universidade de aveiro
theoria poiesis praxis

Mestrado em Engenharia Informática

Teoria Algorítmica da Informação

Professor Doutor Armando J. Pinho

Lab Work nº 3

Parte Bônus:

Reconhecimento Facial pela Webcam

1º Semestre

2019/2020

Borys Chystov - 78198
Dante Marinho - 83672
Francisco Gonçalves – 99788

Índice

Considerações Iniciais

Como uma parte bônus do Lab Work 3, foi-nos proposto a construção de um pequeno programa que, utilizando a câmara do portátil, pudesse nos identificar.

O trabalho começou com pesquisas na internet através deste algo que nos pudesse dar algum embasamento sobre o funcionamento deste tipo de programa. Não foi difícil encontrar soluções em que já estavam completamente prontas para serem feitas o download e testar. E esta foi a etapa inicial. Contudo, houve sempre a dificuldade de correr o programa, por questões dos pacotes de dependência, assim como ficaria aberta a lacuna da falta de base sobre o assunto.

O grupo encontrou um curso online, o qual foi elegido por poder nos trazer algum conhecimento sobre o assunto, inclusive, se tratando também sobre Machine Learning, que é um assunto ainda pouco explorado por alguns dos membros do grupo. O curso utilizou a linguagem Python e a biblioteca OpenCV.

Para além do trabalho proposto, com a utilização da webcam, foi desenvolvido também um script que permite selecionar um ficheiro do computador e o algoritmo tenta reconhecer a face das pessoas que estão na imagem.

O restante deste tópico da parte bônus, buscará resumir todo o aprendizado adquirido e demonstrar o funcionamento do programa.

Informações sobre o Curso Online

Título: Reconhecimento Facial com Python e OpenCV

Link: https://www.udemy.com/course/reconhecimento-facial-com-python-e-opencv/

Duração: 3,5 horas

Autores: Criado por Jones Granatyr, Gabriel Alves

Considerações sobre o Programa de Reconhecimento Facial

Todo o programa foi construído no decorrer do curso, e após a sua finalização foram realizadas algumas melhorias para melhor se adaptar aos nossos interesses de funcionamento. Para isso, foi criado um repositório em que foi reproduzido todo o material ensinado no curso, e posteriormente, foi criado um novo repositório para realizar as alterações para o trabalho apresentado neste documento. Poderá consultar a o repositório do trabalho no seguinte link:

https://github.com/dantovsky/tai-lab-work3-reconhecimento-facial

Foi realizado um vídeo demonstrativo deste trabalho bônus:

https://www.dropbox.com/s/1qedc2lnshaldug/Demo%20Reconhecimento%20Facial.mp4?dl=0

Embasamento Teórico

O ponto principal de partida foi conscientizar sobre o facto de que a visão computacional simula a capacidade do olho humano. O ser humano enxerga o mundo e a sua vivência neste mundo o faz ser capaz de saber identificar todas as coisas e categorizá-las. Deste modo, a visão computacional busca estudar e perceber como uma máquina pode realizar tal fenômeno.

Deteção VS Reconhecimento

- Deteção: ato de identificar algo. Exemplo: discernir se é o objeto identificado é um rosto de um gato ou se é um rosto humano.
- Reconhecimento: além de identificar a presença de tal categoria de objeto, reconhece a identidade daquilo que vê. Exemplo: identifica que é um rosto humano e saber de quem é o rosto.

Aplicações do uso da deteção no mundo real

- Contar pessoas num ambiente
- Alarmes
- Controle de tráfego em rodovias
- Obter o tamanho de um objeto
- Deteção de sorrisos em câmara ou para o ajuste de foco

Aplicações de uso do reconhecimento no mundo real

- Desbloqueio do telemóvel
- Sistemas de segurança
- Validação em cursos online
- Identificar criminoso

Algoritmos do OpenCV

O OpenCV (*Open Source Computer Vision Library)* é uma biblioteca multiplataforma voltada para o desenvolvimento de aplicativos na área da Visão Computacional. [1] No curso estudado foram utilizados alguns algoritmos que se distinguem como classificadores para o reconhecimento facial: o **Eigenfaces**, o **Fisherfaces** e o **LBPH** (*Local Binary Patterns Histograms*).

Etapas para o Processo de Reconhecimento Facial

O processo de reconhecimento facial é composto por quatro etapas principais:

(1) Deteção da face | (2) Coleta das fotos | (3) Treinamento | (4) Reconhecimento

Deteção da face

Primeiramente é preciso detetar a face que está diante de uma câmara. Tecnicamente é utilizado um ficheiro XML (haarcascade-frontalface-default.xml), que é um classificador que permite a deteção de faces. Ao rosto detetato é desenhado um retângulo em volta da face.

Coleta das fotos

Com uma face detetada, o processo de captura da imagem pode ser ativado. No programa desenvolvido, é necessário digitar a tecla “q” e repetir este processo para capturar um total de 25 imagens, o qual serão armazenados no banco de imagens.

Algumas recomendações a seguir durante a coleta de imagens:

- A qualidade de iluminação nas imagens é importante para que na fase do treinamento tenham um reconhecimento eficiente. É importante, portanto, que tenha boa luminosidade no ambiente. Para o caso de utilizar o algoritmo Eigenfaces, este requer boa iluminação.

- Deve retirar fotos com uma variabilidade grande de expressões, exemplo: foto a olhar para a câmara, com a boca aberta, a olhar para esquerda, pra direita, baixo, cima, fazer cara triste, sorrindo, com e sem óculos.
- Ambiente bem iluminado (o algoritmo EigenFaces requer boa iluminação).
- Fotos com luz incindindo no rosto.

Treinamento

Nesta fase o algoritmo utiliza as fotos capturadas para realizar o treinamento (aprendizado de máquina). Ele analisa cada uma das fotos das faces e encontra padrões consoante o tipo de classificador. Ao final do processo o resultado são a criação de **ficheiros de classificações** (extensão YML), que servirão de base para a base reconhecimento.

Reconhecimento

Esta etapa realiza a tentativa de reconhecimento de uma pessoa diante da câmara, onde é novamente realizada a sua deteção e a tentativa de identificação. É necessário especificar um tipo de classificador a utilizar: Eigenfaces, Fisherfaces ou LBPH.

Comparação entre os Classificadores

Cada classificador possui algumas características que favorecem um mais que o outro. O Eigenfaces e o Fisherfaces utilizam os mesmos tipos de parâmetros para realizarem o treinamento das imagens:

- num_components: número de componentes principais a serem gerados (a documentação indica que 50 é quase sempre suficiente). São a assinatura do rosto.
- threshold: limite de confiança / distância de uma face à outra (vizinho mais próximo - KNN).

Para realizar a captura com o Eigenfaces, é preciso boa iluminação. Algumas vezes o processo de captura não funciona ao pressionar a tecla “q”, por estar com baixa iluminação. No script captura.py a captura só é permitira se a média da imagem que está em escala de cinza for maior que o valor de 110 (numa escala de 0 a 255). Ambos trabalham também com o PCA (*Principal Component Analysis*), uma técnica de Machine Learning para trabalhar com a redução de dimensionalidade dos dados, ou conhecido também como "a seleção dos atributos". Com isso, permite remover informação inútil da imagem.

O Fisherfaces utiliza ainda o LDA (*Linear Discriminant Analysis*), que também reduz as dimensões.

O LBPH trabalha com base em uma matriz de uma imagem, onde para cada pixel, analisa os valores a volta desse pixel para classificar em 0 ou 1 e ao final gera um número binário, que daí é obtido o seu decimal respetivo, que por sua vez será utilizado usado para treinar o sistema, gerando um histograma dos valores para cada face. Ou seja, esse algoritmo é responsável por encontrar a estrutura local da imagem por meio dos valores vizinhos. Na parte do reconhecimento ele utiliza o histograma da imagem e compara com os histogramas da base de imagens. O LBPH possui os seguintes parâmetros:

- radius: raio de abrangência para realizar os cálculos na imagem.
    - Raio maior aumenta a abrangência mas pode perder os pontos mais distantes.
    - Quanto maior o raio, mais padrões podem ser codificados, mas aumenta o esforço computacional.

- neighbors: especifica o número de vizinhos para fazer o cálculo e construir um padrão local.
    - Quanto maior o nº de vizinhos maior é o esforço computacional-
- grid_x: especifica o nº de células na horizontal.
    - Quanto mais células, maior é a dimensionalidade do vetor de características (histogramas).
- grid_y: especifica o nº de células na vertical.
    - Se a grade aumentar serão usados menos pixels em cada histograma (ficam mais esparços).
- threshold: limite de confiança (quanto menor, mais confiável). Se este valor for igual a zero, significa que é exatamente a mesma imagem, ou seja, não há distância entre as duas imagens comparadas.

Os experimentos realizados, com os três algoritmos, mostraram que o LBPH tem melhor confiabilidade do que o Fisherfaces e este por sua vez, melhor que o Eigenfaces. Porém, para cada caso de utilização de um sistema deste gênero, é preciso realizar vários testes a fim de encontrar a configuração que melhor se adapta ao problema em questão. Um algoritmo bastante "afinado" também poderá não conseguir reconhecer com muita facilidade.

Desenvolvimento e Código

O programa desenvolvido conta com a seguinte estrutura de pastas e ficheiros:

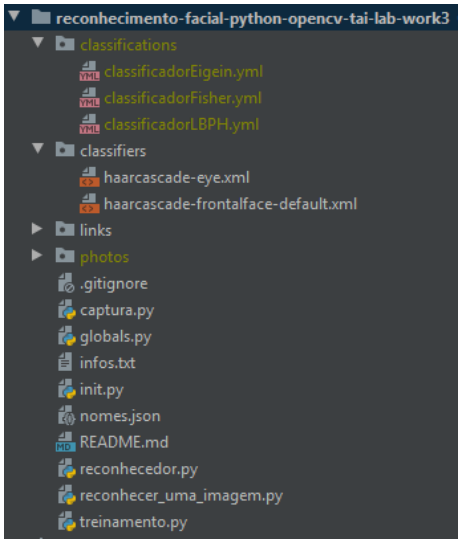

Figura 1 - Estrutura de pastas do programa

A primeira execução do programa, para capturar as primeiras imagens de uma pessoa, pode ser feita sem a existência da pasta "classifications", assim como a pasta "photos". Ao final deste primeiro processo estas duas pastas serão geradas.
Ao executar o ficheiro init.py, será exibido na consola um menu, o qual permitirá ao utilizador uma melhor experiência.

```
~ ~ ~ Reconhecimento Facial ~ ~ ~
1 - Capturar imagens pela webcam
2 - Iniciar treinamento das imagens
3 - Reconhecimento facial pela webcam (Classificador Eigenfaces)
4 - Reconhecimento facial pela webcam (Classificador Fisherface)
5 - Reconhecimento facial pela webcam (Classificador LBPH)
6 - Reconhecimento facial a partir de ficheiro de imagem (Classificador LBPH)
7 - (i) About
0 - SAIR

Opção: 0
```

Figura 2 - Visão do menu principal do programa

A opção de reconhecimento facial a partir de um ficheiro foi utilizada apenas com o classificador LBPH, pois os outros classificadores são exigentes em relação em ter que selecionar imagens com o mesmo tamanho das imagens que foram utilizadas para gerar as classificações. No script de captura foi especificado um tamanho de 200x200 pixels. No caso do LBPH, é possível selecionar uma imagem qualquer e o algoritmo tentará reconhecer todas as faces que forem detetadas na imagem.
A ilustração a seguir demonstra um exemplo do reconhecimento facial com o LBPH em execução.

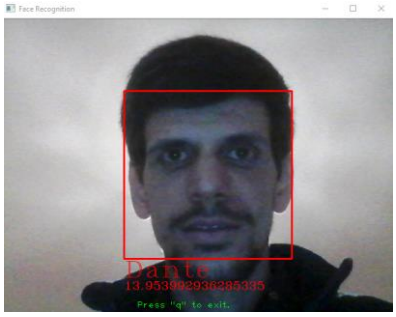

Figura 3 - Reconhecimento facial com LBPH

Os textos em cor vermelho são o nome da pessoa detetada e a variável que representa o limite de confiança. Na próxima ilustração, pode-se ver a deteção de três faces, sendo que em uma dela o algoritmo classificou de forma errada. Mas ainda assim, pode-se considerar como uma boa classificação, pois a imagem do Leonardo foi capturada a partir de uma foto impressa e apenas uma mesma foto para as 25 capturas necessárias.

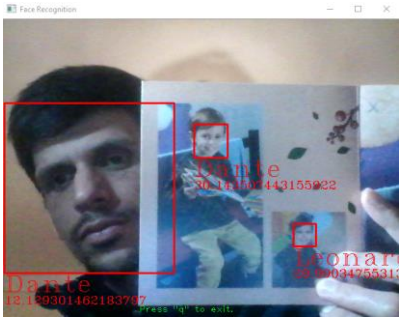

Figura 4 - Algoritmo LBPH a reconhecer várias faces

Para o reconhecimento facial exemplificado na ilustração seguinte, foi feito o treinamento a partir de várias outras imagens do Leonardo. O reconhecimento desta vez teve melhor desempenho, conseguindo descobrir todas as faces corretamente.

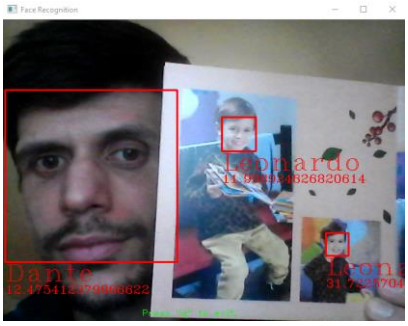

Figura 5 - Algoritmo LBPH a reconhecer várias imagens

A fim de testar melhor a eficiência desta aplicação, foi realizado um experimento para reconhecer as expressões faciais. Apenas foi eliminado qualquer registo na base de imagens e em vez de registar pessoas diferentes foi registado a mesma pessoa, mas para cada registo uma expressão diferente. Neste experimento o melhor resultado de reconhecimento foi com a utilização do classificador LBPH, mas em alguns momentos ele se mostra bastante confuso, não conseguindo permanecer por muito tempo a escolha da expressão detetada.

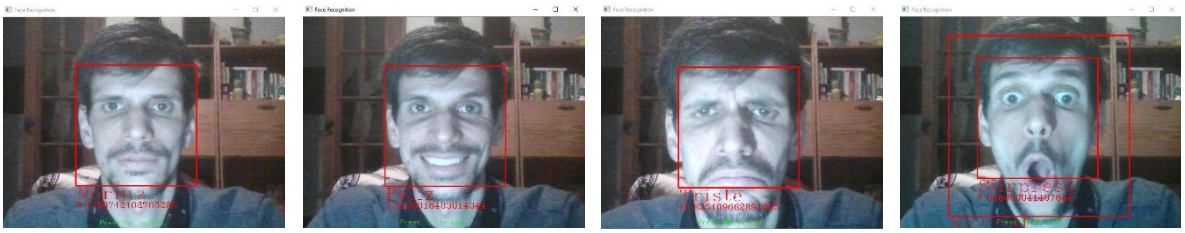

Figura 6 - Utilização do LBPH para reconhecimento de expressões

Considerações Finais

O contato com as técnicas de Machine Learning foi um fator bastante interessante para o grupo. Este trabalho bônus permitiu conhecer três algoritmos classificadores de faces humanas e foi evidente perceber que para que eles tenham um funcionamento aceitável é preciso a realização de vários testes com variações em seus parâmetros de entrada para que melhor atenda a uma necessidade específica de utilização.

Como sugerido pelo autor do curso, existem formas de automatizar os testes dos parâmetros, a fim de realizar um grande número de experimentos e assim poder encontrar a melhor parametrização para uma solução específica. Como sugerido pelo autor do curso do Reconhecimento Facial, o seguinte link contém um script desenvolvido para a realização de testes automatizados dos parâmetros de entrada: https://bitbucket.org/SpikeSL/vision-systems/src/master/

A solução desenvolvida está funcional em ambiente Windows, porém todos os requisitos necessários estão mencionados no ficheiro README.md. Os testes realizados em ambiente Linux ocasionaram erros referentes ao módulo do OpenCV, o qual não houve sucesso de solução. Em ambiente Windows também ocorreram os mesmos erros, porém conseguiu-se solucionar ao instalar os pacotes do OpenCV-Contrib-Python e Pillow pelo instalador de pacotes da IDE Pycharm.

Referências

[1] Wikipedia, “OpenCV,” [Online]. Available: https://pt.wikipedia.org/wiki/OpenCV.