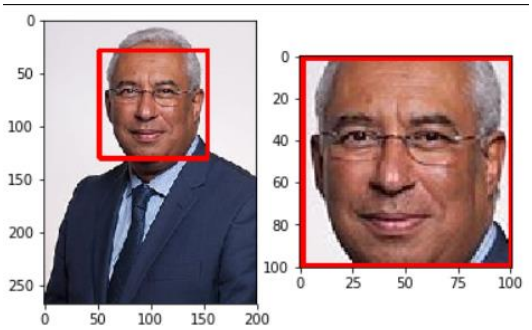


Figura 2- Detecao da face numa foto com Haar Cascade Classifier

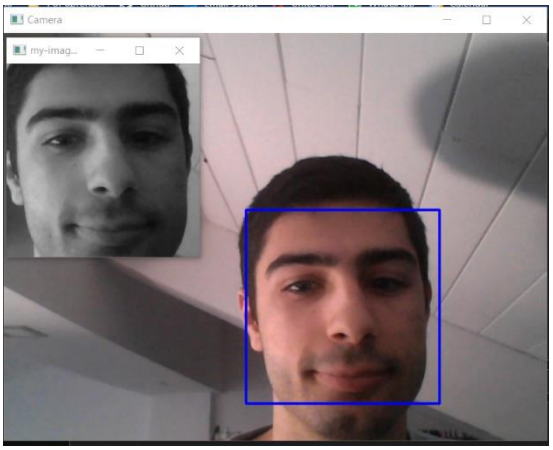


Figura 1- Detenção de faces em tempo real com Haar Cascade Classifier

2.2 Using OpenCV Deep Neural Network module (dnn)

Como a primeira técnica é feita uma deteção facial das pessoas so que com o um modulo pre-treinado de uma Rede Neural Convolucional (CNN). Nesta fase utilizou-se uma biblioteca chamada MTCNN.

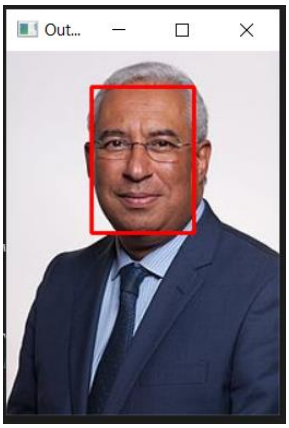


Figura 4- Deteção numa foto com DNN

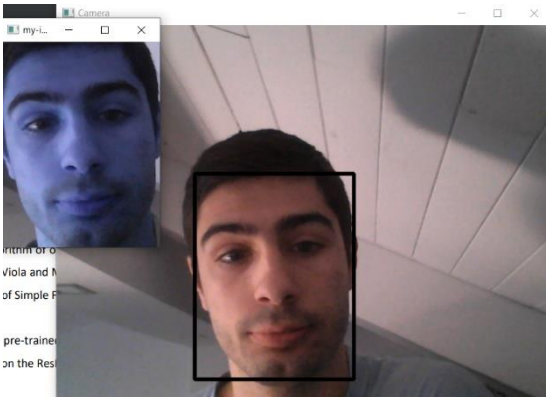


Figura 3- Detenção de faces em tempo real DNN

3 Face Recognition

A segunda parte do projeto consistiu em explorar o reconhecimento das faces pelo que os passos foram os seguintes: 1) criar uma base de dados, 2) normalizar as faces, 3) classificá-las (com os classificadores EigenFaces e FisherFaces) e 4) comprar os resultados dos classificadores de faces.

3.1 Dataset de Faces

Para produzir o processo de reconhecimento de facial é necessário obter algumas imagens para formar um dataset razoável.

PESSOA	NÚMERO DE IMAGENS
ANGELA MERKEL	9
ANGELINA JOLIE	9
ANTONIO COSTA	12
BARAC OBAMA	9
DONALD TRUMP	10
NELSON MANDELA	9
PASSO COELHO	9
PASSO COELHO	9

3.2 Face normalization

Para utilizar os classificadores e ter o melhor desempenho dos mesmos é necessário normalizar o dataset de imagens, pelo motivo de existirem imagens com diferentes dimensões, sem as faces centradas ou com os olhos em diferentes localizações.

Figura 6 - Imagem do dataset com os olhos desnivelados

Figura 5 - Imagem do dataset com a face e olhos desnivelados

Se observarmos algumas imagens do dataset podemos verificar que existem algumas caraterísticas que dificultam os classificadores, pelo que as imagens devem:

1. Ser transformadas em imagens monocromáticas.
2. Ser redimensionadas para uma dimensão de 46 por 56 pixéis.
3. Conter ambos os olhos, esquerdo e direito, perfeitamente alinhados horizontalmente
4. Conter o alinhamento dos olhos (esquerdo e direito) na linha 24, coluna 16 e 31 respetivamente.

Para efetuar este processo todo começou-se por modificar a cor das imagens para cinzento com a funcao **cv2.cvColor().** Apos isto utilizou-se o classificador **Haar Cascade** para fazer o reconhecimento facial e a área dos olhos.

O passo seguinte do processo foi a procura do centro dos olhos, e uma vez que foram esta as coordenadas que revelam o angulo dos olhos com a horizontal, foi possível aplicar a rotação da imagem.

Com os olhos alinhados o último passo foi colocar os olhos na linha 24, coluna 16 e 31, e para isso aplicou-se a translação, rotação e afastamento/aproximação das imagens.

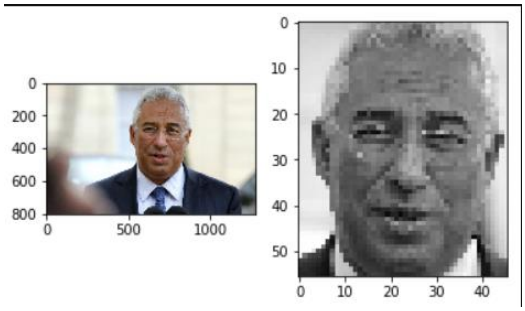


Figura 8 – Antes e depois do processo de normalização.

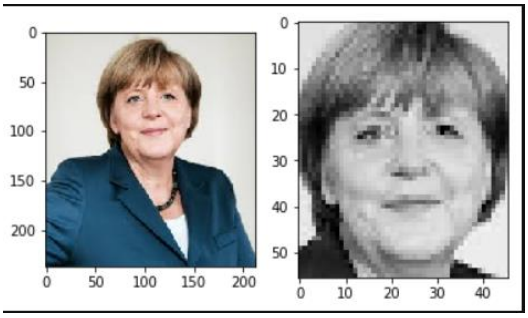


Figura 7 - Antes e depois do processo de normalização.

3.3 EigenFaces

O EigenFaces (algoritmo Principal Compontent Analysis) é uma técnica de redução de dimensionalidade que utiliza Eigenvalues e EigenVectors para reduzir a dimensionalidade e projetar uma amostra/dados de treinamento em um pequeno espaço de características. O algoritmo começar por receber um conjunto de ***m*** imagens de dimensões ***N*N*** (imagens de treino). No primeiro passo convertemos as imagens em vetores de tamanho ***N²***.

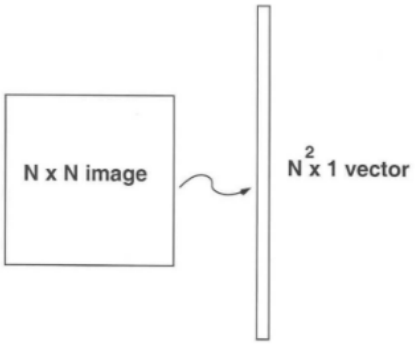


Figura 9 - Conversão das imagens em vetores N²

Com a conversão feita podemos agora calcular a media de todos esses vetores da face e subtrairmos por cada vetor, resultando assim na **mean face**.

$$\varphi = \frac{1}{m}\sum_{i}^{m} x_i$$

$$\mathrm{a}_i = x_i - \psi$$

Figura 10 - Cálculo da mean face

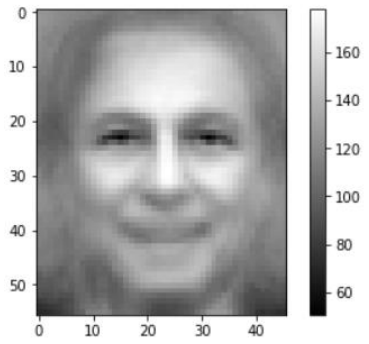


Figura 11 - Mean Face

Agora pegamos todos os vetores de face para obter uma matriz de tamanho ***N² * M.***

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & .... & a_m \end{bmatrix}$$

O próximo passo é encontramos a matriz de covariância multiplicando ***A*** por A^t . A tem dimensões ***N² * M***, portanto **A T** tem dimensões ***M * N²***. Quando multiplicamos isso nos dá uma matriz de ***N² * N***, que nos dá ***N²*** autovetores de tamanho ***N²*** que não é computacionalmente eficiente para calcular. Então, calculamos nossa matriz de covariância multiplicando A^t e ***A***. Isso nos dá a matriz ***M * M*** que tem ***M*** (assumindo que ***M << N²***) autovetores de tamanho ***M***.

$$Cov = A^T A$$

Para calculamos os eigenvalues e eigenvectors da matriz de covariância acima usando a fórmula abaixo.

$$A^T A \nu_i = \lambda_i \nu_i$$

$$A A^T A \nu_i = \lambda_i A \nu_i$$

$$C' u_i = \lambda_i u_i$$

onde,

$C' = AA^T$ e $u_i = A\nu_i$

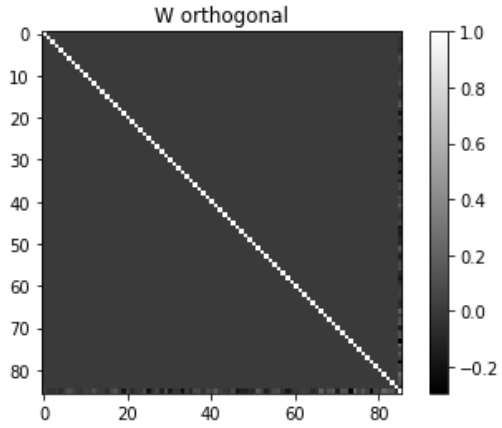
De seguida calculamos o Eigenvectors e Eigenvalues dessa matriz de covariância reduzida e os mapeamos ***C'*** usando a fórmula $u_i = A\nu_i$

Selecionamos os K Eigenvectors de **C'** correspondentes aos K maiores Eigenvectors (onde K < M). Esses Eigenvectors têm tamanho ***N²***.

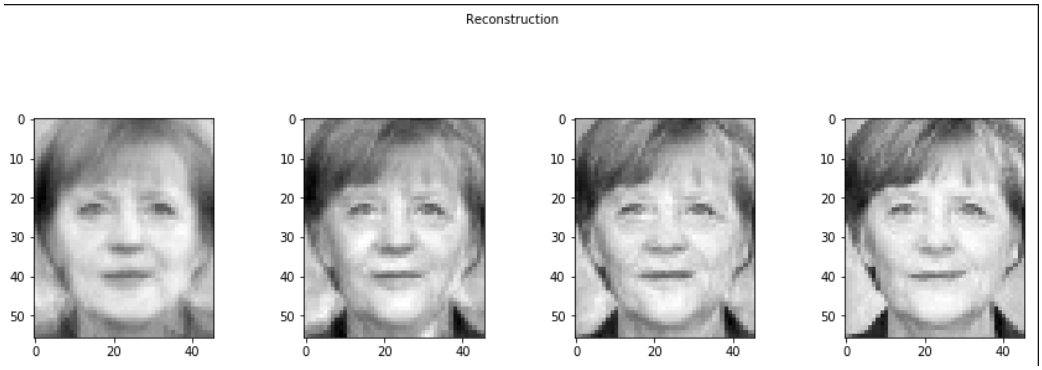
Nesta etapa, usamos os Eigenvectors que obtivemos da etapa anterior e as faces normalizadas (face – Mean face) e representamos cada vetor de face na linear de combinação dos melhores K eigenvectors (como mostrado no diagrama abaixo).

$$\mathrm{x}_i - \psi = \sum_{j=1}^{K} w_j u_j$$

Para determinar se os eigenvetores tem uma base ortogonal, verifica-se a matriz identidade de W e como mostra a figura seguintes podemos ver que formam uma base ortogonal.



As imagens após a redução da dimensionalidade, são projetas num subespaço com menores dimensões, pelo que caso queiramos recuperar temos de proceder a reconstrução de imagens dando origem a face reconstruídas como mostra na seguinte imagem.



3.4 FisherFaces

Ao utilizar o classificador fisherfaces começa-se por utilizar o algoritmo **Principal Compontent Analysis** (PCA) para obter o subespaço intermedio. Com esse resultado procede-se para o algoritmo MDA para obter as direções discriminantes ***c-1***.

O algoritmo MDA começa por receber as faces x1…xn, devidamente alinhadas e classificadas ***c*** (i = 1,..,c), para cada $n_i$elementos e depois determinada a **mean face** e as mean faces de cada classe para determinar a matriz de dispersão $\boldsymbol{S_t}$ ***(N * N).***

De seguida determina-se as matrizes $\boldsymbol{S_b}$***(N * N)*** e $\boldsymbol{S_w}$ ***(N * N)*** calculando-se da seguinte forma:

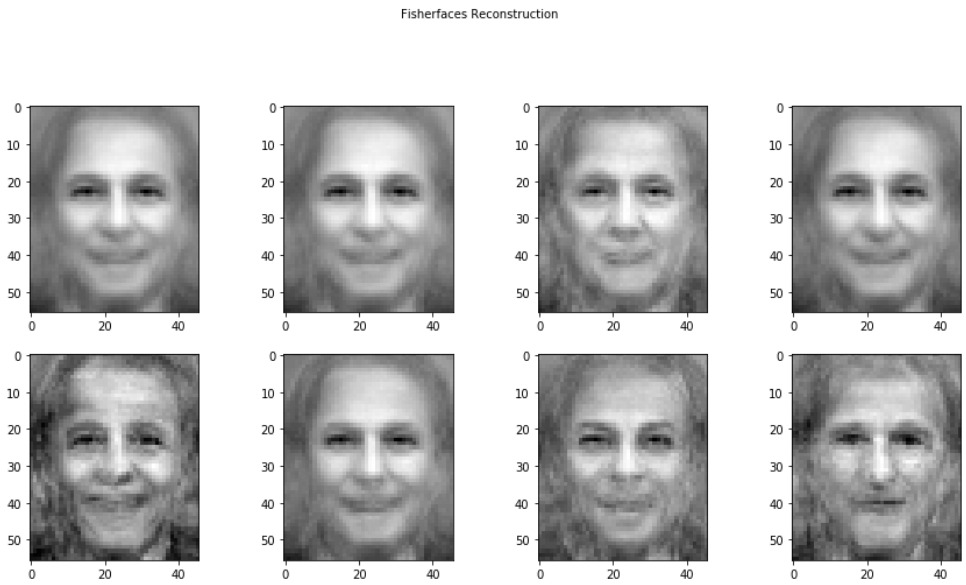
$$\breve{S}_b = W_{pca}^T S_b W_{pca}$$
$$\breve{S}_w = W_{pca}^T S_w W_{pca}$$

Figura 12 - Cálculo das matrizes $\mathbf{S_b}$ *(N * N) e* $\mathbf{S_w}$ *(N * N)*

E por fim são determinados os maiores eigenvectors c-1 da matriz.

$$\breve{S}_w^{-1}\breve{S}_b \longleftarrow m \text{ x } m$$

Apos terminar de classificar, reconstruiu-se as faces dando origem as seguintes imagens:



3.5 Classification

4 Combine real and virtual objects.

No final deste projeto pretendeu-se aplicar objetos ou acessórios virtuais a face presente em tempo real.

Como tal, começou-se por fazer um alinhamento de imagens dos objetos, como foi feito na normalização das faces, detetando e redimensionando faces. Ou seja, antes de utilizar os objetos tem que se fazer uma normalização dos mesmos e identificar face e os olhos para posicionar os objetos, como podemos verificar na seguinte imagem.

Figura 13 - Óculos. Objeto para colocar na face em tempo real

Figura 14 - Chapéu. Objeto para colocar na face em tempo real

Figura 15 - Resultado da face com objetos em tempo real

5 Conclusão

Para concluir, com o recurso da biblioteca OpenCV foi possível neste projeto aprender a trabalhar com a deteção de face com recurso aos classificadores Haar Cascade e Deep Neural Network module (dnn), que demostraram ser de grande performance na deteção de faces e olhos.

Foi possível aprender de forma mais aprofundada com é que os algoritmos de reconhecimento facial, EigenFaces e FisherFaces, funcionam e quais são os paramentos que são possíveis afinar para obter resultados mais realistas.

Por último, foi possível ter precessão de como o mundo das imagens e vídeos funciona ao utilizar a combinação faces do mundo real com objetos do mundo virtual.