視訊聊天軟體

B05902001 資工二 廖彥綸
B03505052 工海四 李吉昌
B03505040 工海四 林後維
B03505039 工海四 李治衡

July 2, 2018

Contents

1 大綱

伺服器端開啟後利用 select 函數不斷聽取客戶端的請求，一開始等待序列為空。有新的客戶端在註冊帳密成功登入後連線，若等待序列為空則將他放入等待序列中，直到有另一個客戶端進入連線。若客戶端連線時等待序列不為空，則會和配對等待序列中的客戶配對。伺服器會告知對方的 IP 位置和可使用的 port。之後伺服器即結束連

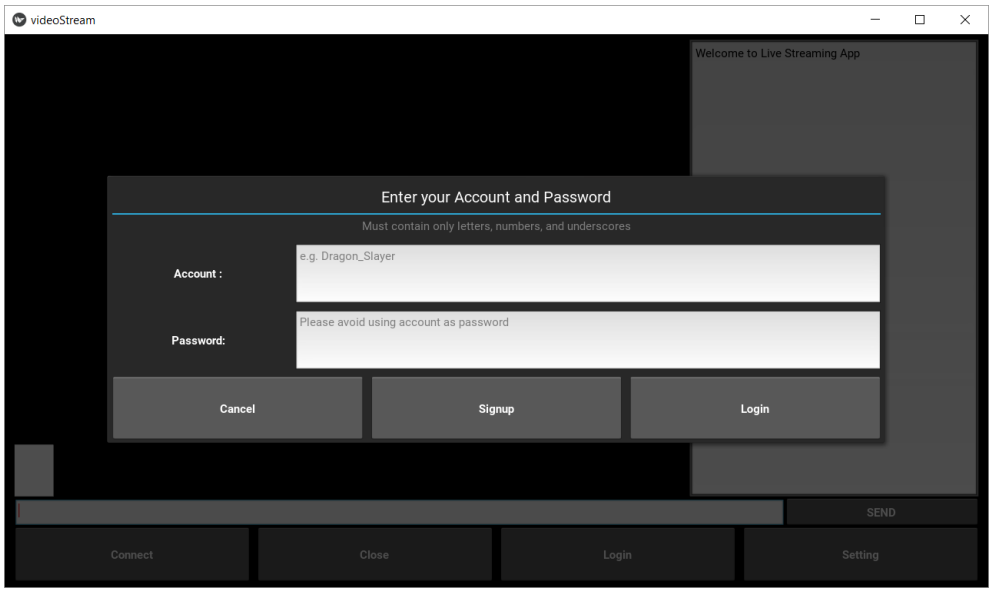
線，改由兩個客戶端以視訊方式連線。伺服器等待下一個客戶進行服務。兩個客戶端的配對除了視訊外提供人臉辨識和文字輸入功能。

2 實作成果

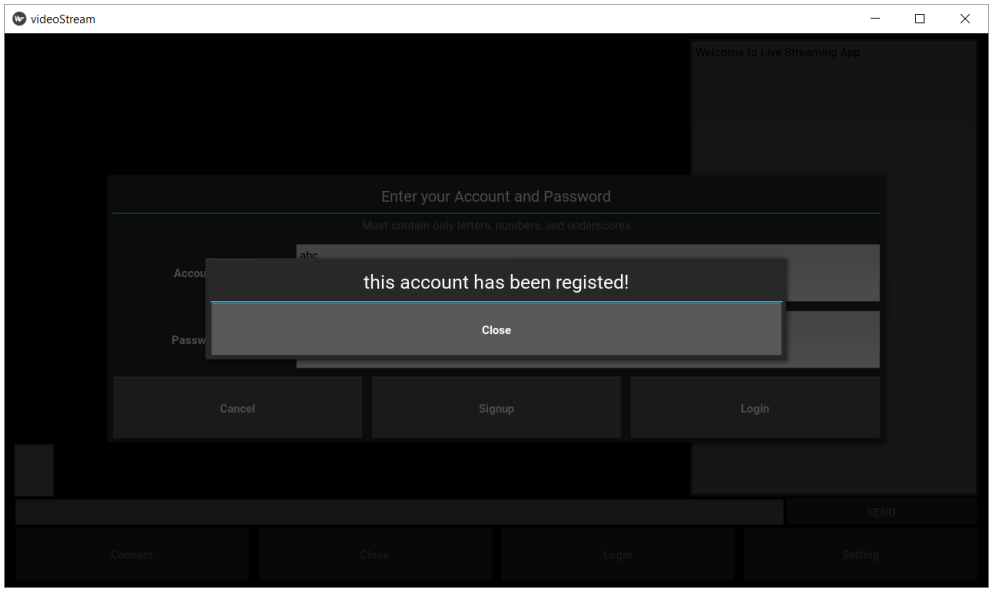
使用者端的執行主要可以分成幾個介面來解釋：

1. 登入介面：

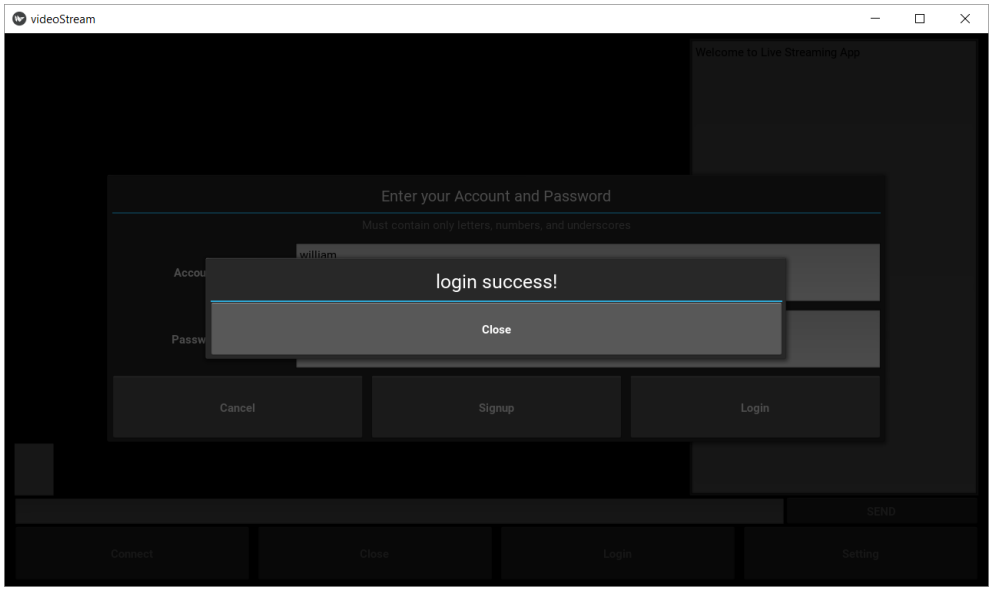
使用者可以此申請新的帳號密碼，或者輸入以前使用過的帳號密碼，此時輸入的帳號將會在接下來與其他使用者連線時顯示在對話框中。



申請帳號時須注意系統只接受英文字母、數字以及底線 (_) 組成的字串為帳號，另外如果該帳號已被其他使用者申請過，那麼就會出現以下錯誤。使用者只能再申請另外一個帳戶。



在登入時也是一樣，server 會回傳是否登入成功，如果登入失敗則會返回登入介面，如果登入成功，則會直接離開登入介面進到主介面。

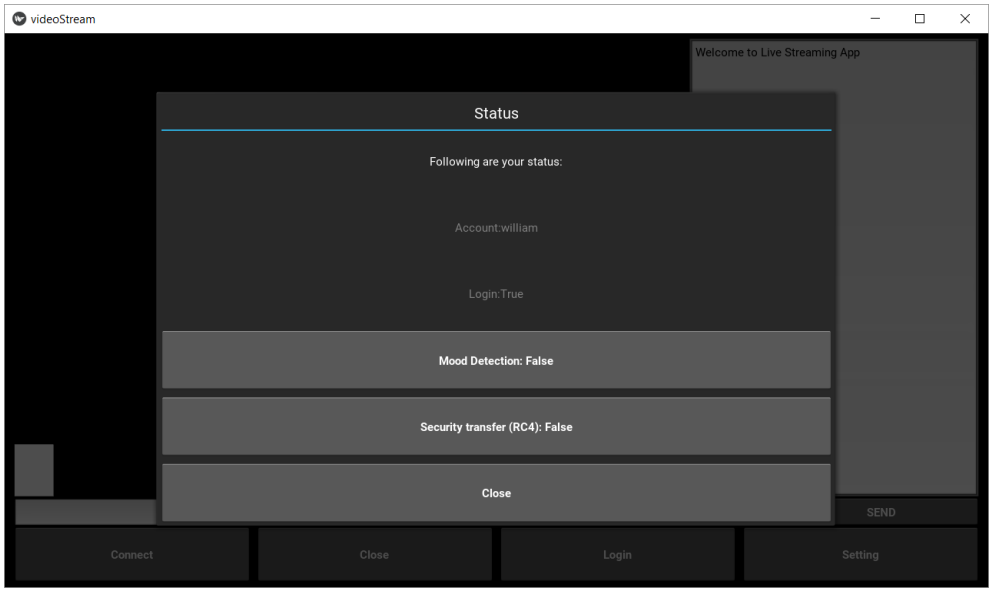


2. 設定介面：

在這裡我們可以看到自己的狀態以及可以操作的一些設定，首先我們可以看到登入時使用的帳號以及登入狀態，如果一開始直接退出登入介面到主介面的話，再進入到設定介面時就可以看到自己的登入狀態為 false。而使用者此時是沒辦法與其他使用者進行連線的。Mood Detection 和 Security transfer 為使用者可以選擇的功能，兩者的初始值皆為 False。

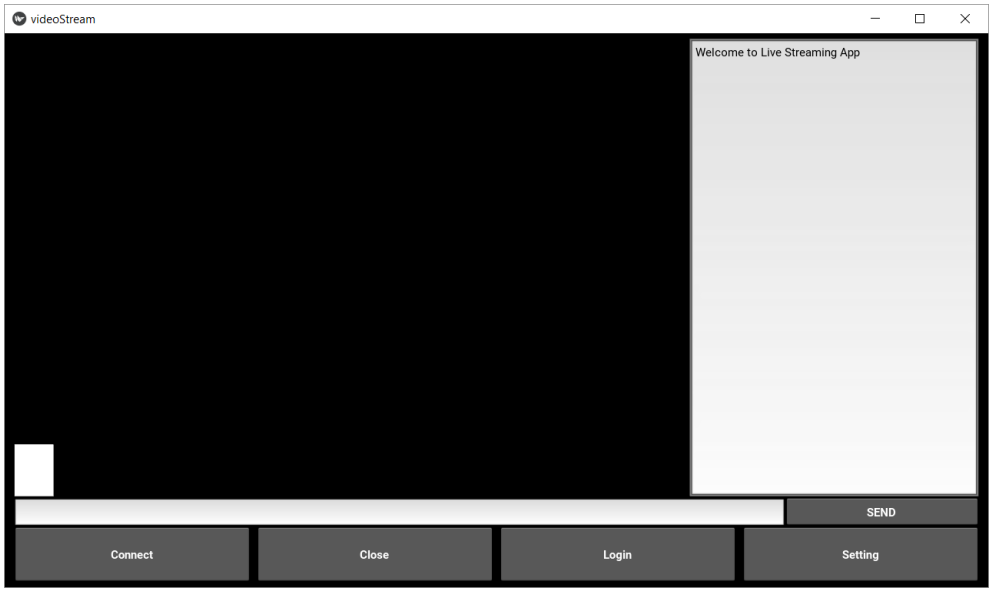
Mood detection 的功能是人臉辨識，會將傳送影像中的人臉標示出來，以方便日後如果要繼續開發時，可以直接截錄使用者的表情，並將它輸入到情緒判斷的 model 中。

Security Transfer (RC4) 的功能是保護使用者的肖像權，將傳輸的影像做 RC4 加密，因為 RC4 的特性，其執行加解密速度較快，可以用在即時的影像傳輸上，讓使用者視訊時不用擔心被網路上其他人擷取影像。(這個功能只能需要雙方同時在連線前啟動才會執行)。



3. 主介面：

也是使用者在登入後最常使用的介面，最下方有四個功能，分別是連線、關閉程式、回到登入介面、進入設定介面。使用者可以在做好登入以及設定好，按下左下角的 Connect 即會在 P2P_server 上等待他人連線，如果湊滿兩個人就可以做視訊連線。



在兩人連線成功後，會呈現如下圖展示的結果，此時可以連線看到彼此的影像，也可以在下方對話欄中輸入字串 (Enter 或右方 SEND 紐皆可)，雙方聊天的內容會展示在右方的對話框中。如果此時使用者開啟 Mood detection 並且程式有偵測到人臉的話，會使用藍色的方框將其標示出來。

3 程式語言、其他套件與支援環境

3.1 開發環境

Window 10, python3.6.4

3.2 第三方套件

1.Numpy-支援各種資料型態的轉換
2.CV2 -支援前鏡頭影像的索取以及影像辨識的功能
3.Kivy -支援使用者介面 (GUI) 的架構

4 執行環境架設

將以下文字打入檔案中
Kivy==1.10.0
Kivy-examples==1.10.0
Kivy-Garden==0.1.4
kivy.deps.angle==0.1.6
kivy.deps.glew==0.1.9
kivy.deps.gstreamer==0.1.12
kivy.deps.sdl2==0.1.17
opencv-python==3.4.1.15
numpy==1.14.3
之後使用 pip install -r + 檔名安裝

5 執行方式

請先使用 python3.6 執行 P2P_server.py 和 account_server.py 兩個 server 檔，兩個 server 需在相同電腦上執行，並擁有相同的 IP address，將前述的 IP address 輸入到 GUI.py 中第 32 行的 ipAddress 宣告中，接下來只要在同個子網域下的任意電腦上執行 GUI.py 即可。

(開發中) 在 window 上有 pyinstaller 可以將 python 檔轉換成 exe 檔，並產生該 exe 檔執行時所需的各種 library 在同個資料夾內，執行時只要直接執行 exe 檔即可，不用再額外下載套件

6 核心通訊方式

除了前述的 recvall 外，我們定義了 customRecv 和 customSend 兩個特殊的傳輸方式，他們倆皆建立在 TCP 連線上。其功能主要是傳一個我們自定義的資料模式，該資料的格式如下圖。

該資料格式可以分成三段，第一段是一個 byte(0-255) 決定了我們傳輸的內容是什麼 (mode)，譬如說傳影像的封包 mode = 50，所以我們只要收到第一個 byte 為 50 的資料，就一定是要傳影像用的，如果程式此時並沒有預期要收到影像的話，那麼就應該將那個資料丟掉。

第二個是 16bytes 長的 Length of payload，主要是將我們主要要傳的資料長度先告訴接收端，這樣接收端才可以正確的接收該筆資料，不會像直接用 recv(1024)，會將後面的幾筆資料一起讀進來，導致程式後續的執行容易出現錯誤。第三個則是我們

宣告好長度的 payload，而用 16 個 bytes 來宣告的長度 (最多 65535 Bytes) 在我們的程式中是很難用完的，因為我們傳的影像都是有經過壓縮的，其大小都在 50000 Bytes 以內，而除了影像之外，我們傳的資料通常只是一小段文字而已，所以這個資料格式對我們十分好用。

customRecv 是架設在 recvall 上的功能，因為我們已經知道我們自定義的資料型態中，前兩個是已經確定的長度 (分別是 1 個 byte 和 16 個 bytes)，所以我們可以先讀取這些資料，在彈性化的讀取第三段的資料，就可以達到我們想要的結果。customSend 則是相反，它先將處理成 bytes 格式的資料讀進來，確定它的長度後，先傳 1 個 byte 的 mode byte，再傳 16 個 bytes 的長度，最後再將主體資料傳出去。

```python
def customRecv(sock):
    mode = recvall(sock, 1)
    length = recvall(sock, 16)
    stringData = recvall(sock, int(length))
    mode = int.from_bytes(mode, byteorder='little')
    return mode, stringData


def customSend(sock, mode, data):
    sock.send(bytes([mode]))
    length = str(len(data)).ljust(16).encode()
    sock.send(length)
    sock.send(data)
```

7 程式碼與輔助套件解析

7.1 伺服器端的通訊

首先向介紹伺服器端程式的運作和流程，根據功能性不同伺服器的程式可以分為幾個部分，第一個是宣告的部分。宣告部分負責處理一些通用參數的宣告。

```python
server = socket.socket()
server_IP = '127.0.0.1'
server_PORT = 3453
server.bind((server_IP, server_PORT))
server.listen(5)
r_list = [server,]
count = 0
wait_queue = queue.Queue()
addr_queue = queue.Queue()
append_port = 7777
```

前五行宣告主機的 IP 位置和 port(伺服器的 IP 使用 127.0.0.1)，r_list 則是使用 select 函式需要的參數，select 在伺服器端 socket 上可以用來檢查是否有新連線，將其他連線加入監聽後，還可以檢查是否有資料寫入或斷線，一開始只監聽伺服器本身。wait_queue 和 addr_queue 用來紀錄目前等待中客戶的資訊，用 queue 紀錄更有利於之後新增功能的可編輯性。

```
readarray, writearray, errorarray = select.select(r_list,[],[],10)
```

上列程式碼是 python 使用 select 函數的用法，可以一次監聽三種狀態，讀取、寫入和錯誤，在這個專題中主要用來判斷是否有新檔案，所以用的是讀取功能，另外兩項未使用到則不選擇監聽。

```
for fd in readarray:
    if fd == server: #新連線
        print(fd)
        clientfd , addr = fd.accept()
        print(addr)
        listx = list(addr)
        listx.pop()
        listx.append(append_port)
        append_port += 1
        addr = tuple(listx)
```

之後需要處理新連線 (客戶) 的處理問題，和一般的 TCP 通訊一樣，伺服器先 accept 要求的連線，之後給予他一個新的 port，由於接受到的 addr 是 tuple 屬性 python 無法直接做更改，所以先轉為 list，更換 port 後再轉成 tuple。

```
if wait_queue.empty():  #序列為空時，直接放入資料
    wait_queue.put(clientfd)
    addr_queue.put(addr)
    r_list.append(clientfd)
else:   #序列不為空時，配對處理
    connfd = wait_queue.get()
    clientfd.sendall(b"find pair")
    connfd.sendall(b"find pair")
    connaddr = addr_queue.get()

    conn_data_string = pickle.dumps(connaddr)
    data_string = pickle.dumps(addr)

    clientfd.sendall(conn_data_string)
    connfd.sendall(data_string)
    time.sleep(1)
    clientfd.sendall(data_string)
    connfd.sendall(conn_data_string)
    r_list.pop()
    clientfd.close()
    connfd.close()
```

當客戶端的連線成功後，系統判斷他是否需要等待其他人。需要等待時儲存他的資料同時監聽 (原因後續說明)，如果是一個新配對系統會互傳他和另一個客戶的 IP 和 port，供他們互相連線，tuple 的格式需要經過 pickle 才允許傳送。結束後，server 則完成他的工作，並結束監聽。

```
else :
    print("one client leave")
    wait_queue.get()
    addr_queue.get()
    r_list.pop()
```

最後處理再配對成功前客戶就離開的情況，由於伺服器端處理的工作較少，可以只用簡單幾行程式就完成。伺服器監聽客戶的 socket 時客戶端不會主動和傳遞資料到伺服器 (除了有其他連線的溝通外)，客戶端的主動聯繫只會是斷線信號的狀況，伺服器接收到斷線通知後還原序列的狀態並取消監聽，以完整保護程式。以下是運行時的虛擬碼，只包含傳輸部分，其他附加功能並未寫入其中。附檔包含完成程式碼。

```
server <- TCP_socket
server <- bind( IP , port )
server <- listen
r_list <- server
while 1:
        readarray <- select
        for fd in readarray:
                if fd == server
                        client , addr = fd.accept()
                        addr <- new port
                        if wait_queue.empty()
                                wait_queue.put(client)
                                addr_queue.put(addr)
                                r_list.append(client)
                        else:
                                conn <- wait_queue.get()
                                connaddr = addr_queue.get()
                                client <- send conn , connaddr
                                conn <- client , addr
                                r_list.pop()
                                wait_queue , addr_queue <- pop
                                client , conn <- close
                else:
                        wait_queue.get()
                        addr_queue.get()
                        r_list.pop()
server.close()
```

7.2 客戶端的通訊

客戶端需要負責 cv2 影像的傳輸，過程較為精細，所以定義一個函式 recvall 用於接收指定長度內的訊息[1]

[1] 參考網頁：http://blog.maxkit.com.tw/2017/07/socket-opencv-client.html

```
def recvall(sock, count):
    buf = b''
    while count:
        newbuf = sock.recv(count)
        if not newbuf: return None
        buf += newbuf
        count -= len(newbuf)
    return buf
```

此函式可以保證接收固定的資料，對於傳輸上幫助很大。

```
capture = cv2.VideoCapture(0)
server_IP = '127.0.0.1'
server_port = 3453
s = socket.socket()
s.connect((server_IP,server_port))  # connect to sever
print(s)
print("serching !!!\n")
msg = s.recv(1024)  # find pair
print(msg.decode())
conn_data = s.recv(1024)
conn_addr = pickle.loads(conn_data)
print(conn_addr)
conn_addr_ip = conn_addr[0]
conn_addr_port = conn_addr[1]
data = s.recv(1024)
addr = pickle.loads(data)
print(addr)
s.close()
```

客戶端先開啟視訊鏡頭，之後連線到伺服器，等待回應後會得到連接對象和自己的 IP 和 port。

```
_thread.start_new_thread( sender, (conn_addr,) )
```

在 P2P 連線的過程中客戶端需要同時負責接收和傳送資料 (視訊)，而由於網路傳輸的特性，彼此之間並不一定存在先後的關係。使用多線程處理這個問題是好辦法，可以將工作區分離、同時執行工作並提升效率。

```
def sender(conn_addr):
    time.sleep(1)
    sock = socket.socket()
    sock.connect(conn_addr)
    print("prepare to send\n")
    encode_param=[int(cv2.IMWRITE_JPEG_QUALITY),90]
    ret, frame = capture.read()
    while ret:
        result, imgencode = cv2.imencode('.jpg', frame, encode_param)
        data = numpy.array(imgencode)
        stringData = data.tostring()
        new_send = str(len(stringData)).ljust(16).encode()
        sock.send( new_send);
        sock.send( stringData );
        ret, frame = capture.read()
        cv2.waitKey(1)
    sock.close()
    cv2.destroyAllWindows()
```

這是負責傳輸的部分[2]。首先，和對方的接收伺服器連線，連線前先等待 1 秒以保證伺服器在連線前建立，cv2 的是頻格式需要透過 numpy 轉成易於傳送的形式，之後不斷地獲取視頻，轉成 jpg 檔之後傳送。

```
receiver = socket.socket()
print("build own receiver")
receiver.bind(addr)
receiver.listen(5)
sock , addr = receiver.accept()
while True:
    print(receiver)
    length = recvall(sock,16)
    stringData = recvall(sock, int(length))
    data = numpy.fromstring(stringData, dtype='uint8')
    decimg=cv2.imdecode(data,1)
    cv2.imshow('CLIENT2',decimg)
    cv2.waitKey(1)
cv2.destroyAllWindows()
print("socket close")
s.close()
```

最後是接收端的部分，接收端建立 socket 的伺服器，用上述的 recvall 函式一次性獲取所有需要的資料，將資料用 numpy 轉回原本的格式後進行之後的處理或直接用 cv2 的播放系統放送對方的視頻。以下是客戶端的虛擬碼，只包含傳輸部分，其他附加功能並未寫入其中。附檔有完成程式碼。sender 和 recvall 函式直接使用上述的定義。

```
cv2.VideoCapture(0)
s <- connect(serverIP , severPort)
receive other side connaddr own addr
s.close()
new thread = sender(connaddr)
recevier <- TCP_socket
recevier <- bind( IP , port )
recevier <- listen
sock , s_addr <- recevier.accept()
while 1 :
```

[2]參考網頁：http://blog.maxkit.com.tw/2017/07/socket-opencv-client.html

```
        length <- recvall(sock,16)
        stringData <-  recvall(sock, int(length))
        data = numpy.fromstring(stringData, dtype='uint8')
        # show image 、 other handle
cv2.destroyAllWindows()
```

7.3 其他重要套件及程式

Face_Detection.py：用 cv2 判讀輸入的影像，如果有抓到人臉的話就將其標示出來並輸出有標示的圖片，如果沒有出現人臉，則將原圖輸出。

```
def face_detect(img):
    face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_alt.xml")
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray,1.1,5,cv2.CASCADE_SCALE_IMAGE,(50,50),(100,100))
    if(len(faces) > 0):
        print("CATCH!")
    else :
        print("NOT YET!")
    for faceRect in faces:
        x,y,w,h = faceRect
        color=[255,0,0]
        img[y:y+5,x:x+w]=color
        img[y+h:y+h+5,x:x+w]=color
        img[y:y+h,x:x+5]=color
        img[y:y+h,x+w:x+w+5]=color
        cv2.rectangle(img.astype(np.int32),(x,y),(x+w,y+h),(255,0,0),2,8,0)

        roi_gray = gray[y:y+h,x:x+w]
        roi_color = img[y:y+h,x:x+w]
    return img
```

RC4.py：實作 RC4 的加解密，雙方使用者要先用其他管道得到相同的 key.txt，此程式會用 key.txt 為金鑰對資料進行加密，接收端只要用相同的金鑰再對密文做一次加密，就可以還原成原文

```
def crypt(PlainBytes, KeyBytes):
    cipherList = []
    keyLen = len(KeyBytes)
    plainLen = len(PlainBytes)
    S = np.arange(256)
    j = 0
    for i in range(256):
        j = (j + S[i] + KeyBytes[i % keyLen]) % 256
        S[i], S[j] = S[j], S[i]
    i = 0
    j = 0
    for m in range(plainLen):
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        k = S[(S[i] + S[j]) % 256]
        cipherList.append(k ^ PlainBytes[m])
    l=len(cipherList)
    return np.array(cipherList,dtype=uint8).reshape(l,1)
```

Account_server.py：主要負責使用者輸出的帳密申請，其中用到 sqlite 做為資料庫來儲存帳密，更值得注意的是我們儲存的帳密是有特別加密過的，因此如果哪天我們的 server 被駭入導致資料庫外流的話，對使用者而言，因為我們儲存的是加鹽資料(salted)，所以駭客無法還原出原始的帳號密碼。

```
conn = sqlite3.connect('CNVIEWER.db')
```

```
while(True):
    client, address = s.accept()
    print(str(address)+" connected")
    try:
        mode, data = customRecv(client)

        if (mode == 5): #SIGNUP
            order = data.decode().split()
            print(order)
            account = order[0]
            password = order[1]
            if (LogIn(account, password, conn)):
                customSend(7,b"signup success!",client)
                    # client.close()
            else:
                customSend(7,b"this account has been registed!",client)
        elif (mode == 6): #LOGIN
            order = data.decode().split()
            print(order)
            account = order[0]
            password = order[1]
            if (SignIn(account, password, conn)):
                customSend(7,b"login success!",client)
                    # client.close()
```

Sound.py 這部分並沒有實作，不過我們已經可以讓程式發出提示音，到未來應該也可以讓視訊可以有聲音的資料傳輸。
GUI.py 這是我們使用者主要執行的程式，因為要搭配 GUI 套件 (Kivy) 的各種限制，所以實做有點麻煩，基本上 main 這個 class 中的每個 function 分別是對應到一個主介面中的一個功能 (像是登入介面的三個按鈕，分別就對應到 closepopup、sendsignup 和 sendlogin 三個 function)，而 TargetCamera 這個 class 則主要是負責雙方的影像以及文字傳輸，由於 kivy 的限制，主介面中的影像一定要用一個 class 來負責，並在該 Object 中執行 blit_buffer 才可以更新圖片。而 Kivy 的最大一個好用的點是它的 Clock.schedule_interval，透過這個指令，它可以達到半個 multithread 的功能，它會自己計時並執行一些功能，像是 blit_buffer 就可以搭配這個指令，使程式每 0.1 秒就檢查 socket 的 receive buffer 並執行更新影像的動作。

8 實作上的困難和解決辦法

1. 分工之後的成品結合問題：

期末報告份量較大，每個組員分工完成一部分的報告，但在結合時出現了很多問題。像是原本伺服器端的工作是最為一個溝通橋梁，連接並同時配對客戶端的同時互傳配對完成客戶的視頻訊息 (使用 select 函數)。伺服器端的工作量比原本還巨大很多，

在更多客戶連線時很可能產生嚴重的延遲。但這和實際的情形是比較符合的，客戶端即使正在和其他客戶端連線，同時也必定和伺服器端保持接觸，測試的結果在少量客戶時也是可行的，但和我們其他部分的工作相容時出現了問題。像是：GUI 等元件不容意透過伺服器傳輸等情況。　　解決方式：只能放棄伺服器為主的傳輸，改成伺服器當作牽線角色的 P2P 連線，結合時出現錯誤是預期的結果，進行一個專題不可能事事順心。在有限時間下修改最少部分已完成整體是較好的方式。之前已伺服器為溝通橋梁的程式碼也有附上，為 old_code 下的 auto_server.py 和 client_windows.py

2. 程式碼保護工作：

費盡天辛萬苦我們做出了基礎的成品並且達到預期內的結果。但這並不代表這部分的工作至此結束，我們的主題是製作軟體，所以必須考慮到任何使用者的使用情形，包括預期外的使用方式或是攻擊等，伺服器端對此相較之下重要的多。像是對偵測並反映客戶端配對前的斷線，若未清除相關資訊其他客戶的配對會出現極大的問題。

但安全性永遠都嫌不足，如果客戶端在配對成功但伺服器主動清除前的一瞬間斷線，程式將會崩潰。但至少我們的程式已經保護了絕大多數正常使用的情況。其他狀況則需要進一步的程式撰寫來保護。

9 未來工作

1. 偏好配對：

目前聊天室使用的是隨機配對，也就是等待序列只會有一個人，當第二個人進入時隨即配對成功。但偏好設定也是一個受歡迎的功能。使用者註冊時需要輸入其他基本資料，可以選擇隨機配對和偏好配對，偏好配對需要設定喜歡的類型。而只有在雙方都符合對方的設定時才能配對，這就是等待設置序列而不是一個參數的原因。例如：有一個 30 歲男偏好為 20 歲以下女進入聊天室，他是第一位所以進入等待。接著有一個 18 歲女偏好設定為 20-25 歲男進入聊天室，因為前者未通過後者的偏好，兩人都進入等待。之後 23 歲男隨機配對進入聊天室，他和 18 歲女互相通過偏好，系統則將兩人配對成功。

偏好設定的工作並不複雜，只需在登入介面記錄幾條欄位和配對的選擇。伺服器是的程式稍微修改，算是可達成機率高的展望。

2. 加密功能的提升：

我們的密碼是使用 MD5 訊息摘要演算法加密，MD5 有一個優點，它就是單向加密技術，也就是說，MD5 密碼是不可以解密的，所以想通過單向解密的方法來解密 MD5 是不可能的，但是 MD5 有個缺點，就是一個字元 MD5 加密密碼是一成不變的，比如 123456 的 32 位 MD5 大寫：E10ADC3949BA59ABBE56E057F20F883E 那麼 12345 的 32 位 MD5 就是 E10ADC3949BA59ABBE56E057F20F883E，它們是對應的，所以，有些人就通過程序生成了龐大的 MD5 字典，通過查詢 MD5 值對應的字元，即可輕鬆知道其對應的密碼。所以 MD5 雖然看似安全，卻仍有被破解的風險，未來若能使用 SHA-2 可能會是較理想的選擇。

3. 傳送速度優化：

由於我們在程式中有較多的判斷，導致影像在傳輸的時候有較多的延遲，雖然已經從剛開始的 3 4 秒，到最後 demo 的時候只剩下 1 秒以內，但仍不及目前主流的 app (如 skype、line、Facebook Messenger... 等等)

4. 音訊及音效：

我們有影像跟文字的傳輸，卻沒有音訊的傳輸。若能傳送聲音的話，可以更廣泛的應用在開會、面試等用途。(我們有成功安裝 pyaudio 並使用 udp 傳送音訊，但礙於

時間不夠，無法將其加入我們最後的程式中。)

我們本來想要加入通知音效，讓使用者知道有無配對成功，避免還沒準備好就開始傳輸影像的尷尬情況。(我們有成功安裝 pygame 並測試一些簡單的程式，但礙於時間不夠，無法將其加入我們最後的程式中。)

10 參考資料

1. http://blog.maxkit.com.tw/2017/07/socket-opencv-client.html
2. https://systemprogrammingatntu.github.io/MP4
3. http://www.how01.com/post_QE9WQyX5yXm9j.html