init.md

CV Homework

> PKU 信息工程學院 2101212850 干皓丞

影像人臉辨識與音源成果 - Demo

bilibili : https://www.bilibili.com/video/bv1jf4y1J7rh

github project : https://github.com/kancheng/kan-cs-report-in-2021/tree/main/CV/pytorch-init-and-face

0. About

Tasks

1. 安裝 PyTorch
2. 執行所有 ipynb 文件
3. 視頻人臉檢測
4. 輸出運行後的結果

> https://pytorch.org/tutorials/beginner/pytorch_with_examples.html

Prepare

1. 建立好屬於 PyTorch 的虛擬環境並安裝。(Windows & Mac)

```
conda create -n pytorch-init python=3.8


conda activate pytorch-init
```

Reference

> https://pytorch.org/get-started/locally/

2. 下載所需要的人臉檢測實驗影片素材

用 Windows Photo 擷取，並用 Mac 的 QuickTime Player 合併。

(0) The mind behind Linux | Linus Torvalds by Youtube Channel - TED

https://www.youtube.com/watch?v=o8NPllzkFhE

> 選用主因 : 自身信仰與影片人臉不複雜，適合 Demo

(1) They Shall Not Grow Old – New Trailer – Now Playing In Theaters by Youtube Channel - Warner Bros. Pictures

https://www.youtube.com/watch?v=IrabKK9Bhds

> 選用主因 : 想知道 OpenCV 在修復後的紀錄片效果

(2) 1917 | The Battlefield Run in 4K HDR - by Youtube Channel - Universal Pictures

https://www.youtube.com/watch?v=WYCo-3pw52o

> 選用主因 : 想知道 OpenCV 在大量人物複雜背景下的效果

(3) DUNKIRK - OFFICIAL MAIN TRAILER [HD] by Youtube Channel - Warner Bros. Pictures

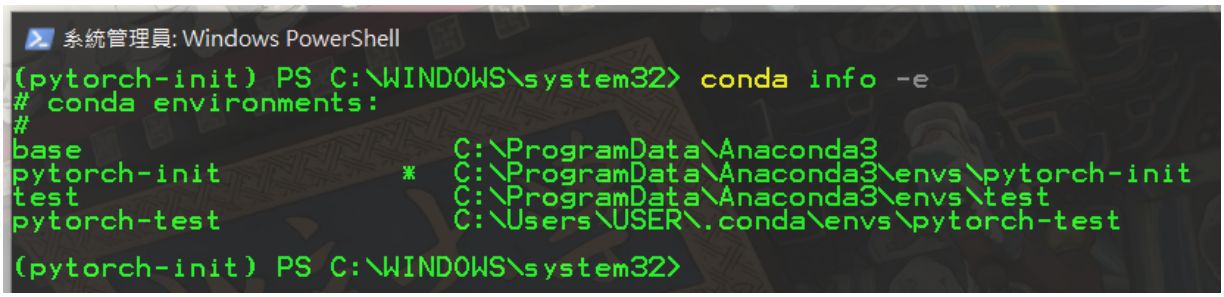
https://www.youtube.com/watch?v=T7O7BtBnsG4

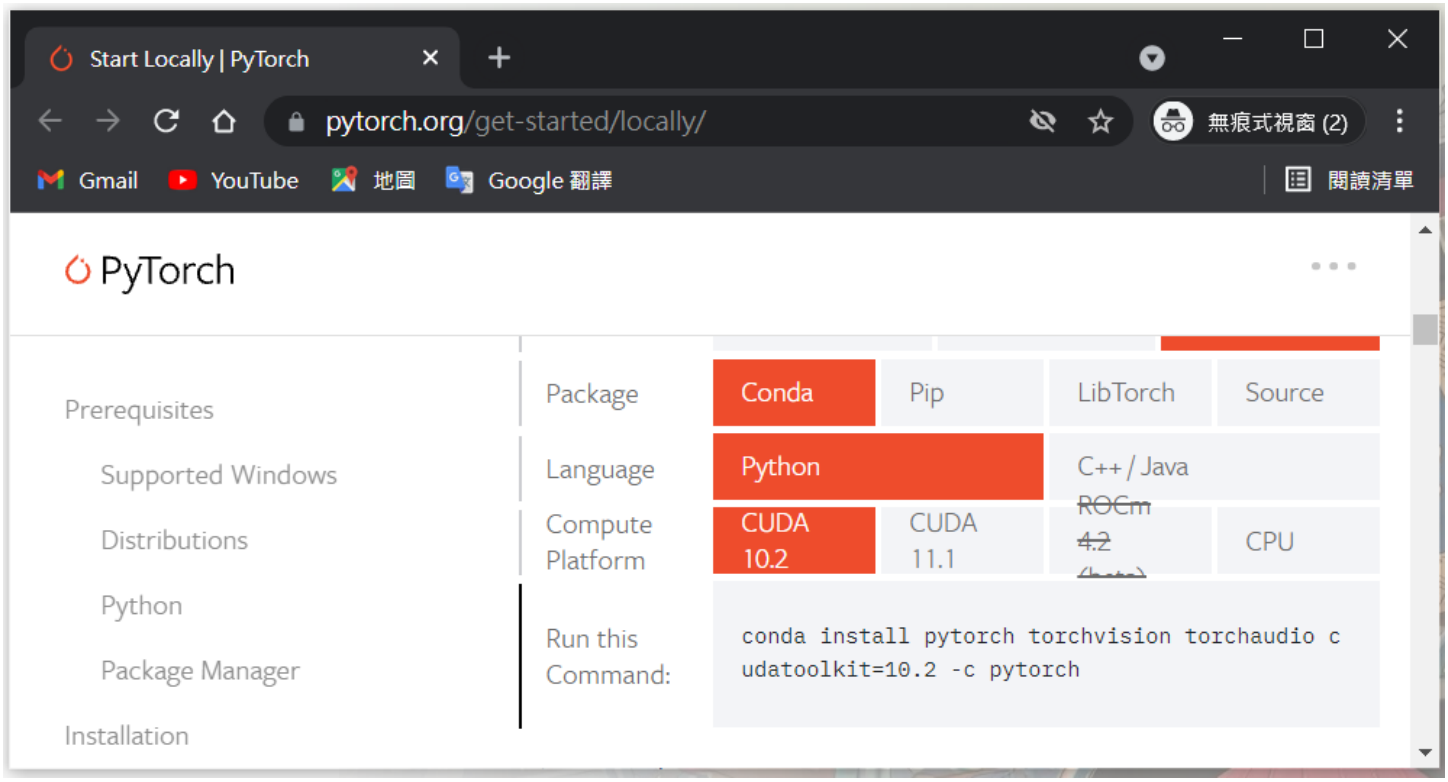
> 選用主因：想知道 OpenCV 在臉部特寫影片下的效果

1. Details

安裝 PyTorch

```
# windows
conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch

# mac
conda install pytorch torchvision torchaudio -c pytorch
```





執行所有 ipynb 文件

程式碼與匯出的檔案結果於專案的 `code` 目錄下，而為了課堂呈現匯出 PDF 於專案的 `pdf` 目錄下。在解釋各 `*.ipynb` 執行狀況。

1. W2_Numpy.ipynb

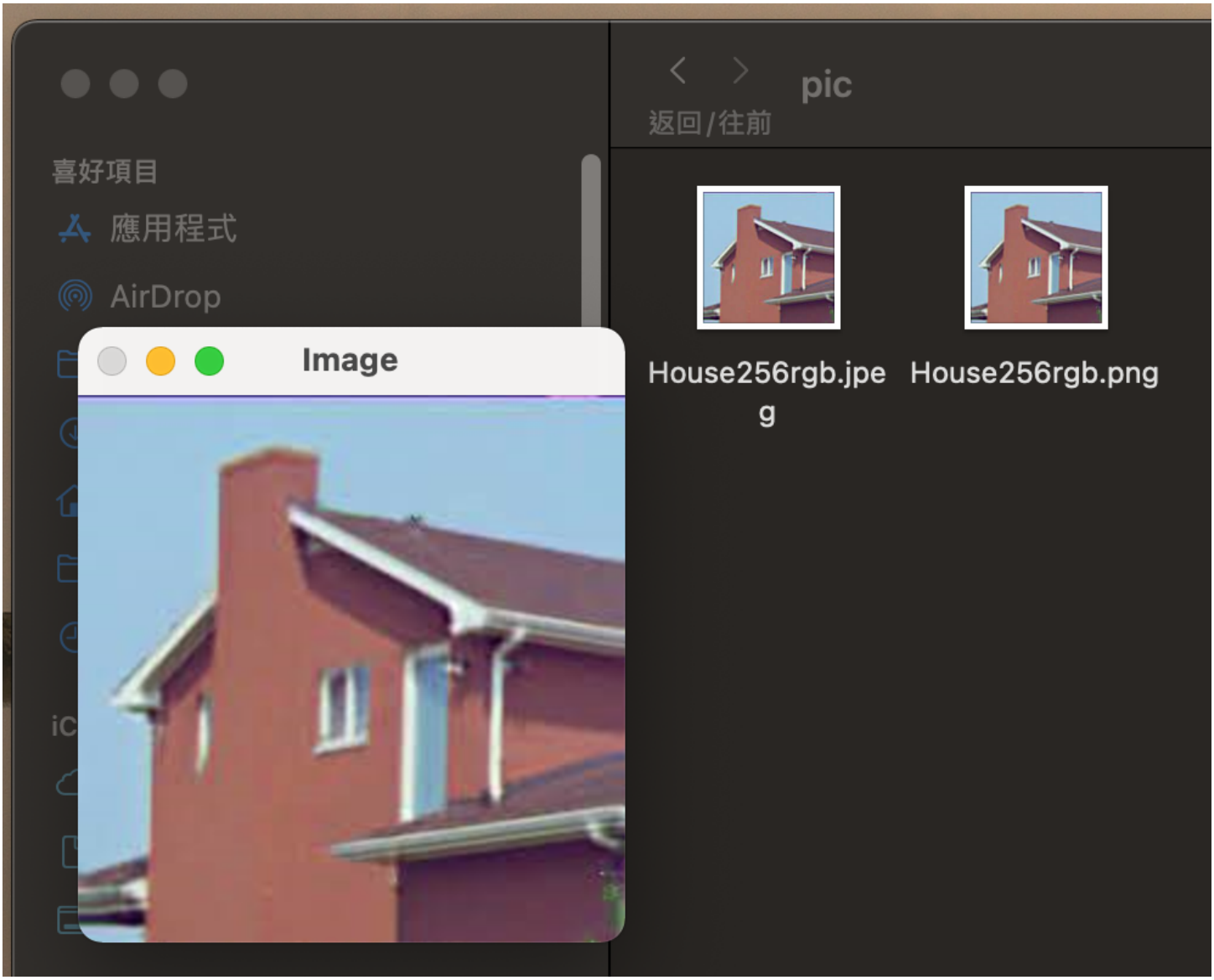
為 CS 231n Python & NumPy Tutorial 的教程。

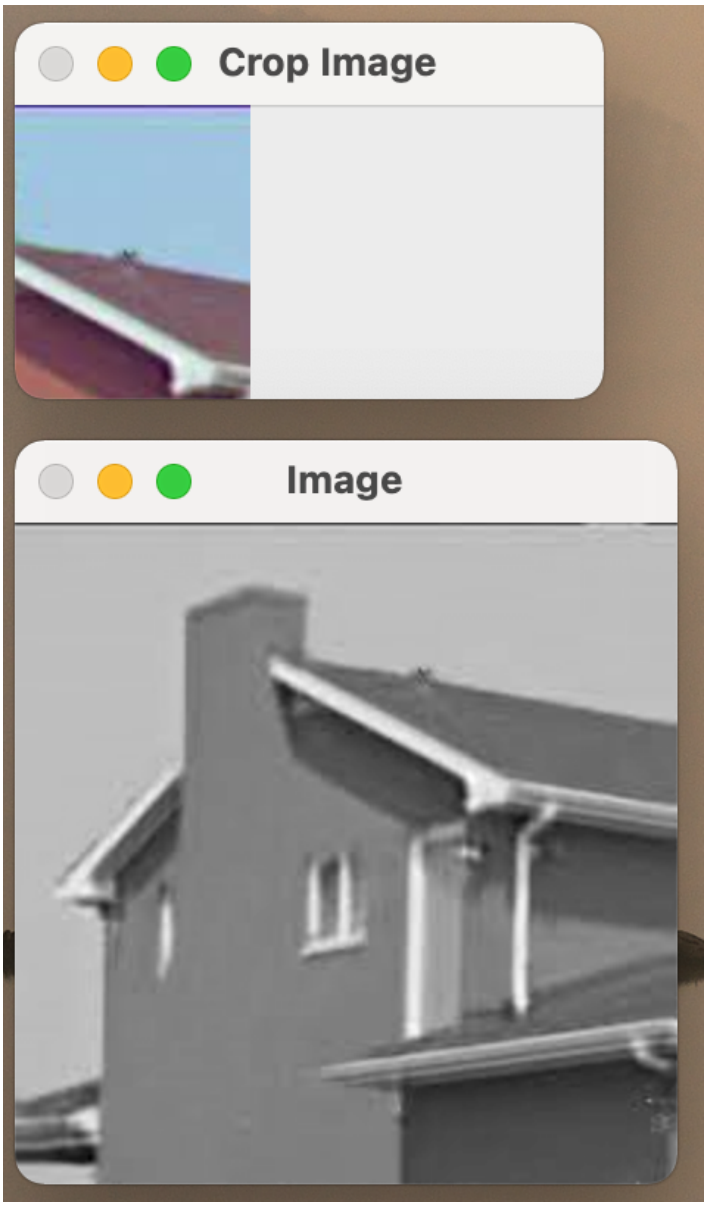
2. W2_OpenCV.ipynb

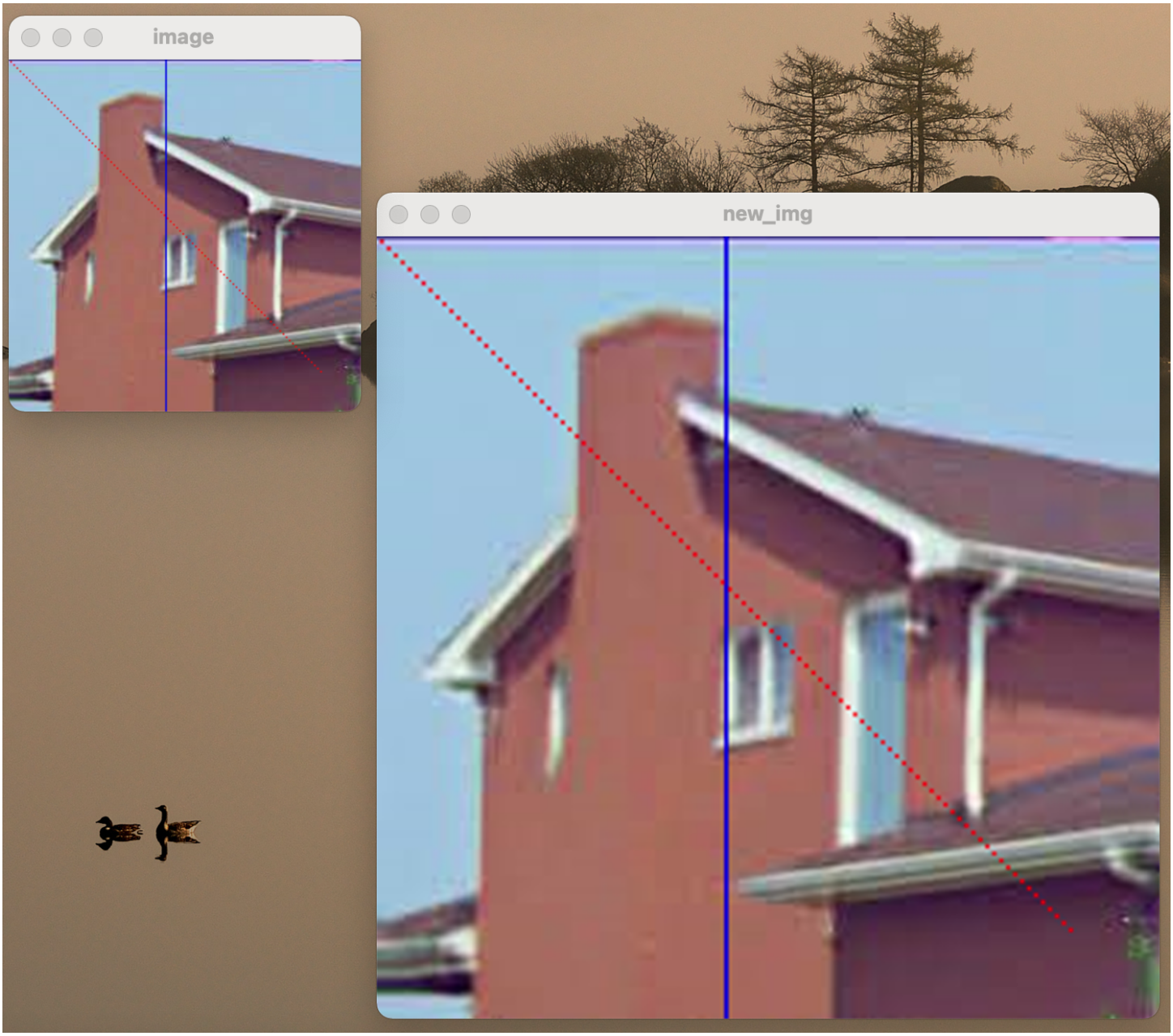
OpenCV 的呈現，當中的圖檔並沒有附上，所以自己額外再去 Google 下載， `House256rgb.png` 的圖檔，加入專案中，同時修改路徑。

最後面畫紅點與藍線的部分，由於紅點並不明顯，而且一開始的部分有 Bug，查完錯後，自己額外修改成畫多點的虛線。

```
for k in range(0, 225):
    img[k, 100, :] = [225, 0, 0]
for g in range(1,200,2):
    img[g, g] = [ 0, 0, 255]

cv2.imshow('image', img)
cv2.waitKey(0)
```


喜好項目
應用程式
AirDrop
返回/往前
pic
Image
House256rgb.jpeg
House256rgb.png


Crop Image
Image



3. W2_Python-Class.ipynb

Python 類別

4. W3_PyTorch_Basic.ipynb

PyTorch 基礎，過程中有報錯，修正完畢。

5. W3_Regression_Python.ipynb

Regression 圖像

6. W3_Tensor_Tutorial.ipynb

Tensor 實作

視頻人臉檢測

解決邏輯：OpenCV 匯出匯入影像 & OpenCV 影像人臉識別 -> OpenCV 匯入影像進行人臉識別後匯出無音源影像 -> 嘗試事先分離音源後合併前者影像成果。

事前測試

先使用電影 1917 的部分來測試

> 該程式碼皆為自己臨時趕出來的測試範例，僅單純能夠呈現，過程中還有一些錯誤待解決。

1. 官方範例修改，單純輸入轉檔

影像寬高與路徑很重要，當然該影像沒有聲音，因為 OpenCV 沒有負責這個，只有 FFmpeg 有。

```
import cv2

# cap = cv2.VideoCapture(0)
cap = cv2.VideoCapture('mov/t1-1917-the-battlefield-run4.mp4')
# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'MJPG')
out = cv2.VideoWriter('output.avi',fourcc, 20.0, (1920,1080))
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    # 水平上下翻轉影像
    #frame = cv2.flip(frame, 0)
    # write the flipped frame
    out.write(frame)
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) == ord('q'):
        break
# Release everything if job is finished
cap.release()
out.release()
cv2.destroyAllWindows()
```

3. 單純影像人臉辨識

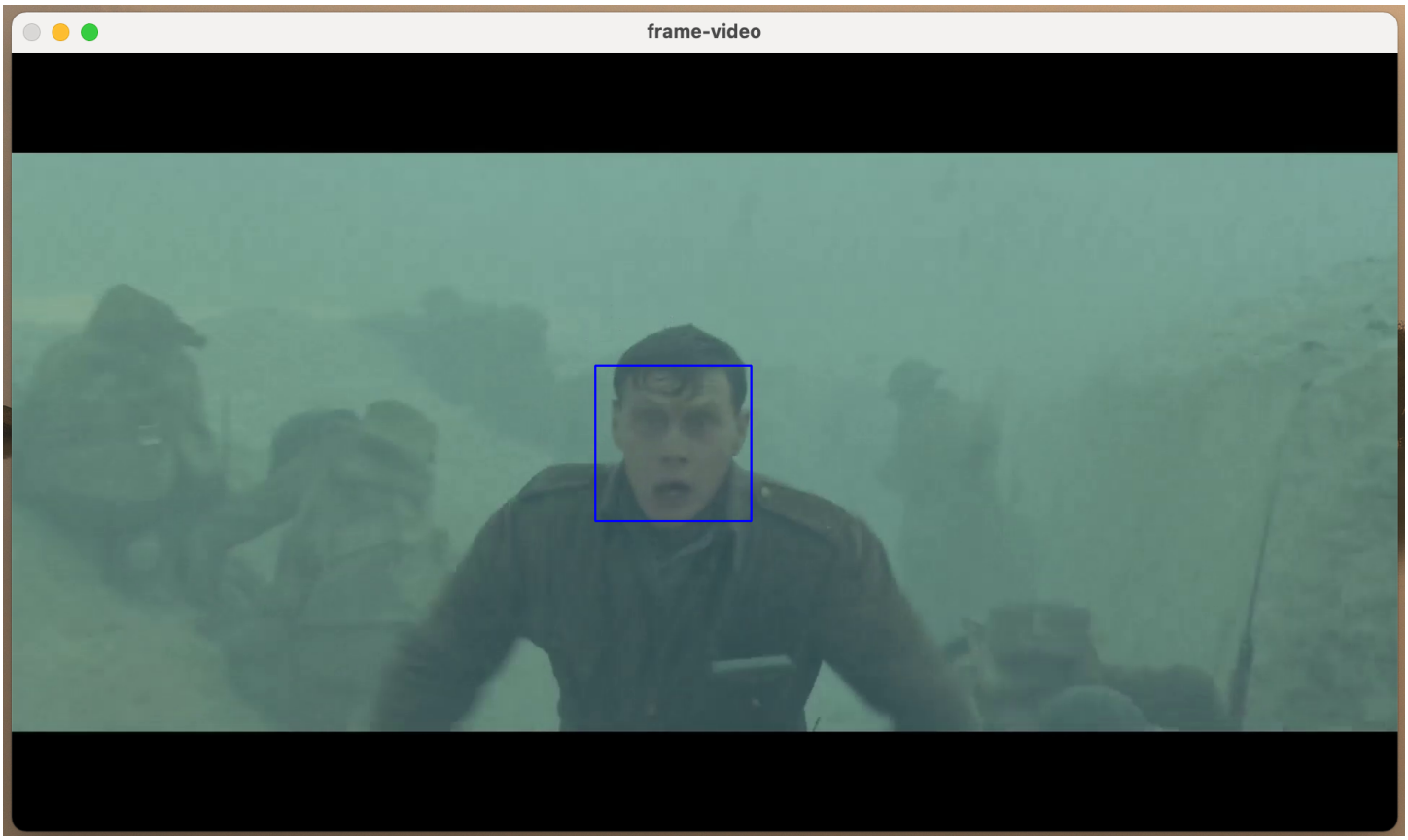
```
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# To capture video from webcam.
cap = cv2.VideoCapture(0)
# To use a video file as input
# cap = cv2.VideoCapture('filename.mp4')

while True:
    # Read the frame
    _, img = cap.read()
    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Detect the faces
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
    # Draw the rectangle around each face
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
    # Display
    cv2.imshow('img', img)
    # Stop if escape key is pressed
    k = cv2.waitKey(30) & 0xff
    if k==27:
        break
# Release the VideoCapture object
cap.release()
```

4. 影像人臉辨識並輸出成無音源檔案

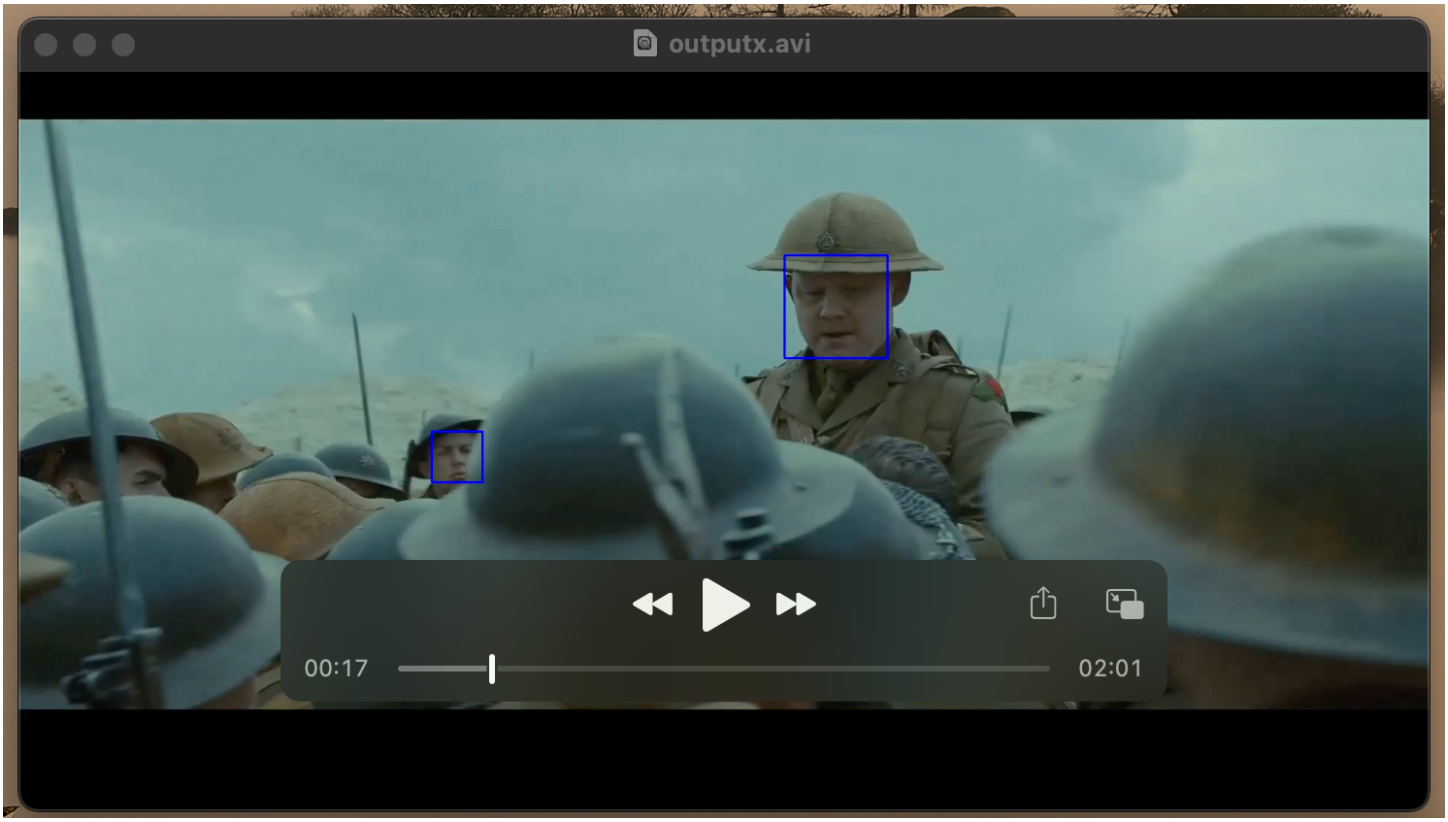
```
import numpy as np
import cv2 as cv

# Load the cascade
face_cascade = cv.CascadeClassifier('opcv-face/haarcascade_frontalface_default.xml')
# cap = cv.VideoCapture(0)
cap = cv.VideoCapture('mov/t1-1917-the-battlefield-run4.mp4')
# Define the codec and create VideoWriter object
# fourcc = cv.VideoWriter_fourcc(*'XVID')
```

```
# out = cv.VideoWriter('output.avi', fourcc, 20.0, (640,  480))
fourcc = cv.VideoWriter_fourcc(*'MJPG')
out = cv.VideoWriter('/Users/kancheng/py-work/outputx.avi',fourcc, 20.0, (1920,1080))
while cap.isOpened():
    ret, frame = cap.read()
    # Convert to grayscale
    gray = cv.cvtColor( frame, cv.COLOR_BGR2GRAY)
    # Detect the faces
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
    # Draw the rectangle around each face
    for (x, y, w, h) in faces:
        cv.rectangle( frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    # 水平上下翻轉影像
    # frame = cv.flip(frame, 0)
    # write the flipped frame
    out.write(frame)
    cv.imshow('frame-video', frame)
    if cv.waitKey(1) == ord('q'):
        break
# Release everything if job is finished
cap.release()
out.release()
cv2.destroyAllWindows()
```

已經可以從預計的部分測試得知，1917 - The Battlefield Run，可以成功辨識出人臉，但面對複雜背景跟人，很有可能會辨識錯誤，比如將士兵的水壺與壕溝邊的碎石面當成人臉。



當結束 Jupyter Notebook 時，可以看到已經匯出的 avi 檔案。



切記！ 每個影片的寬高必須要特別注意！

ffmpeg install - mac

```
brew install ffmpeg
```

ffmpeg - mp4 to wav

```
ffmpeg -i $ID.mp4 -ac 1 ar 16000 $ID.wav
```

ffmpeg 合併影像與音源

```
import subprocess
cmd = 'ffmpeg -i new_music.wav -i star_gray.mp4 out.mp4'
subprocess.call(cmd)#返回‘0’就说明合并成功了
```

ffmpeg 合併多個影像

先建立一個設定檔 `merge-video.txt`

```
file '/path/to/file1.mp4'
file '/path/to/file2.mp4'
file '/path/to/file3.mp4'
```

合併指令

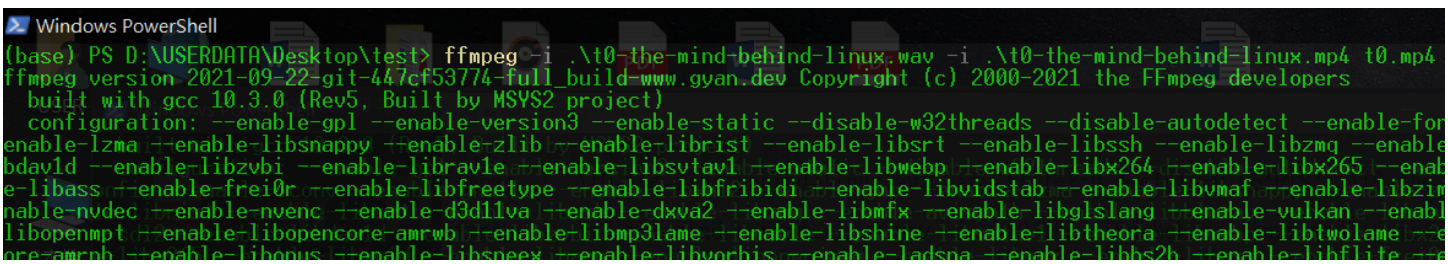
```
ffmpeg -f concat -safe 0 -i merge_video.txt -c copy output.mp4
```

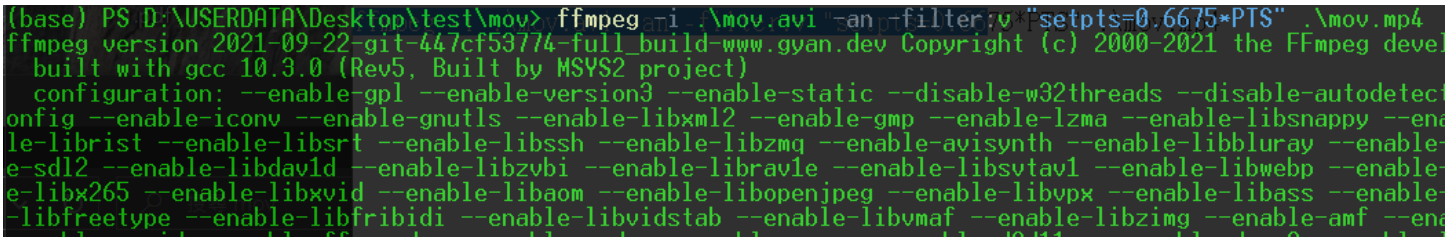
5. 音源與影像檔案合併

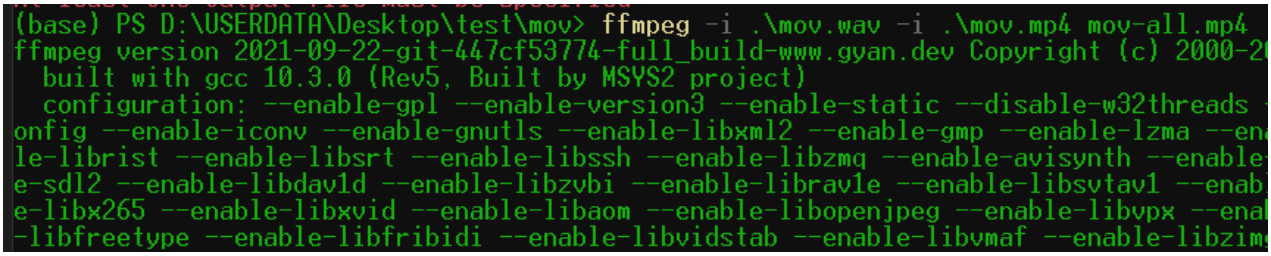
由於 Linus 的 TED 影片寬高與後面電影的影片寬高不一，所以分為兩類各自拆開處理。實際上原本的音檔與影像人臉辨識處理過後的檔案並沒有對上。所以只能用 PTS 直接處理。

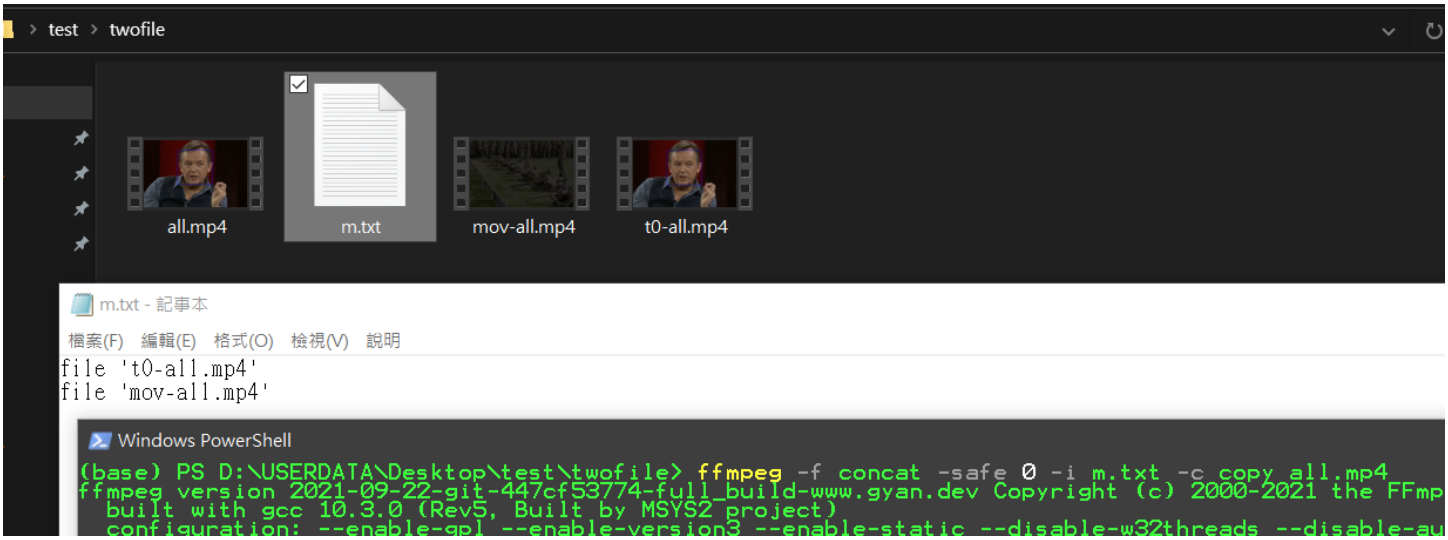
> 音源跟影像的幀數應該有更好的方式，但由於在此音源不是重點，一秒內的誤差可以接受。

```
ffmpeg -i .\mov-raw.mp4 -ac 1 -ar 16000 audio.wav

ffmpeg -i .\t0-the-mind-behind-linux.avi -an -filter:v "setpts=0.66*PTS" .\t0-the-mind-behind-linux.mp4

ffmpeg -i .\t0-the-mind-behind-linux.wav -i .\t0-the-mind-behind-linux.mp4 t0.mp4

ffmpeg -i .\mov.avi -an -filter:v "setpts=0.6675*PTS" .\mov.mp4
```









Reference

由於這次作業涉略大量的知識，考量到可以用在日後不同課程的問題上，在此列出作為紀錄。

> https://docs.opencv.org/master/dd/d43/tutorial_py_video_display.html
>
> https://github.com/adarsh1021/facedetection
>
> https://blog.csdn.net/JNingWei/article/details/78753607
>
> https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81
>
> https://stackoverflow.com/questions/30509573/writing-an-mp4-video-using-python-opencv
>
> https://zhuanlan.zhihu.com/p/41738479
>
> https://blog.csdn.net/duan19920101/article/details/53317197
>
> https://www.jianshu.com/p/88f70f0e6b14
>
> https://blog.csdn.net/IT_zxl001/article/details/117353780
>
> https://www.youtube.com/watch?v=sz25xxF_AVE
>
> https://yanwei-liu.medium.com/python%E5%BD%B1%E5%83%8F%E8%BE%A8%E8%AD%98%E7%AD%86%E8%A8%98-%E4%B8%80-%E4%BD%BF%E7%94%A8open-cv%E8%BE%A8%E8%AD%98%E5%9C%96%E7%89%87%E5%8F%8A%E5%BD%B1%E7%89%87%E4%B8%AD%E7%9A%84%E4%BA%BA%E8%87%89-527ef48f3a86
>
> http://www.4k8k.xyz/article/qq_40575024/105908013

> https://www.zhihu.com/question/49558804
>
> https://blog.csdn.net/ayouleyang/article/details/104101049
>
> https://www.youtube.com/watch?v=8nbuqYw2OCw
>
> https://www.wongwonggoods.com/linux/ffmpeg/ffmpeg_merge_video/

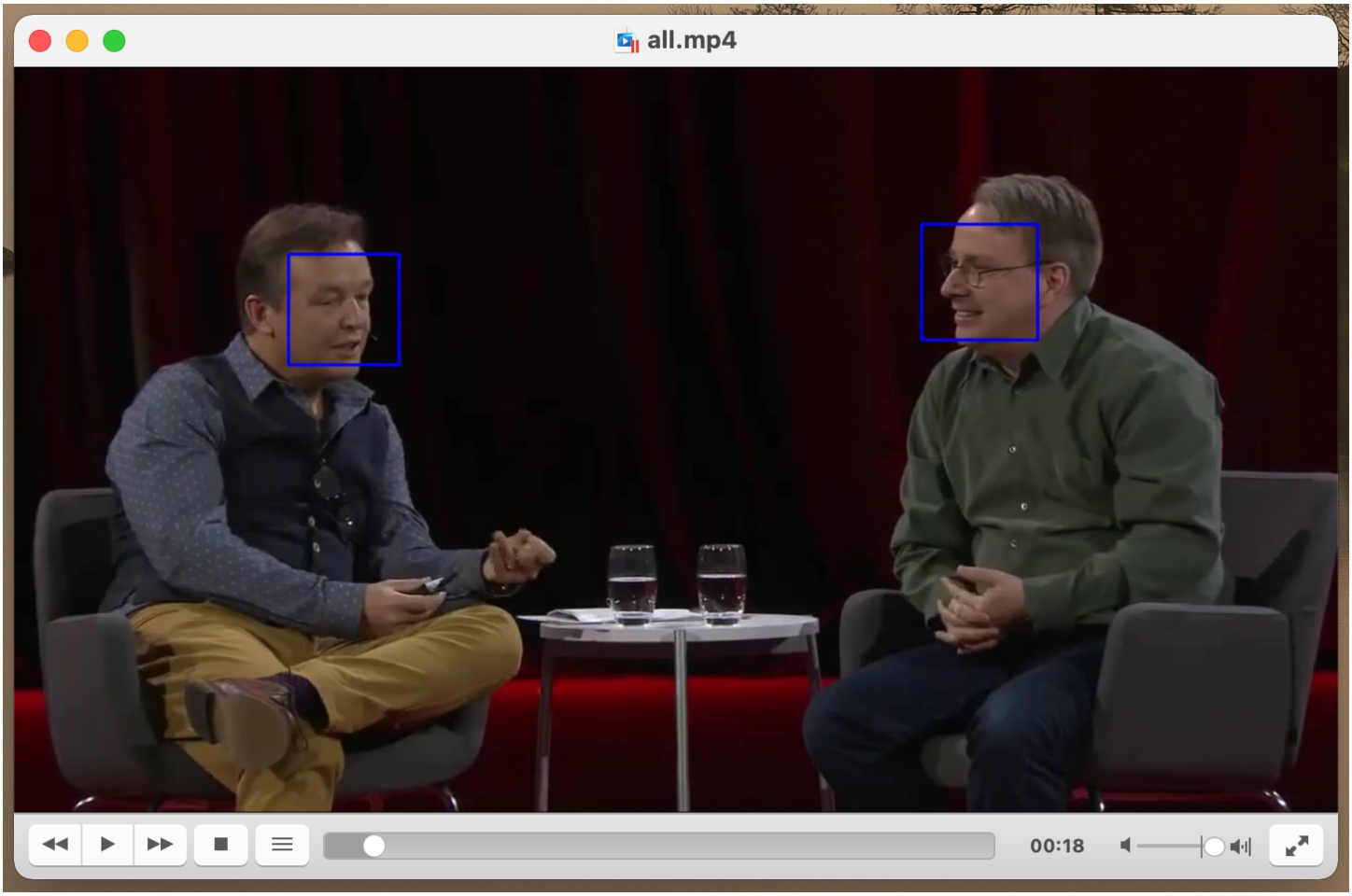
輸出運行後的結果

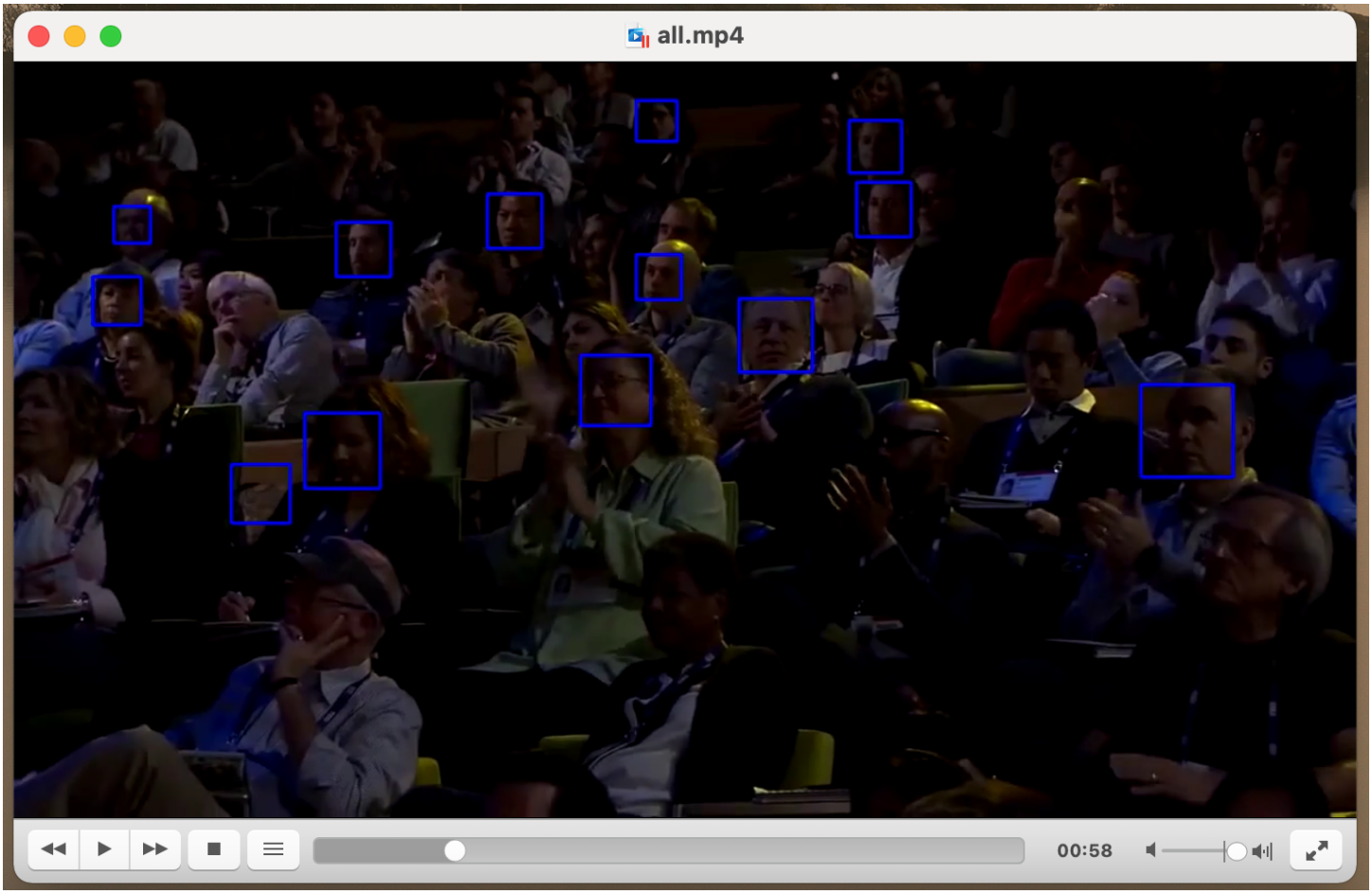
0. 影像人臉辨識與音源成果 - Demo

bilibili : https://www.bilibili.com/video/bv1jf4y1J7rh

1. The mind behind Linux | Linus Torvalds

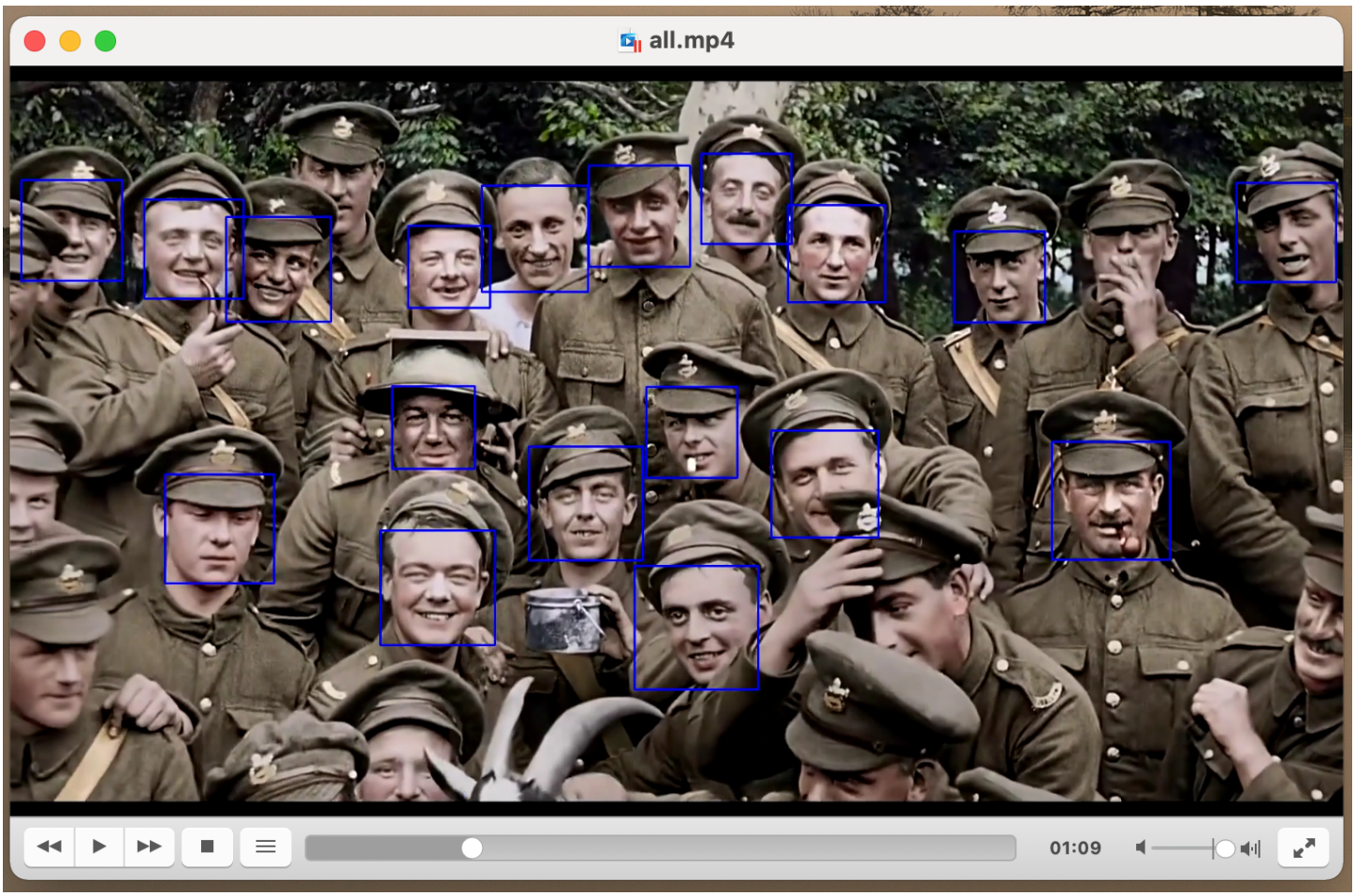
TED 講座影片，人臉辨識是當中最為穩定的段落，當然一拉開距離到遠景，人臉就很難辨識。





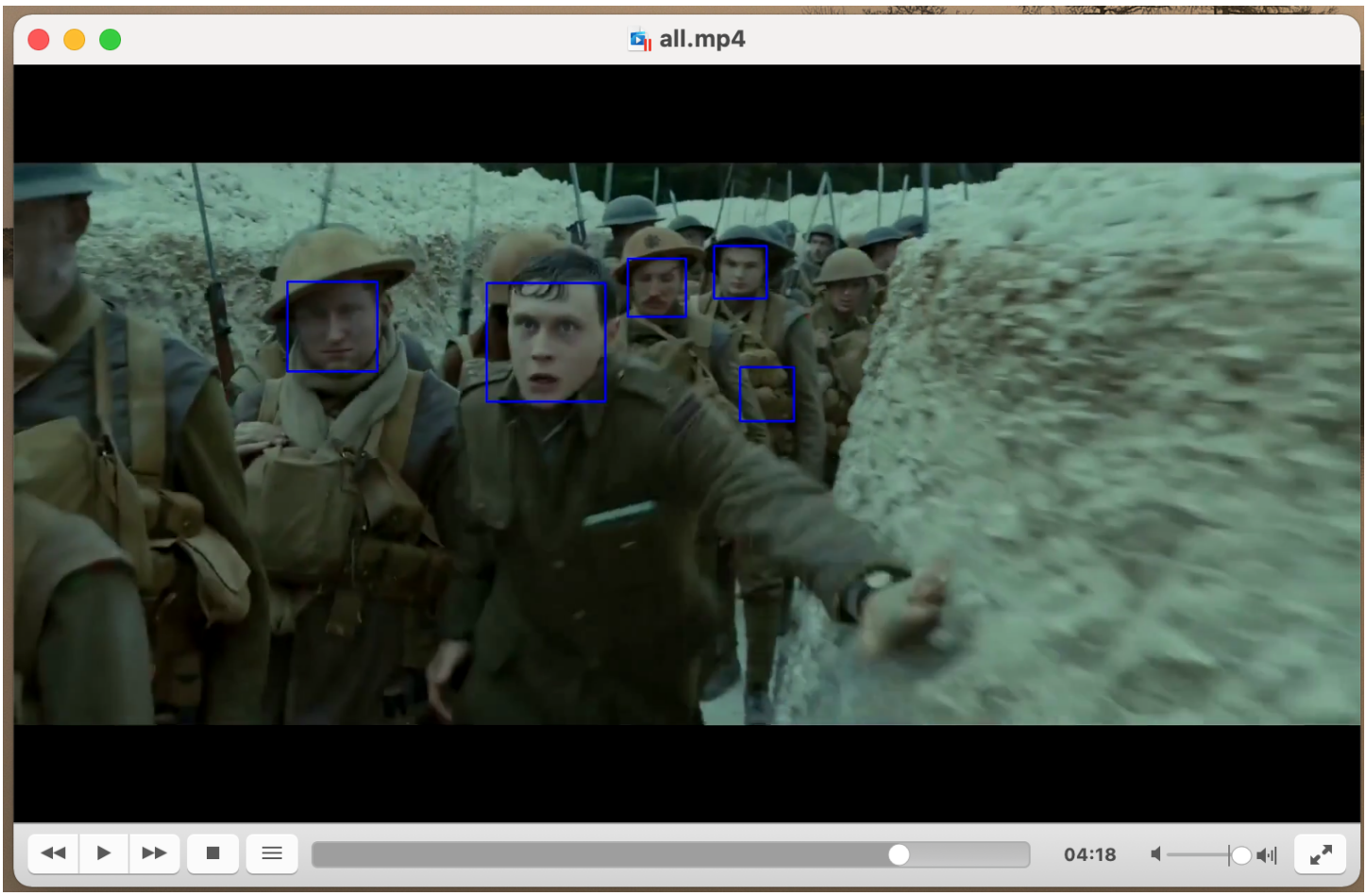
2. They Shall Not Grow Old

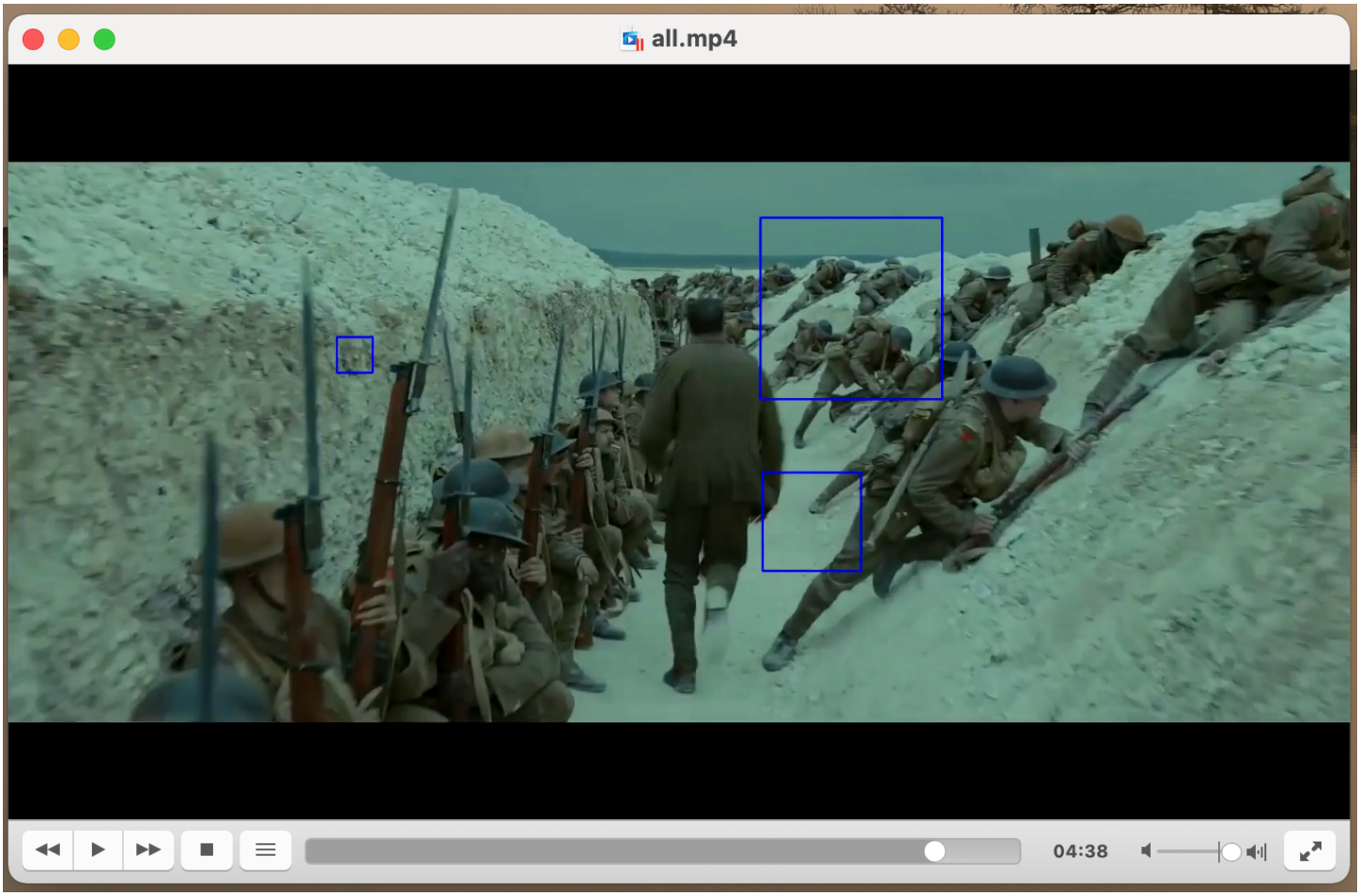
一戰紀錄片，雖然是修復過後的影片，人群正面的段落，人臉辨識非常的穩定，但是複雜的背景會被誤人為人臉，而模糊的部分則不會被辨識。



3. 1917 - The Battlefield Run

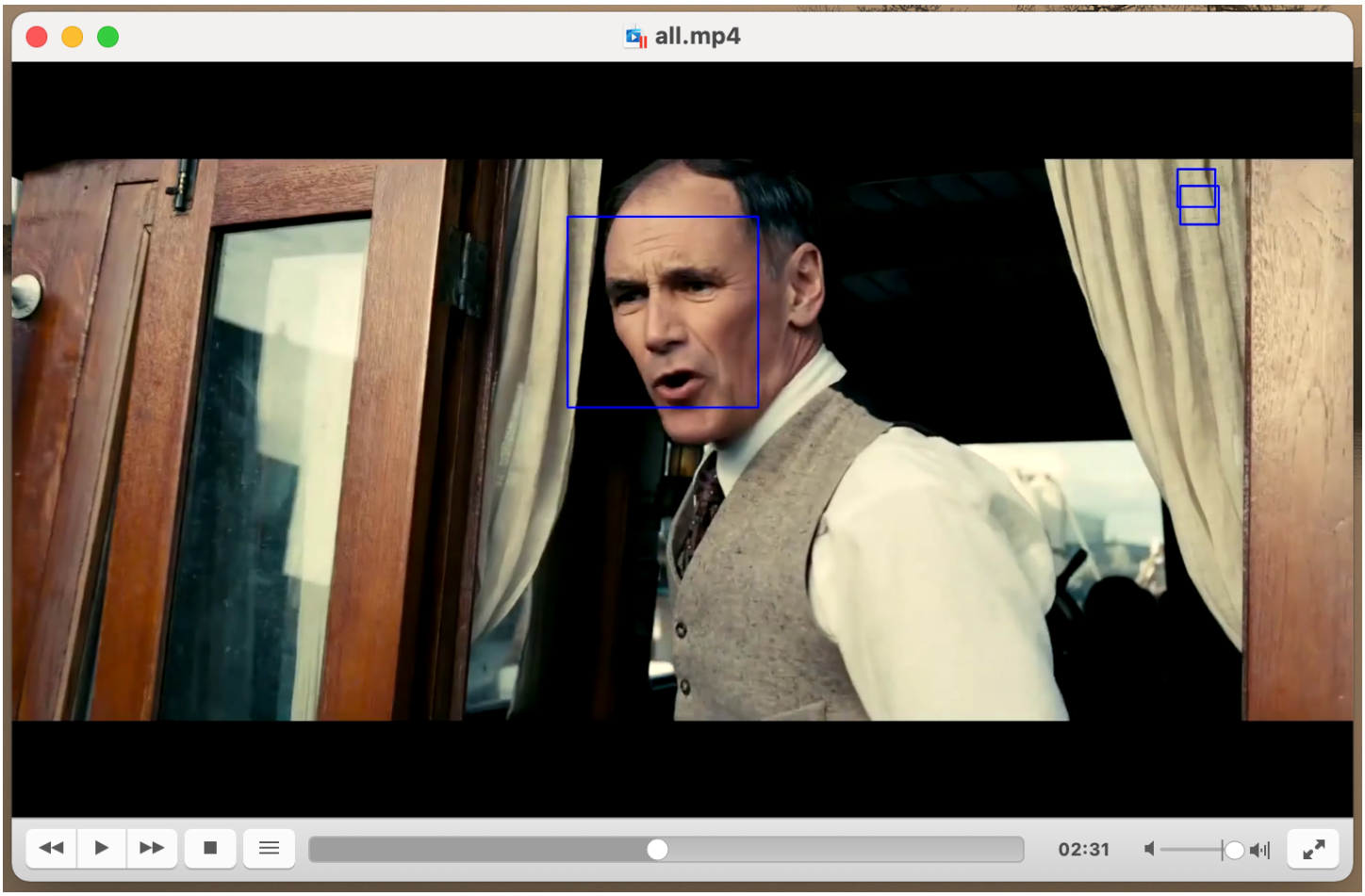
真實還原一戰的電影，但由於背景壕溝石子牆面，容易被視為人臉，而人臉若在光源不穩定，天氣景色偏暗的情況下可能無法辨識。而背景構圖複雜也容易被誤認為是人臉。





4. DUNKIRK

人物的近景，或者光線偏暗會讓人臉無法辨識。




all.mp4
03:37