國立台北大學資訊工程學系專題報告

結合光學點補正與軸控自動化之點膠機系統

Desktop dispenser

專題組員:蔡汶霖、王昱順、劉柏廷、蔡欣甫

執行期間:102 年 9 月 至 104 年 5 月

1. 摘要

點膠機就是可以對流體進行控制，並將流體點滴或塗覆於產品的自動化機器那他可以用來實現打點、畫線、圓形、弧形....等等功能。

工業上點膠機的使用相當廣泛，諸如 IC 封裝、光電廠、LCD 廠 LCD 框膠、LED 廠 LED 封裝 LED 灌膠、電腦手機外殼封裝、筆記型電腦膠合、SMT 零件使用、過錫爐前點膠固定、印刷電路板組裝等等。

而我們所側重的核心，則是在軸控自動化和光學點補正上。前者能節省人力，加快速度；後者則是增加良率與產能。

關鍵詞: 點膠機、軸控自動化、
光學點補正

2. 簡介

本專案是和台灣樂業公司進行產學合作所共同開發的點膠機操控程式。目標是以自動化機械取代人力，進行精密及重複的點膠工作。主要預期效益是能夠提供客製化操作、高精度、簡潔直觀的介面操作。我們主要功能為：

a.人性化介面

假定操作員是完全不了解這個程式，如何能讓操作員在最短的時間內上手操作的直覺式 UI 設計，同時避免人為操作疏失等的防呆功能。

b.拼板功能

拼板功能利用類似迴圈的概念，能以簡短的指令進行枯燥反覆的工作，減少操作員的工作，同時也能簡化指令，降低指令複雜度，間接降低出錯率。

c.光學點

利用圖形比對的方式，能夠精確的辨別不同的板子樣式，從而在每塊不同的板子上進行同樣精準的點膠。

d.點膠程式編輯

客製化的指令編輯。透過各種點膠、移動指令組合，能提供多樣化的點膠操作，而不是寫死一種動作。

e. I/O、與機器的連線

同時所有的指令設定、光學點參數都提供存讀檔功能。

3. 專題進行方式

3.1 人員配置與職責

我們的點膠程序主要可以分成兩個部分，軸控(點膠程序編輯)和光學點。

劉柏廷、王昱順負責和光學點以及相機相關的所有功能，例如：

a.壞板偵測編輯：將偵測錯誤的拼板設定為壞板的設定編輯

b.點膠點偵測編輯：在特定區域下膠作為條件判斷的依據

c.標準光學點定位編輯：二值化後將光學點設定作為點轉換的依據

d.點轉換功能：將光學點測試後和原本設定好的光學點做線性轉換後做出自動補正的功能。

還有除了光學點之外，為了因應廠商需求而加入的“登入控制”（將設定好的帳號密碼加密後存檔，在開啟程式後進行登入）和“安全設定”（撰寫不同的權限與配置）。

蔡汶霖負責點膠程序編輯頁面的功能和拼板編輯頁面的功能，主要是點膠程序新功能的追加和單一拼板編輯的部分。

蔡欣甫負責頁面切換以及所有 Form 的 UI 設計/製作，亦負責中英文轉換功能，以及專題製作前期負責整合大家的功能（後期大家改用 Git 來維護專案）。現階段專注於 UX 使用者體驗，目標誘導使用者以正確的方式操作而不會對機器或程式造成問題，並盡量改善客戶的使用需求以及操作體驗。

3.2 時程規劃

- 102 年 9 月～103 年 7 月

  專案雛形，包含簡易相機功能與簡易軸控功能。

- 103 年 8 月～104 年 1 月

  新增較為複雜的相機功能與軸控功能。

- 104 年 2 月~104 年 5 月

  補上微量天平、雷射測高等功能，亦增加系統中/英語言切換功能與觸控相關功能。

3.3 合作對象

台灣樂業科技股份有限公司

3.4 系統架構

點膠機程序由數個功能不同的頁面組成。

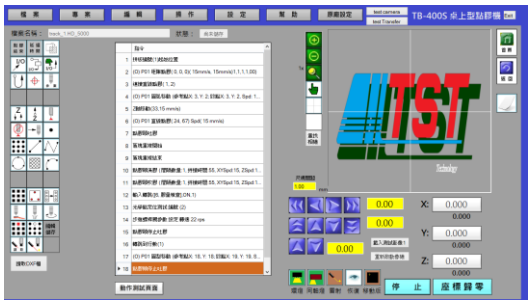
檔案頁面

因為想要避免每次對同樣的一組零件做點膠，都還要重新設定各種光學點或點膠指令的情況，所以就有這個存檔並讀檔的功能，程式中所有頁面下的設定都可以儲存在當前開啟的設定檔。採用的方式是物件序列化，可以直接將物件的狀態直接儲存，.net 提供了幾個類別來幫我們完成序列化的工作像是 BinaryFormatter, SoapFormatter, XmlSerializer，最後採用的方式是 XML serialization

（圖 1，檔案頁面）

點膠程序編輯頁面

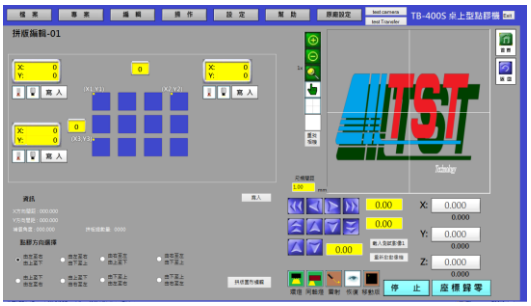
點擊圖形指令區會跳出參數設定視窗，設定完成後指令會依序在中間的"執行列"列出，最後可以點擊動作測試頁面進入測試畫面來執行這些指令，中間的指令列也支援右鍵選單，有新增、刪除、修改、複製、剪下、貼上的功能，右邊的 picturebox 顯示的是當前相機畫面。



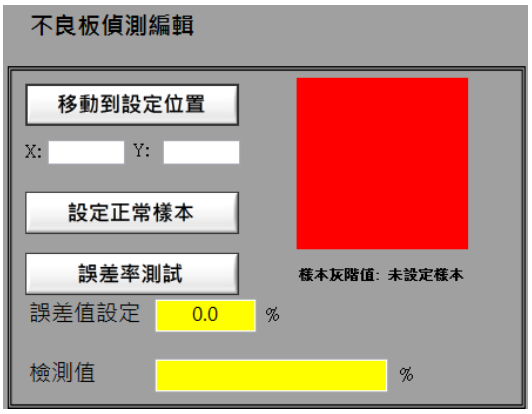
（圖 2，點膠程序編輯頁面）

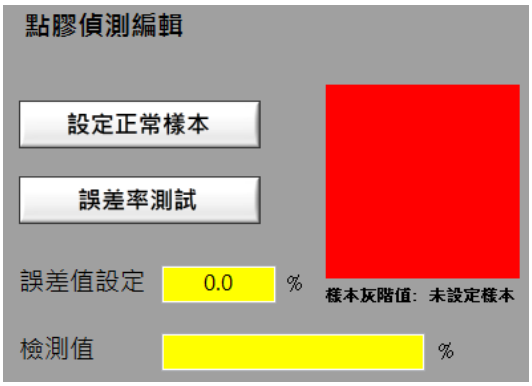
拼板編輯頁面

藉由設定這份拼板的左上，右上，左下三個點的座標以及拼板的數量，就可以推出每塊拼板跟基準版之間的 X,Y 間距，在執行點膠動作時就能將原本的指令座標加上該拼板與基準版的間距，藉此將重複的點膠指令畫在不同座標的零件上。除此之外還有對個別拼板做設定，也就是單一拼板編輯，可以略過不想做到重複點膠指令的拼板。



（圖 3，拼板編輯頁面）





（圖 4、5，不良版偵測編輯與點膠偵測編輯）

不良版偵測可以判斷出是否有事前設定好的壞版記號，如果有，在整個自動化的流程中就不會對有記號的版子做操作；點膠點偵測可以判斷是

否這次的點膠有作完全。兩者皆是由圖型比對來實現

3.5 主要困難與解決之道

相機 c++部分

問題:相機API與主程式C#部分的結合

說明:相機原廠提供的API為C++版的，與主程式C#為不同語言。

解決方法:曾嘗試過 IPC、management dll 等方式，不過因技術與穩定度問題，最後採用封裝 dll，以 unmanegement dll 的方式呼叫

C++函數

OpenCamera(); //call opencamera api

CloseCamera(); //call closecamera api

GetOnePhtot(); //call getonepicture，之後轉成 byte array 將取得的影像以 byte array 回傳 C#

相機 C#部分

主要將影像顯示在 picturebox

1.問題:multithread 同步問題

說明:要不干涉到主 UI 同時重複顯示相機影像，必須使用無限迴圈的 multithread，而產生同步、記憶體錯誤等問題

同步問題解決方法:

因高頻率跨 thread 存取變數(byte array)會發生錯誤，曾嘗試用單一 bool 值控制，但效果不彰。

之後改用 mutex 控制，但會產生 dead lock，最後改用 timer 來實踐。

UI 存取問題解決方法:因跨 thread 存取 UI 會產生錯誤，因此必須使用 invoke 委派 main thread 刷新 picturebox

但是 invoke 委派為非同步函數，與 mutex 會有極低機率發生 dead lock

最後改用 windows form timer，因為 timer 與 main thread 同 thread，故錯誤就不會發生。

2.multithread 的效能問題

問題:背景無限 while 迴圈效能不佳

說明:無限迴圈非常吃 CPU，約占 CPU50%

解決方法:曾使用使用 Sleep()，可降至大約 10%CPU，停越久降越多，可是 frame 數會下降，仍不是最佳解

最後使用使用 timer，考慮到刷新影像不需要嚴格要求同步，即使掉一張，下一張繼續顯示便不會有問題，不需要使用無限迴圈

使用 timer 即使某一次 tick 出錯，下一次 tick 正確的話仍可正常顯示。同時在未 tick 時完全釋放 CPU 資源，最終 CPU 使用率低於 5%，且保留最佳 frame 數、容錯率

3.禎數問題

問題:picturebox 並不是專門用來處理 video 的元件，對影片支援不佳

說明:若以內建的方法不斷快速刷新圖片，會導致禎數下降或一些不可預期的錯誤

解決方法:使用 graphics class，用底層 API 將相片畫在 picturebox 的 buffer 上，然後刷新整個元件以顯示圖片，可以正常快速顯示且不出錯

4.byte array ↔ bitmap
問題:C#函數原生處理速度過慢
解決方式:若使用 Bitmap 原生的函數處理，會因函數重複呼叫過多，而效率低下
之後嘗試使用 stream 的方式，但因圖片過大超過 C#限制，記憶體溢出
最後使用 unsafe 關鍵字，設定好 stride、width、height、pixel 等，以雙迴圈和指標方式直接讀寫 Bitmap 底層陣列，可讓效能最佳化

5.光學點存檔
問題:因為必須儲存圖片，所以採用 binary serilization。
說明:維護上困難，因為與其他組員儲存方式不同所以捨棄
解決方法:使用 XML serilization。將圖片轉成 byte array，以字串形式存於 XML，讀取時再反向構成圖片

6.光學點範圍選取
問題:相機影像與手繪的框會互相干涉
解決方法:參考了一些小畫家實作範例，以第二層 picturebox 當作透明圖層覆蓋在相機上，然後以 graphics 畫在第二層圖層上，彼此不會互相干擾
問題:若想以拖拉方式作畫，需不斷清空畫布，造成已畫好的圖形消失
解決方法:利用 Z 值決定繪畫先後順序，並在每清空畫布後依 Z 值重繪

3.6 實作平台與技術

本程序開發採用 C#，編譯環境為 Visual studio 2013。

4. 主要成果與評估

i.FOV 設定值 Field of View 與準心:
目的:
由於電腦是無法得知實際上影像下所得的真實距離，因此藉由設定相機上取得的特定像素長度，再提供影像的真實長度數據來提供轉換式，例如說 1 像素是幾公釐。
實際上也可藉由改變相機準心樣式，使用者可以直接利用 FOV 設定好的數值來判定現在的影像的真實大小事多少。
方法:
藉由在程式介面上提供拉取正方形框,再提供真實距離的長度來達成，藉由換算出長度的對應，便可以算出轉換比例。
應用:
這是非常基本的參數，幾乎有用到相機與所有和軸控相關的的數值都需要用到。
ii.點轉換方程式:
目的:
R2 to R2 線性轉換，用到旋轉和平移的方法，可以將任意平面上的點或是向量，轉換到另一個平面上。
方法:
旋轉用矩陣:
$\cos\theta$,$\sin\theta$
$\sin\theta$,$\cos\theta$
平移
加上平移量。
應用:
配合光學點來將軸控的位置修正。

iii. 光學點定位

目的:

藉由判定光學點後，利用 ii 的點轉換方程式，將測試取得的點帶入換算取得新的下膠點，達成自動定位的效果，只要光學點判定的物件沒有超過相機的取像範圍，都可以成功的提供高準度下膠。

標準定位:

標準定位使用的是二值化加上去雜點的圖形辨識技巧來達成，然後使用質心的位置差提供點轉換方程式作為參數。

圖形定位:

使用 Pattern Match 的方法，判斷影像內存在的圖案是否一致，然後使用影像的位置差來提供點轉換參數。

方法:

設定好兩個光學點，決定影像處理的方法，接著將影像的取向範圍設定，必要時提供延遲時間，整體偏移，以及按照方法不同，提供雜點去除程度或者是 Pattern 設定，然後在測試完後沒有偏移就是成功的設定。

應用:

利用取得光學點後，將其應用在拼板之下，可以做成拼板自動判定並補正的效果，是點膠機能夠在高產下也能提供高準度的關鍵。

iv. 壞版與點膠點偵測:

目的:

壞板:

在光學點定位後會發現出現錯誤，因此會將光學點錯誤的拼板位置設定為壞板，將特定的膠下在此位置後，藉由壞板設定的數值來標記壞板或提供壞板判定。

點膠點偵測:

用來檢查點是否有點好的判定，如果沒有點好，在位置上就會找不到膠，所以需要點膠點判定。

方法:

特定位置下統整出灰階的平均值，藉由取得膠的灰階數值來將其和測試結果比較，來做判定。

應用:

壞板用在標記上，可以清楚提供其他機器或是工作人員輕易地得知此板是否需要進型作業。

點膠點偵測用在檢查，利用判定點膠是否存在來決定重新點膠或是告知工作人員檢查膠閥是否有問題或出膠高度不正確。

5. 結語與展望

為了驗證我們光學點補正後修正的座標精準度，我們藉由輸出執行時的軸控座標到 AutoCAD 的方式來實測，誤差在 5~6 位小數點內，不但滿足精準度上的需求，執行中的辨識處理速度也相當快，成功達成客戶所希望的，又快又準的需求。目前也已經成功匯出了第一版系統與機器，將來還會繼續維護第二版的程序以及新增其他功能，也希望能繼續加強光學點補正系統。

6. 銘謝

感謝指導教授曾俊元老師提供舒適環境供我們能全力投入專題製作。

感謝台灣樂業曾世明先生多次帶我們前往公司見學點膠機實際運作過程和操作以及討論新功能。

感謝王聖超學長、郭皓德學長、莊文立學長在我們遇到瓶頸時提供各方面的協助。

7.參考文獻

[1]研華科技,2012, “PCI-1245/1265用户手冊中文”,研華科技,第一版

[2]研華科技,2012, “PCI-1245/1265用户手冊英文”,研華科技,第一版