機器視覺作業報告

# 題目

姓名:吳俊威

學號:01157061

日期:4/4/2025

# 方法

First Process the Image Processing

```
# Import Youtube URL
youtube_url = "https://www.youtube.com/watch?v=PHqhEgkGfrs"
data = urllib.parse.urlparse(youtube_url)
query = urllib.parse.parse_qs(data.query)
ID = query["v"][0]
video = 'https://www.youtube.com/watch?v={}'.format(str(ID))
```

Function to process the video from the File Source

```
def process_image(image):
    # Resize the resolution of the video become 600 x 500
    resize = cv2.resize(image, (600,500))

    # Convert the video BGR (Blue-Green-Red) color space to grayscale
    gray = cv2.cvtColor(resize, cv2.COLOR_BGR2GRAY)

    # Convert the video color back to a 3-channel (RGB) image
    gray_3d = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)

    # Apply Gaussian filter to the video
    gaussian = cv2.GaussianBlur(resize, (31,31), 0)

    # Convert the video to the HSV color space using tensorflow
    vid = tf.convert_to_tensor(resize, dtype=tf.float32) / 250.0
    hsv = tf.image.rgb_to_hsv(vid)

    # Convert the scaled HSV tensor back to the uint8 data type or else it won't be visible
    hsv = tf.cast(hsv * 250, dtype=tf.uint8)
    hsv_numpy = hsv.numpy()

    combined = np.hstack((resize, gaussian, hsv_numpy))
    return combined
```

Function to process the video from Youtube URL

```
while True and capture.get(cv2.CAP_PROP_POS_MSEC)<=total*mili:
    ret, image = capture.read()
    if not ret:
        break

    # Resize the resolution of the video become 600 x 500
    resize = cv2.resize(image, (600,500))

    # Creates a 5x5 kernel with all elements set to 1
    kernel = np.array([[1, 1, 1, 1, 1],
                       [1, 1, 1, 1, 1],
                       [1, 1, 1, 1, 1],
                       [1, 1, 1, 1, 1],
                       [1, 1, 1, 1, 1]])

    # Normalizes the kernel
    kernel = kernel/sum(kernel)

    # Apply low-pass filtering to the video
    low = cv2.filter2D(resize,-1,kernel)

    # Apply high-pass filtering to the video
    high = resize - cv2.GaussianBlur(resize, (21, 21), 3)+127

    # Apply histogram equalization
    equalized = cv2.equalizeHist(cv2.cvtColor(resize, cv2.COLOR_BGR2GRAY))

    # Convert the video color back to a 3-channel (RGB) image
    equaliz = cv2.cvtColor(equalized, cv2.COLOR_GRAY2BGR)

    # Stack videos in a 3x2 grid
    combined = np.hstack((low, high, equaliz))

    frames.append(combined)
```

```
# Input video
video1 = VideoFileClip('homework_1_test_video.mp4').fl_image(process_image)
video2 = VideoClip(lambda t: frames[int(t*20)], duration=len(frames)/20.0)

# Make sure the videos have the same duration
duration = min(video1.duration, video2.duration)
clip1 = video1.subclip(0, duration)
clip2 = video2.subclip(0, duration)

# Merge the videos side by side
final_clip = clips_array([[clip1],[clip2]])

# Write the result to a file
final_clip.write_videofile('YoutubeOutput.mp4', fps = 30)
```

Next for the Subtitles and Text To Speech

```python
#Subtitles
subtitles = '''
幾乎整個影片幾乎完全由人工智慧生成,
都是用python程式碼寫的,
視頻來源是在左上角視頻,
中間頂部的影片使用 GAUSSIAN BLUR 進行過濾,
右上角影片使用tensorflow轉換為HSV,
在左下角，影片使用 KERNEL 過濾為 Low-Pass Filtering,
在底部中間，影片被過濾為 High-Pass Filtering,
最後右下角的影片透過 Histogram Equalization 進行了增強.
Thanks for watching! Catch you in Homework 2. Bye-bye!
'''

source_video_filename     = "YoutubeOutput.mp4"
background_music_filename = "calm-background-for-video-121519.mp3"
target_video_with_subtitle= "Final_Output.mp4"

lines = [msg for msg in subtitles.split('\n') if len(msg)>0]
speech= []

# Text To Speech Loop reading every line in the lines
for i,msg in enumerate(lines):
    gttsObj = gTTS(text = msg, lang = 'zh', slow =False)
    gttsObj.save('/content/voice{:04d}.mp3'.format(i))
    speech.append(AudioFileClip('/content/voice{:04d}.mp3'.format(i)))

# Calculate the start and end time of each narration
duration       = np.array([0]+[s.duration for s in speech])
cumduration    = np.cumsum(duration)
total_duration = int(cumduration[-1])+4

# Generate narration subtitles, using msjh font
generator = lambda txt: TextClip(txt, font='/content/msjh.ttc', fontsize = 55, color = 'white')
subtitles = SubtitlesClip([((cumduration[i],cumduration[i+1]),s) for i,s in enumerate(lines)], generator)

# Adjust the video duration so that the video playback time is longer than the entire narration.
clip = VideoFileClip(source_video_filename)
clip = clip.fx(vfx.speedx,clip.duration/total_duration)

# Generate the final video with subtitles.
final_clip = CompositeVideoClip([clip, subtitles.set_pos(('center','bottom'))])

# Adding Background music, length is same as video and turn down the volume.
bgm = AudioFileClip(background_music_filename)
bgaudio = bgm.subclip(bgm.duration-total_duration).volumex(0.2)


# Combine the audio of the background music and the Narrative
voices = concatenate_audioclips(speech)
clip.audio = CompositeAudioClip([bgaudio,voices])
final_clip = final_clip.set_audio(clip.audio)

# Output
final_clip.write_videofile(target_video_with_subtitle)
```

## 結果

Output Video:

https://youtu.be/YmRDsGZIYsg

Python File:

https://github.com/aaronwu2/MachineVision

## 結論

我在做作業時獲得了很多新知識，首先我學會瞭如何使用 moviepy，我認為 moviepy 是一個非常有用的庫，特別是對於圖像/視頻處理，比如調整視頻幀的大小，就像我在代碼中所做的那樣使用我將視訊大小調整為 600 x 500。困難的部分是當我想添加字幕和文字到語音時。 首先，我想使用我們導師提供的一個庫，是 pytts，但是我似乎無法使用它（錯誤），所以我使用了 gTTs，這也是一個文字轉語音庫，而且效果很好。 總的來說。我也覺得 opencv 有滿多各種各樣的 library 我們可以用。我真的很喜歡做這個作業。

## 參考文獻


Font msjh

https://github.com/taveevut/Windows-10-Fonts-Default/blob/master/msjh.ttc

Background Music calm background for video 121519

https://pixabay.com/music/corporate-calm-background-for-video-121519/

Youtube Video 聖稜-雪山的脊樑©

https://www.youtube.com/watch?v=PHqhEgkGfrs&t=6s&ab_channel=%E9%9B%AA%E9%9C%B8%E5%9C%8B%E5%AE%B6%E5%85%AC%E5%9C%92Shei-PaNationalPark

```
!pip install pyttsx3
!pip install moviepy
!pip install gTTS
!pip install yt-dlp
!pip install moviepy

!apt install imagemagick

!apt install libmagick++-dev

!cat /etc/ImageMagick-6/policy.xml | sed 's/none/read,write/g'> /etc/ImageMagick-6/policy.xml
!pip install espeak-py
!wget https://github.com/taveevut/Windows-10-Fonts-Default/raw/master/msjh.ttc
```