適用於線上會議之虛擬替身臉部姿態軟體開發

3D Avatar software for video conference

指導老師：黃朝宗　　組別：B240　　組員：林孟平

領域：資工領域

1. Abstract

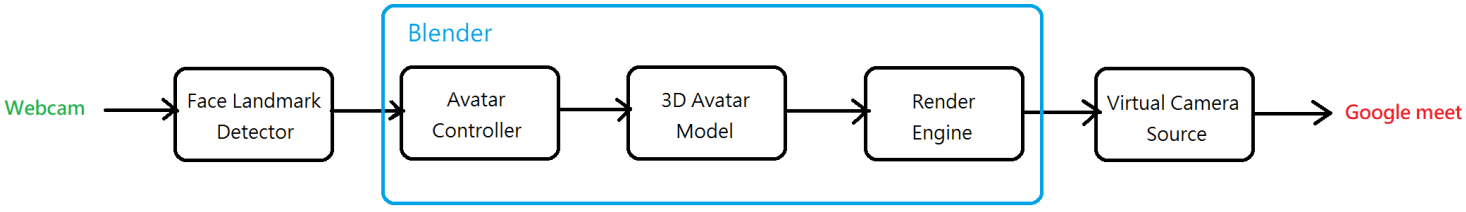
在後疫情時代，為了順暢地進行居家工作，此時的人們經常使用 Google meet、Teams 等軟體進行線上視訊會議。然而，由於人們在家裡工作時通常疏於化妝打扮，因此對於開啟視訊鏡頭難免存在顧忌。因此，我們希望能開發一個能夠在 Google meet 等會議軟體上輸出虛擬替身的軟體，此替身能夠代替人們呈現臉部表請，使得人們在不需在意自身形象的狀況下，放心地開啟鏡頭和其他會議參與人互動。

本專題的目的在於用軟體建立一個實際可運行的系統，為了能夠讓電腦自動辨識真人臉部表情，將使用 Python 作為程式語言，驅動訓練好的深度學習模型以用於偵測臉部特徵點，並實作出電腦視覺領域之相機、世界參考坐標系轉換演算法，計算出臉部特徵點的 3D 資訊，最後搭配 Blender 建模軟體，製作出 3D 虛擬替身並控制其臉部姿態，整個系統將可以應用在 Google meet、Teams 等會議軟體上。

由於單一攝影機只能偵測以相機為參考點之 2D 座標，且無法預測頭部對於鏡頭之旋轉平移關係，因此為了還原出 3D 特徵點之資訊，我們通常仰賴多個不同視角之攝影機進行預測。本專題較為困難之點在於：我們必須在使用無法進行深度預測之單一攝影機下，也能還原出 3D 臉部特徵點位置。而為了達成這個目標，我們對於開會情境下之使用者位置進行了諸多假設，以實現數學模型並降低其計算複雜度，並在最後用實驗結果證明此假設之可行性。

2. System Overview

本系統必須達到以下三個核心目標：
(1) 使用真人臉部表情資訊控制虛擬替身(3D Avatar)
(2) 能夠傳輸虛擬替身之動態影像至 Google meet
(3) 整個系統作為 Webcam 及 Google meet 之間的 Bypass Software
因此，整個系統將使用 Python 語言，且用 Blender 作為虛體替身之建模軟體，



Face Landmark Detector: 偵測臉部多個位置之 2D 特徵點位置資訊
Avatar Controller: 透過計算臉部相對於鏡頭之旋轉、平移關係，從 2D 特徵點還原出 3D 資訊。
3D Avatar Model: 接收 **Avatar Controller** 所計算出的資訊，驅動虛擬替身輸出動態影像
Render Engine: 對虛擬替身進行圖片渲染，將動態影像轉換至可讀取之影格。
Virtual Camera Source: 建立虛擬攝影機來源，並將影格送至會議軟體。

3. System Specification

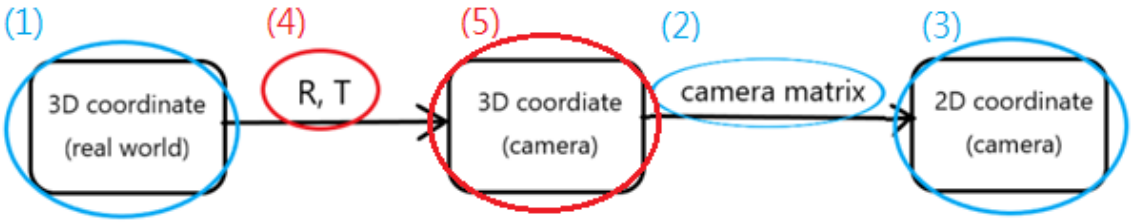
3.1 Face Landmark Detector

為了讓電腦理解真人臉部的表情，在本專題中，我們使用 Google 之開源臉部辨識神經網路 Mediapipe，以偵測多達 468 個臉部特徵點。

除了能顯示特徵點於螢幕上，由於此神經網路偵測之準確度及穩定性高，因此我們可將這些特徵點於輸出影像上的位置，作為臉部某器官位於影像上之座標。然而，由於此座標僅為相機坐標系中之 2D 座標，為了避免臉部相對於鏡頭的旋轉、平移造成偵測誤差過大，我們無法直接透過此座標判斷臉部的表情資訊。因此我們必須透過 **Avatar Controller** 之數學模型將其還原成同座標系之 3D 座標。

3.2.1 Avatar Controller

本部分為電腦視覺演算法之實現，將分為理論與實驗兩部分說明。在 **Face Landmark Detector** 中偵測出之特徵點為以鏡頭為參考點之 2D 位置資訊，然而，為了準確地讀取臉部表情資訊，我們不僅需要同坐標系下之 3D 座標，還必須計算世界參考(臉部)坐標系與相機坐標系之間的旋轉、平移關係。由於在會議進行中，使用者的頭部中心約略坐落在鏡頭正前方，且並不會有過大的移動，因此我們可將使用者的頭部中心定義為世界參考坐標系之原點。除此之外，由於頭部各器官相對於鼻子之相對位置皆相同，我們也可同時得知這些器官位於世界參考坐標系上之位置。除此之外，我們假設頭部在鏡頭前的位置約等於筆電的寬度，因此已知焦距的狀況下，可得知此相機之 Camera Matrix。以下將透過這些已知資訊預測臉部特徵點位於相機座標系之 3D 座標，以及頭部相對於鏡頭之旋轉、平移關係等未知資訊。關係圖如下：



(1) 特徵點位於世界參考坐標系之 3D 座標
(2) 鏡頭之相機矩陣
(3) 特徵點位於相機坐標系之 2D 座標
(4) 世界參考坐標系與相機坐標系之旋轉、平移矩陣
(5) 特徵點位於相機坐標系之 3D 座標

(1) 臉部器官之世界參考坐標系位置：
我們以鼻子中心為世界坐標系之原點，則 X, Y, Z 方向之旋轉角度可分別命名為 Pitch、Yaw、Raw。

由於臉部器官彼此之間相對位置不會改變，我們可假設臉部各器官坐落於此坐標系中固定位置，假設座標如下：

$$P_w(x, y, z) = (P_x, P_y, P_z)$$

Table 3.2-1
各器官之 World Reference 座標

(Unit: Pixel)

Organ	Px	Py	Pz
Nose	0	0	0
Chin	0	-300	-50
Left Eye	200	150	-130
Right Eye	-200	150	-130
Mouth (Left corner)	150	-150	-130
Mouth (Right corner)	-150	-150	-130

(2) 相機矩陣之定義

在相機坐標系中，相機矩陣代表著 3D 座標與 2D 座標之間的轉換關係：

$$P' = KP$$

其中:

$P' = [\mathrm{x}', \mathrm{y}', 1]^{\mathrm{T}}$ 為相機座標系中之 2D 座標

$P = [\mathrm{x}, \mathrm{y}, \mathrm{z}, 1]^{\mathrm{T}}$ 為相機坐標系中之 3D 座標

已知頭部於鏡頭前的位置下，可得知此相機矩陣之參數 α。

$$K = \begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

其中:

$\alpha =$ *number of pixels bewteen the head and the camera* (1080 in this case)

(3) 特徵點位於相機坐標系之位置(2D)

在 **Face Landmark Detector** 所偵測到之特徵點可透過 Python 函數，將之轉換到相機坐標系中之 2D 座標。

$$P_i = P_i(x, y)$$

其中: P_i 為第 i 個特徵點位於相機坐標系之 2D 座標

(4) 計算姿態矩陣、還原 3D 特徵點座標

已知(1)~(3)之資訊後，即可透過 SolvePnP(OpenCV Library)之方法，得到相機坐標系與世界坐標系之間的姿態矩陣 M，矩陣正是代表臉部相對於鏡頭的旋轉、平移關係。

$$P^C = MP^w$$

$$M = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

其中:

$P^c = [\mathrm{x}^{\mathrm{C}}, \mathrm{y}^{\mathrm{C}}, \mathrm{z}^{\mathrm{C}}, 1]^{\mathrm{T}}$ 為相機坐標系中之 3D 座標

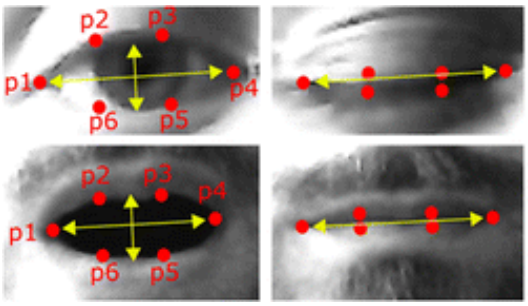
$P^w = [\mathrm{x}, \mathrm{y}, \mathrm{z}, 1]^{\mathrm{T}}$ 為世界坐標系中之 3D 座標

R 為 x, y, z 三個維度之旋轉矩陣乘積

$T = [t_x, t_y, t_z]^T$ 為兩坐標系原點間之平移向量

(5) Mouth and Eye Aspect Ratio

透過 SolvePnp 所求得之姿態矩陣，我們可以轉換 Table 1 中之假設座標，找到它們在相機坐標系中的對應點，並且可透過 Mouth(Ear) Aspect Ratio 的方法，計算出嘴巴、眼睛等器官的張開、閉合狀態。



$$\mathrm{EAR} = \mathrm{MAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Reference: https://www.researchgate.net/figure/The-eye-aspect-ratio-is-practically-identical-to-the-mouth-aspect-radio_fig22_323953620

其中: p_i為各特徵點於相機坐標系中之 3D 座標

(6) Euler angle

由於我們已求出兩坐標系間之姿態矩陣，因此透過分解 R 矩陣，即可得知兩座標系間的旋轉關係。

$$R = R(\alpha)R(\beta)R(\gamma)$$

$$R(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad R(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \quad R(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

其中: α、β、γ分別代表相機坐標系與世界坐標系在 x, y, z 方向的旋轉角度。

3.3 3D Avatar Model:

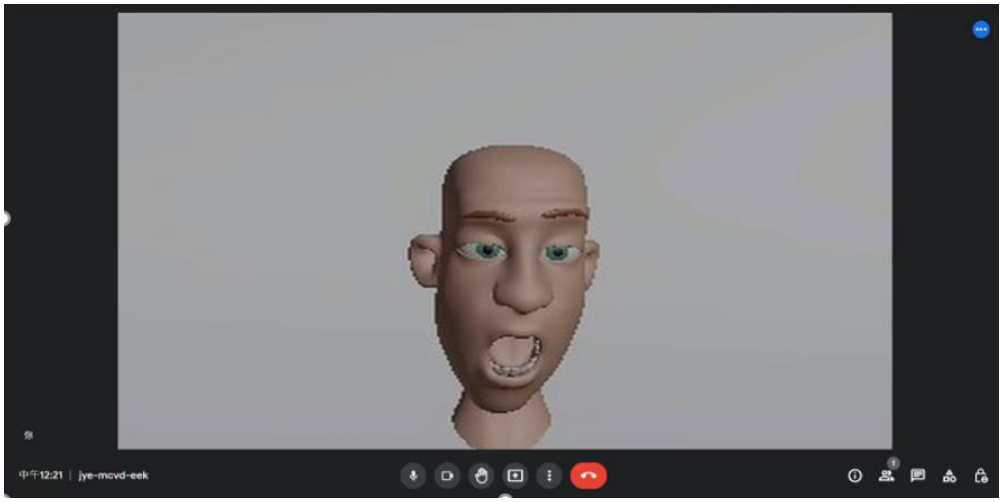
利用 Blender 建模軟體，我們能繪製出動態的客製化虛擬替身。此虛擬替身可接收由 **Avatar Controller** 所計算出之嘴巴、眼睛閉合狀態以及頭部旋轉尤拉角，表現出不同的臉部姿態，並同時輸出動態影像。

3.4 Render Engine

Blender 內的 Render Engine 將對虛擬替身進行圖片渲染(Render)，將虛擬替身動態影像轉換至能存取到 Google meet 之影格(Frame)。

3.5 Virtual Camera Source

為了讓 Python 控制模組將渲染圖片傳送至 Google Meet，我們使用開源的 Open Broadcaster Software(OBS)中，建構虛擬攝影機之原始碼，此部分程式碼將在 Google Meet 上建立 OBS 虛擬攝影機來源，並且將 **Render Engine** 所產生之渲染圖片作為 OBS 的影像輸入，便可在 Google Meet 上看到 3D Avatar 的動態影像：



4. Problem and Discussion

由整個系統必須倚賴 Google meet、Python 控制模組、Blender、OBS 等軟體同時運行，且當中神經網路偵測、圖片渲染之過程較為耗費硬體計算資源，因此在輸出影像之每秒幀數(FPS)表現較為不理想，以下是我認為在同樣之系統架構下，此專題可繼續改進之方向：

(1) 訓練(使用)較為簡單之神經網路：Google Mediapipe 之神經網路雖然有偵測準確度高、偵測特徵點數量多之優點，但多達 468 個特徵點之偵測十分耗費電腦之硬體資源。在本專題中，僅需大約 30 個特徵點給予之資訊即可操縱虛擬替身之臉部姿態。若未來可針對此目標，進行專用之神經網路訓練，或許可改善偵測耗費時間較長之問題。

(2) 使用開源 GPU (Colab)進行圖片渲染：Blender 之圖片渲染(Render Engine)十分耗費計算資源，在不能保證所有使用者皆擁有強大之繪圖顯示卡下，若能使用開源之 GPU 進行渲染，則可大幅提升效率並提升影像輸出效率。然而，目前僅能透過此方法渲染已錄下之影像片段，尚未找到在開啟鏡頭之狀況下，同步(Real-time)進行渲染的實作方式。未來希望能針對此目標開發專門之模組，讓每一位使用者皆能在強大的硬體支援下運行整個系統。

5. Conclusion

從前述之結果顯示，使用 Mediapipe 神經網路進行 2D 特徵點偵測，並使用 Computer Vision 之數學方法還原成 3D 資訊，可有效地操縱虛擬替身之臉部姿態，並將將此作為 Google meet 的影像輸出。然而，整個數學模型須建立在使用者頭部幾何中心無大量移動的狀況下，估計出之座標和旋轉角度才會較為準確。因此，若是我們能另行開發一個模組先行預測使用者的頭部幾何中心之位置，就可以突破人臉必須位在鏡頭正前方的限制。不僅如此，尤於現今之神經網路技術已可同時辨識影像中多個人像之特徵點，因此我們還能透過此方法一次估計多個使用者的臉部姿態，讓多個虛擬替身在會議軟體上顯現。

6. Project Progress and Management

在上個學期，我們先進行了題目的設定，參考完諸多電腦視覺領域之論文及開源程式碼後，決定以「可在會議軟體上運行之虛擬替身軟體」為此專題之題目。而為了發想出可實作的演算法，我們進行了電腦視覺知識之學習，包括閱讀 Richard Szeliski 教授編寫之教科書以及參考孫民教授開設之計算機視覺課程教材。最後進行了系統架構的規劃，包括發想出系統各模組大致的功能及需要的軟體資源。

在這個學期，我們的主要目標是實作出系統中各模組的功能。包括利用 Python 進行了電腦視覺演算法的驗證實驗，以及完成神經網路偵測、驅動建模軟體與虛擬攝影機所需要之程式碼。最後我們除了將這些模組搭建成一個可運行的完整系統外，也嘗試透過諸多方法嘗試優化原本的系統架構，讓軟體有更佳的使用體驗。

7. Thought and Reflection

自己在加入這個專題以前，對於電腦視覺領域的知識沒有太多的瞭解，再加上對於 Python 的熟悉度較為不足，對於各個套件、函式庫的使用概念也十分缺乏，因此起初並沒有把握能實作出什麼樣的系統，在題目設定上花費了不少心力。

由於是自行發想題目之專題，從演算法發想到搭建系統的過程中，時常會碰到看似無法解決的問題，而這些問題多半自己與學長姊甚至是老師先前不曾接觸過的，因此我花費不少時間並不斷嘗試各種方法才得以找到答案。但也因為必須面對這些困難，我在這個專題當中不只學到了如何解決問題，更學到了如何去定義一個可解決的問題本身。除此之外，由於每兩週會舉行一次專題會議，因此我在這個專題中也累積了大量的報告經驗。相比於其他的必選修課學到的課本知識與解題技巧，我認為這個專題的收穫是更加可貴的。

感謝上個學期跟我一起修實作專題一的隊友們，在設定題目、演算法發想與架構設計的部份幫了我不少忙，讓我在這學期能夠花大部分的心力在實作程式功能。也很感謝朝宗老師在這一年當中的指導，每次碰到問題找不到解決方法時，老師在專題會議中給予的建議除了幫助我找到可執行的方向外，也讓體認到自己還有許多需要精進的地方，最重要的是告訴了我們解決問題須具備的正確心態，這些寶貴的建議都使我在這一路上獲益良多，最後才得以完成這個專題。