

107-2 Introduction to Computer Networks

# Final Project

## 螢幕分享程式

第十六組 蕭如芸、林緯璋、詹欣玥

# 目錄

<b>一、功能簡介</b>	<b>3</b>
<b>二、如何使用</b>	<b>3</b>
<b>三、實作成果</b>	<b>4</b>
<b>四、技術說明</b>	<b>6</b>
<b>五、遇到的問題和解決方法</b>	<b>11</b>
<b>六、Future Work</b>	<b>14</b>
<b>七、Demo Video</b>	<b>14</b>

## 一、功能簡介

「一個有助於線上討論作業、project的影片串流服務。」

1. 傳送螢幕畫面
2. 傳送鍵盤輸入文字
3. 傳送語音轉成的文字
4. 網路feature

計算傳送所需的時間

若時間超過設定的閾值，就降低下一次傳送的圖片畫質

## 二、如何使用

Server端：python server.py

Client端： python client.py

*On Mac OS 10.14:*

Server端：python server.py

\$OBJC\_DISABLE\_INITIALIZE\_FORK\_SAFETY=YES

Client端： python client.py

\$OBJC\_DISABLE\_INITIALIZE\_FORK\_SAFETY=YES

\* Server端需固定IP，並將client端46行、server端129行 HOST 對應之IP改成Server所在IP

\* 由於使用multiprocessing，在Mac OS 10.14以後必須手動輸入

OBJC\_DISABLE\_INITIALIZE\_FORK\_SAFETY=YES指令，將限制multiprocessing的設定關掉，才能順利執行。也可以在.bash\_profile中export

OBJC\_DISABLE\_INITIALIZE\_FORK\_SAFETY=YES設定。

### 三、實作成果

#### Server端

The screenshot shows a Mac OS X desktop environment. In the center is a terminal window titled 'demo2\_server' with the following code:

```

11 print('\n=====')
12 print('= Intro to Computer Network =')
13 print('=====\\n')
14
15 for i in range(10):
16     for j in range(i):
17         print('*', end='')
18     print()
19
20 #####
21 # Type here #
22 #####
23 ...
24
25

```

To the left of the terminal, a processing application window displays a video feed from a camera. The processing code in the sketch is:

```

Say something
processing
除了語音辨識的套件之外都

Say something
processing
Please say it again

Say something
processing
好想吃拉麵

Say something

```

On the right side of the desktop, there is a vertical stack of files and folders:

- CVhw2
- pt\_
- IEEE xplor...ensor.pdf
- recording

This screenshot shows a similar setup to the previous one, but the processing application window now displays the text 'Please say it again'.

The terminal window contains the same code as before:

```

1
2
3
4
5 #####
6 # This is a code for demo #
7 #####
8
9
10
11 print('\n=====')
12 print('= Intro to Computer Network =')
13 print('=====\\n')
14
15 for i in range(10):
16     for j in range(i):
17         print('*', end='')
18     print()
19
20 #####
21 # Type here #
22 #####
23 ...
24
25

```

The processing application window now shows the text 'Please say it again' and the previous speech recognition results:

```

Say something
processing
今天真是一个好日子

Say something
processing
電腦網路導論

Say something
processing
Please say it again

Say something

```

A small 'pygame window' is visible at the bottom of the processing window, with the text 'Type at line 24'.

右邊的白色視窗顯示的是語音辨識的結果。下方的白色視窗顯示的是server鍵盤輸入的訊息。

## Client端

畫面顯示的是server端的螢幕畫面，畫面上方的字是server鍵盤輸入的訊息，畫面下方的字則是server語音輸入的訊息。

This is a demo code

```

16     for j in range(i):
17         print('*', end='')
18     print()
19
20     #####
21     # Type here #
22     #####
23     ...
24
25     ...
26

```

拉麵超好吃的

(x=338, y=254) ~ R:28 G:28 B:28

Type at line 24

```

13     print('=====\\n')
14
15     for i in range(10):
16         for j in range(i):
17             print('*', end='')
18         print()
19
20         #####
21         # Type here #
22         #####
23         ...
24     testing ...
25     typing here ...
26     ...
27

```

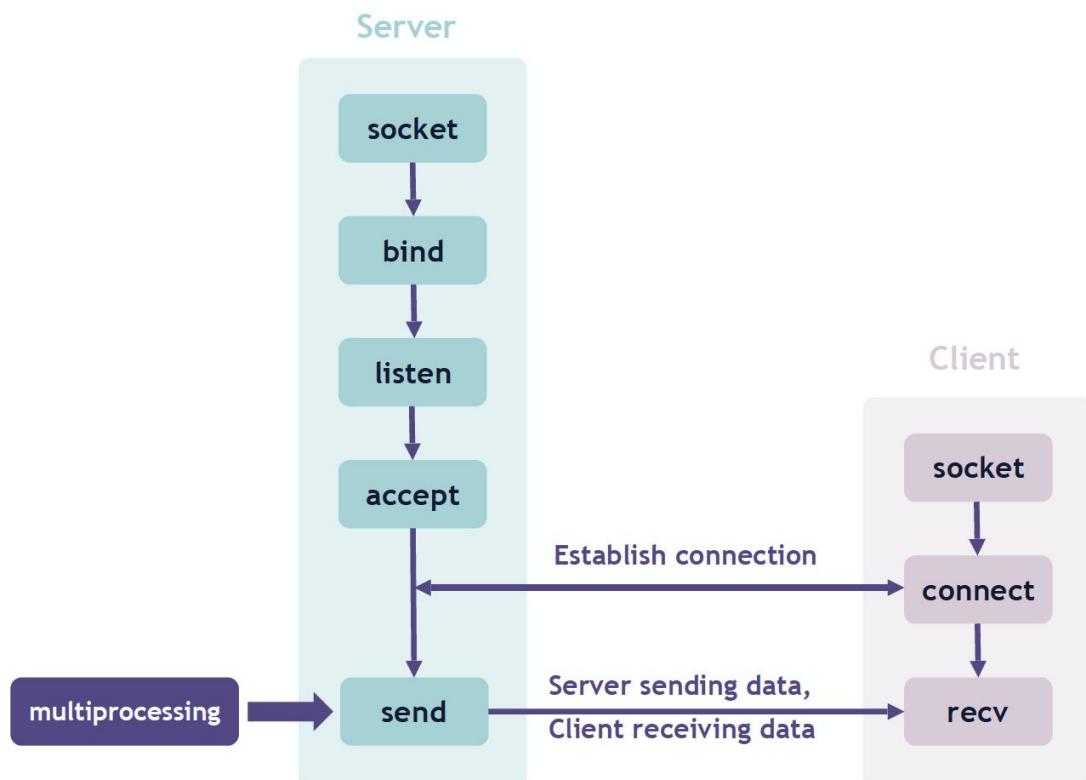
但我們還是做出來了

(x=378, y=233) ~ R:28 G:28 B:28

## 四、技術說明

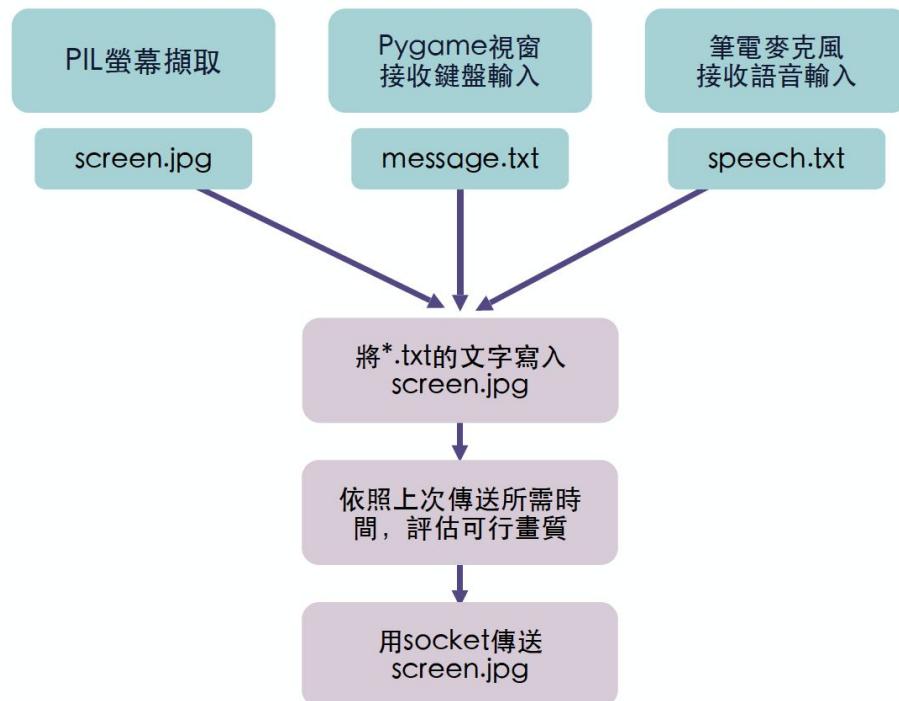
### 整體概述

- 程式語言：Python 3.6
- 使用套件
  - numpy == 1.16.4
  - opencv-python == 4.1.0.25
  - Pillow == 6.0.0
  - PyAudio == 0.2.11
  - pygame == 1.9.6
  - SpeechRecognition == 3.8.1
- 使用Python內建的multiprocessing同時執行多個程序，達到同時傳送螢幕畫面和文字訊息的目的。
- 利用TCP Socket傳送資料，目的是以python從最底層開始實作傳送畫面。
- 流程圖



## Server端

- 流程圖



- 螢幕畫面
  - 用Pillow套件擷取螢幕畫面
  - 確認是否有文字訊息要一併傳送，若有，將訊息寫在螢幕截圖上
  - 將螢幕截圖儲存成screen.jpg
  - 將screen.jpg透過socket傳送給client
  - 計算傳送所需的時間，若時間超過設定的閾值，就降低下一次傳送的圖片畫質
- 文字訊息
  - 鍵盤輸入
    - 用pygame套件開一個專門的視窗讀取鍵盤輸入，儲存輸入的訊息於message.txt，顯示於螢幕截圖上方
  - 語音輸入
    - 將筆電麥克風接收到的聲音輸入用SpeechRecognition套件做語音辨識，儲存辨識後的文字訊息於speech.txt，顯示於螢幕截圖下方

- Multiprocessing同時處理螢幕畫面與文字訊息
- 程式碼及註解

```

import socket
import cv2
import time
import speech_recognition
import numpy as np
from PIL import Image, ImageGrab, ImageFont, ImageDraw
from multiprocessing import Process
import pygame
from pygame.locals import *
import os

def video():
    font = ImageFont.truetype('SimSun-Bold.ttf', 28)

    # 傳送螢幕畫面
    def sndscreen():
        resolution = 20      # 影像畫質

        while(True):
            # 開啟要顯示的鍵盤輸入訊息
            try:
                f = open('message.txt')
                message = f.read()
                f.close()
            except:
                message = ""
            # 開啟要顯示的語音輸入訊息
            try:
                f = open('speech.txt')
                speech = f.read()
                f.close()
            except:
                speech = ""

            # 螢幕截圖
            screen = ImageGrab.grab(bbox=(640, 400, 1920, 1200)).resize((640,
400))

            # 在圖片上加入語音輸入的訊息 (speech)
            draw = ImageDraw.Draw(screen)
            draw.text((10, 350), speech, font = font, fill = (51, 153, 255,
1))

            screen = cv2.cvtColor(np.array(screen), cv2.COLOR_BGR2RGB)
            # 在圖片上加入鍵盤輸入的訊息 (message)
            cv2.putText(screen, message, (10, 40), cv2.FONT_HERSHEY_COMPLEX,
0.75, (255,153,51), 2, cv2.LINE_AA)
            # 將圖片儲存為'screen.jpg'
            cv2.imwrite('screen.jpg', screen, [cv2.IMWRITE_JPEG_QUALITY,
resolution])

    try:
        start = time.time()
        # 將'screen.jpg'傳送給client
        with open('screen.jpg', 'rb') as f:

```

```

        client.sendall(f.read())

        # 若圖片傳送時間超過1e-3秒，降低圖片畫質 # 網路Feature
        sample = time.time() - start
        if sample > 1e-3:
            resolution = 10
        else:
            resolution = 20

    except:
        pass

    sndscreen()

# 鍵盤輸入訊息
def type():
    temp = ''
    # 開新的視窗
    pygame.init()
    screen = pygame.display.set_mode((480, 120))
    font = pygame.font.Font(None, 30)

    while True:
        # 讀取鍵盤輸入，將結果寫入'message.txt'
        for evt in pygame.event.get():
            if evt.type == KEYDOWN:
                if evt.key == K_SPACE:
                    temp += ' '
                elif evt.key == K_BACKSPACE:
                    temp = temp[:-1]
                elif evt.key == K_RETURN:
                    f = open('message.txt', 'w+')
                    f.write(temp)
                    f.close()
                    temp = ''
                else:
                    temp += evt.unicode
            elif evt.type == QUIT:
                return

        # 將輸入的訊息顯示在視窗中
        screen.fill((255, 255, 255))
        block = font.render(temp, True, (0, 0, 0))
        rect = block.get_rect()
        rect.center = screen.get_rect().center
        screen.blit(block, rect)
        pygame.display.flip()

# 語音輸入訊息
def recognition():
    r = speech_recognition.Recognizer()
    with speech_recognition.Microphone() as source:
        print('\n')
        while 1:
            # 讀取麥克風輸入
            r.adjust_for_ambient_noise(source)
            print("\n-----")
            print("Say something")

```

```

audio=r.listen(source)
try:
    # 執行語音辨識，將結果寫入'speech.txt'
    print("Processing")
    global a
    a = r.recognize_google(audio, language='zh-TW') #zh-CN
    f = open('speech.txt', 'w+')
    f.write(a)
    f.close()
    print(a)
except speech_recognition.UnknownValueError:
    # 無法辨識
    print("Please say it again")
    pass

if __name__ == '__main__':
    HOST, PORT = "127.0.0.1", 61677
    # HOST, PORT = "140.112.226.236", 61677
    # 建立和client互連的socket
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    s.bind((HOST, PORT))
    print('waiting...')
    s.listen(1)

    client, address = s.accept()
    print('%s connected' % str(address))

    if os.path.exists('message.txt'):
        os.remove('message.txt')
    if os.path.exists('speech.txt'):
        os.remove('speech.txt')

    # 開始多個Process
    Process(target = type).start()
    Process(target = recognition).start()
    Process(target = video).start()

```

## Client端

- 接收server傳送的資料，儲存成save.jpg
- 將save.jpg顯示在螢幕上
- 程式碼及註解

```

import cv2
import socket
import time
from PIL import Image
import numpy as np
from random import randint
from multiprocessing import Process

MAX_BUFFER_SIZE = 1000000000
def video():
    # 接收螢幕畫面

```

```

def recscreen():
    while 1:
        # 用多個buffer接收一張圖片
        data = s.recv(MAX_BUFFER_SIZE)
        for i in range(10):
            data += s.recv(MAX_BUFFER_SIZE)
        try:
            # 將收到的資料寫入'save.jpg'
            with open('save.jpg', 'wb') as f:
                f.write(data)
        try:
            # 顯示'save.jpg' (server的螢幕畫面)
            global frame
            frame = cv2.imread('save.jpg')
            cv2.imshow('Server', frame)

            if cv2.waitKey(100) == ord('q'):
                cv2.destroyAllWindows()
                break

        except:
            pass

        except:
            pass

    recscreen()

if __name__ == "__main__":
    HOST, PORT = "127.0.0.1", 61677
    # HOST, PORT = "140.112.226.236", 61677

    # 建立和server互連的socket
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((HOST, PORT))

    print('connected to %s' % HOST)

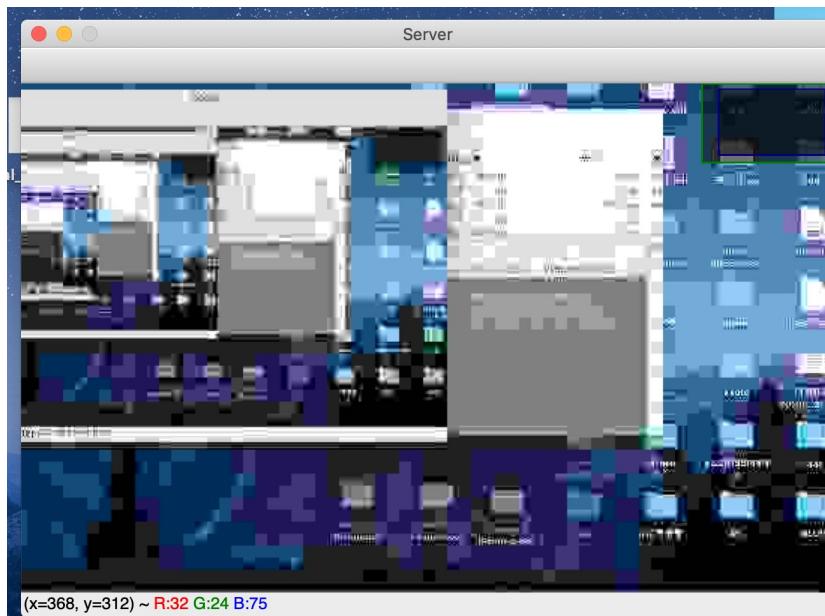
    # 開始Process
    Process(target = video).start()

```

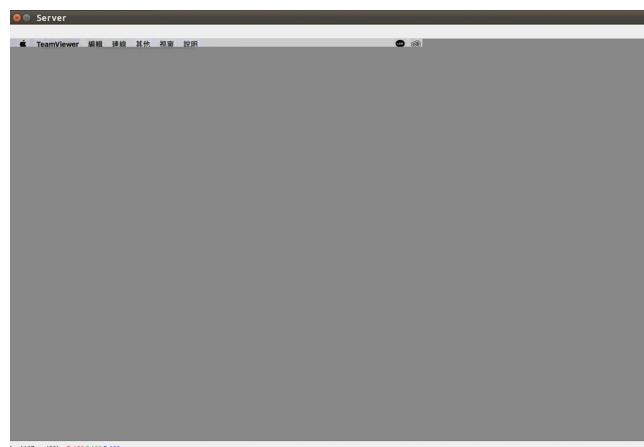
## 五、遇到的問題和解決方法

- 不同電腦之間差異極大
  - 問題描述：
    - 我們組員有兩人是用macbook pro 2017/2018，一個人是macbook pro 2014

- 用2014無論當client還是server, 效果都遠比另外兩台電腦差

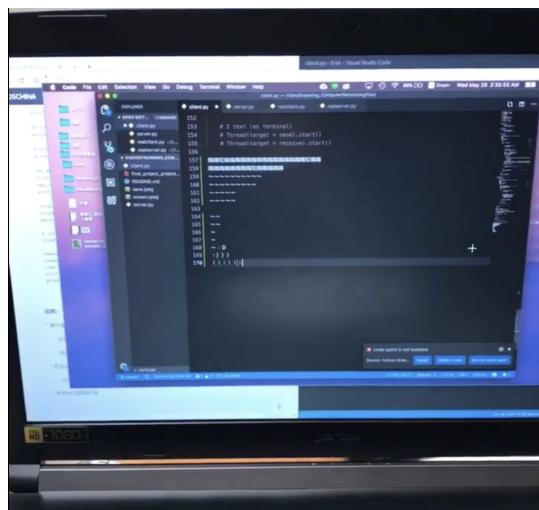


- 猜測原因 :
  - 改版後整體效能大幅提升。
  - 一開始使用的python套件都是在新電腦上運行較順。（有試過使用其它套件的話，2014版可以跑得順暢很多，但後來還是採用現行套件）
- 解決方法 :
  - 在舊電腦上跑的時候盡可能再壓低畫質等等，主要還是以較新的電腦測試。
- 網路好壞影響極大
  - 問題描述 :
    - 某一種狀況非常流暢，但其他人的電腦當client時，整個視窗只能出現約1/30的畫面。



- 可順利運行的環境描述：

- 套件：截圖部分改用wxPython
- server: 家裡一般router更改設定以固定ip
- client: 宿舍網路線
  - 高畫質、打字完全像是在本機打字般流暢
  - 中間粉紫色是遠端筆電



- 猜測原因：

- 網路線比使用wifi快很多。
- 可流暢運行的client方電腦有GTX1060顯示卡，是比較大台的筆電。

- 解決方法：

- 還是以一般client能運行為主，將這個case視為特例。

- 圖片傳不完，畫面下半部灰色

- 問題描述：

- 在server端存下的圖片是完整的，但是傳到client端後，就只剩半張圖（灰色部分會忽大忽小，很少能顯示出完整圖片）



- 在本地(127.0.0.1)上面測的時候很順暢，但跨網域就不行了
- 猜測原因：
  - 因為在127.0.0.1上面測很順暢，猜測跟網速有很大的關係。
  - 猜測是因為package drop嚴重的話，在一個recv接收完一張圖前，就已經停止接收、並顯示圖片。
- 解決方法：
  - 在client接收的地方，增加recv的數量。

```
# 用多個buffer接收一張圖片
data = s.recv(MAX_BUFFER_SIZE)
for i in range(10):
    data += s.recv(MAX_BUFFER_SIZE)
```

  - 大約每增加3個recv就會有顯著的改善。
  - 進行實驗發現，似乎跟MAX\_BUFFER\_SIZE的大小關係較小，跟BUFFER數量較有關。

## 六、Future Work

- 增加多使用者情境：用Server來分配Port，讓不同Client可以互傳對方資料。
- 試著用更高等的方式實作傳圖片，來增加視訊穩定程度。
- 增加「開始」與「結束」的圖形化按鈕。

## 七、Demo Video

[影片連結1](#)

[影片連結2](#)