

DIP Term Project

第 4 組：

傅敏桓 B01902008

陳柏屹 B03902096

周家宇 B03902032

在這次的學期作品中我們實作了上次提案之中的其中兩種功能，分別為 **facial remapping** 和 **offspring prediction**，以下分別介紹：

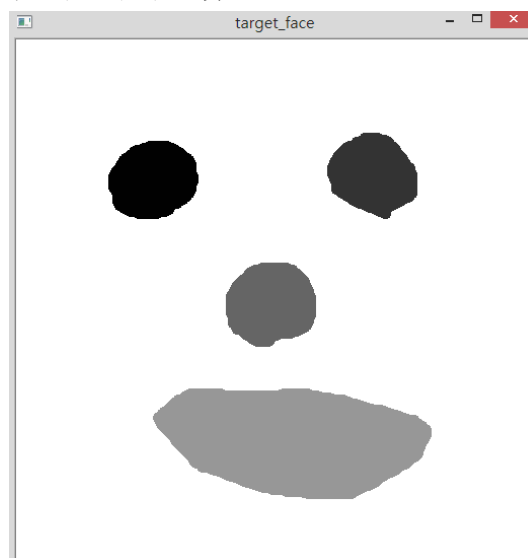
I. Facial Remapping

這部分則全部都是利用 **openCV** 內建的函式來達成。過程可以分為圖畫五官布局、臉部五官抽取和布局至新圖三個步驟。主要是透過 **setMouseCallback()** 來作畫，並用 **CascadeClassifier** 來偵測出臉部五官的位置，最後再用 **Nearest Neighbor** 來作矩形縮放。另外執行時可以輸入 0、1、2 來分別選擇成果背景。

a. 圖畫五官布局 (Target Feature Layout)

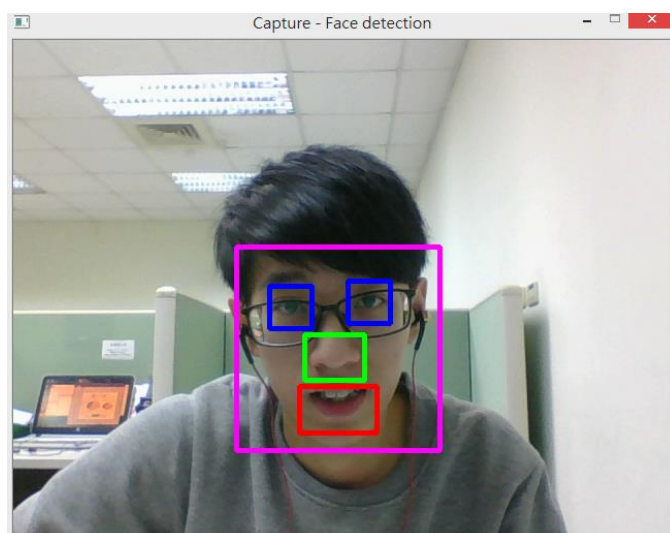
首先要畫出五官的相對位置、大小、數量，這個部分我們是利用 **openCV** 內建函式 **setMouseCallback()** 來實作。我們先開設一新的 **Mat** 當作臉的畫布，並開始設定每個滑鼠 **event** 對應到的動作，並記錄每個經過的點，每個點都是利用 **line()** 函式相接而成，最後再將起點和終點連起，就得到圖形的輪廓。接著利用內建函式 **fillPoly()** 來塗滿輪廓內的區域。整體而言，在設定完滑鼠的 **event** 之後，我們進入一個 **while()** 迴圈讓使用者可以持續作畫，並可以作圖出多重五官直到 **cvWaitKey()** 偵測到按鍵“a”，才會換到下一個五官(順序為左眼、右眼、鼻、口)和標記顏色，最後一次偵測到按鍵則會結束作畫。

範例畫布，顏色由深至淺為左眼、右眼、鼻、口



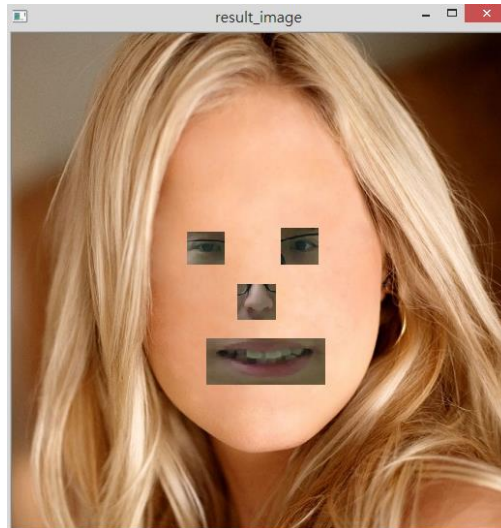
b. 臉部五官抽取 (Face Feature Extraction)

在這次的臉部辨識中我們試著使用 openCV 內建的 CascadeClassifier 來取得五官位置。一開始先用 openCV 內建函式中的 VideoCapture 類別來讀取影片/影像，而影像來源的部分我們選擇使用電腦的預設鏡頭，因此在 capture() 函式中輸入為 0。接著將來源影片串流使用 VideoCapture 類別中的 read() 來將影片取得瞬間擷圖並讀出成為單個 Mat，並對此 Mat 進行操作。我們先將得到的這個 Mat 轉換成灰階色彩空間並利用 histogram equalization 提高影像對比，接著使用 openCV 中的 CascadeClassifier 針對各部位(臉、眼、鼻、口)的訓練集做 training，然後對影像做 detection，得到的資料包含 detect 到幾個物件，以及每個物件的左上角的座標和物件的長與寬。然後使用 rectangle() 將每個物件框起來(紀錄方形各點座標以及長寬)。其中 CascadeClassifier 類別中的 load(String)，用來選擇要對哪個訓練集做訓練，而這些訓練集是由 openCV 官方網站所提供，用來分別辨識各個五官或臉；detectMultiScale() 則是對影像進行 detect，並將資訊存到 vector<Rect> 的資料結構中。如此我們就得到了輸入影片中使用者五官的位置(框起五官方形的各點座標以及長與寬)。



c. 布局至新圖 (Implement Layout)

在得到了五官布局以及來源五官的座標後，我們要將兩者合併結果對應到一張新的臉上，其中我們本來的理想是可以任意變形五官，例如將眼睛做成彎月形、或任意的奇形怪狀，然而在 openCV 中找不到相對應的函式，因此我們雖然有 contour 的座標群，但卻只能做出基本的矩形縮放。縮放的方式使採用 Nearest Neighbor 的方法，先取得畫布上各顏色區塊的上下左右極值，再將來源臉部五官用顏色對應器官來分配到相對應的位置上，最後就得到我們的成果圖，背景則可以預設成空白臉或任意物品。



II. Offspring Prediction

我們在這個部分是透過影像變形 (morphing) 的方式實作。變形在影像處理上應用面很廣，也有許多發展成熟的技術，而我們在這次的 project 只實作簡單的臉部變形，希望透過兩張臉的混和結果，可以在微笑四方上面推廣各種應用，獲得一些娛樂效果。將兩張圖片直接疊合會產生不預期的效果，因此必須找出哪些地方要和哪些地方疊合，這些過程主要可以概括為：臉部特徵抽取、三角平面分割 (triangulation) 以及仿射變換 (affine transformation)，以下分別就這幾個部分概述。

a. 臉部特徵點抽取 (Face Landmark Detection)

臉部辨識已經是長期被研究的領域，各影像處理工具庫都有提供相關的功能。其中 OpenCV 雖然有內建人臉偵測，但只會標出包含臉部或五官的 windows，並沒有提供更精確的特徵點位置。這些特徵點雖然可以透過 window 的相對位置算出 (例如嘴角應該是包含嘴部 window 的兩短邊之中點等)，但是怎麼定義這些特徵是個問題，且臉型的部分也很難精確地描述，而有可能在這個過程中花費額外的時間。

在這個部分我們使用的是機器學習的 dlib 函式庫來幫助我們抽取臉部特徵點。dlib 的人臉偵測除了會找出人臉的位置以外，還會標註出描述臉型、五官等特徵的 68 個特徵點，在後續的平面分割過程中也可以使用這些點。這個臉部特徵點抽取的過程是藉由一群迴歸樹

(regression trees) 來達成；dlib 聲稱這是一個幾乎可以在 real time 完成的演算法，也可以直接使用於 webcam 影像上，我們認為應用於微笑四方也很合適。主要用到的東西如下。

`get_frontal_face_detector()`

回傳 `frontal_face_detector` 物件，可以呼叫取得所有包含臉部影像的區塊

`shape_predictor` 物件

輸入取得的臉部影像區塊，抽取每一張臉的特徵點

b. 三角平面分割 (Triangulation)

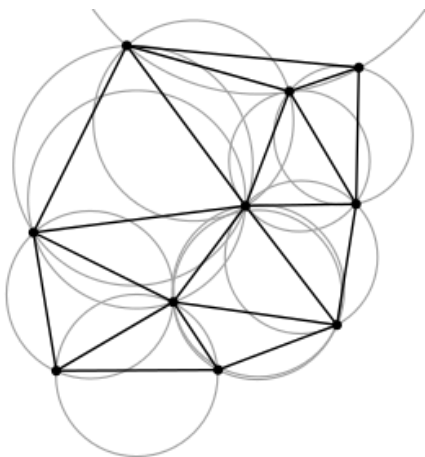
得到臉部的特徵點對我們很大的意義：我們可以藉此在臉上切割出代表不同部位的區塊，有了這些資訊我們才知道不同臉之間哪些區塊可以互相疊加，而不致產生奇怪的結果。對於平面上任意分布的點有不同的連線方式使得平面被切割成數個三角形，這邊採用的方法是 Delaunay Triangulation，可以保證這些點不會出現在切出的任一三角形的外接圓內部，也就是不會切出過度尖銳的三角形。這個部分我們可以透過 OpenCV 的函式來實作，使用的特徵點除了用 dlib 抓出的 68 個之外，還加入代表圖片邊界的 8 個點 (四個頂點、四邊中點)。用到的主要有以下幾個東西。

Subdiv2D 物件

`Subdiv2D(Rect rect)`會建立一個空的 Delaunay Subdivsion，可以透過 `insert()`加入包含於這個子區域中任何二維空間的點，然後用下面的 function 來取得分割出的三角形的各頂點。

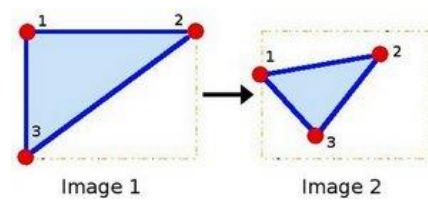
`getTriangleList()`

回傳三角平面分割中頂點位置的 tuple list。

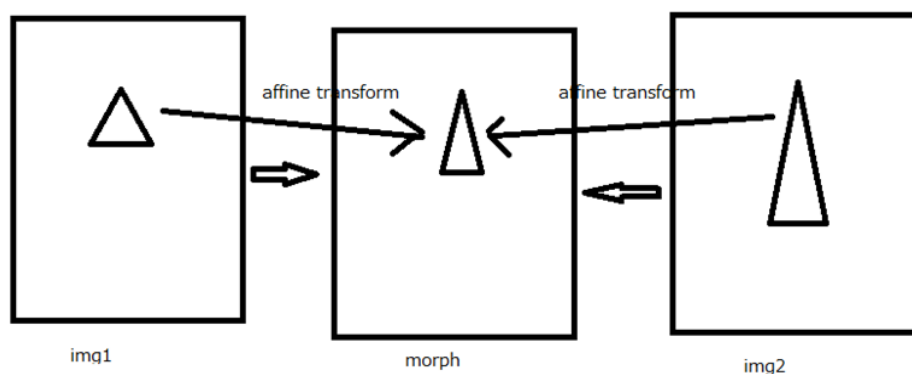


c. 仿射變換 (Affine Transformation)

仿射變換是指在向量空間進行一次線性變換後進行一次平移，或者在矩陣的表示法下經過一次的矩陣乘法和一次的向量加法。對於平面空間中任兩個三角形，我們可以透過一次仿射變換互相轉換成對方，而這將決定我們如何將不同形狀的三角區塊在同一個形狀中疊加。考慮兩個平面空間中的三角形，頂點分別為 (x_1, y_1) , (x_2, y_2) , (x_3, y_3) 及 (x'_1, y'_1) , (x'_2, y'_2) , (x'_3, y'_3) ，我們可以透過以下的關係式逆求得轉換的矩陣 `map_matrix`，接著就可以對區塊中的每個點進行轉換。

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \text{map_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$


我們定義融合係數 α 為兩張影像在變形的過程中所佔的比例。在整個變形的過程中，首先必須決定目標影像相對於來源影像特徵點的對應位置，比方說新的影像的眼角位置必須要和兩張來源影像的眼角位置有一些相對位置關係，比方說是在兩者連線上的某個位置，這可以透過一個簡單的內插法來實現。接著我們就可以決定來源影像和目標影像的對應區塊之間的相對位置關係，並算出仿射轉換的矩陣，如下圖所示。



這邊我們同樣使用 OpenCV 的 function 來實現仿射變換：

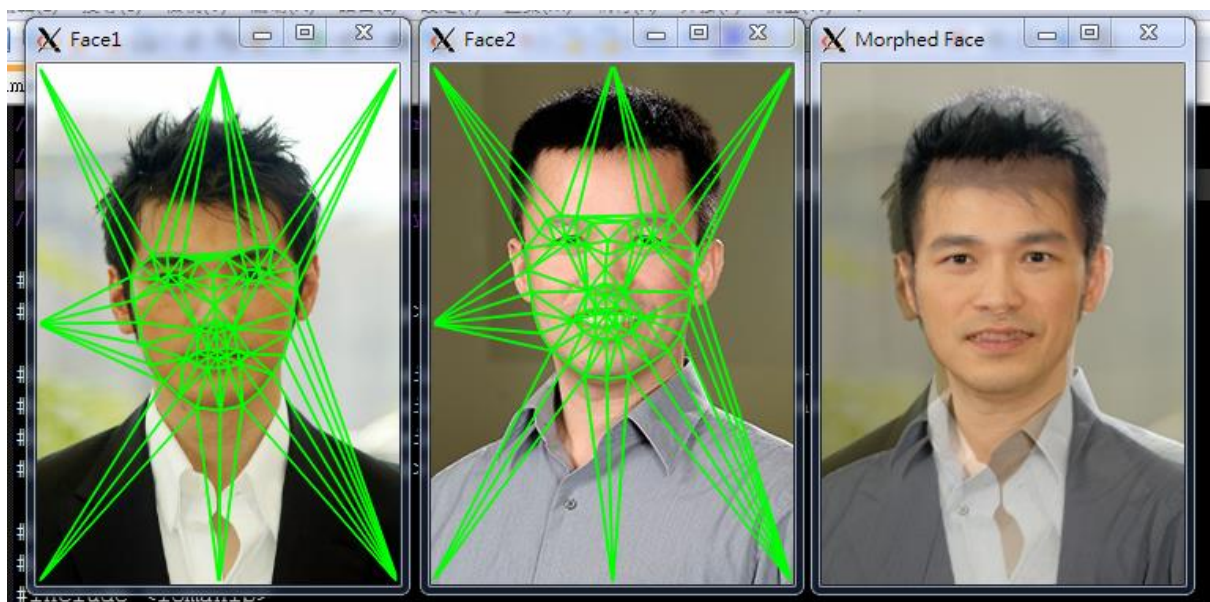
`getAffineTransform()`

輸入兩組三角形的頂點，回傳對應的仿射變換矩陣。

`warpAffine()`

根據上面的結果進行仿射轉換。

對於每一組來源圖片中互相對應的三角形區塊，我們可以得到他們轉換到目標影像的結果，然後在目標影像中根據融合係數 α 的比例進行疊加，得到最終的結果。



Reference:

www.learnopencv.com/face-morph-using-opencv-cpp-python

Dlib, OpenCV Reference

Wikipedia