國　立　清　華　大　學
工業工程與工程管理學系

智慧化企業整合
Intelligent Integration of Enterprise

人體動作辨識結合語音助理於組裝現場應用

110034538 許淨嵐

指導教授：　　邱銘傳　　博士

中　華　民　國　一　一　一　年　一　月

目錄

圖目錄

表目錄

壹、簡介

一、背景與動機

在工業4.0的浪潮下，在智慧工廠中結合物聯網、人工智慧等相關技術及應用數見不鮮，機器人取代作業員的議題也甚囂塵上。然而，人類的工作特性，仍有眾多機器人無法企及之處，例如更彈性的派工，抑或是細微的組裝動作及定位。因此，去人化並非新型態人機互動的主旨，而是機器人與作業員的良性協作，或稱人機協同，才有機會使自動化發揮更好的潛能。

在深度學習應用於智慧工廠的實例當中，作業員之管理與監控，抑或是作業員本身工作品質的量化，少有建樹。作業員針對機器的操作與介入，常是產線運作資訊中，無法量化且不可視的部分。因此，如何透過人工智慧及影像分析方法擷取製造現場人員操作所隱含之資訊，並針對作業員本身亦或是生產流程優化，是人體辨識應用於智慧製造上之重要領域。

二、研究目的

為提高組裝工作站中人機互動的自主性，以及考慮到人類觀察者必須在組裝員行為監測上所花費的時間和精力，本研究欲探討並開發一以電腦視覺技術驅動，語音交互為基礎的人機協同介面。基於電腦視覺的人體骨架辨識模型可以幫助持續且自動地監視工作人員，而以語音作為媒介的人機互動方式則能徹底解放組裝員的雙手，改變以往運用雙手操作鍵盤、滑鼠之輸入方式，降低組裝員操作系統所需之注意力需求。組裝員將可以透過與提出之系統互動，客觀審視自身的工作表現與精度，並在需要工作指引時，即時透過語音互動得到改進工作表現的建議。

三、5W1H 分析

What	人體動作辨識
When	組裝作業進行時、新進人員教育訓練
Where	組裝現場
Why	提升組裝作業效率與準確率
Who	組裝員
How	運用深度學習方法整合語音助理

貳、資料集介紹

本研究募集八位同學協助進行組裝影片拍攝，影片幀率設定為30幀/秒，畫質設定為1280*720，每部影片紀錄一次完整的組裝流程，每一位同學重複完成組裝流程十次，共蒐集80部影片。影片拍攝完成後，將依照單一標準動作流程進行切割，供後續訓練模型使用，圖 1 為組裝影片示意圖。

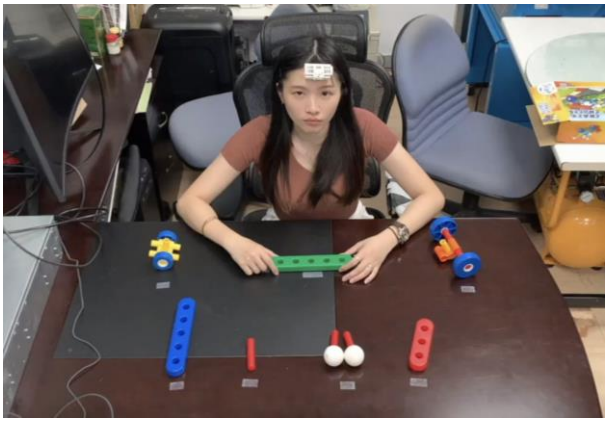


圖 1、組裝影片示意圖

參、研究方法

本研究重心著眼於建立一整合性的系統，將組裝現場拍攝之影像，進行實時的分析，並透過雲端服務及語音助理之整合，將組裝流程中不可見的資訊量化並創價。為達成此一目標，首先制訂一系列符合情境之手部組裝動作，並透過攝影

機對這些動作影片進行拍攝及蒐集。所有收集之組裝影像均透過 OpenPose 套件將人體影像資訊轉成骨架之時序資料(Skeleton Spatial-Temporal, SST)，並進行人工影像貼標，標註各個人體標準動作。待所有資料經過必要的處理後，將被投入 ST-GCN 模型作為深度學習訓練用之影像資料。

同時亦使用亞馬遜網路服務建置一符合需求之環境，一旦 ST-GCN 模型能夠準確識別作業員之動作，模型計算之資料將會傳送並儲存於過亞馬遜網路服務之雲端載體，最後透過語音助理對作業員的回饋做為輸出。整體研究方法如圖 2。

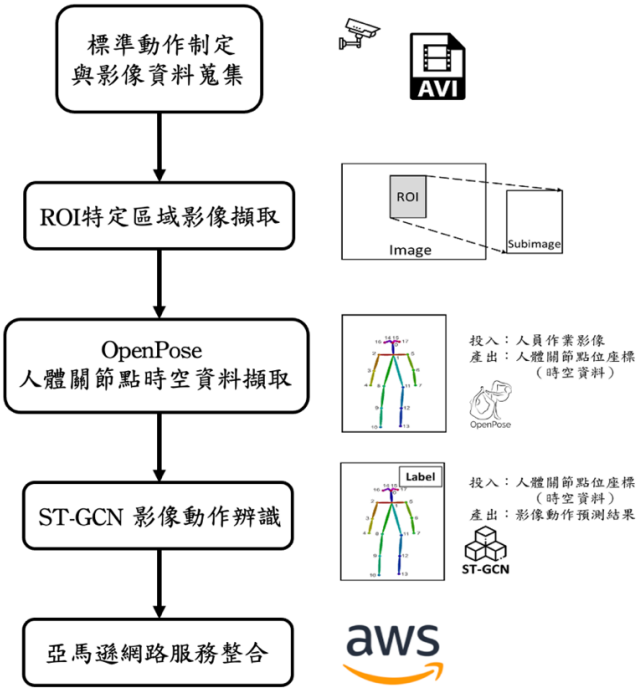


圖 2、研究方法流程圖

一、標準動作制定與影像資料蒐集

本研究以「Crazy Craft」積木玩具模擬一組裝之作業流程，並針對其制定標準動作且架設作業環境與攝影機。而後，本研究安排數位同學重複進行指定動作，並進行錄製。此階段將盡可能蒐集大量影像資料，並確保個體間的差異以避免後階段模型過度擬合的情況。

此外，本研究也將引進 Region Of Interest (ROI) 的效率改善方法。ROI 屬智慧視頻編碼技術的一種，可在不損失圖像品質的前提下，優化視頻編碼

性能。在影像處理領域，ROI 是從圖像中選擇特定區域，這個區域將成為圖像分析所關注的焦點，而不被選擇的區域降低其碼率和圖像品質，抑或是不傳輸這部分區域圖像。本研究使用 ROI 選定輸入資料圖像目標區域，以減少處理時間，增加精度。

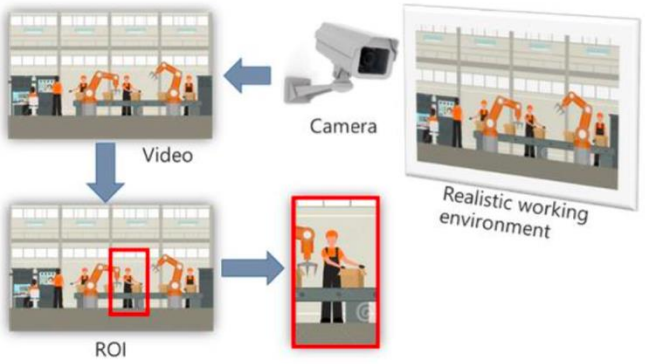


圖 3、ROI 技術改善影像資料擷取效率

二、OpenPose 人體關節點時空資料擷取

在眾多人體動作辨識演算法中，其主要可以分為兩種思路，一是「由上而下（Top-down）」意指先辨識人體區域，再檢測區域內的人體關鍵點；另一方面，「由下而上（Bottom-up）」則為先辨識影像中所有的人體關鍵點，接著將這些關鍵點對應至不同的人物個體身上。然而，前者因需針對辨識出之每個人體區域，分別進行前向關鍵點檢測，運算效能較差。本研究目前僅著重單人辨識之情況，因而導入使用第二種方案之OpenPose套件作為人體關節點時空資訊擷取之工具。

此Openpose套件透過兩分支多階段的CNN，對關節點進行偵測。如圖 4 所示，第一個分支中的每個階段都預測信心圖（Confidence map），係人體特定部位在影像中的像素座標；第二個分支中的每個階段都預測局部關聯向量場（Part affinity field），屬於每個軀幹的向量場，能夠保存關節的位置和走向資訊。在每個階段運算完成後，來自兩個分支的預測以及圖像特徵將被串聯到下一個階段。

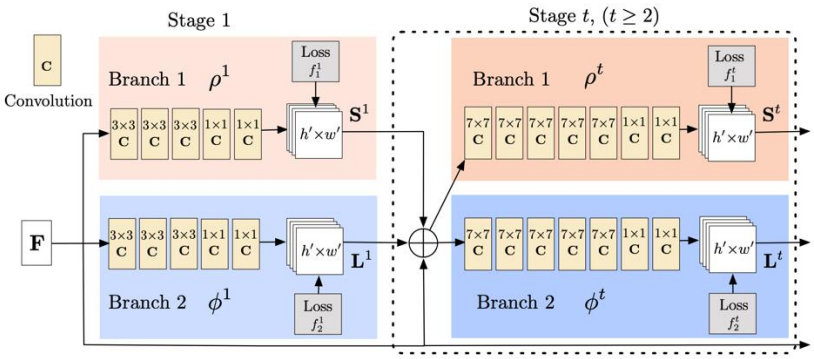


圖 4、OpenPose中兩分支多階段CNN結構

最後網路將求出人體每個關節點於像素座標系中的2D 座標（X，Y），以及18個人體關節的置信度分數C（Confidence score）。因此，每個關節點被用以（X，Y，C）表示，而骨架則以一個含有18個值的數組紀錄。這些骨架的時空資訊將作為本研究引入之ST-GCN模型之訓練及測試資料。

三、ST-GCN 影像動作辨識

ST-GCN相對於傳統GCN骨架辨識方法模型的差異在於，傳統GCN 骨架辨識方法只考慮圖與圖之間的空間關係，而ST-GCN則多考慮了圖與圖之間的時間關係，如圖 5所示。因此，本研究將前一階段OpenPose套件輸出之骨架資訊以時間序列為基準分割，並根據邊緣的重要性在卷積時對其進行加權，透過跨時域的卷積處理對人物進行時序動作評估。此時，每一卷積層之輸出為一時空圖，而圖上每一節點皆保有特徵向量，並可構建人體骨架的時空圖，逐步找出原始影像中的特徵。

由ST-GCN所獲得之每個序列的特徵向量，將被輸入SoftMax分類器，利用隨機梯度下降學習模型，將其分類為相對應的動作類別，將多分類的計算結果以概率表示，且進行動作名稱標示。

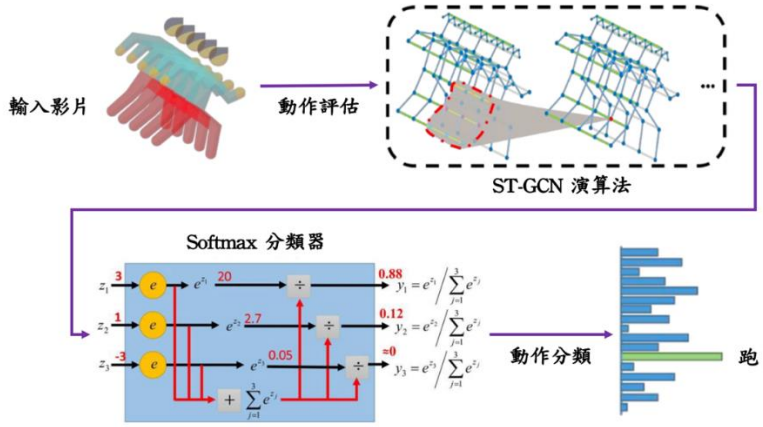


圖 5、ST-GCN 運算示意圖

四、超參數優化 (Hyperparameter tuning)

受限於硬體設備之影像隨機記憶體(VRAM)大小，本研究將訓練批量(Batch size) 設置為硬體可負荷之最大值 7，並將訓練週期(Epoch)設置為 100 以確保模型能夠順利收斂。

在固定訓練批量及訓練週期的條件下，本研究調整主要影響 ST-GCN 模型準確率的三個超參數：丟棄率(Dropout rate)、基礎學習率(Base learning rate)、及梯度下降的步伐(Step)，並將每次訓練結果以最後一訓練周期之訓練誤差如 Top1、Top5 及 Mean loss 作為衡量標準進行比較，結果如表 1 所示。

表 1、不同超參數設定之模型評估

#	Dropout	Base_lr	Step	Top1	Top5	Mean_loss
1	0.1	0.1	[80,90,100]	80.58%	100.00%	0.4127432756
2	0.3	0.1	[80,90,100]	76.70%	100.00%	0.4480777343
3	0.3	0.1	[80,90,95,100]	84.47%	100.00%	0.3942492694
4	0.1	0.1	[80,90,95,100]	73.79%	100.00%	0.5054964473
5	0.1	0.01	[80,90,95,100]	99.03%	100.00%	0.0215713551
6	0	0.1	[80,90,95,100]	100.00%	100.00%	0.0115661198

最後，本研究選擇 Top1、Top5 最高且 Mean loss 最低的模型 6 做為最

終的模型，以其進行後續之模型驗證及系統建構。

五、亞馬遜網路服務整合

亞馬遜服務平台之各式功能，將被用於整合ST-GCN及SoftMax 的分類結果，將本地端運算結果以Alexa語音回饋給組裝員。首先利用 AWS IAM取得本地端電腦對AWS資源及AWS服務之間的存取權限。而後，本研究將本地端骨架辨識結果上傳至Amazon S3雲端空間，再透過AWS Lambda存取S3資料，並進行Alexa語音助理主程式回覆資料庫的編輯，最後將語音助理主程式回傳至Alexa Skill Kit，完成語音助理開發。整體架構如圖 6所示。

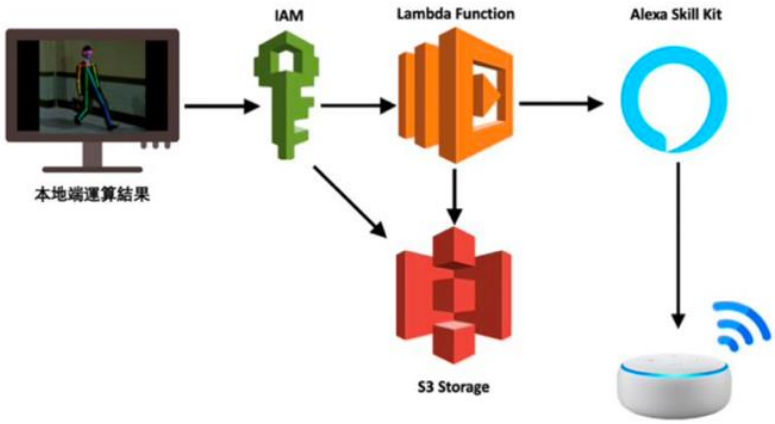


圖 6、整合亞馬遜服務應用架構

肆、實驗流程與結果

一、動作資料蒐集與肢體辨識

(一) 標準動作制定

本研究以「Crazy Craft」積木玩具模擬一組裝之作業流程，並建構出以下包含七個動作之組裝順序。

拿取紅色短板	組裝眼睛-1	組裝眼睛-2	組裝紅色棍棒	組裝藍色長板	組裝前輪	組裝後輪

表 2、組裝步驟說明

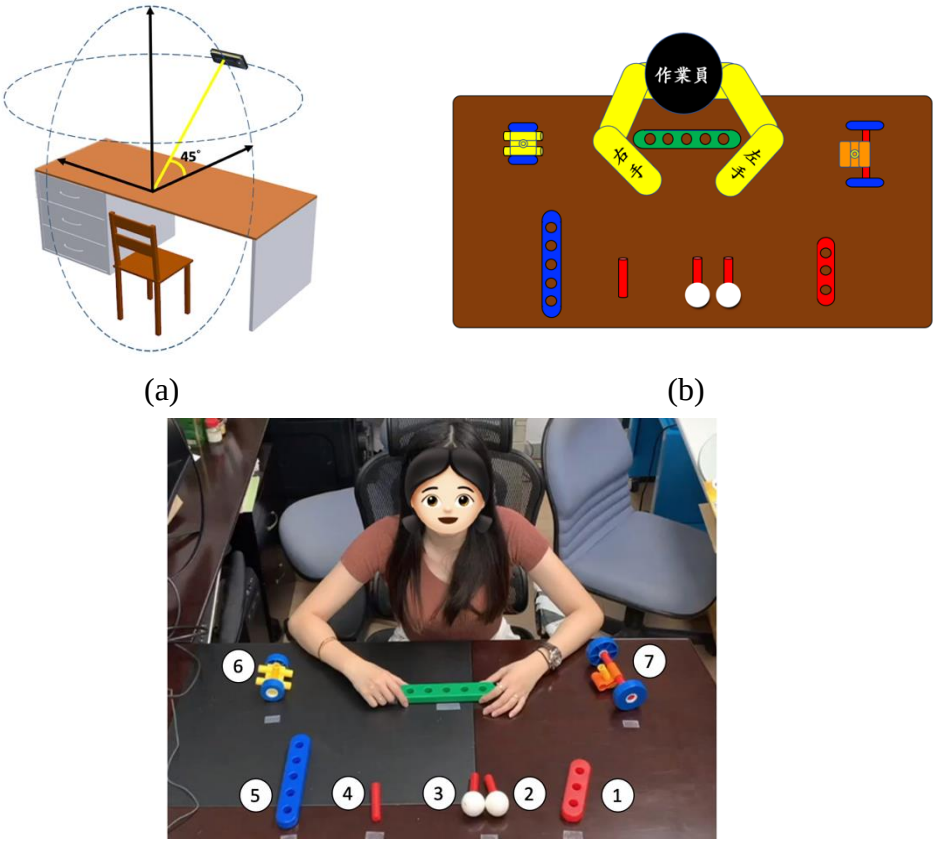
標準動作	組裝說明
1. 拿取紅色長板	左手拿取紅色短板，右手穩定綠色長板，並將紅色短板對準綠色長板中心孔後疊在其上。
2. 組裝眼睛-1	右手穩定半成品，左手拿取眼睛 1 對準紅色短板最左側的孔組裝。
3. 組裝眼睛-2	左手穩定半成品，右手拿取眼睛 2 對準紅色短板最右側的孔組裝，並雙手將組裝完眼睛後之半成品翻面向下壓緊。
4. 組裝紅色棍棒	左手穩定半成品，右手拿取紅色棍棒對準綠色長板中心孔組裝。
5. 組裝藍色短板	左手穩定半成品，右手拿取藍色長板並將其中心孔對準紅色棍棒組裝。
6. 組裝前輪	左手穩定半成品，右手拿取前輪零件對準藍色長板最前方的孔組裝。
7. 組裝後輪	右手穩定半成品，左手拿取後輪零件對準藍色長板最後方的孔組裝，並翻面完成一成品。

圖 7、Crazy Craft 組裝完成圖

(二) 組裝環境定義與影像資料蒐集

根據先前制定的標準動作，本研究設計了一假設情境中之組裝站。組裝員以坐姿進行動作，並伸手至零件存放處拿取零件且組裝，完成組裝後會將成品放至完成區。此工作站為影像資料蒐集用之第一階段工作站，故使用智慧型手機作為攝影裝置，並架設於組裝員正上方45度角處，並預期在未來引入視頻串流設

備，以及語音助理音箱。攝影機相對位置如圖 8 (a)所示，工作站俯視圖如圖 8 (b)，實際架設成果如圖 8 (c)。 而後，根據先前制定的標準動作，蒐集共80部組裝影片。



(a) (b)

(c)

圖 8、組裝站架設圖，(a) 攝影機相對位置 (b) 工作站俯視圖 (c) 架設成果

(三) 資料切割

為了進行ST-GCN模型訓練與驗證，首先將所蒐集之影像資料分為三群：訓練集（Training set）、驗證集（Validation set）、測試集（Testing set），比例為5:1:1。其中訓練集與驗證集中的資料將投入ST-GCN模型訓練，而測試集則將作為模型訓練完成後進行驗證之資料來源。

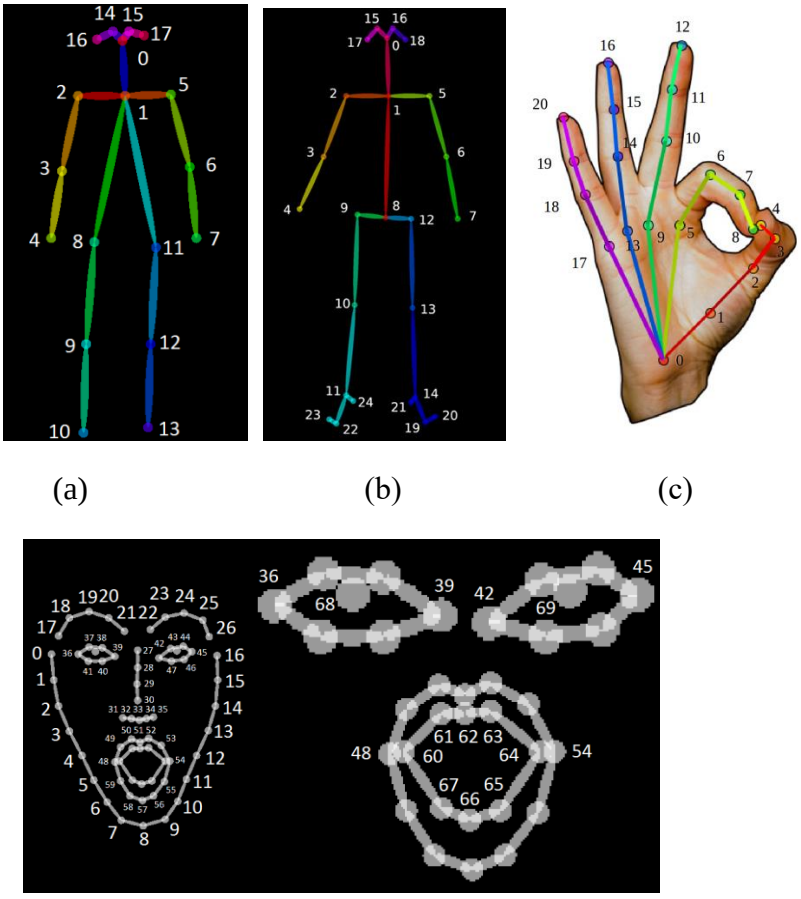
在先前定義的七個標準動作當中，每個組裝動作之持續時間不盡相同，並直接造成OpenPose輸出各動作辨識資料中的幀數差異極大。而訓練集與驗證集中動作間資料量的落差，可能導致模型無法準確進行動作的分類。因而本研究

將動作時間較短的影片重複投入訓練，使得七個標準動作之資料量相當，以提升模型分類動作之效能。

(四) 導入OpenPose 肢體辨識套件

1. 原始關節點設定及應用調整

在OpenPose開源的程式碼中，其提供的肢體辨識分為幾個部分，其中包含全身、臉部以及手部，如圖 9所示。本研究欲使用此肢體辨識套件於手部動作密集的組裝動作，因此透過修正全身辨識（含手部）之輸出資料，將下半身關節點資訊去除，此部分將在後續章節詳述。



(a) (b) (c)

(d)

圖 9、骨架節點示意圖，(a) 全身關節點辨識不含腳趾 (b) 全身關節點辨識含腳趾 (c) 手部關節點辨識 (d) 臉部辨識

2. 運用OpenPose 套件進行肢體辨識

此階段係待軟硬體建置完成後，開始使用OpenPose套件進行肢體辨識。由於需要對所有搜集完成的影像資料進行運算，因此本研究透過批次檔將此流程自動化，關鍵程式碼如表 3。

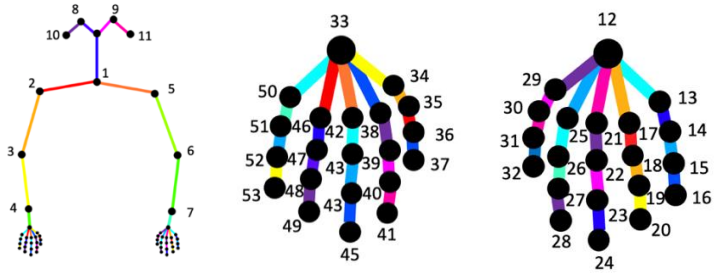
表 3、肢體辨識流程自動化之關鍵程式碼說明

關鍵程式碼	步驟說明
	在指定路徑下，對所有的影片以 COCO 模式使用 OpenPose 進行關節點辨識運算，並將辨識資料以 JSON 檔案寫出。

(五) 導入ST-GCN 模型

1. 關節點座標及標籤JSON格式檔案

如前述，本研究之應用情境只需考量上半身及手指關節點，故在此階段刪除所有下肢關節點座標，如圖 10所示。



(a) (b) (c)

圖 10、本研究使用之骨架圖 (a) 針對此專題修正之人體骨架圖 (b) 左手關節點 (c) 右手關節點

又因ST-GCN模型要求特定JSON格式檔案作為輸入，因此進一步對OpenPose辨識結果之輸出節點進行處理以符合導入模型的格式。OpenPose原

始針對輸入影片之每一幀輸出一JSON檔案，其中包含多項資訊，如臉部、手部、全身之姿勢關鍵點資訊等。為使OpenPose之輸出結果合於ST-GCN之輸入規格，此處透過一系列之資料預處理過程，使得最終完成之單一JSON檔案包含每一幀所對應的所有關節點之座標及辨識信心分數，以及該標準動作之標籤，資料處理流程如圖 11所示。

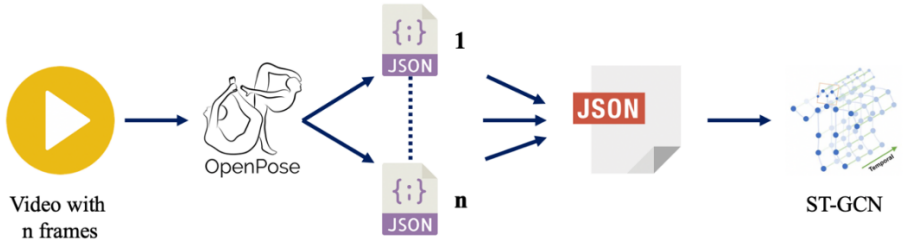
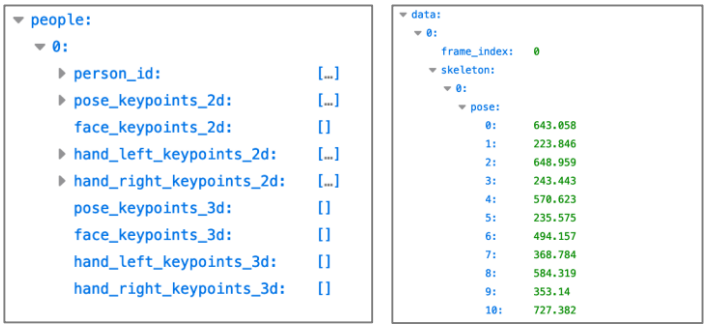


圖 11、資料處理流程圖

如圖 11所示，OpenPose原輸出之JSON檔案僅包含一幀中的多項資訊。本研究進而將單一影片之所有JSON檔案合併，即保留所有幀之所有資訊於該一檔案中，且只保留體態姿勢及左右手的二維關鍵點（pose_keypoints_2d, hand_left_keypoints_2d, hand_right_keypoints_2d）。最終成果如圖 12。



(a) (b)

圖 12、JSON處理前後比較，(a)處理前內容 (b)處理後部分內容

2. 骨架圖設置

ST-GCN模型預設針對不同資料集以及常用資料格式預先設定三種不同

空間上的骨架連接方式，分別為18個資料點的OpenPose、25個資料點的NTU-rgb+d以及24個資料點的NTU_edge。然而，本研究所需之骨架圖設定並非上述三種，因此新增一組包含54個資料點之骨架圖設定。關鍵程式碼如表 4，骨架圖視覺化結果如前述圖 10之骨架節點。

表 4、骨架圖設置之關鍵程式碼說明

關鍵程式碼	步驟說明

`elif layout=='prettyAlenawithHands':`
`  self.num_node=54`
`  self_link=[(i,i)for i in range(self.num_node)]`
`  neighbor_link=[(0,1),(8,0),(9,0),(10,8),(9,11),(1,2),(2,3),(3,4),(1,5),(5,6),(6,7),`
`  (7,12),(12,13),(13,14),(14,15),(15,16),`
`  (12,17),(17,18),(18,19),(19,20),`
`  (12,21),(21,22),(22,23),(23,24),`
`  (12,25),(25,26),(26,27),(27,28),`
`  (12,29),(29,30),(30,31),(31,32),`
`  (4,33),(33,34),(34,35),(35,36),(36,37),`
`  (33,38),(38,39),(39,40),(40,41),`
`  (33,42),(42,43),(43,44),(44,45),`
`  (33,46),(46,47),(47,48),(48,49),`
`  (33,50),(50,51),(51,52),(52,53)]`
`  self.edge = self_link+neighbor_link`
`  self.center=1`
	將資料點數量設定為 54 個，並在定義每個點的自我連接後定義空間邊的連接。

3. ST-GCN 模型訓練與結果驗證

為驗證模型辨識能力，本研究將每個動作之測試集中包含 7 個動作之 112 部影片作為驗證資料來源。ST-GCN 模型在進行卷預測時，會將 4 個影格的骨架圖合成一個預測結果，因此，本團隊紀錄每個預測結果並製作混淆矩陣，計算公式如下：

$$Confusion\_Matrix_{ij} = \frac{N_{ij}}{\sum_j N_{ij}}$$

N_{ij}：實際為第 i 個組裝步驟而預測結果為第 j 個組裝步驟的數量

將每一種動作的每一個預測結果經上述公式計算後，得出每一種預測結果佔每一種動作的比例，對角線即單一動作召回率（Recall）。混淆矩陣如表 5。

表 5、單一動作辨識混淆矩陣

		Predicted						
		1	2	3	4	5	6	7
Actual	1	80.129%	0.000%	0.000%	8.682%	0.000%	0.000%	31.190%
	2	9.605%	85.429%	12.712%	4.802%	1.977%	1.695%	6.780%
	3	11.416%	0.000%	75.572%	10.163%	0.907%	0.000%	31.942%
	4	5.674%	1.064%	6.738%	84.794%	15.957%	1.418%	0.355%
	5	8.333%	0.000%	1.087%	4.710%	88.783%	1.087%	0.000%
	6	2.644%	5.048%	3.846%	20.933%	0.962%	80.231%	10.337%
	7	12.596%	0.641%	4.327%	8.013%	0.000%	0.481%	77.942%

為能夠更深入分析模型辨識能力，本研究亦利用下列公式得到了本模型之準確率（Accuracy）。

$$Accuracy = \frac{\Sigma_i \Sigma_j (N_{ij} \times k)}{\Sigma_i \Sigma_j N_{ij}}，其中\ k = \begin{cases} 1, & if\ \ \mathrm{i} = \mathrm{j} \\ 0, & if\ \ \mathrm{i} \neq \mathrm{j} \end{cases}$$

經計算，此模型之準確率為 84.5%，而後分析每一動作之辨識狀況，亦針對辨識結果計算召回率、精確率（Precision）、F1-Score 指標作為模型效能探討之評估依據，計算結果如表 6。

表 6、各動作辨識率比較

#	Recall	Precision	F1-Score
1	80.129%	35.283%	70.008%
2	85.429%	88.755%	87.060%
3	75.572%	64.052%	69.336%
4	84.794%	39.271%	63.120%
5	88.783%	79.322%	83.786%
6	80.231%	92.000%	85.713%
7	77.942%	53.919%	63.742%

二、整合雲端服務與語音助理

(一) 整合 AWS 雲端服務

本研究導入亞馬遜網路服務（Amazon Web Service, AWS），結合AWS IAM、Lambda Function以及S3 Storage，並將AWS存取區域設定為美國東部（維吉尼亞北部）us-east-1，供本地端骨架辨識結果輸出至AWS中。

1. AWS IAM 權限設定

(1) 使用者建立

透過建立使用者 ASK-CLI 取得存取金鑰，使本地端可透過該金鑰進行 AWS 服務的存取，並設定連結政策存取 Lambda、S3 及 Alexa 之 FullAccess，將許可存取之 AWS 服務連結至 IAM，供後續環境設定使用，設定過程如圖 13 所示。

圖 13、IAM 使用者 ASK-CLI

(2) 角色建立

建立 IAM 角色 simsla，供後續欲使用之 AWS 服務進行連接，透過連結政策之設定，使得 IAM、Lambda、S3 及 Alexa 之 FullAccess 可互相串聯存取，如圖 14。

圖 14、IAM 角色 simsla

2. AWS S3 資料庫設定

建立儲存貯體（Bucket）simsla，連接前述 IAM 角色 simsla，供後續本地端運算結果能即時上傳至 S3。

圖 15、S3 儲存貯體 simsla

3. AWS Lambda 函式建立

建立名為 happy_Lambda 之 Lambda 函式，連接前述 IAM 角色 simsla，執行語言為 Python3.6，並串聯 S3 及 Alexa Skills Kit。此函式主要的用途為存取 S3 運算結果，將運算結果轉換成語音助理回覆形式，如圖 16 所示。

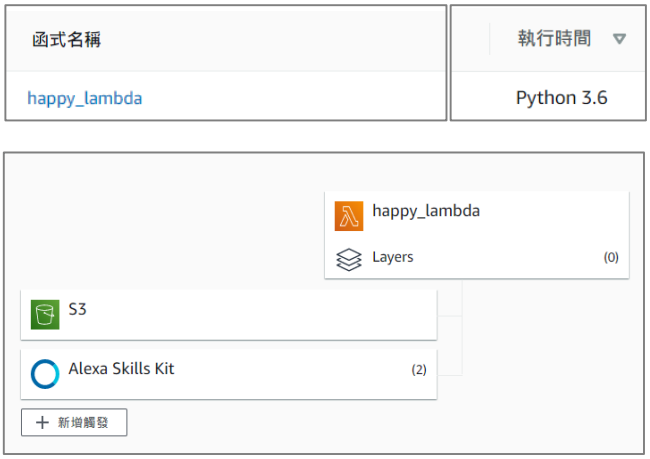


圖 16、Lambda 函式 happy_Lambda

表 7、happy_Lambda 之關鍵程式碼說明

關鍵程式碼	步驟說明
`#-----建立s3------` `s3 = boto3.client('s3')`	建立與 S3 的連線。

``# 問數量
def get_amount_response():
    data = s3.get_object(Bucket='happyla', Key='amount.txt')
    contents = data['Body'].read()
    contents=str(contents, encoding = "utf-8")
    session_attributes = {}
    card_title = "amount"
    speech_output = "You have done"+ contents
    reprompt_text = "You never responded to the first test message. Sending another one."
    should_end_session = True
    return build_response(session_attributes, build_speechlet_response(
        card_title, speech_output, reprompt_text, should_end_session))```
	讀取 S3 Bucket 為 happyla 之 amount.txt 將運算結果轉換成語音助理輸出形式，作為組裝員欲詢問做幾個產品之回覆。

``# -----下個步驟的回覆------
def get_nextStep_response():
    speak=""
    if step ==0:
        speak ="take the red board."
    elif step ==1:
        speak ="install the first eye to the left hole."
    elif step ==2:
        speak ="install the second eye to the left hole and flip upside down."
    elif step ==3:
        speak ="install the red stick to the middle hole."
    elif step ==4:
        speak ="take the blue board and install the red stick to its middle hole."
    elif step ==5:
        speak ="install the front wheels to the blue board."
    elif step ==6:
            speak ="install the back wheels to the blue board and flip upside down."
    session_attributes = {}
    card_title = "NextStep"
    speech_output = speak
    should_end_session = True
    return build_response(session_attributes, build_speechlet_response(
        card_title, speech_output, reprompt_text, should_end_session))```
	將運算結果之 step 轉換成對應語音助理輸出形式，作為組裝員欲詢問下個步驟之回覆。

``# -----做錯什麼的回覆------
def get_wrong_response():
    speak=""
    if wrong ==0:
        speak ="1"
    elif wrong ==1:
        speak ="2"
    elif wrong ==2:
        speak ="3"
    elif wrong ==3:
        speak ="4"
    elif wrong ==4:
        speak ="5"
    elif wrong ==5:
        speak ="6"
    elif wrong ==6:
        speak ="7"
    session_attributes = {}
    card_title = "Process"
    speech_output = "You might be wrong after process"+speak+
                    ",if you want the correct video, please say open trigger C M D."
    should_end_session = True
    return build_response(session_attributes, build_speechlet_response(
        card_title, speech_output, reprompt_text, should_end_session))```
	將運算結果之wrong轉換成語音助理輸出形式，作為組裝員組裝錯誤的回覆，並提醒欲觀看正確組裝流程影片，可以呼叫 Trigger CMD。

``# -----對應Alexa設定之Intent-----
def on_intent(intent_request, session):
 	
    intent = intent_request['intent']
    intent_name = intent_request['intent']['name']
 	
    if intent_name == "test":
        return get_test_response()
    elif intent_name =='Amount':
        return get_amount_response()
    elif intent_name =='NextStep':
        return get_nextStep_response()
    elif intent_name =="Video":
        return get_video_response()
    elif intent_name =='Process':
        return get_wrong_response()
    elif intent_name == "AMAZON.HelpIntent":
        return get_welcome_response()
    elif intent_name == "AMAZON.CancelIntent" or intent_name == "AMAZON.StopIntent":
        return handle_session_end_request()
    else:
        raise ValueError("Invalid intent")```
	整合前述函式至對應 Alexa 設定之 Intent，當詢問 Alexa 對應 Intent 時，即回覆函式內的語音輸出結果。

4. 本地端環境建立

使用 AWS 提供之命令列界面(CLI)，首先於本地端終端機輸入安裝 AWS CLI 指令建立環境。其次，輸入存取金鑰及相關設定指令碼，建立與 AWS 服

務之連線。

表 8、建立雲端連線之關鍵程式碼說明

關鍵程式碼	步驟說明
`~$ pip install awscli`	於本地終端機安裝 AWS CLI。

`export AWS_ACCESS_KEY_ID=AKIAUYYI7WYGOQLJDWUT`
`export AWS_SECRET_ACCESS_KEY=9gmIYsSQYP2SVZeB`
`export AWS_DEFAULT_REGION=us-east-1`
	輸入存取權限相關設定指令。

(二) 整合 Alexa 語音助理

為達成語音助理回覆之特定目的，採用Amazon提供之Alexa Skills Kit (ASK)，其中包含 Application Programming Interface (API)、開發工具、規格文件與程式碼範例。

1. Alexa Skill 建立

使用 ASK 中之客製化 (Custom) 模板進行 Skill 的建立，以 Alexa-Hosted 作為後端輸出資源，並將 Skill 名稱命名為 happy toy，如圖 17。

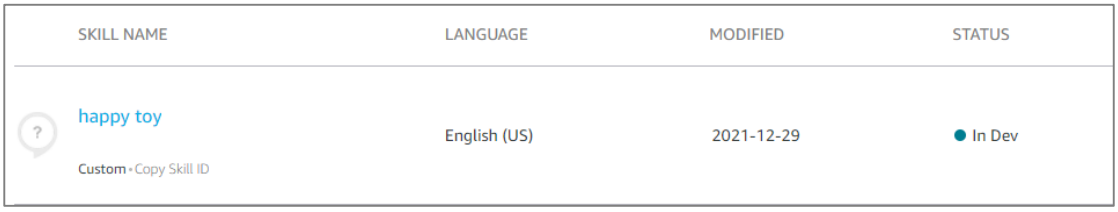


圖 17、Alexa Skill happy toy

2. Invocation 設定

Invocation 為呼叫 Alexa Skill 所需之喚醒關鍵字，將 Invocation 命名為 happy toy，如圖 18。當組裝員欲喚醒時，可說“Alexa, open happy toy”。

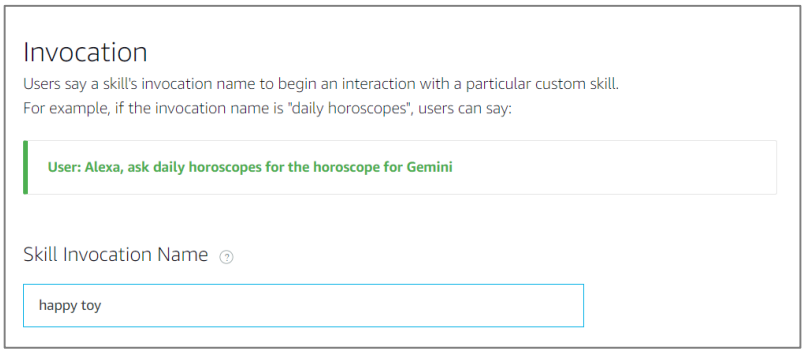


圖 18、Invocation 設定

3. Intents 設定

Intents 作為對話語句的分類，一個 Intents 可對應多種相同目標的語句，如前述 Lambda 函式關鍵程式碼，當呼叫 Intents 中語句時，即輸出其在函式中對應的回覆內容。Alexa Skill 預設四種例外狀況的 Intents，本研究建立三個 Intents，Amount、Process 及 NextStep，如圖 19，分別對應 Lambda 函式中的 get_amount_response()、get_wrong_response()及 get_nextStep_response()。

NAME	UTTERANCES	SLOTS	SLOT TYPE	ACTIONS
AMAZON.CancelIntent	-	-	Required	Edit
AMAZON.HelpIntent	-	-	Required	Edit
AMAZON.StopIntent	-	-	Required	Edit
AMAZON.NavigateHomeIntent	-	-	Required	Edit
Amount	2	-	Custom	Edit | Delete
Process	5	-	Custom	Edit | Delete
NextStep	8	-	Custom	Edit | Delete

圖 19、Intents 設定

4. 內容設計

與 Alexa 對話之語句皆基於規則（Rule-based），因此所有語句皆須經由人為設計，將各種可能的說法列出，以提高喚醒的成功率。本研究為三個 Intents，Amount、Process 及 NextStep 設計相關語句，分別如圖 20、圖 21

及圖 22 所示。

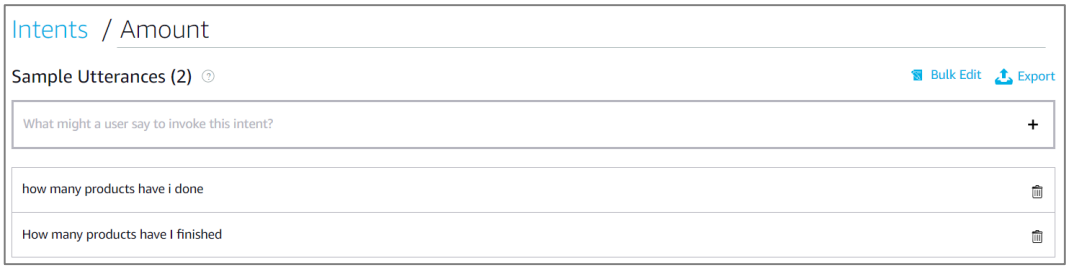


圖 20、Amount Intents 內容設計

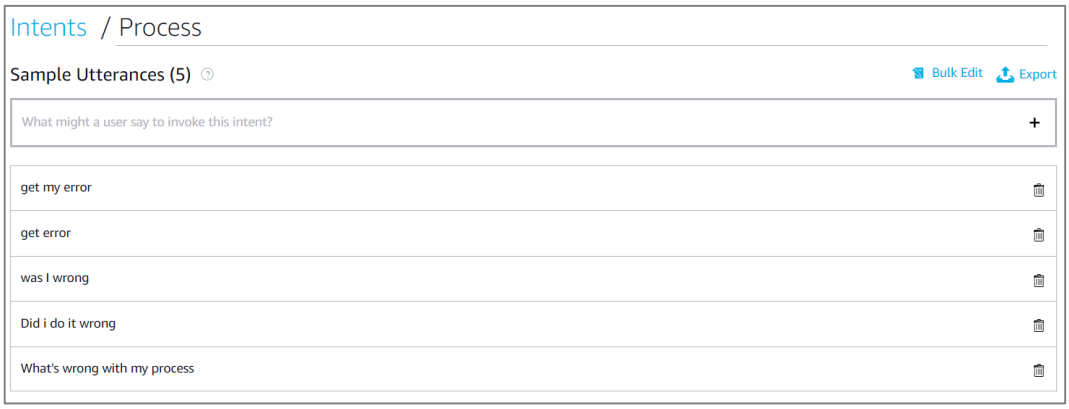


圖 21、Process Intents 內容設計

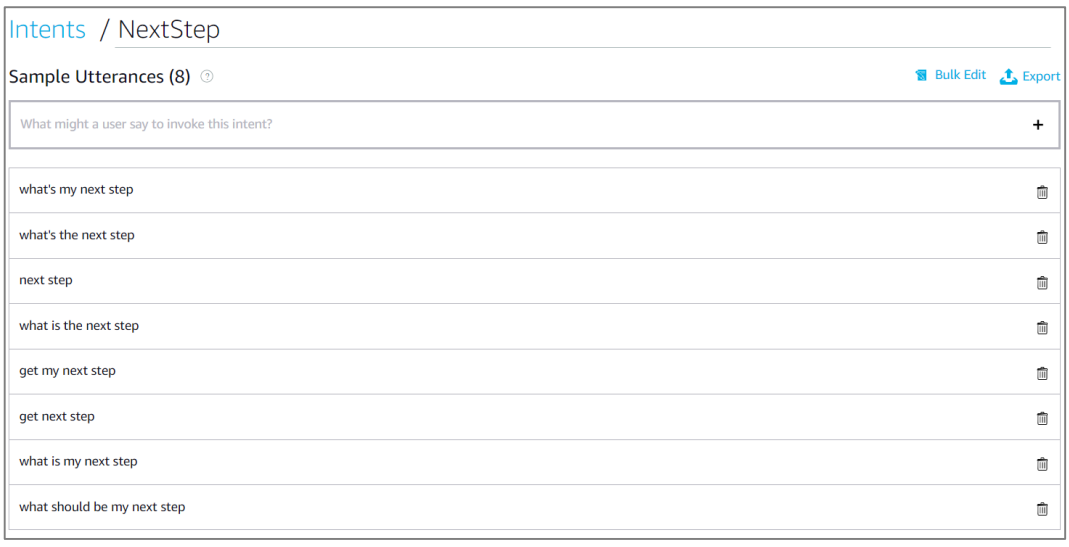


圖 22、NextStep Intents 內容設計

5. Alexa Skill 設定

將 Alexa Skill 連接前述完成之 Lambda 函式，並將 Lambda 函式中的 ARN 輸入至 ASK 的 Endpoint 中，如圖 23 所示，並將 Alexa Skill 的 ARN

輸入至 Lambda 函式中，如圖 24 所示。

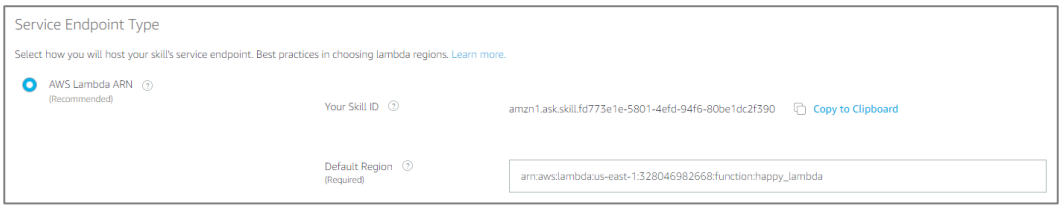


圖 23、ASK Endpoint 設定

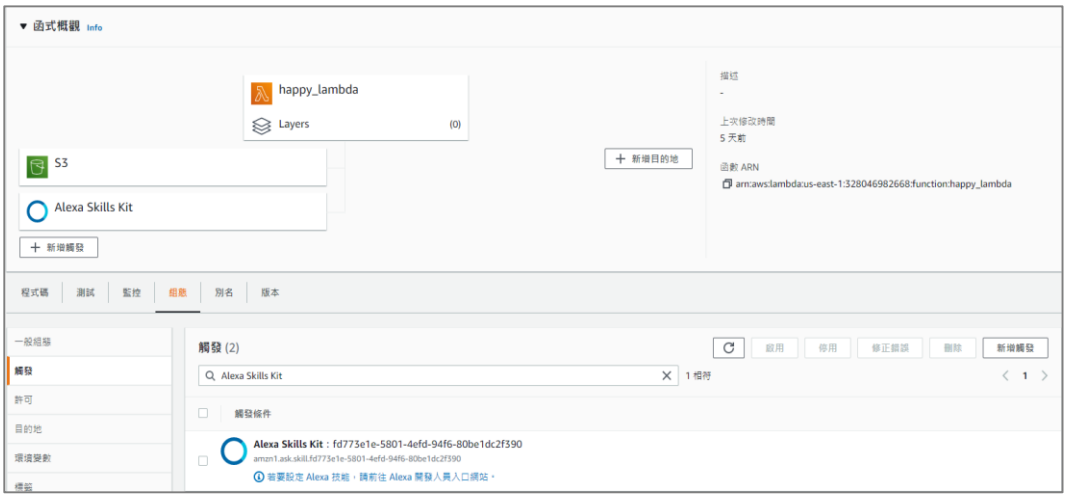


圖 24、Lambda 函式設定

6. Alexa Skill 測試

使用 ASK 提供之 Alexa Skill 測試模擬器，以語音及文字呈現測試結果。語音助理使用客製化技能之回覆結果如圖 25 所示。

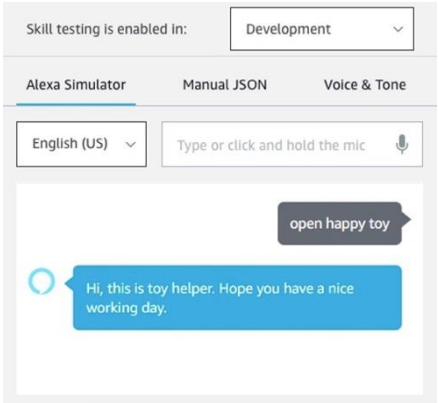


圖 25、Alexa Skill 測試結果

伍、結論

一、整體貢獻

本研究建立了一套由電腦視覺技術驅動、語音助理為互動介面之人機協同系統，能使組裝員於工作期間喚醒語音助理，並獲得工作效能之資訊。

二、研究限制

礙於時間與硬體設備限制，此次的研究未考量攝影機架設在不同角度下對模型訓練的結果差異，且因AWS有付費機制，本研究並未實現即時互動的研究成果。

三、應用方面

任何組裝環境現場，可透過標準動作制定後錄製影片，投入本研究模型，並結合語音助理，一可作為組裝現場之組裝員的輔助工具，二可提供管理者導入該系統作為新進人員之教育訓練。

四、未來展望

綜上所述，未來可考慮多角度影像的資料集，針對資料集做更完善的處理。另外，也能針對雲端串聯的即時性進一步進行改善，以期能達到語音助理即時回覆的目的。

參考資料

[1] H.-Y. Huang. "GCN(Graph Convolutional Network)的理解."
https://purelyvivid.github.io/2019/07/07/GCN_1/ (accessed.

[2] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii. USA, July 22 - 25, 2017 2017, pp. 7291-7299.

[3] OpenCV. "OpenCV." https://opencv.org/ (accessed.

[4] Amazon. "Amazon AWS." https://aws.amazon.com/tw/ (accessed.

[5] CMU-Perceptual-Computing-Lab. "openpose." https://github.com/CMU-Perceptual-Computing-Lab/openpose (accessed.

[6] S. Yan, Y. Xiong, and D. Lin, "Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition," presented at the Thirty-Second AAAI Conference on Artificial Intelligence, Hilton New Orleans Riverside, NewOrleans, Louisiana, USA, February 2 - 7, 2018.