國立中興大學電機工程學系

碩士學位論文

透過單目相機在 Unity 還原 3D 行人

3D Human Motion Reconstruction in Unity with Monocular Camera

指導教授 : 林維亮 Wei-Liang Lin

研 究 生 : 陳泰瑋 Tai-Wei Chen

中華民國 一百零九 年 七 月

國立中興大學電機工程學系

碩士學位論文

題目(中文)：透過單目相機在 Unity 還原 3D 行人

題目(英文)：3D Human Motion Reconstruction in Unity with Monocular Camera

姓名：陳泰瑋　　學號：7107064337

經　口　試　通　過　特　此　證　明

論文指導教授　林維高

論文考試委員　林光浩

陳彩希

中　華　民　國　109　年　7　月　31

誌謝辭

其實當初有想過直接工作，沒有要讀研究所，但在家人和女朋友的鼓勵下，還是來了中興大學。也因為他們，這兩年我也確實學到了很多東西，不僅僅是學術，我覺得更多是與人相處，還有獨立解決問題的能力。

由於這屆教授只收了我一位學生，很多事情必須靠自己完成，要感謝修課時認識的同學，雖然來自別的實驗室、不同領域，但我們一起進步。在硬體方面的課，他們教我，程式設計我則拿手，若沒有認識這些朋友，我想我修課的過程不會這麼順利。

後來升上二年級，實驗室多了三個新血，氣氛也變得活絡起來，我覺有人互相幫助、督促是一件好事。以往我常常下午就會離開實驗室，但後來當大家都在的時候，我們時常一起待到很晚，碩二這年就過得很充實。

最後要感謝我的指導教授，我們教授其實很照顧研究生，常常交代完事情又擔心我們被交代的的工作壓得喘不過氣，且也會和我們一起研究，meeting 時互相分享新知，比起教授倒覺得比較像朋友。而雖然這間實驗室做的是 AI 領域，對於當初的我來說是一個未知的世界，但現在我覺得我沒有選錯。

摘要

本文使用 3D 姿態預測器來預測人類 3D 姿態。透過把這些 3D 姿態訊息組合成動作捕捉，我們就可以在 Unity 中以任何外觀重建人體運動。而其潛在的應用層面是蒐集緊湊的人類 3D 活動資訊。

Abstract

This paper using a 3D pose estimator to predict human 3D poses. By combining the pose sequence information as a motion capture, we could reconstruct the human motion in Unity with any appearance. A potential application is collecting a compact human 3D activity dataset.

目次

表目次

圖目次

第一章 序論

1.1 研究背景

近年來深度學習開始被廣為運用，不論是物件偵測、物件追蹤、自動駕駛、路徑追蹤等等，眾多領域都脫離不了大量的資料集，且需要的資料不僅僅是影像，甚至有雷射、光達(Lidar)等資訊。然而一個好的結果，除了完善的網路架構，訓練資料量的多寡往往也是一大因素。而人工取得大量的資料是一件費時費力的事，除了拍攝還需要對資料進行標註。而這些繁瑣的工作在遊戲引擎(Unity3D、Unreal)中，都可以輕易完成。因此我們想在虛擬世界中還原真實世界的人體運動資訊，這樣不僅有了現成的 Ground Truth，甚至可以套用任意外觀到已有的運動資訊上，藉此增加資料的多樣性。

1.2 Uniy3D 簡介

在眾多遊戲引擎中 Unity3D 不僅在畫面呈現上非常的擬真，更能搭配 C# 實現許多複雜的操作，甚至是現有的演算法。一些精緻的場景、人物模型都可以在 Unity Asset Store 取得(圖 1)。而畫面中每個物件的位置，甚至是人體上某個關節點，在 Unity 中都可以直接拿來作為 Ground Truth 使用，不須再人工標記。

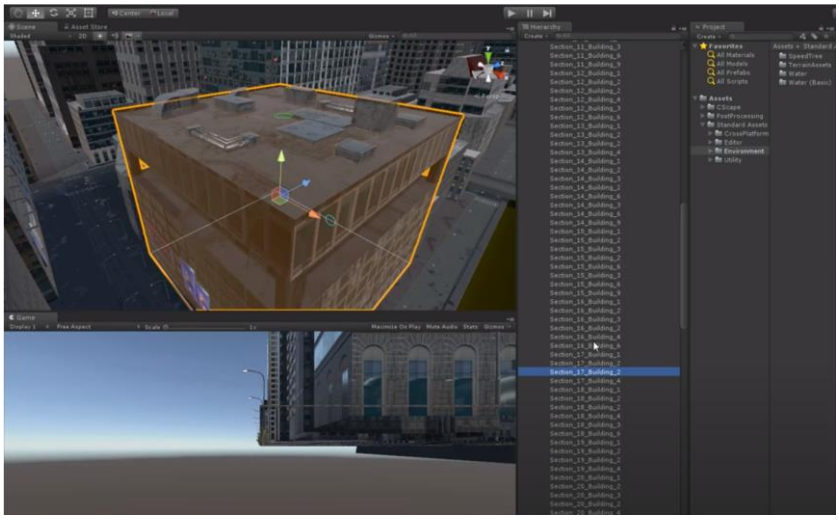

圖 1 Unity Asset Store 中的都市模型

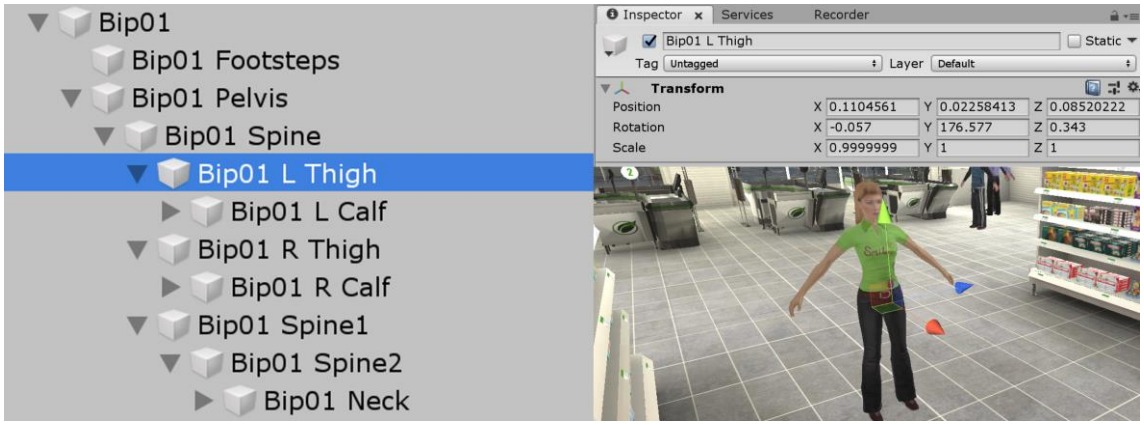

圖 2 Unity 人物模型的骨架資訊

圖 2 展示的座標資訊會隨著人物的運動即刻變化，在蒐集資料的時只需搭配簡易的程式便可輕鬆取得 Ground Truth，加上人物外觀可以隨意替換，因此我們為了製造含大量資料的資料集，便開始了「重現人物動作」這項工作。

第二章 相關工具介紹

2.1 AlphaPose 姿態估計[1]

為了得到人體運動的資訊，姿態估計是一項不可或缺的工作，首先需要做的前處理就是得到 2D Pose。現今 2D 姿態估計的技術已經相當成熟，已經許多開源的系統如 OpenPose[2]、Mask-RCNN[3]、AlphaPose 等等，本篇之所以選用 AlphaPose 是因為其準確率及處理速度都優於其他兩者。

圖 3 為 AlphaPose 的網路架構，不同於 OpenPose Botton-up 的作法，它採用 Top-down，也就是先偵測框，得到行人的 proposal 後在做姿態估測，這樣的好處是讓結果更精準，但速度會較慢。

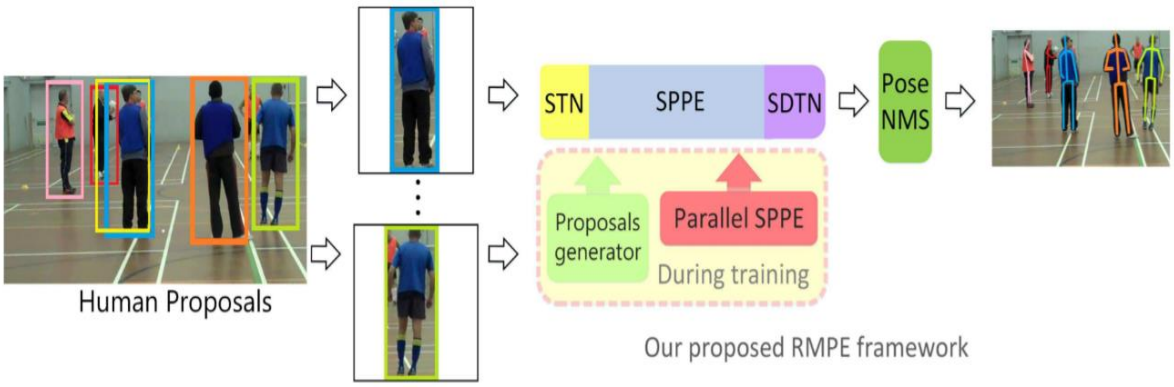

圖 3 AlphaPose 網路架構

而 AlphaPose 之所以用了 Top-down 運算還能夠這麼快，是因為現在的版本使用 Pytorch 框架，torch 產生的 tensor 放在 GPU 中運算，等同 Numpy 的 array 在 CPU 中運算，運算速度上自然有了落差。而大家已經習慣的 Numpy 格式也能和 troch 的格式自由轉換。

2.1.1 AlphaPose 輸出結果

AlphaPose 影像輸出如圖 4。

圖 4 AlphaPose 輸出結果

輸出結果包含照片 ID，人物 ID，關節點，評分四項資訊。其中關節點又細分成 17 項(圖 5)。

```
[
 // for person_1 in image_1
 {
    "image_id" : string, image_1_name,
    "category_id" : int, 1 for person
    "keypoints" : [x1,y1,c1,...,xk,yk,ck],
    "score" : float,
 },
```

```
{0,  "Nose"},
{1,  "LEye"},
{2,  "REye"},
{3,  "LEar"},
{4,  "REar"},
{5,  "LShoulder"},
{6,  "RShoulder"},
{7,  "LElbow"},
{8,  "RElbow"},
{9,  "LWrist"},
{10, "RWrist"},
{11, "LHip"},
{12, "RHip"},
{13, "LKnee"},
{14, "Rknee"},
{15, "LAnkle"},
{16, "RAnkle"},
```

圖 5 AlphaPose 輸出格式

2.1.2 Oks 指標

不同於物件偵測以 IOU 來計算 mAP，在姿態估計方面採用另一種指標，Object keypoint similarity，對預測出的關節點位置與標註位置之間的相似度進行評分(表 1)。

開源系統	準確率(mAp)	Fps (1080 Ti)
OpenPose	60	10
Mask-RCNN	67	5
AlphaPose	**71**	**20**

表 1 三種框架在 COCO 資料集上的表現

2.2 人物追蹤

要還原連續的動作，前提是我們從每個 frame 所取出的 3D 資訊是同一個人的，為了達到此目的，追蹤的技術是不可或缺的。

2.2.1 PoseFlow

PoseFlow[4]是一種 Tracking by detection 的技術，它基於 AlphaPose 的結果先計算出行人的 bounding box，再結合前後數個 frame 的信息來完成追蹤。然而它存在一項缺點，做追蹤時人物，若兩人在攝影機前重疊後又分開，此演算法會將兩人視為新的行人，並產生新的 ID，這對我們還原人體動作有很大的影響(圖 6)。

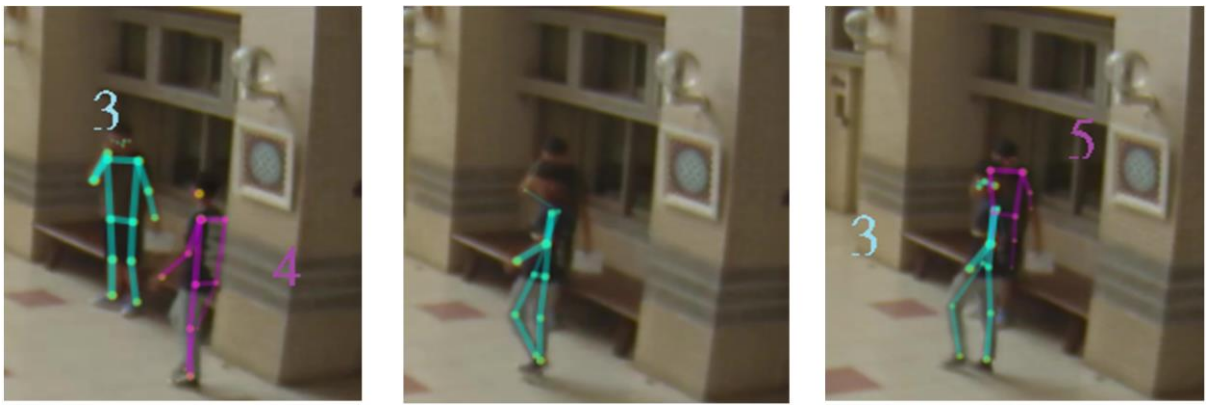

圖 6 人物交錯導致 Tracking 產生誤差

2.2.2 MOTDT [5]

MOTDT 也是一項結合物件偵測的追蹤技術，藉由比較追蹤預測的結果和物件偵測的結果，進行評分，這樣做不僅較精確也能解決行人重疊而造成 ID 丟失的問題(圖 7)。

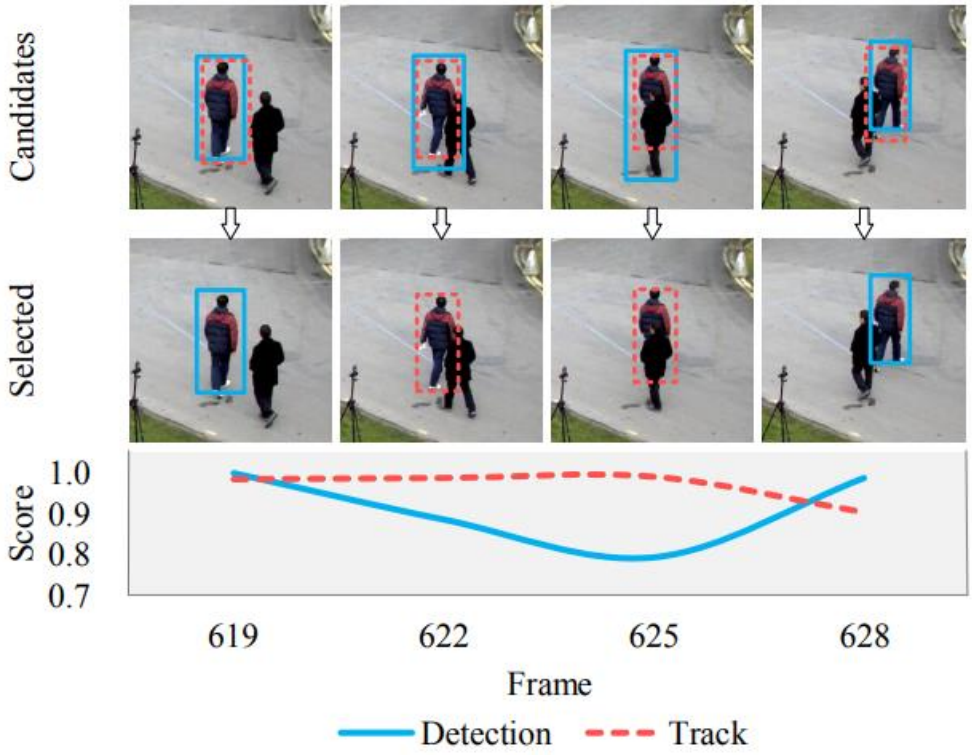

圖 7 MOTDT 演算法在人物重疊時的表現

由於軌跡(Trajectory)是由 bounding box 關聯組成，MOTDT 演算法引入卡爾曼濾波器預測下一個 frame 人物的位置，在一定程度上解決 bounding box 不準確及遮擋的情況。而軌跡是由 Tracklet(軌跡段)組成，卡爾曼濾波器只利用最後一段 Tracklet，這邊也針對了卡爾曼濾波器的準確度制定了一個軌跡可信度S_{trk}（式 1）。

$$S_{trk} = \max(1 - log(1 + \alpha L_{trk}), 0) * \mathbf{1}(L_{det} \geq 2) \qquad \text{式 1}$$

其中L_{det}指的是關聯到此 Tracklet 的 detection 個數，其中$\mathbf{1}$是一個函數，達成條件則回傳 1，否則回傳 0，若這個軌跡段中透過偵測框得到的資訊太少(小於兩個 frame)，則可信度為 0。而L_{trk}指的是此 Tracklet 中最後一個 detection 關聯後來自 track prediction 的個數。最終的評分方式如式 2 所示

$$s = P * (\mathbf{1}(\boldsymbol{x} \in B_{det}) + \mathrm{S_{trk}}\mathbf{1}(\boldsymbol{x} \in B_{trk})) \qquad \text{式 2}$$

$\mathrm{B_{det}}$和$\mathrm{B_{trk}}$分別代表 bounding box 來自物件偵測或濾波器的預測，P表示 ROI 的分類機率，本文最後選用此方法，得到行人追蹤結果以及像素座標的移動軌跡，詳見 3.4 節。

2.3 3D Pose Estimator

現今已經有許多 3D 人物重建的論文，但大部分著重於使用 SMPL[7]模型還原出人物的網格(mesh)，甚至是衣著[9][10]。但我們需要的僅是 3D 關節，為了增加資料量我們只會用到骨骼運動資訊，並將骨骼換上任意的外觀。我們採用[8]提出的 3D Pose Estimator，並捨去重建 mesh 的部分提升運算速度，圖 8 是示意圖。

圖 8 3D 骨架預測框架

2.4 相關動畫格式和建模軟體介紹

如果有一連串的 3D joints，我們就可以把它組合成連續的動作，而

其中儲存的格式非常多樣，本文會用輕量化的 BVH 格式作為儲存，再轉換成 Unity 中能運作的 FBX 格式。

2.4.1 BVH 格式

此格式分為兩個部分，前半段是對於骨骼的定義，描述了各個關節點之間的層級關係還有初始位置。其中初始位置又包含了座標，以及旋轉角度兩項(圖 9)。

```
HIERARCHY
ROOT Hips
{
    OFFSET  0.00    0.00    0.00
    CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
    JOINT Chest
    {
        OFFSET   0.00    5.21    0.00
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT Neck
        {
            OFFSET   0.00    18.65   0.00
            CHANNELS 3 Zrotation Xrotation Yrotation
            JOINT Head
            {
                OFFSET   0.00    5.45    0.00
                CHANNELS 3 Zrotation Xrotation Yrotation
                End Site
                {
                    OFFSET   0.00    3.87    0.00
                }
            }
        }
```

圖 9 骨骼初始位置定義

後半部分則是數據，決定了該動作包含幾個 frame，以及每個 frame 播放的時間，且描述了各關節之間的偏移和旋轉(圖 10)。

```
MOTION
Frames:    2
Frame Time: 0.033333
 8.03    35.01   88.36  -3.41    14.78  -164.35
97.95  -23.53  -2.14   -101.86 -80.77  -98.91
-1.57    0.69    0.02    15.00   22.78  -5.92
0.00   -23.95   0.00
 7.81    35.10   86.47  -3.78    12.94  -166.97
93.12  -9.69   -9.43    132.67 -81.86   136.80
-1.63    0.95    0.03    13.16   15.44  -3.56
0.00   -25.93   0.00
```

圖 10 關節點運動數據

2.4.2 Blender

Blender 是現在一個被廣為運用的 3D 建模軟體，它支援各種 3D 模型格式的匯入以及轉換(圖 11)，且可配合 Python 完成許多建模時重複性高的動作，相當方便。

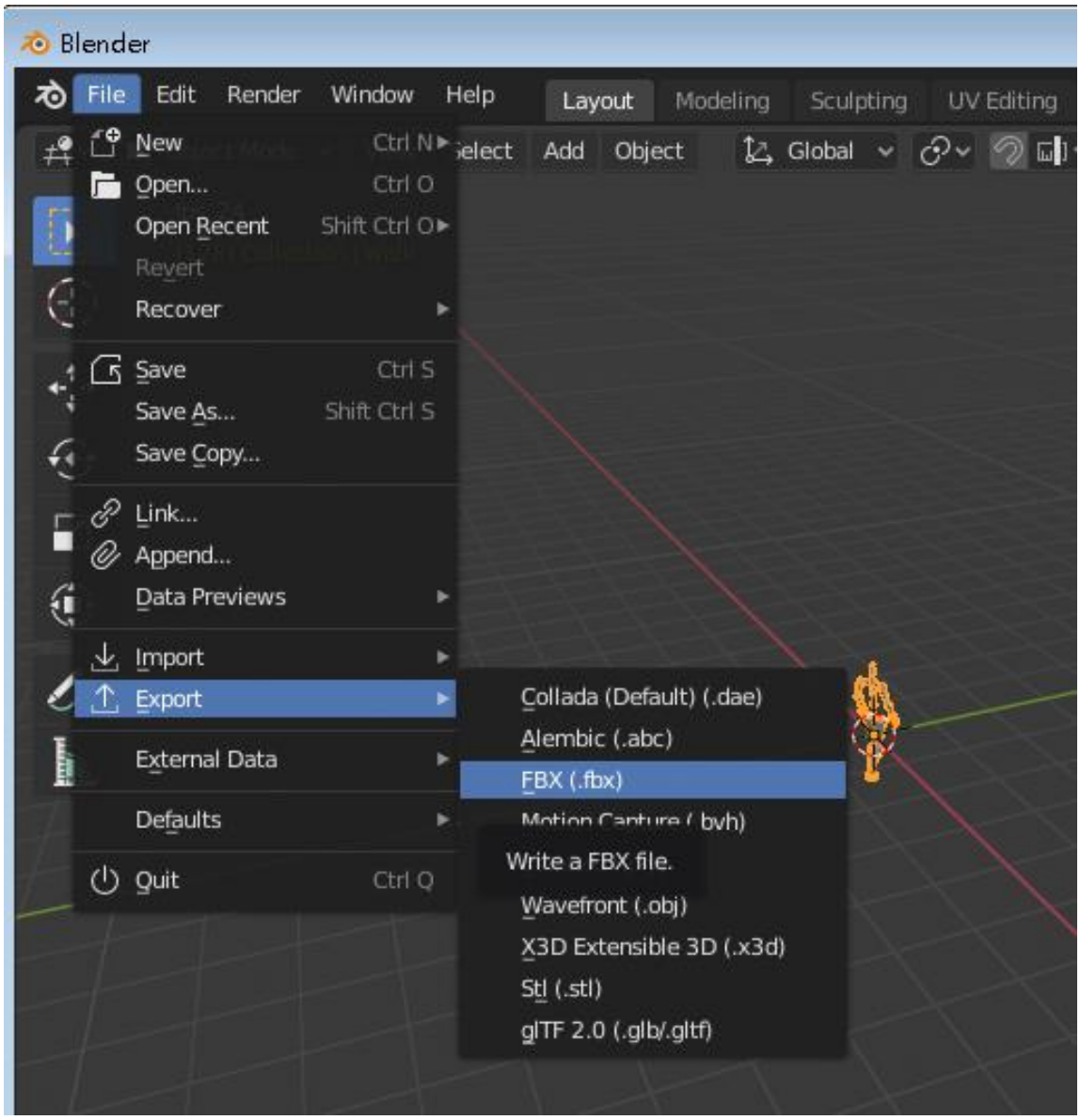

圖 11 Blender 轉換各種格式

2.4.3 FBX 格式

FBX 格式可用於 Unity，不同於 BVH 只有骨骼定義和運動資訊，它還能包含人物的網格(mesh)、紋理(texture)等等，但從 BVH 轉換過去的 FBX 並不會有這些資訊。而在 Unity 中使用的座標系較特別，是左手座標系統，但 BVH 是右手座標系統，所以透過程式進行轉換時要特別注意(圖 12)。

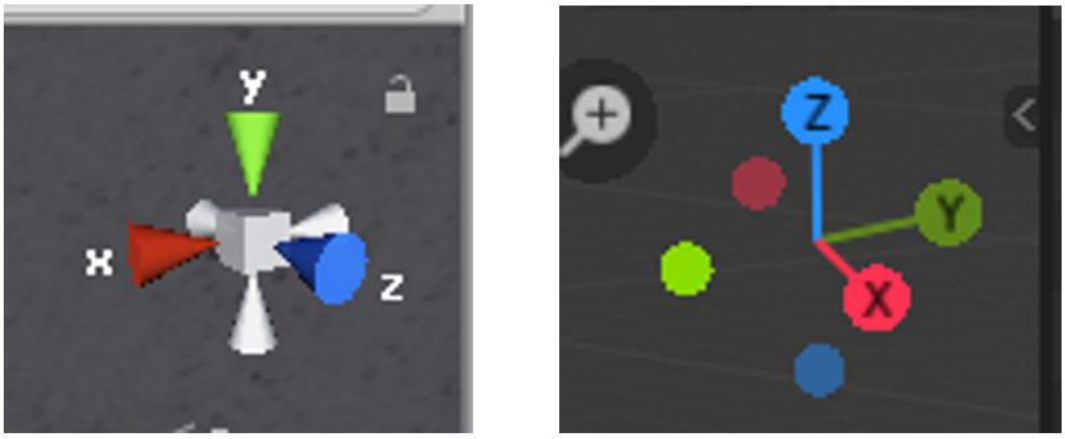

圖 12 座標系統差異，左為 Unity，右為 Blender

2.5 張氏相機標定[11]

我們要做的事就是從相機拍攝到的圖像訊息，得到物體對應真實世界的 3D 訊息。而其中相機的內外參數就是關鍵，透過相機標定不僅可以得到相機的內外參數，甚至能矯正透鏡的畸變。

2.5.1 座標系介紹

各個座標之間的對應關係如圖 13 所示。

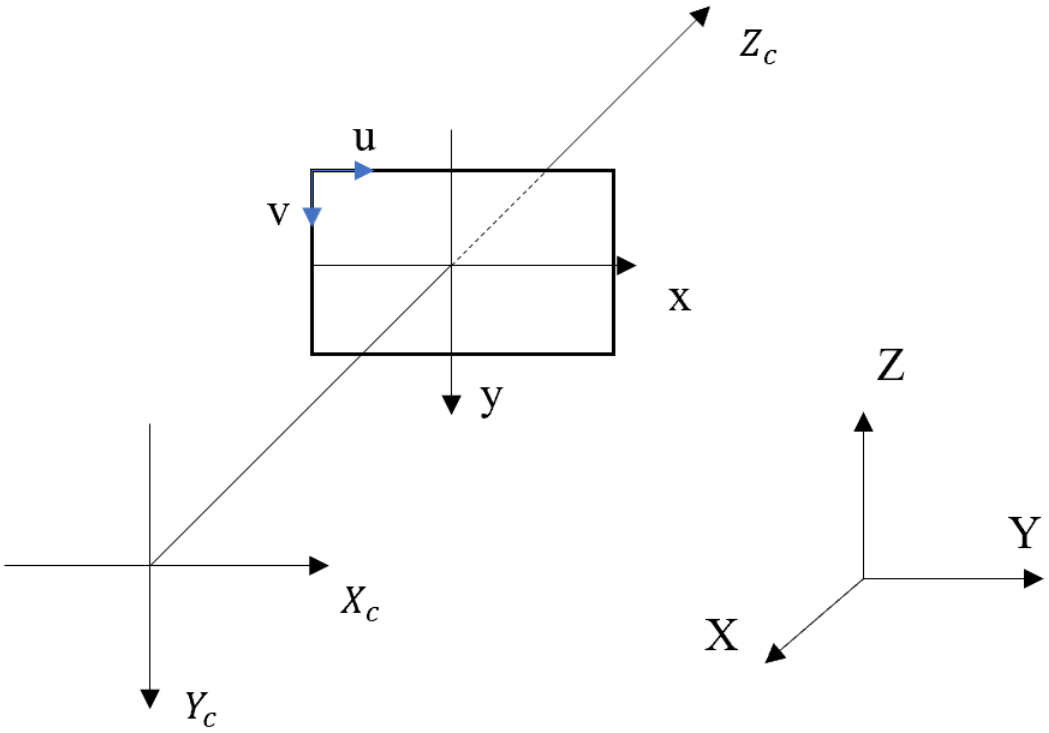

圖 13 各座標系之間示意圖

2.5.1.1 世界座標系(X Y Z)

自行定義的三維座標系統，用來代表目標物在真實世界的位置，單位通常用公尺(m)。

2.5.1.2 相機座標系(X_C Y_c Z_c)

在相機上建立的座標系統，主要用來從相機的角度描述物體的位置，單位一樣也是用公尺(m)。

2.5.1.3 圖像座標系(x y)

為方便描述物體從相機座標系的投影關係，因此定義了此座標系，用以得到像素座標，單位公尺(m)。

2.5.1.4 像素座標系(u v)

我們真正從相片上讀取到的信息，單位為個(像素)。

2.5.2 座標系之間轉換

關於座標系之間的轉換，會開始引入矩陣運算，我們從轉換公式往前看(圖 14)。

$$S\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{\mathrm{d}y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} * [R \mid T] * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

圖 14 像素座標系對應世界座標系的轉換公式

2.5.2.1 世界座標轉相機座標

透過旋轉矩陣 R 和平移矩陣 T 轉換世界座標和相機座標，做法相對直觀，其中擴展矩陣[R|T]為 3*4 的矩陣。

2.5.2.2 從相機座標到圖像座標

這一步就是簡單的針孔成像，此時的轉換矩陣只跟焦距 f 有關。

2.5.2.3 從圖像座標到像素座標

這裡的d_X d_y代表每個像素點在圖像座標系的尺寸，圖像座標的原點在像素座標下為u_0 v_0，最後一樣透過轉換矩陣反推像素座標。而當我們假設要觀測的物體通過世界座標 X-Y 平面，可以忽略擴展矩陣的第三行及 Z 座標，因為此時我們已經把物品假想於世界座標系下的地面上了。故整個公式可以簡化成像素座標 u,v 到世界座標 X,Y 的轉換，而 Z 座標為 0。

第三章 資料蒐集與實驗流程

3.1 原始影像蒐集

本文使用的資料為電機大樓一樓的監視攝影機畫面，申請為時一周的監控紀錄做為研究用途。

3.2 事前工作

3.2.1 相機標定

印製 9 乘 5 的棋盤格作為標定板(圖 15)。

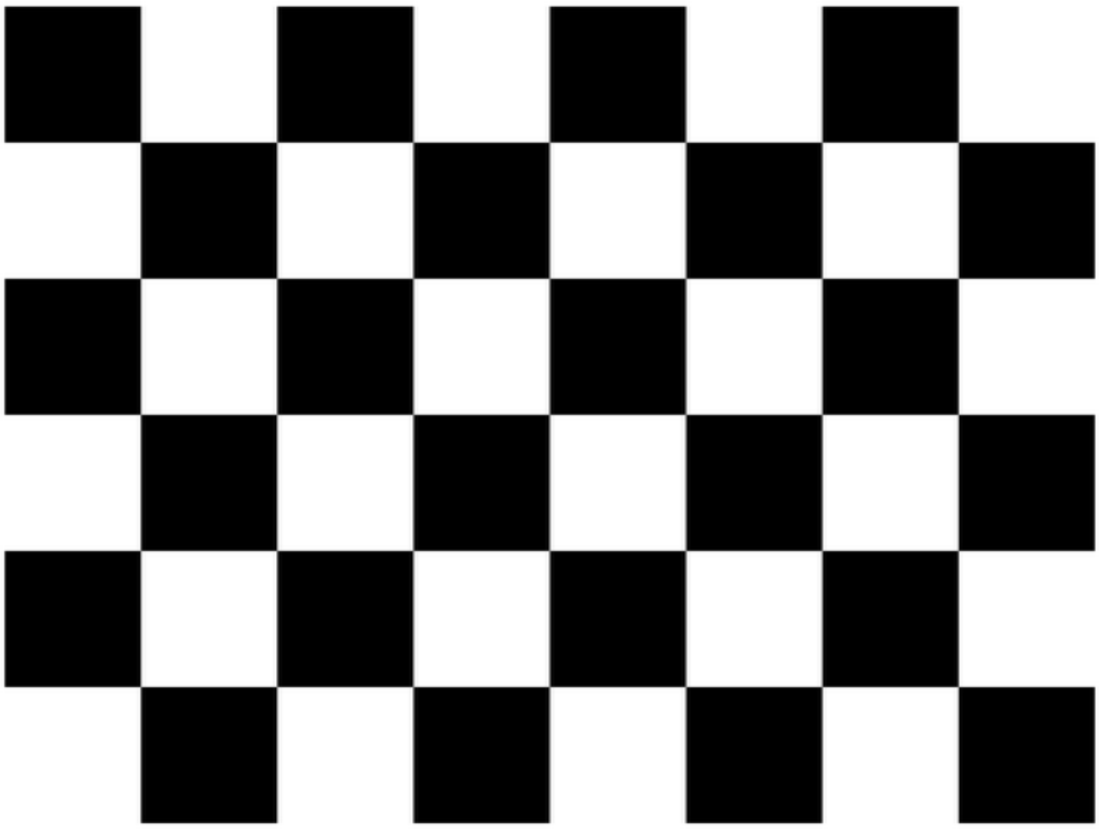

圖 15 標定用棋盤格

3.2.2 透過各個角度棋盤格計算相機內外參數

每個棋盤格都會對應一組相機外參數，由於我們要求的路徑在地面上，故最後選用地上的棋盤格，這樣便能簡化計算公式，忽略 Z 座標。

圖 16 各角度下的標定版

3.2.3 資料分割

對於蒐集到的影片，先將其全部裁切成照片。這一步可以透過程式設定好每秒幾個 frame，但由於我們事先不知道監視器的取樣率，所以不知道 fps 的情況下，本文選擇把所有的 frame 都分割出來。

3.3 AlphaPose 預處理

這一步主要用以得到 2D 資訊，並藉此計算行人的 bounding box，作為後續 tracking 演算法的輸入(圖 17)。

圖 17 從 Pose 算出行人 Bounding Box

此框的計算方法是根據2D關節點像素值而得,儲存方式用左上角座標、框的寬高來儲存。相比傳統的物件偵測如 Yolo[12]、SSD[13]等，以此法得到的 bounding box 不會忽大忽小。而 3D 動作還原需要輸入連續的影像，穩定大小的輸入對我們來說非常重要。

3.3.1 預處理結果的儲存格式

做 tracking 需要同時輸入原始照片和偵測結果,而偵測結果我們以數據儲存格式如圖 18，其中 id 為-1 ，表示沒有 tracking 的 ground truth，<x> <y> <z>則是用來描述 3D box 情況的中心，所以不會用到。

```
<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <x>, <y>, <z>
```

```
1,-1,1359.1,413.27,120.26,362.77,2.3092,-1,-1,-1
1,-1,571.03,402.13,104.56,315.68,1.5028,-1,-1,-1
1,-1,650.8,455.86,63.98,193.94,0.33276,-1,-1,-1
1,-1,721.23,446.86,41.871,127.61,0.27401,-1,-1,-1
1,-1,454.06,434.36,97.492,294.47,0.20818,-1,-1,-1
1,-1,1254.6,446.72,33.822,103.47,0.14776,-1,-1,-1
1,-1,1301.1,237.38,195.98,589.95,0.051818,-1,-1,-1
```

圖 18 Bounding box 結果儲存格式

接下來會用到的追蹤技術需要同時輸入原始影像、物件偵測結果，而物件偵測的結果這邊用 Pose 計算出的結果取代，經實驗發現這樣更能得到穩定的輸出，詳見 3.4 節的描述。

3.4 Tracking by Detection

3.4.1 輸入與輸出

輸入為 Object detection 結果、連續的影像。輸出為預測的 bounding box 和行人 ID。圖 19 左為 MOTDT[5]結果，中間是先前透過 AlphaPose 算出來的 bounding box，右為 AlphaPose 的骨骼資訊。左右腳踝上像素平均，得到連續的 pixel 路徑，相比於用 bounding box 底部 pixel 算出來的路徑還要準確。

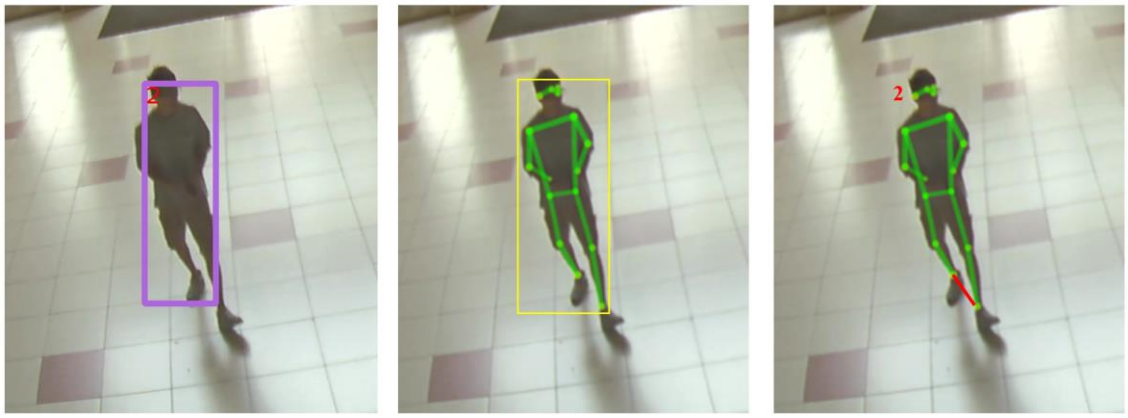

圖 19 從 tracking 結果回推腳踝座標

使用 MOTDT 演算法，得到追蹤後的人物 ID，同時記錄該 ID 對應的 bounding box，並將此 bounding box 與透過 pose 算出來的 bounding box 做比較，藉此反推回兩腳踝中心的像素，以此作為準確的 2D 路徑。

3.4.2 用 Tracking 結果分割行人

人物追蹤不僅是為了取得行人的 pixel 路經，同時也將不同 ID 的行人從原始影像中分割出來，準備做為 3D Pose Estimator[6]的輸入。分割出來的影像大小為 224*224，在相片大小固定的情況下，對於遠近不同的行人我們會根據一個尺度進行縮放，讓行人在畫面中的占比是固定的。而這個縮放的尺度則由 bounding box 的高(占 pixel 數)決定。我們希望行人的高度占用 150 個 pixels，所以這個尺度就是 150/bounding box 高。

這邊 bounding box 的高非常關鍵，由於我們 detection 的結果框是由 Pose 而得，所以結果一樣很穩定，若用傳統的 object detection 會出現 scale 忽大忽小的情況(圖 20)。

圖 20 用傳統 object detection 導致 scale 的誤差

3.5 預測 3D 關節點並還原動作

為了在 Unity 還原現實中的人，我們使用[6]訓練好的 model，並移除 mesh 重建、可視化等工作，僅保留 3D 關節點，增加其運算速度，圖 21 為大致流程。

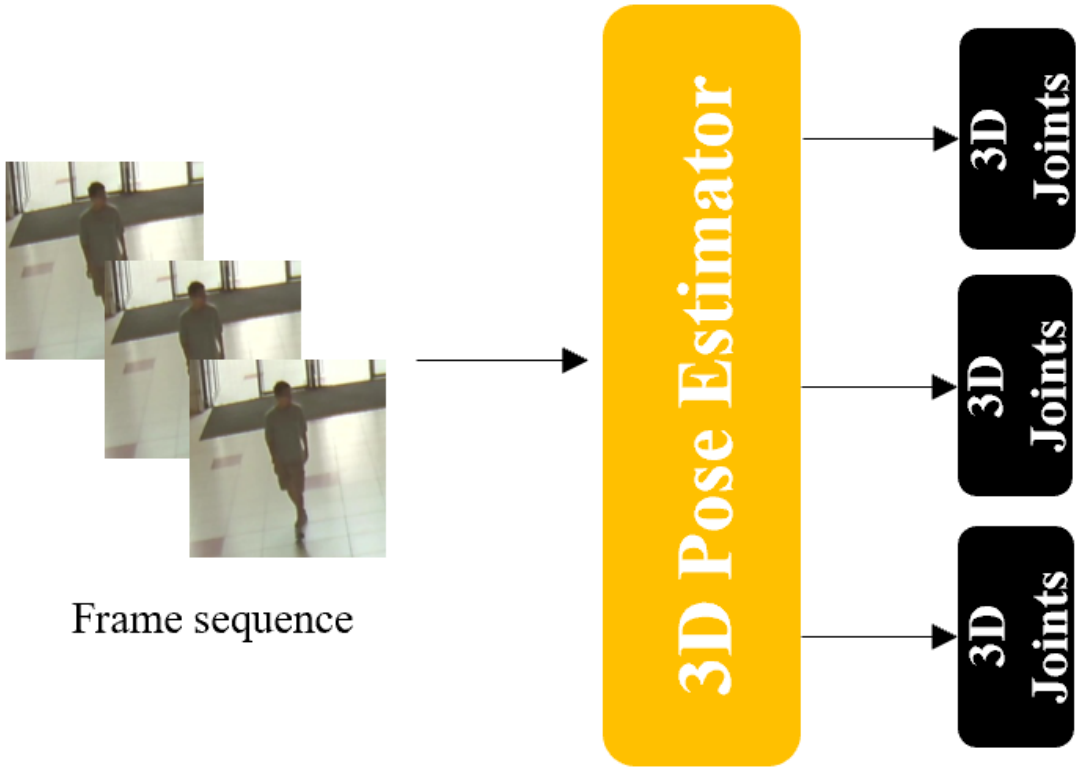

圖 21 3D Pose Estimator 輸入輸出示意圖

我們透過組合這些 3D Joints，可以還原一連串的骨骼運動資訊，值得一提的是，我們記錄下每個行人 ID 以及起始 frame 和結束的 frame，藉此作為動畫的檔名，便於之後在虛擬世界(Unity)重現。

3.5.1 透過 Blender 把關節點組合為 BVH 格式

Blender 可搭配 Python 操作，這裡透過 Python 讀取 3D Joints，且搭配事先定義好的骨骼位置(圖 22.23)，將每個 frame 產生的數據對應在正確的關節上，完成 BVH 格式的動畫，而這也是我們儲存數據的主要格式。

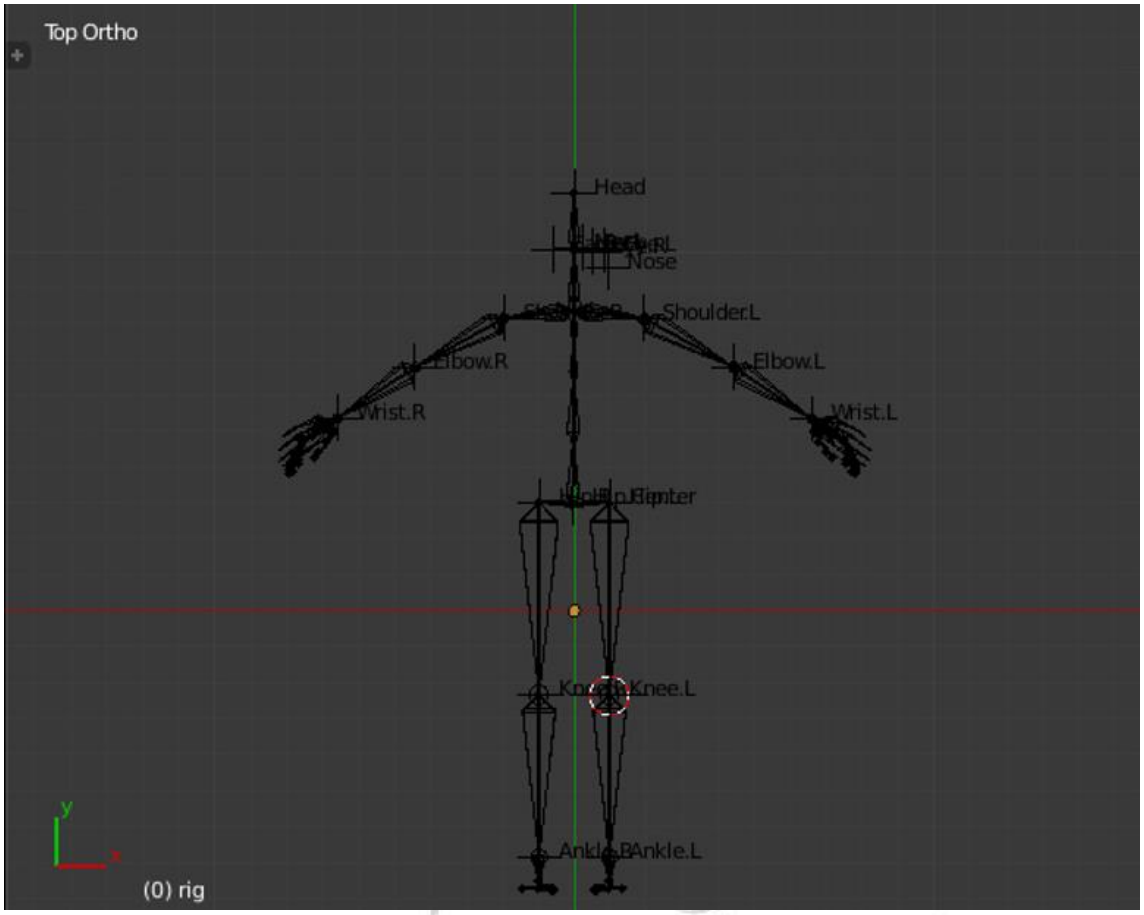

圖 22 事前定義好的 BVH 模板

圖 23 各項關節定義

3.5.2 透過 Blender 把 BVH 格式轉成 FBX 格式

如同 2.4.3 提到的，兩者座標系不同，因此在做轉換的時候，要確實指定好骨架的前方、上方分別對應哪個座標軸，如此轉換出來的動畫才不會出錯。表 2 比較了在 150 個影像中出現的三個人，在不同格式下儲存所占用的空間，可以發現 BVH 在沒有編碼的情況下已經大大減少了佔用空間。

	File size (Mb)	***Compression ratio (%)***	***3D information***
Photos from film (JPG)	26.1	-	No
BVH	1.1	**95.8**	Yes
FBX	1.4	94.6	Yes

表 2 各格式空間占用比較

而 BVH 和 FBX 兩種格式在功能上幾乎沒有差異，都包含骨骼定義、運動資訊，選擇使用 FBX 是因為它在套用外觀時較容易、且我們使用的遊戲引擎(Unity3D)只支援 FBX(圖 24)。

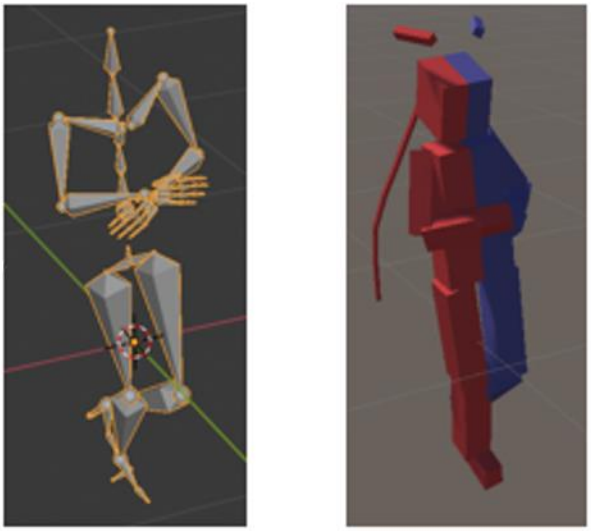

圖 24 左為 BVH、右為 Unity 中變成木偶的樣子

3.5.3 還原 3D 路徑

最後把 3.4 提到的路徑經由相機標定得到的內外參數矩陣，還原成 3D 路徑，如此便可在 Unity 重現完整動作。

第四章 實驗結果

4.1 動作還原的結果

由於本文用只有使用一個攝影機畫面，所以在多人的環境下，被遮擋的人物沒有辦法還原出 3D 座標，而在虛擬場景控制各個行人出現及消失的時機也是一個難題。

從不同角度觀察還原情形，發現有些行人與實際方向是相反的，其中的原因是單目相機受到光影影響，畫面並不清晰，導致 3D Pose Estimator 不能準確的判定人物的正反面，圖 25 展示正反面視角。

圖 25 在 Unity 中不同視角的結果

從圖中可以看出行人在背光的時候特徵變得很不行明顯，幾乎只剩輪廓的資訊，這是目前遇到的一項難題。

4.2 路徑還原的結果

這部分會受到相機標定結果影響，若相機標定結果準確則路徑準確，下表為座標轉換與實際值的誤差。

	實際位移(m)	預測位移(m)	誤差(%)
X 方向	4	3.16	21
Y 方向	1.5	1.05	29.3

表 3 行人路徑的誤差

這裡相機標定的結果不盡理想，我們用錄影的方式並在攝影機前移動棋盤，最後得到約 500 張可用的棋盤格相片進行標定，但在這 500 張影像中其實重複性略高，而一些邊角的部分較難偵測到、高處如牆上等地方也沒有角點資訊，這可能是導致誤差的原因。

隨後又考慮了座標轉換中 scale 的問題，人工校正後調整 scale 大小後，誤差減少至 6%以內。

	實際位移(m)	預測位移(m)	誤差(%)
X 方向	4	4.2	4.7
Y 方向	1.5	1.42	5.3

表 4 修正 scale 後誤差

第五章 結語與未來展望

我認為還原多人場景是一項很具有挑戰性的題目，當中牽涉到許多不同層面的工作，我學習了很多。結果方面還有很多可以改善的地方，像是 3D 動畫還原的精準度、被遮擋人物的 3D 關節預測、座標轉換的精準度等等。或許我們可以透過多鏡頭解決遮擋問題，又或者對影像做更多的前處理來解決監視器畫面不清晰等等，很多細節都值得思考。

本文最大的貢獻是完整了一個從多人影像到多人 3D 動畫的系統，其中有很多應用層面，像是多了時間軸的 4D 地圖，導航系統中行人的模擬等等，能延伸的範圍很廣，我認為都會是很好的題目。

參考書目

[1] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In ICCV, 2017.

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2018. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields.

[3] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322

[4] . Xiu, Y., Li, J., Wang, H., Fang, Y., Lu, C.: Pose flow: Efficient online pose tracking.

[5] C Long, A Haizhou, Z Zijie, and S Chong. 2018. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. ICME

[6] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In CVPR, 2018J

[7] M. Loper, N. Mahmood, J. Romero, G. Pons-moll, and M. J. Black, SMPL: A Skinned Multi-Person Linear Model, ACM Transactions on Graphics, vol.34, issue.6, pp.1-248, 2015.

[8] A. Kanazawa, M. J. Black, D. W. Jacobs and J. Malik, "End-to-End Recovery of Human Shape and Pose," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 7122-7131, doi: 10.1109/CVPR.2018.00744.

[9] hiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar,Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single RGB camera. In IEEEConference on Computer Vision and Pattern Recognition,pages 1175–1186, 2019.

[10] Chongyang Ma, Hao Li, and Shigeo Morishima. Siclope:Silhouette-based clothed people. In IEEE Conference onComputer Vision and Pattern Recognition, pages 4480–4490,2019.

[11] Z. Zhang, "A flexible new technique for camera calibration," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000, doi: 10.1109/34.888718.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection.In CVPR, 2016

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed.SSD: single shot multibox detector. CoRR, abs/1512.02325,2015.