國立高雄應用科技大學

100-101 年度獎勵科技大學及技術學院

典範科大計畫

產學及研究成果轉專題製作教材

-(The Kinect motion analysis principle Discussion)

Kinect 動作分析原理之探討

Kinect 動作分析原理之探討

(科系)(指導教授)(學生)
電機系 黃敬群老師 學生 陳義仁
黃柏輝

教卓計畫:優化教學品質與活
100C9097A

摘要

使用 Kinect 擷取人體,透過 Kinect 的深度資料取得輪廓,經過前處理使的輪廓更加穩定,接著使用水平投影辨識動作,之後由人體骨架中求得特徵向量,再經類神經網路的輸入輸出可辨識出人體姿態。

1. 動機及目的

在現今這忙碌的社會中,休閒娛樂是身心俱疲的我們一項不可或缺的舒壓活動,而其中體感遊戲可說是相當的熱門一種,在某次機緣巧合下,我們在同學家玩過 XBOX360 後,發現了大部分遊戲都是藉由對人體姿態的辨識來進行比對,判斷出所相對應的結果,並在遊戲中呈現出相對的動作反應,我們對這一部份感興趣,於是投入研究中。

研究中我們學習到應用 Kinect 來做人體動作的辨識,並應用背景相減法、骨架資訊、水平投影、類神經網路、特徵提取等技術,來辨識姿勢。當 Kinect 擷取到人體影像後,藉由 Kinect 深度的資訊,然後經過前處理(侵蝕或膨脹)取得較佳的人體輪廓。接著用 Kinect 骨架資訊取得人體的特徵向量,此特徵向量與人體深度資訊為類神經網路(Neural Network)的輸入;接著類神經網路的輸出便可辨識出人體姿態。如有相似特徵,會藉由水平投影,計算出人體長度與寬度比率再加以辨識,以提高辨識率。

2. 研究方法

推演流程:

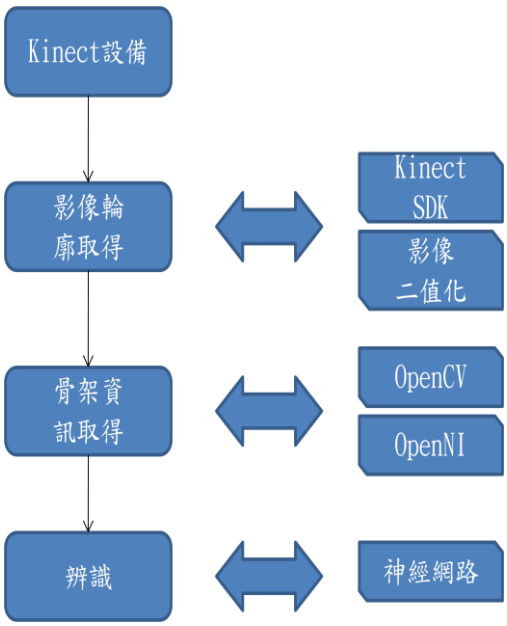


圖 1、理論推演流程圖

(1)Kinect 設備:

a、Kinect 硬體與原理介紹:

圖 2、Kinect 硬體

Kinect 可以取得下列資訊:彩色影像(中間的鏡頭)、3D 深度影像(左右 2 顆鏡頭)、紅外線發射器和紅外線 CMOS 攝影機(左右 2 顆鏡頭)、聲音(麥克風陣列)。

Kinect 也支援追焦功能,底座碼達會隨著人物轉動[15]。

b、Kinect 開發環境與設定:

在進行 Kincet 程式開發之前,需要準備的軟硬體如下:

作業系統:Windows7(x86/x64)
硬體:CPU:雙核心 2.66GHz 以上
RAM:2GB 以上
顯示卡:支援 DirectX 9.0c 以上
Kinect 感應器

軟體：Visual Studio 2010
或 Visual C# 2010 Express
NET Framework 4.0
Kinect SDK for Windows
安裝 Kinect 感應器：
1.確定 Internet 連線正常
2.接上 Kinect 外接電源
3.接上電腦 USB 接口
4.會自動下載必要驅動程式
安裝 Coding4Fun Kinect Toolkit：
這個工具包主要是將一些在開發 Kinect 應用程式時會使用到的程式碼整理成擴充方法，可在開發時減少程式碼的撰寫。

c、Kinect for Windows 架構：

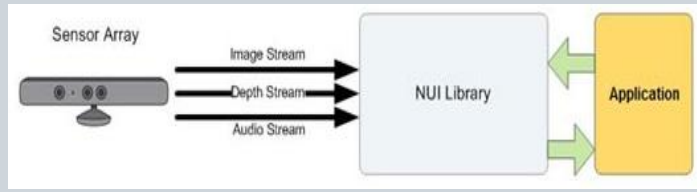


圖 2、Kinect 架構圖

Kinect 的 NUI 程式庫提供應用程式取得 Kinect 感應器傳送至主機的三種訊號串流，必須在初始化 API 時指定要接收哪幾種串流，彩色影像串流、深度影像串流、聲音串流。

d、NUI API 初始化：

要使用 Kinect API 接收感應器的資訊，是透過 Runtime 物件，因此 Kinect 應用程式的第一步就是建立一個 Runtime 物件來準備接收感應器的資料，並呼叫 Initialize 方法進行初始化(指定接收哪種類型的資料)，在應用程式結束時要呼叫 Uninitialize 方法，關閉 Kinect 設備。

(2)影像輪廓取得：

在輪廓比對上，我們必須要取得輪廓，而使用的方法為背景相減法，如連續影像相減法(Frame Differencing)等…，這種方法是使用事先建立場景做為參考背景影像，利用連續兩張或相隔一段距離之影像進行相減，本方法不易受到光線的影響，在動態的環境的偵測上有不錯的偵測效果；但在物體靜止時就無法找出其中的差異，所以我們尋求另外一種方法為背景相減法(Static Background Subtraction)，這個方法是使用事先建立好的背景影像和目前所輸入的影像做相減的動作，但容易受雜訊影響。之後我們修改 Kinect SDK 的應用程式與影像二值化，可以替代前者兩種方法，大幅減少我們擷取輪廓資訊的速度。

此專題的影像處理極為重要，一個未處理的影像可能含有雜訊、模糊不清…等問題，這些狀況不好的影像會大大的影響接下來系統的判斷。如果我們要得到一個良好的影像品質就必須加上一些處理，影像處理的流程大致上分為(影像前處理、影像二值化、型態學處理等)。

a.影像前處理：

所謂的灰階就是將影像強度表現出來，被灰階過後的影像會顯現出亮跟暗的兩種表現，影像中光源越強的地方顯現出的光源就會越強，反之則越暗。灰階影像通常以 8-bit 的整數來儲存，所以可表示出 256 個灰階。灰階值如果是 255 代表全白，而 0 表全黑。我們利用 Kinect 紅外線 CMOS 攝影機來取得影像如圖 3，再將影像灰階化。

圖 3、由 Kinect 紅外線 CMOS 攝影機所擷取之影像

b.影像二值化：

影像二值化是將經過灰階化的影像在做二值化的動作，所謂的二值化是將所有低於臨界值的像素都設為黑色，大於臨界值的設為白色，將影像二值化的好處可減少資料的儲存跟執行效率的提升。二值化的重點就在於臨界值的設定，隨著使用環境

的臨界值也跟著不同如果外在的環境光線太強臨界值的設定就必須提升,否則所產生的影像將會不方便辨識。

影像二值化的公式表示如下：

$$f(x,y)=\begin{cases}255, \text{IF } f(x,y)\geq T\\ 0 \quad, \text{IF } f(x,y)<T\end{cases} \text{ or } f(x,y)=\begin{cases}255, \text{IF } f(x,y)<T\\ 0 \quad, \text{IF } f(x,y)\geq T\end{cases} \quad (1)$$

臨界值調好之後接下來就是將影像二值化,如圖 4。

圖 4、二值化影像

c.形態學處理：

二值化後的影像只是把亮度不同部分區分出來而已並沒辦法消除整個影像中其他地方的雜訊,這些雜訊可能會影響往後影像處理的結果,消除影像中雜訊的方法很多，有移動平均法、中值濾波法…等。而我們利用型態學中的侵蝕跟膨脹，將影像中多餘的雜點去除除此之外如果將原影像跟侵蝕過後的影像做相減的處理的話還可得知影像的邊緣。

侵蝕(Erision)

侵蝕的目的是為了將影像中不必要的雜質去除掉，以本專題來說，因行車記錄器在夜間模式會利用紅外線做燈光補強之動作,進而影響到輸出影像，為了解決此現象，我們將紅外線遮住，但周圍還是會發射出一點微量的光源,造成影像在做二值化轉換時產生過多的雜點,而這些雜點會影響影像判斷時的結果,這時候我們就可以利用侵蝕的功能，將這些雜點去除以達到影像判斷的最佳效果。

膨脹(Dilation)

通常膨脹的用途在於將侵蝕過後的影像恢復成原來的影像,因為侵蝕過後的影像除了會使雜點消除外,還會造成影像中的物體可能會因侵蝕的作用隨之變小,所以做了幾次侵蝕,就要膨脹幾次彌補回來。

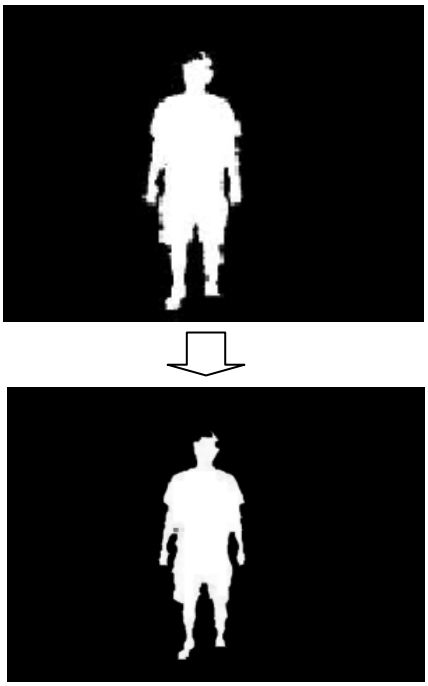


圖 5-1、 影像經由一次侵蝕之結果

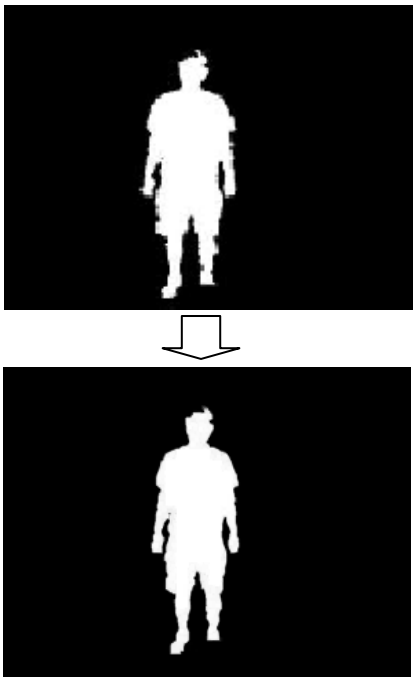


圖 5-2、影像經由一次膨脹之結果

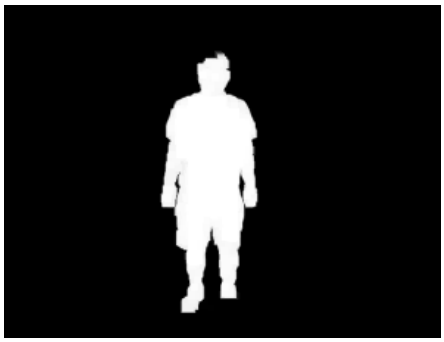


圖 5-3、影像經由先 2 次侵蝕後再膨脹之結果

透過圖 5-3 顯示，我們發現 2 次侵蝕後再膨脹的圖確實跟單純二值化的影像相比變得更佳的清楚而且邊緣的部分也很明顯的產生出來。

(3)骨架資訊取得：

OpenNI 的人體骨架基本上是由「關節」(joint)來構成的，而每一個關節都有位置（position）和方向（orientation）兩種資料[10]；同時，這兩者也都還包含了對於這個值的「信賴度」(confidence)，可以讓程式開發者知道 middleware 所判斷出來的這個關節資訊有多大的可信度。

圖 6、人形骨架與深度資訊

而在目前的 OpenNI 裡，他以列舉型別的方式總共定義了 24 個關節（XnSkeletonJoint）。不過雖然 OpenNI 定義了這麼多的關節，但是實際上在透過 NITE 這個 middleware 分析骨架時，其實能用的只有上面十五個（可以透過 xn::SkeletonCapability 的成員函式 EnumerateActiveJoints()取得可用的關節列表），再把這些關節連成線畫出來，結果就會是類似圖 6 最左上方的那張圖裡的樣子了。

要能夠在 OpenNI 的環境裡建立人體骨架，基本上是要靠所謂的 User Generator、也就是 xn::UserGenerator；而由於他的資料來源是深度影像（depth map），所以要使用的話，同時也要建立一個可用的 Depth Generator 出來才行。在 OpenNI 的文件中，有提及「Production Chain」這個概念（請參考《Kinect 的軟體開發方案：OpenNI 簡介》），在實作時，只要同時有 User Generator 以及 Depth Generator 這兩種 production node，就會自動讓 User Generator 去存取 Depth Generator 的資料；而 User Generator 的使用方法必須要透過 callback function 的機制，來做事件（event）的處理；而在 OpenNI 裡面，要為一個 production node 加上 callback function，基本上就是透過各種 node 提供、名稱為 RegisterXXXCallbacks() 的成員函式，來「註冊」(register)該 node 的 callback function；如果以這邊要使用 xn::UserGenerator 來說，就是 RegisterUserCallbacks() 這個函式。

另外由於骨架的判斷、以及判斷骨架時需要的姿勢偵測在 OpenNI 都是屬於延伸功能的「Capability，所以在這裡所使用的 user generator 也必須要有支援 Skeleton 和 Pose Detection 這兩個 capability 才行；不過現階段所使用的 user generator 應該都是 NITE 所提供的，所以都有支援。

NITE 的 StickFigure 這個範例程式裡，用來建立人體骨架的標準流程，人體骨架分析的流程大致如圖 7 所示。

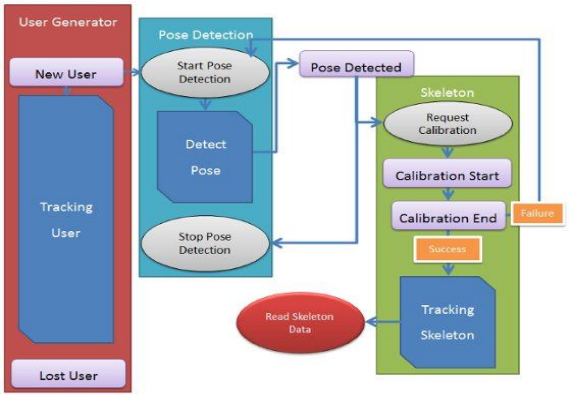


圖 7、建立人體骨架的流程

在上方的流程圖中，最左邊的紅色方塊是代表 user generator（xn::UserGenerator），裡面的

「New User」和「Lost User」則是代表他的兩個事件的 callback function；這兩個函示分別會在「畫面內偵測到新的使用者」、「使用者離開可偵測範圍一段時間」時被呼叫。而中間偏左的藍色方塊則是 pose detection 這個 capability（xn::PoseDetectionCapability）。在 user generator 偵測到有新的使用者、呼叫「New User」這個 callback function 時，「New User」的程式會去呼叫 pose detection 的「Start Pose Detection」、讓 pose detection 開始偵測 NITE 預先定義的校正用姿勢：「Psi」（如圖 8）。

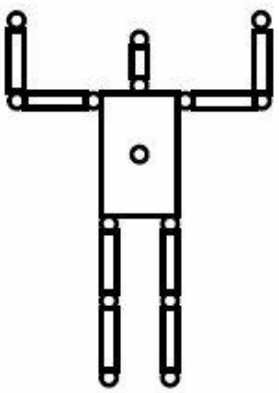


圖 8、Psi 姿勢

在呼叫「Start Pose Detection」前，pose detection 是不會進行姿勢偵測的動作的。當 pose detection 偵測到使用者擺出「Psi」這個姿勢後，他就會去呼叫自己的「Pose Detected」這個 callback function、以進行下一階段的動作；在這個例子裡，「Pose Detected」會去做兩件事，一個是去呼叫自己的「Stop Pose Detection」來停止繼續偵測使用者的動作、另一個則是去呼叫 skeleton 這個 capability(xn::SkeletonCapability)的「Request Calibration」函式，要求 skeleton 開始進行人體骨架的校正、分析。在 xn::SkeletonCapability 的「Request Calibration」被呼叫後，skeleton 就會開始進行骨架的校正、分析。當開始進行骨架校正的時候，skeleton 會去呼叫「Calibration Start」這個 callback function，而當骨架校正完後，則是會去呼叫「Calibration End」這個 callback function。不過，當「Calibration End」被呼叫的時候，只代表骨架的校正、辨識的階段工作結束了，並不代表骨架辨識一定成功。如果成功的話，就進入下一個階段、呼叫 xn::SkeletonCapability 的「StartTracking()」函式，讓系統開始去追蹤校正成功的骨架資料；而如果失敗的話，則是要再讓 pose detection 重新偵測校正姿勢，等到有偵測到校正姿勢後，再進行下一次的骨架校正。而在骨架校正成功、並開始進行追蹤骨架後就可以讀取到最新的關節相關資訊，並建立整個人體的骨架資料。

再來我們利用 Cmake 將 OpenCV 與 OpenNI 結合，重新編譯出新的 Opencv：

所需程式軟體：OpenNI、NITE、Sensor、SensorKinect、OpenCV、Cmake、Tbb

a、使用 Cmake 編譯 OpenCV：

- 修改 OpenCV 的 cmake 文件(OpenCVFindOpenNI.cmake)
- 原因:環境變數的路徑與原本 32 位元多了 64

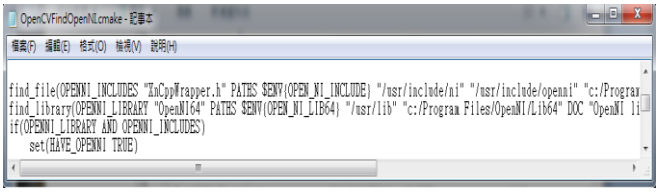


圖 9、修改 OpenCV 的 cmake 文件

- 執行 cmake-gui.exe
- 選擇 Visual Studio 10 Win64
- 下面選項不用更動

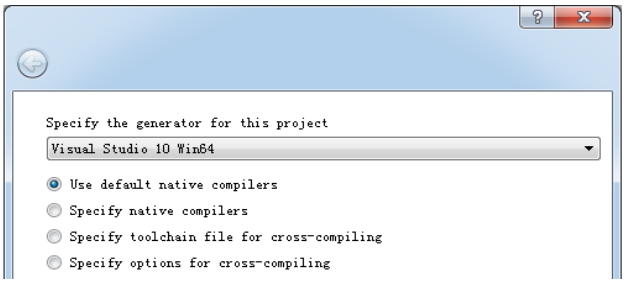


圖 10、選擇 Visual Studio 10 Win64

- where is the source code 路徑為 OpenCV 安裝路徑
- where to build the binaries 為 Cmake 編譯

後存放路徑

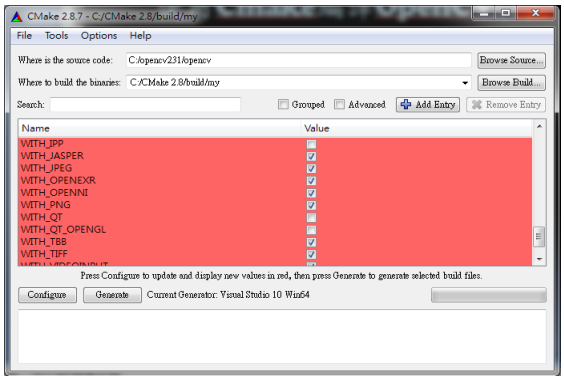


圖 11、勾選所要的選項(完後按下 Configure)

- WITH_OPENNI:整合 OpenCV 和 OpenNI
- WITH_TBB:某些函數需要 TBB 的程式
- 設定好 Tbb、OpenNI 路徑並檢查 (OpenNI 會自動掛上)
- 再按下 Configure

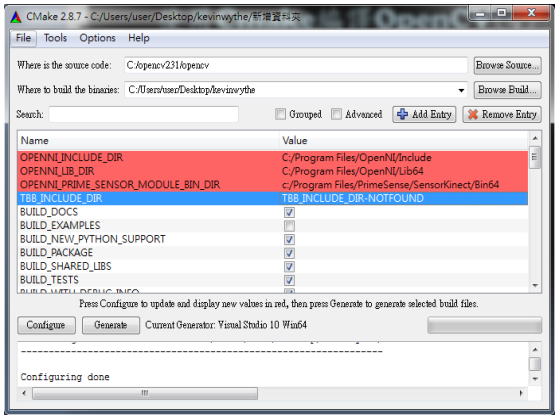


圖 12、設定好 Tbb、OpenNI 路徑

- 檢查紅色選項路徑 (TBB_LIB_DIR 會自動掛上)
- 再按下 Configure

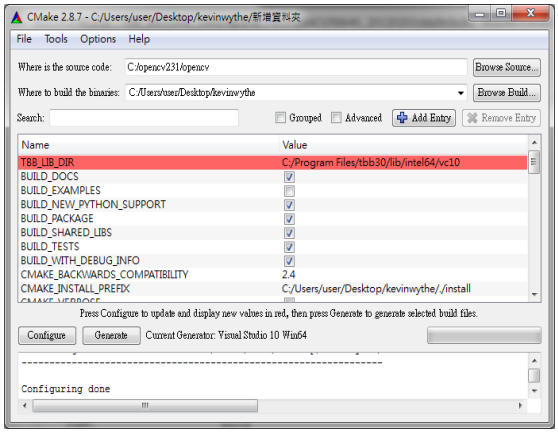


圖 13、檢查紅色選項路徑

- 檢查 Cmake 配置結果 OpenNI、Tbb 是否成功(YES)
- 之後點擊 Generate 就 OK 了

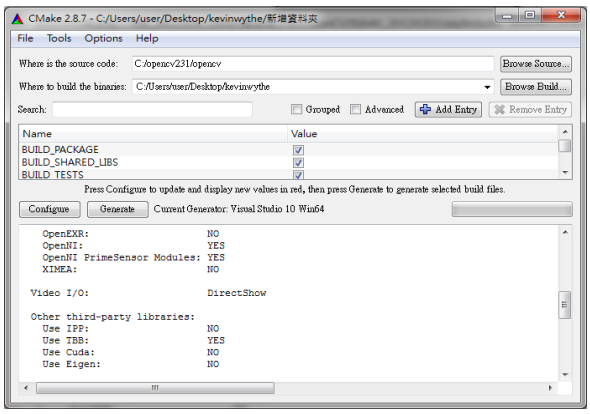


圖 14、檢查 Cmake 配置結果是否成功

b、編譯 Cmake 產生的檔案：

- 檔案在 where to build the binaries 所設定的路徑.
- 使用 Visual Studio 10, Build OpenCV.sln
- 點擊 All_BUILD 進行 Build(Debug、Release 都要)
- 點擊 INSTALL 進行 Build(Debug、Release 都要)

c、新 OpenCV 環境配置：

- 上個步驟結束後會生成一個 install 資料夾(新 OpenCV)
- 設定系統環境變數：
  - C:\...\install\bin
  - C:\...\tbb30\bin\intel64\vc10
- 建構一個 VS2010 的 OpenCV + OpenNI 程式
- 不需建構空的 project

d、新 OpenCV 路徑配置：

- 設定 Include 路徑:
  - C:\install(新的 Opencv 資料夾)\include\opencv;
  - C:\install(新的 Opencv 資料夾)\include\opencv2;
- 設定 Library 路徑:
  - C:\install(新的 Opencv 資料夾)\lib;

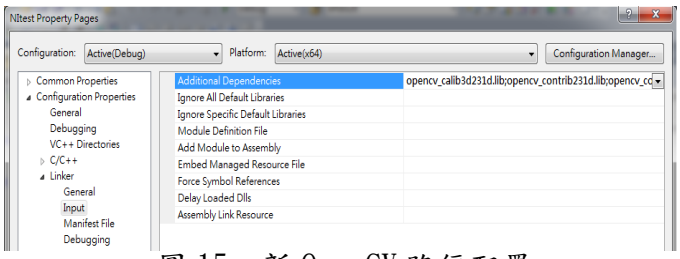


圖 15、新 OpenCV 路徑配置

e、OpenNI 路徑配置：

- C/C++的 Additional Include Directories: 加入---> $(OPEN_NI_INCLUDE);
- Linker 的 Additional Library Directories: 加入---> $(OPEN_NI_LIB64);
- Linker 下的 Input 裡的 Additional Directories:加入---> openNI64.lib;

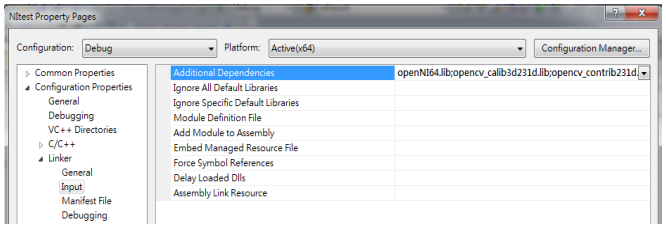


圖 16、OpenNI 路徑配置

(4)辨識：

所謂類神經網路，它是一種平行計算系統[12]，包含硬體與軟體，它使用大量的相連人工神經元來模仿生物神經網路能力。現今電腦雖然擅長執行高速的複雜計算，而且所得結果具有高度精確與可靠性，但還是有許多工作人腦比電腦強的多，例如辨識就是其中之一，因此我們使用類神經網路來模仿生物神經網路的資訊處理系統。類神經網路架構如下：

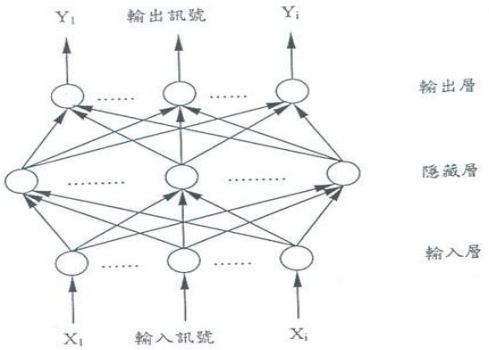


圖 17、類神經網路

神經網路的學習演算法基本上都是由「能量函數」(Energy Function) 推導而得。能量函數是用來衡量網路的學習效果， 此網路的學習過程即是使能量函數最小化的過程。學習演算法可分為三類，其中以監督式學習(Supervised Learning)裡的到傳遞神經網路(Backpropagation Neural Network)應用在辨識的議題較為廣泛且準確。下面為倒傳遞神經網路步驟：

(一)輸入範本及目標結果

(二)設置網路架構、參數

1. 設定網路的隱藏層數目及各層的神經元數目。
2. 設定網路的學習率(Learning Rate)及最大可容忍誤差。
3. 以隨機亂數初始化權重值。

(三)計算輸出

1. 將訓練範例由輸入層進入網路，計算引藏層和輸出層的輸出值。

$$Y_j = f(\sum_i W_{ij} X_i - \theta_j) \qquad (2)$$

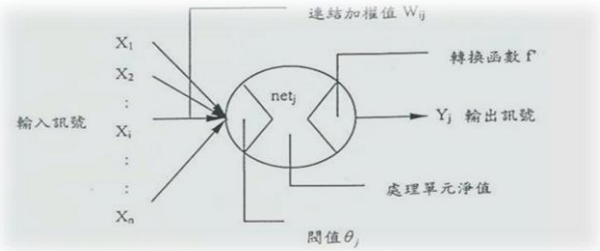


圖 18、處理單元示意圖

i：輸入層之神經元數

j：輸出層之神經元數

Y_j：模仿生物神經元模型的輸出訊號。

$f()$：模仿生物神經元模型的轉換函數(Transfer function)是一個用以將從其他處理單元輸入的加權值轉換成處理單元輸出值的數學公式。

W_{ij}：模仿生物神經元模型的神經元強度，又稱連結加權值，各示第 i 個處理單元對第 j 個處理單元之影響強度。

X_i：模仿生物神經元模型的輸入訊號。

θ_j：模仿生物神經元模型的閥值。

2. 控制輸出單元形成的雙曲線轉移函數。

$$Y = \frac{1}{1+e^{-net_j}} \quad (3)$$

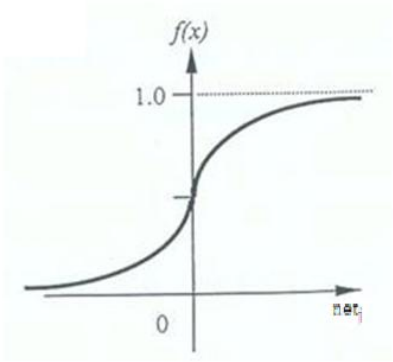


圖 19、雙曲線轉移函數示意圖

（四）計算誤差函數

使用此能量函數(誤差平方函數)來表示實際輸出值與目標輸出值差異量。

$$E = \frac{1}{2}\sum_j^M (T_j - Y_j)^2 \quad (4)$$

M=神經元個數

T_j=第 j 個神經元的目標輸出值

Y_j=第 j 個神經元的實際輸出值

（五）修正權重

當 E 值大於最大容忍誤差，則開始反向傳遞，使用最坡陡降法(Steepest Descent Method)，計算權重修正量，使 E 值下降。

$$\Delta W = -\eta \frac{\partial E}{\partial W} \quad (5)$$

W=各層神經元連結權重

∂E=各層神經元連結權重修正量

η=學習率

◆ 計算輸出層和隱藏層差距量：

輸出層：$\delta_j = Y_j \times (1 - Y_j) \times (T_j - Y_j)$ (6)

隱藏層：$\delta_h = H_h \times (1 - H_h) \times \sum_{j-1}^{M} W_{hj}\delta_j$ (7)

δ_j=輸出層第 j 個神經元的差距量

δ_h=隱藏層第 h 個神經元的差距量

◆ 各層間的權重值修正量：

$\Delta W_{hj} = \eta\delta_j H_h + \alpha \times \Delta W_{hj}$ (8)

$\Delta W_{ih} = \eta\delta_h X_i + \alpha \times \Delta W_{ih}$ (9)

α=慣性項，目的在於改善收練過程中震盪情形

◆ 更新權重：

$W_{hj} = W_{hj} + \Delta W_{hj}$ (10)

$W_{ih} = W_{ih} + \Delta W_{ih}$ (11)

(六)判斷是否符合目標

判斷誤差誤差函數，若不符合目標，便重複訓練直到達到標準或收斂，亦或是達到最大訓練次數。

3. 實驗結果

建立實作流程圖：

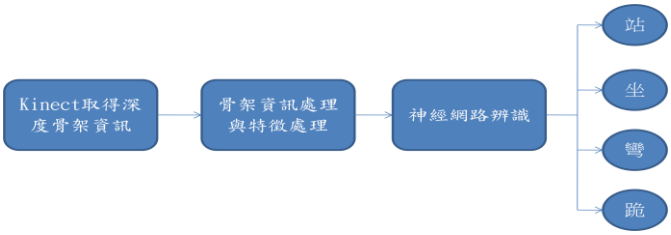


圖 20、實作流程圖

(1)Kinect 取得深度骨架資訊：

在此步驟之前嘗試了使用背景相減法，利用目前的影像與背景相減來偵測出變化的區域，變動的區域即為目標物的位置，但得出的結果不如預期，且耗時，之後改由 Kinect 深度影像的改變，取得人體輪廓，如圖 21-1、21-2。

圖 21-1、深度影像

圖 21-2、人體輪廓

取得人體輪廓後透過 OpenNI 與 Nite(middleware)取得人形骨架,能用的只有十五個，再把這些關節連成線畫 出來，結果如圖 22。

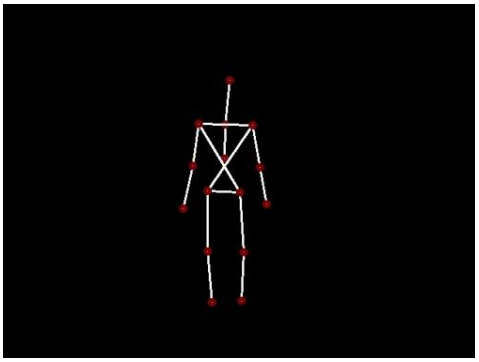


圖 22、人體骨架

(2)骨架資訊處理與特徵處理：

我們將骨架資訊取 6 個點如圖 23，將骨架的中心點做為我們的參考點，並分成 5 組輸入。第一組的輸入為頭部骨架點的 X 軸之值和雙腳骨架點計算出中間點的 X 軸之值的差值,第 2 組的輸入為左手掌骨架點與人體中心骨架點的距離。第 3 祖的輸入為右手掌骨架點與人體中心骨架點的距離,第 4 組及第 5 組的輸入為左、右腳掌骨架點與中心點的距離。將這 5 組資訊做為我們的輸入的特徵。

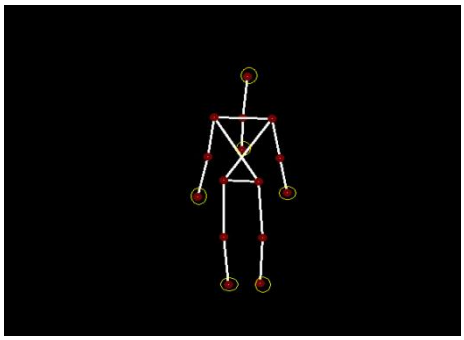


圖 23、骨架特徵點

(3)神經網路辨識：

將上一步驟中所取得的特徵做為神經網路的輸入，如圖 24。

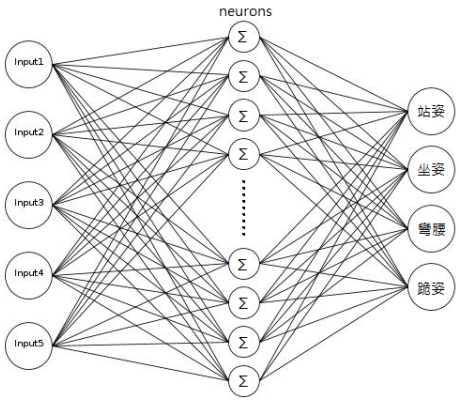
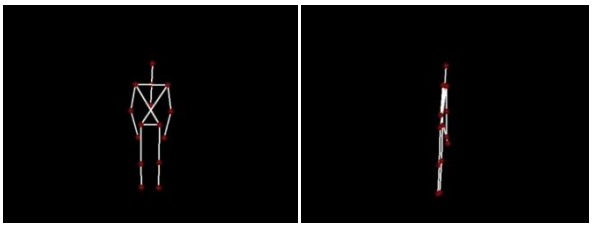


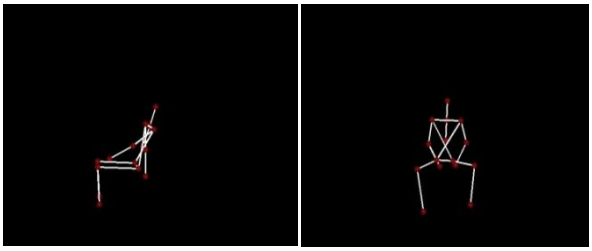
圖 24、神經網路

(4)辨識結果：

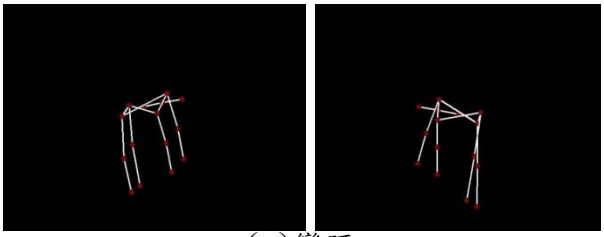
神經網路辨識結束後,我們辨識人的四種姿勢：站、坐、彎、跪，每個姿勢有幾個不同的模式，如圖 25。測試資料每個姿勢總過辨識 50 次，但坐姿有 80 次，成功的辨識率如表 2。



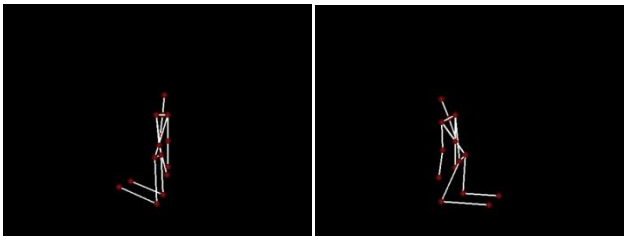
(a)站姿



(b)坐姿



(c)彎腰



(d)跪姿

圖 25、四種不同的姿勢(a)站姿(b)坐姿(c)彎腰(d)跪姿

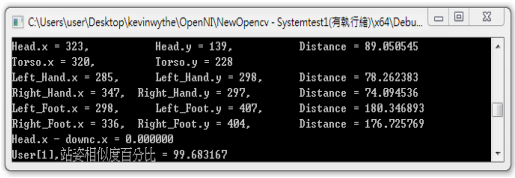
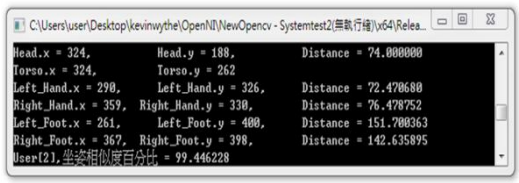
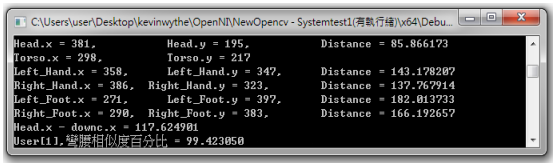
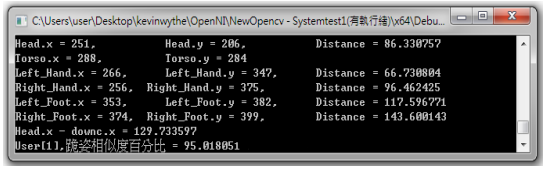
辨識姿勢	辨識率
站姿	
坐姿	
彎腰	
跪姿	

表 2、人體動作辨識率

這項研究使用了 Kinect 捕捉人體影像，經過了各項技術與演算法，包含了：前處理、水平投影分析、骨架資訊、類神經網路、Cmake、特徵提取、OpenCV、OpenNI 等技術，最終達到的人體姿態辨識的目的，而辨識的成功率更是達到了 90%以上。

最困難的部分還是在 Kinect 的程式碼，包括如何與 OpenCV 和 OpenNI 整合，以及如何儲存連續影像、如何取得特徵點…等，這些都還要持續的上網搜尋資訊與相關的程式。此外運動模型以及訓練 data 的建立，還有辨識的方法等，也是要上網找一些相關的 paper 來看。另外在 OpenCV 的安裝也費了很大的功夫，尤其是在環境的建立、路徑的設置、如何運行，也從老師、網路和書本上取得了資訊。

而另外的五項困難就是[13]：

1. 光線因素:雖然擷取影像是在室內但室外光線投射入室內所造成之陰影變化，以及人體運動過程中各部位造成之陰影都會造成辨識的困難。
2. 背景因素：當人物跳動影響地面引起的 Kinect震動會產生背景的變化，背景的改變會造成影像校正困難。
3. 人物的衣著:如果所穿著的服裝與背景顏色太過雷同，則會影響
   系統的辨識率。
4. 即時性:必須快速的對影像做分析處理，否則將造成系統的實用性下降。
5. 由於人體所作之運動屬於形變物體的運動，要對形變物體進行動作偵測，也是這個研究的困難。

4. 參考文獻

[1] C. C. Li and Y. Y. Chen, “Human posture Recognition by simple rules,” *in Proceedings of IEEE Systems Man Cybernetics*, pp. 3237-3240, 2006.

[2] B. Castiello, T. D’ Orazio, A. M. Fanelli, P. Spagnolo, and M. A. Torsello, “A model-free approach for posture classification,” *in Proceedings of IEEE Advanced Video and Signal Based Surveillance*, pp. 276-28, 2005.

[3] C. H. Chuang, J. W. Hsieh, L.-W. Tsai, and

K.-C. Fan, “Human action recognition using star templates and delaunay triangulation,” *in Proceedings of IEEE Intelligent Information Hiding and Multimedia Signal Processing*, pp. 179-182, 2008.

[4] C. F. Juang and C. T. Lin, “An online self-constructing neural fuzzy inference network and its applications,” *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–32, Feb. 1998.

[5] C. F. Juang, and C.-M Chang, “Human body posture classification by a Neural fuzzy network and home care system application,” *IEEE Trans. Syst., Man, and Cyber., Part A: Systems and Humans*, vol. 37, no. 6, pp. 984-994, Jan. 2007.

[6] H.-S. Chen, H.-T. Chen, Y.-W. Chen, and S.-Y. Lee, “Human action recognition using star skeleton,” *in Proceedings of International Conference on 4th ACM Video surveillance and sensor networks*, pp. 171-178, 2006.

[7] C. F. Juang, C. M. Chang, J. R. Wu, and D. M. Lee, “Computer vision-based human body segmentation and posture estimation,” *IEEE Trans. Syst., Man, and Cyber., Part A: Systems and Humans*, vol. 39, no. 1, pp. 119-133, Jan. 2009.

[8] PrimeSense. 21 June ,2011. <http://www.primesense.tw/>

[9] 李乾丞，“Fast Human Posture Recognition by Heuristic Rules”，國立台灣大學碩士論文 ，2006年

[10]VIML. <http://viml.nchc.org.tw/home/>

[11]吳居穆，“利用轉換狀態圖及模型建立進行人體運動姿勢辨識”國立中央大學碩士論文，2006年

[12] 石傑方、吳姿儀、潘茂森，“類神經網路於介質穿透之應用”，2005年

[13]顏羽君，“視覺式體操動作辨識系統 Vision-based Gymnastics Motion Recognition System”，國立臺灣師範大學，2008年

[14] Shih-Fu Huang and Wen-June Wang, “Human’s Postures Recognition by Using Kinect ,” *國立暨南國際大學*,2011.

[15]msdn.<http://msdn.microsoft.com/zh-tw/hh367958.aspx>