運用工業用機械手臂之自動拼圖系統的實現

廖珫洲
朝陽科技大學資訊工程系
教授
hcliao@cyut.edu.tw

黃冠瑜 葉凌承 陳膨琛
朝陽科技大學
資訊工程系

摘要

工業化機械手臂在現在自動化製造的流程中佔有很重要的角色,其也是工業 4.0 發展趨勢中重要的一環。因此，在本論文中將機械手臂、輸送帶、感測器以及攝影機進行結合，並設計出一套系統讓機械手臂可以自動化的完成積木拼圖。在系統運作上，機械手臂不只是進行固定的動作，它可以經由取得攝影機影像之後進行圖形識別功能，以判斷出目前的積木塊是拼圖的哪一塊，這樣就可達到了自行拼圖的動作。系統也展示出了機械手臂的精準度以及運作效率，以呈現手臂在未來各種應用的發展可能性。

關鍵詞：機械手臂、圖形識別、輸送帶、感測器

Abstract

Industrial robotic arms play an important role in the course of manufacturing automation. They are also important to the trend of Industry 4.0. Therefore, a system is designed and implemented by using a robotic arm, conveyer, photo interrupter and camera to finish the task of assembling a child's puzzle in this project. The robotic arm is not only controlled to perform pre-defined fixed operations. It can also be controlled to perform dynamic operations according to the identified block in a puzzle. The system demonstrates the precision and efficiency of a robotic arm and many possible applications in the future.

Keywords: mechanical arm, pattern recognition, conveyor belts, sensors

1. 簡介

在現代工業製造產業中，機械手臂已經成為了普遍使用的設備之一，在人力密集時代，一條作業線會需要好幾百位以上作業員，這也使得營運成本較高，不過隨著工業用機械手臂的出現，讓得作業線所需作業員人數大幅降低，除此之外，機械手臂的精準度以及 24 小時的運作，提高工廠的生產效率與品質，上述各種原因使得機械手臂具有取代人力的成本效益。

本論文藉由輸送帶、工業用機械手臂和圖形識別系統的結合，可以把規格相同的積木完成拼圖的動作，一開始將積木放置於輸送帶上方，等積木被光感應器偵測到時會立即停止輸送帶，接著透過攝影機擷取積木影像，再透過圖形識別技術來判斷該積木屬於拼圖中的哪一個部分，然後將積木放置到對應位置上，重複上述流程以達成自動拼圖的目標。

2. 系統流程

為了達成自動拼圖的目標,本論文設計的流程圖如圖 1 所示，一開始先讓輸送帶將積木往前送，而感測器會判斷是否感測到積木，當感測到積木時輸送帶就會停止將積木往前送，然後開始執行積木識別。

積木識別步驟中,首先攝影機會將所拍攝到的影像中積木的影像擷取出來，再將擷取出來的積木影像與事先建立的樣板進行逐一匹配，以得知積木屬於拼圖中的哪一個位置，這時就會開始運算機械手臂夾取與移動積木的路徑。

由於欲完成的拼圖是以垂直方式呈現,因此當圖形識別比對結束時，會將得到的目前積木的位置進行判斷是否可以直接放上目前的積木之上，若是無法直接放在目前積木之上，系統會先將順序不符的積木移動到暫時區放置,接著放入目前的積木之後，再將暫時存放區的積木放回目前積木的上方。

當完成以上步驟後系統會判斷是否已將所有的積木放入拼圖中，若是尚未完成，就會讓輸送帶繼續將剩餘的積木往前送以重複以上步驟。若是已經完成拼圖之後,系統會將機械手臂的移動速度降低，接著以緩慢的速度推動拼圖，將垂直拼圖之間的間隙去除以完成拼圖。

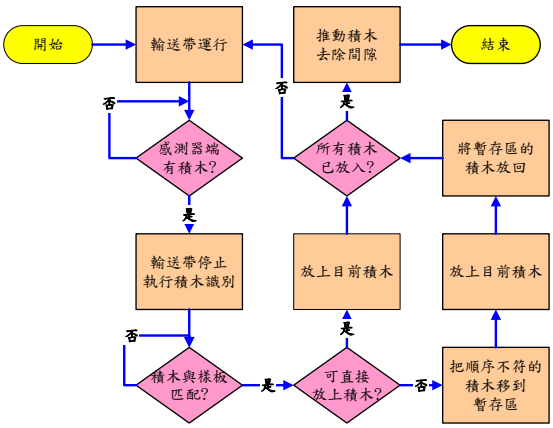


圖 1: 主要系統流程圖

下面內容中將針對系統流程的主要三個部分：包含輸送帶運作流程，圖形識別運作流程以及機械手臂運作流程，分別進行說明。

2.1 輸送帶運作流程

輸送帶運作流程如圖 2 所示,當輸送帶運行後，在運行過程中會判斷拼圖所需的積木數量是否已被

到達，若是的話，輸送帶就會結束運作，反之感測器就會持續感測是否有積木到達，接著執行圖形識別對目前的積木進行辨識，機械手臂就會依照其辨識結果將積木夾取並移動到正確位置上，最後判斷是否完成了整個拼圖，直到拼圖完成為止。

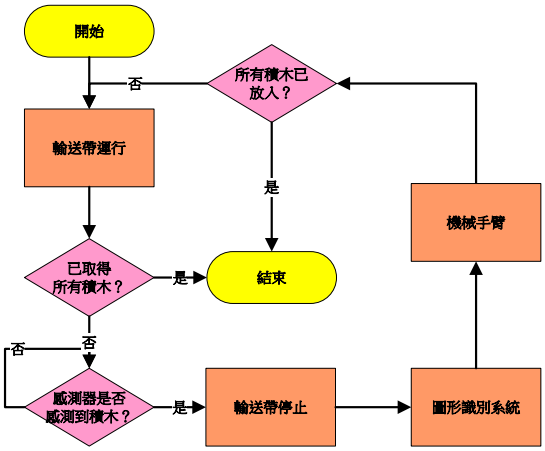


圖 2: 輸送帶運作流程圖

2.2 圖形識別運作流程

圖形識別流程如圖 3 所示，一開始先處於等待狀態，等待輸送帶停止將積木往前送。接著攝影機會將拍攝到的積木影像擷取出來，並與事先建立一組樣板逐一進行匹配，當匹配成功就會依據結果進行機械手臂的控制，將積木夾取並移動到正確位置上，直到完成所有積木的辨識為止。

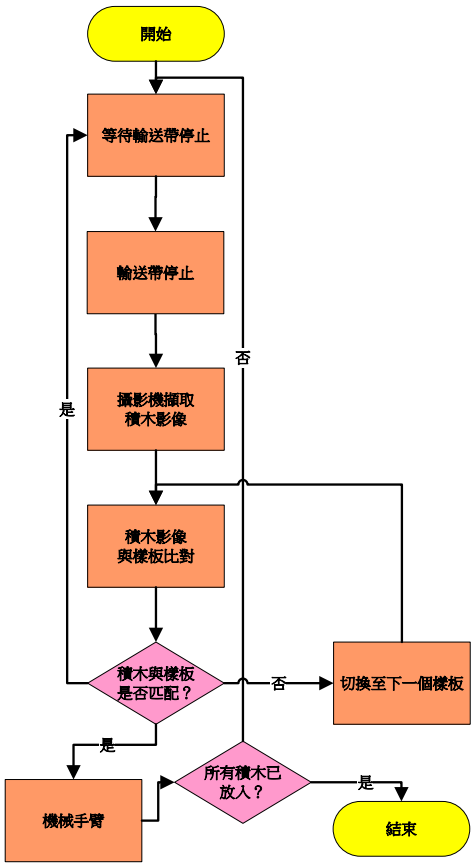


圖 3: 圖形識別運作流程圖

2.3 機械手臂運作流程

流程圖如圖 4 所示，一開始等待圖形識別辨識的結果。在得到結果之後，系統會判斷目前的積木是否可以直接放上某一個拼圖柱中(該拼圖為 4×4，故有 4 個拼圖柱)，若是可以放入就直接控制手臂夾取積木放入。不然，就會把順序不符的積木由拼圖柱中取出並放在暫存區中，接著將目前的積木放入後，再將暫存區中的積木取回。一旦所有積木都已經放入，就會將手臂移動速度降低，以緩慢的速度推動拼圖柱去除柱子之間的間隙以完成拼圖。

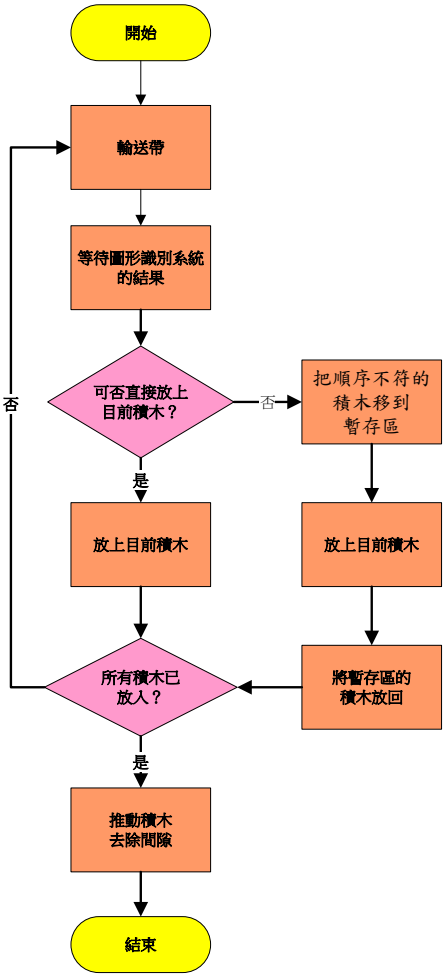


圖 4: 機械手臂運作流程圖

3. 圖形識別的方法

這裡將在下面幾個小節中說明圖形識別流程中所使用的方法內容：

3.1 擷取積木的影像

主要使用 OpenCV 的 cv::VideoCapture 類別來取得攝影機的即時影像，接著使用類別中的函式 isOpened 來判斷是否有找到並開啟攝影機。當感測器偵測到積木並停止輸送帶之後，接著將積木所在的範圍擷取出來並將其存儲成新的.jpg 檔，如圖 5 範例所示。

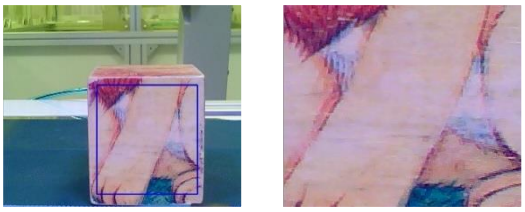


圖 5: (a)攝影機影像 (b)取出積木影像

3.2 積木與樣板的匹配

當取得目前積木的影像後就會進行樣板匹配的步驟，這裡使用 OpenCV 裡的 matchTemplate 函式，主要針對進行匹配的原始圖與目標圖，在原始圖上將目標圖會不斷地由左到右由上到下的滑動，以得到原始圖上各個位置的匹配值，接著找出最大值以定位出目標圖的位置。

另外，在 matchTemplate 函式裡有 6 種相似的方法，本系統使用 CV_TM_CCOEFF_NORMED (標準相關係數匹配法)作為樣板匹配方法。其運算時透過減去各自的平均值，以及除以各自的方差（標準差的平方），可以將積木影像與樣板影像都標準化，確保影像的光照亮度比較不會影響計算結果。下面公式(1)為標準相關係數匹配方法計算方法，T'為樣板影像，I'為即時積木影像，(x, y)為某個欲匹配的左上角座標，(x', y')為樣板影像的寬與高座標。一個積木影像與樣板匹配的範例如圖 6 所示：

$$R(x,y)=\frac{\sum_{x',y'}(T'(x',y')\cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'}T'(x',y')^2\cdot\sum_{x',y'}I'(x+x',y+y')^2}} \quad (1)$$

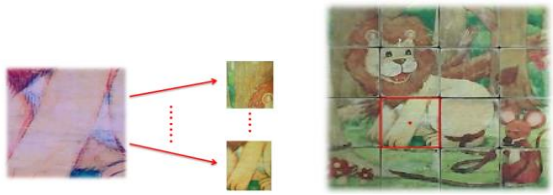


圖 6: (a)積木與樣板進行匹配 (b)匹配結果

4. 機械手臂擺放積木與完成拼圖

4.1 取得目前積木的位置

目前設定是以 4 個柱子，每一柱可以放入的積木數為 4 個的方式來擺放積木並完成拼圖，柱的設定從左到右為(A, B, C, D)，每一柱由上到下以 C 柱為例是 (9, 10, 11, 12)。如圖 7 所示：

圖 7: 拼圖擺放方式

透過圖形識別的方式可以找出目前所要放入積木的位置，亦即目前積木是屬於那一柱的那個位置的積木，得知這項資料後就可以計算出機械手臂的動作步驟。

4.2 計算機械手臂動作的步驟

當得知目前判斷的積木是屬於 x 柱中第 y 個位置後，這個步驟的計算動作主要分為三種情況：

1. 判斷在 x 柱中是否有屬於該柱的積木已放入其中，如果沒有積木在其中，就可直接將目前的積木放入該柱中
2. 若是 x 柱中最上方的積木編號是否大於目前要放入的積木，若是的話，表示可以直接將目前的積木放到柱中最上方的積木之上
3. 若是 x 柱中最上方的積木編號是否小於目前要放入的積木，若是的話，會將柱中最上方的積木夾取到暫存區，等目前的積木放入柱中後，再將暫存區積木取出並放回柱中。

這三種狀況分別如圖 8 到圖 10 所示。

圖 8: 無積木在柱中時直接放入的實例

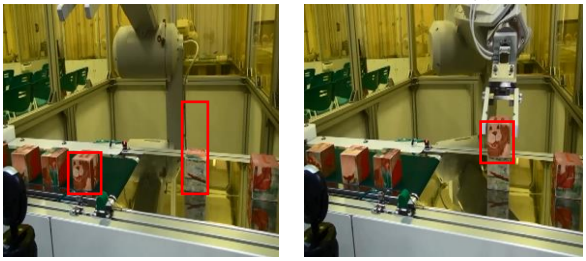
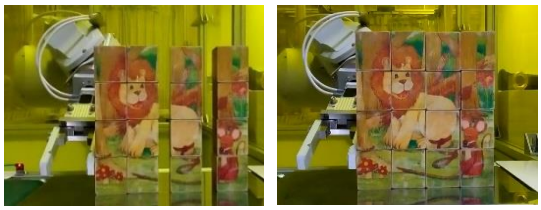


圖 9: 有積木在柱中但可直接放入的實例

圖 10: 有積木在柱中但不可直接放入的實例

4.3 機械手臂推動拼圖柱完成拼圖

當機械手臂已經將所有的積木都擺放到拼圖柱中後，拼圖柱之間還存在著一些間隙，如圖 7 所示，為了去除柱與柱之間的間隙，系統會控制機械手臂的移動速度，以緩慢的速度推動拼圖柱 A 向右靠攏以去除間隙及完成拼圖，如圖 11 所示。



(a) (b)

圖 11: (a)推動拼圖柱去除間隙 (b)完成拼圖

除了現在我們所設計的機械手臂移動並擺放積木的方式外，只要符合目前設備的需求也可以設計其他不同的方式，因此，接著將說明如何設定機械手臂的動作與調整其移動速度。

5. 機械手臂的設定

5.1 點位與機械手臂移動的設定

要控制機械手臂可以進行移動來完成一個動作，首先需要以設定點位的方式使機械手臂可以從點 A 移動到點 B。如圖 12 所示：

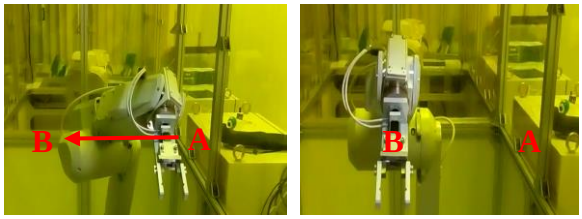


圖 12: 點 A 到點 B 示意圖

在設定點位時，主要使用手臂三菱原廠提供的 RT-Toolbox，來將機械手臂目前的坐標記錄下來，再將記錄的坐標放入變數(P1)裡，如圖 13 所示：

```
P1=(617.190,22.210,701.410,-178.610,63.890,-179.200)(7,0)
P2=(617.190,22.210,902.920,-178.610,63.890,-179.200)(7,0)
Pa_0=(573.200,201.430,902.940,-178.610,63.890,-179.200)(7,0)
Pa_1=(573.200,201.430,699.130,-178.610,63.890,-179.200)(7,0)
Pa_2=(573.200,201.430,745.880,-178.610,63.890,-179.200)(7,0)
Pa_3=(573.200,201.430,789.920,-178.610,63.890,-179.200)(7,0)
Pa_4=(573.200,201.430,835.420,-178.610,63.890,-179.200)(7,0)
```

圖 13: 點位坐標示意圖

設定好一組機械手臂的點位後，系統就可以透過程式呼叫並將預定移動到哪個坐標位置的指令傳送給機械手臂，在接收到指令後機械手臂會移動至指定的坐標位置。如圖 14 所示：

```
*S10MOVE
    Mvs P1
    Wait M_Fbd<0.02
Rerurn
```

圖 14: 機械手臂移動指令程式碼示意圖

解決點位坐標的設定問題後，接著又因為所能使用的點位坐標只有 7 個而受到限制，因此查閱原廠提供的手冊後，發現程式設定中只使用了 3 條輸入信號源，導致只有 7 個點位可以使用，因此，嘗試新增 3 條輸入信號源，讓可使用的輸入訊號達到 000001 至 111111，亦即有 63 個輸入訊號，如圖 15 所示：

```
(a)  case RobotCode::StartPos:
         m_pOUT[2]->SetValue(0);
         m_pOUT[3]->SetValue(0);
         m_pOUT[4]->SetValue(1);
         break;

(b)  case RobotCode::StartPos:
         m_pOUT[2]->SetValue(0);
         m_pOUT[3]->SetValue(0);
         m_pOUT[4]->SetValue(0);
         m_pOUT[5]->SetValue(0);
         m_pOUT[6]->SetValue(0);
         m_pOUT[7]->SetValue(1);
         break;
```

圖 15: (a)原始訊號設定 (b)修改後訊號設定示意圖

5.2 自動控制機械手臂移動的速度

系統最後會降低機械手臂的移動速度，讓其以緩慢的速度推動積木柱去除柱與柱之間的間隙並完成拼圖。同樣透過原廠手冊找到在 Basic Stamp 程式碼中可以使用 Spd 指令讓機械手臂可自動的提高與降低其移動速度。如圖 16 所示：

```
*S300MOVE
    Spd 10
    Mvs Push1
    Wait M_Fbd<0.02
Return
```

圖 16: 控制手臂移動速度 10%的程式碼

6. 結論

本論文將工業用機械手臂、輸送帶、感測器以及攝影機進行結合，一般既有的設備在結合圖形識別系統後，讓機械手臂具備視覺能力，能夠依據實際狀況自行判斷與運作，大幅降低人為的控制。

目前所學習到的技術還有進步的空間，可以更進一步針對各個步驟進行精進，以提升系統整合及各項技術的能力，讓機械手臂的用途更多元化。

在邁入工業 4.0 的時代，透過學校建置的工業用機械手臂系統，可以在學校接觸到業界所使用的設備平台，並進行相關系統的開發，透過這樣的訓練機會可以在踏入業界前累積更多經驗，收穫相當豐富。

7. 參考文獻

[1] matchTemplate 函式介紹, http://monkeycoding.com/%E7%9B%B4%E6%96%B9%E5%9C%96%E6%90%9C%E5%B0%8Bmatchtemplate%E3%80%81minmaxloc/，http://docs.opencv.org/2.4/modules/imgproc/doc/object_detection.html

[2] minMaxLoc 函式介紹, http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#void minMaxLoc(InputArray src, double* minVal, double* maxVal, Point* minLoc, Point* maxLoc, InputArray mask)

[3] OpenCV- 維基百科自由的百科全書，http://zh.wikipedia.org/wiki/OpenCV