Mestrado Engenharia Informática e Multimédia

Visão Artífical e Realidade Mista

Trabalho 1

Data: July 16, 2023

Docente:
Eng. Pedro Jorge

Realizado por:
Mihail Ababii - 46435

Índice de Conteúdos

Lista de Figuras

1 Introdução

O seguinte projeto visa a deteção e o reconhecimento facial para realidade aumentada. Realidade aumentada é a combinação entre objetos digitais com os sentidos humanos em tempo real, que causa o digital a estar anexado ao espaço físico (registo 3D). Realidade aumentada deve conter as seguintes características:

- Combinar o real com o virtual.
- Interação em tempo real.
- Registar em 3D.

Será utilizada a linguagem de programação Python para o desenvolvimento da aplicação que deteta, reconhece e utiliza faces para realidade aumentada. De forma a auxiliar o desenvolvimento, será também utilizada a biblioteca OpenCV, uma biblioteca extremamente poderosa no que diz respeito a processamento de imagens em Python. O produto final, será uma aplicação que deteta faces e, consoante a face detetada adiciona um objeto virtual à mesma. É de notar que serão utilizados os algoritmos Eigenface e Fischerface para processar as faces e, um classificador KNN (K-Nearest-Neighbour), que serão aprofundados posteriormente.

Sendo assim, o projeto segue os seguintes pontos:

- Deteção Facial.
- Normalização da face.
- Algoritmos Eigenface e Fischerface.
- Classificação - KNN.
- Objetos virtuais com a realidade.

2 VARM - Visão artificial e Realidade aumentada

2.1 Deteção Facial

Numa fase inicial do desenvolvimento deste projeto é necessário detetar faces presentes. Esta tarefa pode ser realizada de diferentes formas, no entanto foi-se recomendado a utilização de uma ou mais das seguintes formas: • Haar Cascade Classifier • Deep Neural Networks • Dlib - HOG

2.1.1 Haar Cascade Classifier

Este algoritmo de deteção é baseado em *Haar-features*, que consistem em padrões retangulares de pequena dimensão, que permitem calcular certas características. Estas características podem ser calculados rapidamente, permitindo uma deteção eficiente em aplicativos em tempo real

Para criar um *Haar cascade classifier (HCC)*, é necessário treinar este mesmo classificador, atribuindo-lhe diversas imagens de treino, positivas e negativas, sendo estas imagens que contêm as características desejadas e imagens que não as contêm, respetivamente. Durante o processo de treinamento, é-se usado uma técnica denominada de *AdaBoost* para selecionar os melhores padrões que podem efetivamente distinguir entre amostras positivas e negativas. Em seguida, combina estes padrões numa estrutura em cascata, onde cada nível da cascata consiste em vários classificadores fracos.

Durante a fase de deteção, o classificador aplica diferentes regiões de uma imagem de entrada, pelos vários níveis da cascata, pelo que, se região falhar um dado nível, esta região é descartada e não é mais considerada para futuros níveis. Assim, as regiões que passam por todos os níveis são consideradas deteções das características procuradas. (HCC23)

2.1.2 Deep Neural Networks

Deep Neural Networks (DNNs) tenta emular o comportamento do cérebro humano criando neurónios artificiais, em camadas inter-conectadas, onde cada neurónio recebe valores dos neurónios anteriores, executa uma computação sobre os valores obtidos e produz uma saída que é passada para a próxima camada.

Á exceção da primeira camada, que apenas recebe os dados brutos, como imagens, texto ou áudio, as restantes camadas aplicam transformações matemáticas aos dados de entrada, sendo estas transformação constituídas pelo calculo duma soma ponderada das suas entradas e aplicando uma função de ativação para introduzir não linearidade e produz uma saída. Os pesos que os neurónios têm nos neurónios das camadas seguintes é calculado durante uma fazê de treinamento, por forma a minimizar a diferença entre os resultados previstos e as resultados desejados.

2.1.3 Dlib - Histogram of Oriented Gradients

O Dlib é um conjunto de ferramentas que contém algoritmos e ferramentas de aprendizagem automática, que possui um conjunto de métodos para realizar deteção de faces e pontos de referência faciais, incluindo a estimativa de até 68 pontos de características faciais.

Ao utilizar o HOG, a imagem é dividida em pequenas regiões conectadas chamadas células, e para cada pixel dentro de uma célula, é calculado um histograma das direções de gradiente. O descritor HOG é obtido pela concatenação desses histogramas de todas as células. Esse descritor fornece informações sobre os padrões locais de gradientes, que podem ser úteis para tarefas de deteção e reconhecimento de objetos. (HOG16)

2.2 Normalização

Após obter as faces capturadas pela câmara, é necessário normalizar as mesmas, pois as técnicas de deteção facial presenta valores de largura e comprimento facial variáveis. Assim sendo, para realizar a normalização das caras será necessário realizar transformações como rotação, escalamento e seleção, por forma a obter rostos que estejam em conformidade com o formato definido pela recomendação MPEG-7. Ou seja, cada rosto será representado por uma imagem monocromática (256 níveis de cinza) com 56 linhas e 46 colunas, em que ambos os olhos, direito e esquerdo, estejam perfeitamente alinhados horizontalmente e localizados na linha 24, nas colunas 16 e 31, respetivamente.

2.3 Eigenfaces

Eigenfaces é uma técnica de reconhecimento facial baseada em análise de componentes principais (PCA) que é usada para representar e comparar faces presentes num espaço de alta dimensão. O conceito apresentado é que as faces humanas têm características únicas que podem ser expressas em um conjunto limitado de vetores de características chamado Eigenfaces.

Para obter as Eigenfaces, é necessário um conjunto de imagens de faces de treino, para serem utilizados em durante a aplicação da técnica PCA. Ao utilizar a técnica de PCA estamos a obter os principais direções da variação das características faciais, criando assim uma matriz de pesos correspondentes os eiganvectors com maior valor.

Após calcular as Eigenfaces, é possível usar as mesmas para reconhecer novas faces. Este processo consistem em projetar a nova face nas Eigenfaces, sendo que esta projeção retorna os coeficientes que representam a semelhança entre as Eigenfaces e a nova face. Ao obter estes coeficiente podemos comparar os mesmos aos coeficientes previamente adquiridos para as faces de treino, para associar um dada pessoa á nova face.

2.4 Fisherfaces

A técnica de reconhecimento facial Fisherfaces, é bastante similar ao Eigenfaces, pois é uma extensão do mesmo. A diferença entre ambas as técnicas consiste no facto que Eigenfaces procura obter a estrutura global das faces, enquanto Fisherfaces procura extrair as características, por forma a maximizar a separação entre as classes. Assim, para implementar Fisherfaces é necessário realizar as seguintes operações:

- Reduzir dimensão das imagens, utilizando a técnica de PCA, mantendo as principais caraacteristicas faciais
- Fisher Linear Discriminant (FLD): FLD procur encontrar a projeção na qual a proporçãao entre a distancia entre classes e a distancia intra classe seja maximizada
- Implementar uma classificação

2.4.1 Fisher Linear Discriminant

Contrariamente aos restantes passos, este é o único que se diferencia relativamente ao Eigenfaces, assim sendo, requer uma melhor dissertação. Este passo é constituído pelos seguintes sub-passos:

- Obter dados relativo ás faces de treino, sendo estes constituídos pelos seguintes dados: media de todas as faces; media das faces de cada classe; matriz de dispersão entre vetores da mesma class (inter class); matriz de dispersão entre as diversas classes (intra class);
- obter o *"Fisher Criterion"* que é a proporção entre a matriz de dispersão inter-class e a matriz de dispersão intra-class
- calcular os novos eigenvalues, com base nos eigenvalues PCA, que representam o espaço de características que maximiza o *"Fisher Criterion"*.

2.5 Classificador K-Nearest-Neighbours

Neste projeto, o classificador requerido é o K-Nearest-Neighbours (KNN) , sendo este utilizado para nos algoritmos anteriormente mencionados.

K-Nearest-Neighbours é um algoritmo que armazena as instâncias de treino em memória, para que estas possam posteriormente ser utilizadas. O modo de funcionamento deste algoritmo consiste em procurar pelos k vizinhos mais próximos de um determinado vetor a predizer, sendo esta distancia calculada através da distância Euclidiana ou da distancia de cosseno. Por fim, de entre os k vizinhos mais próximos, a classe que se apresenta em maior numero é selecionada como previsão para o novo vetor.

2.6 Realidade Aumentada

Realidade Aumentada (RA) é uma tecnologia que integra componentes digitais e do mundo real para fornecer uma experiência imersiva e interativa, permitindo que os dados virtuais sejam incorporados ao ambiente real em tempo real. Assim sendo, para implementar esta ideia é necessário ajustar e posicionar os objetos de forma adequada.

3 Desenvolvimento

3.1 Deteção Facial

Ao serem recomendados os métodos de deteção facial anteriormente mencionados, decidiu-se implementar uma classe, denominada de *FaceDetector*, que implementa os 3 métodos recomendados. Assim sendo é importante mencionar que todos métodos foram implementados utilizado modelos já treinados. Estes métodos foram implementados por forma a retornarem as respetivas deteções num mesmo formato, retornando o conjunto de valores representativos da face e dos olhos.

As deteções retornadas pelo **HCC** consistem num array constituído pelo para (x,y) do canto superior esquerdo e pelo par largura e comprimento (w, h), **DNN** retoram os valores de x e/ou y que constringem a face,e o metodo **dlib** retorna um objeto com as funções que retornam os valores de x e/ou y que constringem a face. Por forma a normalizar a estrutura de dados que o *FaceDetector* retorna, os resultados obtidos são formatados para apresentar a seguinte estrutura (x, y, w, h).

É também importante normalizar os resultados obtidos na deteção dos olhos, pois **HCC** retorna um array constituído pelo para (x,y) do canto superior esquerdo e pelo par largura e comprimento (w, h), no entanto o método **dlib** retorna um conjunto de pontos encontrados na face em questão, sendo os pontos 36-41 e os pontos 42-47 representando o olho esquerdo e o olho direito respetivamente. Assim, por forma a normalizar os dados é retornado um array com o par dos centros dos olhos [centro do olho esquerdo, centro do olho direito], onde cada centro é representado por (x,y). O **DNN**, não apresenta a capacidade de detectar olhos, pelo que utilizou-se **HCC**, para a deteção dos olhos.

Assim todos os métodos implementados retornam 1 array com as faces e 1 array com os pares de centros de olhos, existindo uma relação de 1 para 1 entre o numero de faces e o numero de pares de olhos.

3.2 Noramlização

Após a deteção facial, é necessário realizar uma normalização sobre cada face, assim sendo, é necessário realizar uma rotação da face por forma a alinhar os olhos, seguido de um recorte.

Ao realizar o alinhamento dos olhos é necessário calcular o ângulo atual dos olhos. Para tal, calcula-se o declive de uma reta entre 2 pontos(centro dos olhos), seguido do calculo do ângulo que o declive presenta.

$$m = (y2 - y1)/(x2 - x1)$$
$$angulo = atang(m)$$

Seguidamente é necessário rodar a face, tarefa que se decompõe em cv2.getRotationMatrix2D, que retorna uma matriz de rotação (matriz com os dados referentes á rotaçãao de 1 ponto) seguido de cv2.warpAffine que utiliza a matriz calculada para rodar a frame.

O segundo passo na normalização consiste no escalamento. Este passo é realizado calculando a proporção entre a distancia aspectável (15) e a distancia entre os olhos, e usando esta proporção para escalar a frame.

Em seguida, é preciso recortar a frame a fim de obter apenas a face normalizada. Este corte é realizado com base na nova posição dos olhos, cortando 24 pontos acima de ambos os olho, 32 pontos abaixo e 16 pontos para cada lado dos olhos.

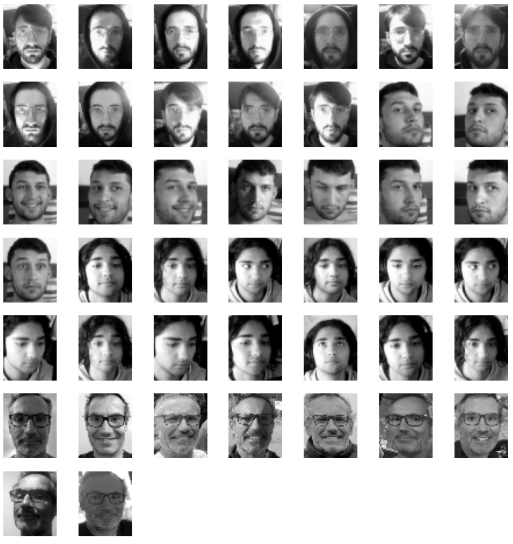

Fig. 1: Instâncias de treino

3.3 Reconhecimento facial

Fig. 2: Mean face

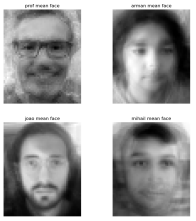

Fig. 3: Mean faces

3.3.1 Eiganfaces

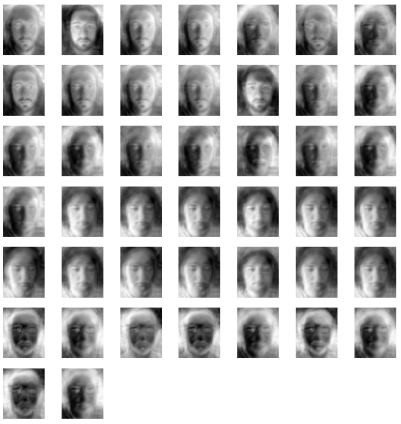

Fig. 4: Eiganfaces - Faces Reconstruidas

Fig. 5: Eiganfaces - Faces Erro

3.3.2 fisherfaces

Fig. 6: Fisherfaces - Faces Reconstruidas

Fig. 7: Fisherfaces - Faces Erro

3.4 Classificador KNN

Tal como descrito anteriormente o classificador KNN, é um classificador que guarda os parâmetros de treino para posteriormente comparar com o paramento a predizer, assim, neste projeto os parâmetros de entrada consistem na projeção das faces de treino e estes são comparados com a projeção da face a predizer.

3.5 Realidade Augmentada

Para aplicar objetos virtuais ao espaço real, é primeiramente necessário aplicar transformações como rotação e escalamento, com base na posição e ângulo da face.

Neste projeto a adição de objetos virtuais é realizada através de mascaras. Está máscara tem como funcionalidade remover o background do objeto a adicionar, tornando o mesmo transparente. Esta mesma mascara é posteriormente invertida e utilizada na zona de interesse (ROI). Com a soma do resultado anteriores obtemos a zona de interesse com a marca desejada, como podemos ver na figura 10.

Fig. 8: ROI com mascara invertida

Fig. 9: Marca com mascara

Fig. 10: Soma das figuras 11 e 9

Tal com podemos observar, ainda existem certos erro, no entanto este problema pode ser resolvido aplicando transformações de imagem , como erosão ou dilatação da mascara.

Após obter a soma esta deve substituir a zona de interesse original.

3.6 Resultado Final

Podemos observar que ao utilizar uma classificador KNN obtemos resultados esperados relativamente á deteção da face, bem como a adição de objetos virtuais á imagem original.

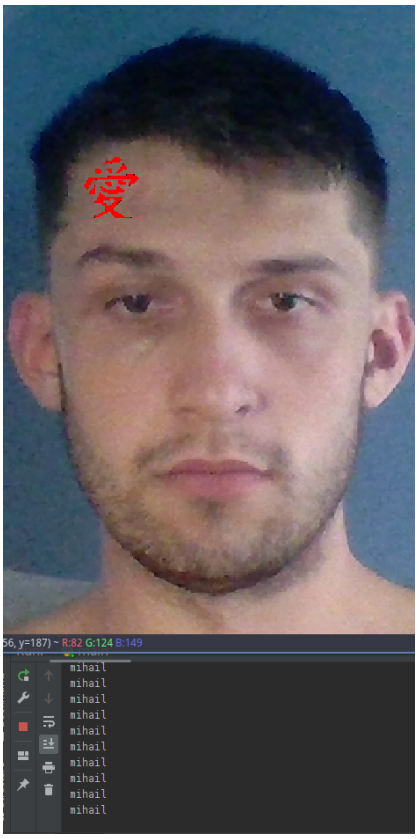

Fig. 11: ROI com mascara invertida

4 Conclusão

O presente projeto possibilitou uma compreensão mais aprofundada e abrangente dos conceitos de realidade aumentada e reconhecimento facial.

O desenvolvimento do projeto decorreu de forma fluida, apesar de pequenos desafios que foram superados com pesquisa adicional. Novos conceitos emergiram durante o trabalho, especialmente na área de reconhecimento facial.

Assim sendo implementou-se diversas técnicas de deteção facial, bem como técnicas de reconhecimento facial, neste caso Eiganfaces e Fisherfaces. Ao realizara este projeto foram estudadas as técnicas anteriores, tendo-se à chegado conclusão que a técnica Eigenfaces procura obter a variância entre todos os elementos do dataset, enquanto a técnica Fisherfaces procura aumentar a distancia entre as diversas classes.

Em resumo, o projeto foi bem aproveitado, alcançando plenamente todos os resultados propostos pelo enunciado, e permitindo um avanço significativo no entendimento e aplicação dos conceitos de realidade aumentada e reconhecimento facial.

References

[HCC23] Haar cascade classifier. *https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html*, 2023.

[HOG16] Histogram of oriented gradients. *https://learnopencv.com/histogram-of-oriented-gradients/*, 2016.