逢 甲 大 學

資 訊 工 程 學 系

專題研究報告

鋼琴樂譜辨識以及播放系統

指導教授：張貴忠

學 生：周蔓君(資訊四丙)

中 華 民 國 一百 年 五 月

摘要

在影像辨識的技術日漸發達與普及,可以廣泛的將此技術應用在各種生活與學習上,因此在本專題中,嘗試了去辨識一個在生活中算是可以常見到的圖片類型,也就是樂譜的辨識,期望能夠設計出一個簡單播放音樂與簡易鋼琴鍵盤教學的系統,但是受限在樂譜的表達方式不同以及複雜度的關係上,因此辨識的方面還沒辦法非常完整的辨識。

在此專題中,運用的都是一些利用圖形上黑白界線區別的切割方式,然後在藉由找尋出樂譜上各種特徵來進行開發,而開發的程式是使用 JAVA,因此可以在安裝有 java 的各種硬體上使用,並且能夠在持續往手機等方面發展。

目錄

圖表目錄

第一章 緒論

1.1 研究動機

隨著現在各種科技的發達以及，人們對於物質上也有著越來越多的要求，對於“美”的定義也越來越嚴格，因此不再只是滿足於眼睛所能見的享受，同時對於聽的感受也是有著十分的要求，音樂幾乎是充斥在我們生活的四周，因此可以知道音樂對於現在人們的生活而言儼然已是不可或缺的一部分了，許多東西如果配合了音樂，就彷彿有了靈魂，就好像一部影片，如果只是單純的影像，那麼觀看的過程中就會感覺無趣，相反的配上了適當的音樂，會讓影片的趣味性更加的吸引人。

隨著音樂的發展，使得人們對於音樂的理解也越來越深入，也因而有著越來越多的興趣，並且也不在滿足於單純的聆聽他人所創造、演奏出來的音樂，有更多的人願意去學習並且演奏出屬於自己的音樂，並且在加上現在環境的影響以及許可，也有著許多的父母願意花錢讓自己的孩子去學習一門才藝，而這一門才藝往往也以音樂演奏方面的才藝為主，並且在演奏的樂器中，鋼琴是一個許多的人會選擇的一個樂器。

因此在現在計算機普及的情況之下，便希望如果能夠有一個系統可以輔助於這一些願意學習演奏的人進行學習，因此在這次專題中，選擇了辨識樂譜的一個方式來做此系統，因為樂譜的閱讀會是學習者最早遇到的一個問題，並且也希望可以藉由此系統，讓人可以了解樂譜所表達音樂，而且希望在未來可以應用在手機上，這樣就可以在鋼琴邊就可以使用，而不需要用到電腦。

1.2 研究目標

在學習任何樂器演奏的過程中，初學者首先遇到的問題，就是必須能夠閱讀並了解此樂器音樂的表達方式，但是在看到樂譜的時候，卻只是一張圖片，除非是能夠非常理解樂譜的人，否則很難從看譜的過程中，了解到這一個樂譜演奏出來的成果，因此在本專題中，主要的目的就是希望可以藉由讀取樂譜的圖片，來進行音樂的撥放，這樣就可以讓使用者了解到讀取的樂譜撥放出來的音樂聽起來的效果為何。

在專題中，是藉由判斷出樂譜中由黑色像素以及白色像素值所呈現的各種音符不同的位置以及特徵碼來判斷音樂和速度，在使用 midi 來撥放出音樂。有音樂中最基礎的撥放、停止以及暫停功能，可以讓使用者選擇，並且也可以讓使用者根據需求以及自行了解的狀況來選擇撥放的速度，以便於可以讓使用者更加的了解樂譜的音樂。

第二章 相關研究

2.1 BMP 檔案格式

BMP 文件通常是不壓縮的，所以它們通常比同一幅圖像的壓縮圖像文件格式要大很多，但是也因為是非經過壓縮圖像的關係，所以其包含的資訊與其他經過壓縮後的圖像相比也較完整，因此較其他壓縮圖像文件來的適合使用於影像辨識，BMP 檔案結構通常包含下面幾個數據塊：

1. 點陣圖頭：保存點陣圖文件的總體信息。
2. 點陣圖信息：保存點陣圖圖像的詳細信息。
3. 調色板：保存所用顏色的定義。
4. 點陣圖數據：保存一個又一個像素的實際圖像。

點陣圖頭的 14 個位元再加上點陣圖信息的 40 個位元，合起來總共為 54 個位元，而這 54 個位元在 BMP 檔案中稱之為文件頭，也是檔案開啟後會最先讀到的 54 個位元。

點陣圖頭這部分是識別信息，典型的應用程序會首先普通讀取這部分數據以確保的確是點陣圖文件並且沒有損壞，其內容如下：

位元組 #0-1 保存點陣圖文件的標識符，這兩個位元組的典型數據是 **BM**。

位元組 #2-5 使用一個 dword 保存點陣圖文件大小。

位元組 #6-9 是保留部分，留做以後的擴展使用，對實際的解碼格式沒有影響。

位元組 #10-13 保存點陣圖數據位置的地址偏移，也就是起始地址。

點陣圖訊息這部分則告訴應用程序圖像的詳細信息,在螢幕上顯示圖像將會使用這些信息，它從文件的第 15 個位元組開始，其內容如下：

位元組 #14-17 定義以下用來描述影像的區塊（BitmapInfoHeader）的大小。它的值是：**40** - Windows 3.2、95、NT、**12** - OS/2 1.x、240 - OS/2 2.x

位元組 #18-21 保存點陣圖寬度（以像素個數表示）。

位元組 #22-25 保存點陣圖高度（以像素個數表示）。

位元組 #26-27 保存所用彩色位面的個數。不經常使用。

位元組 #28-29 保存每個像素的位數，它是圖像的顏色深度。常用值是 1、4、8（灰階）和 24（彩色）。

位元組 #30-33 定義所用的壓縮演算法。允許的值是 0、1、2、3、4、5。

位元組 #34-37 保存圖像大小。這是原始點陣圖數據的大小。

位元組 #38-41 保存圖像水平方向解析度。

位元組 #42-45 保存圖像豎值方向解析度。

位元組 #46-49 保存所用顏色數目。

位元組 #50-53 保存所用重要顏色數目。當每個顏色都重要時這個值與顏色數目相等。

而在調色盤這部分定義了圖像中所用的顏色。如上所述，點陣圖圖像一個像素接著一個像素儲存，每個像素使用一個或者多個位元組的值表示，所以調色板的目的就是要告訴應用程序這些值所對應的實際顏色。典型的點陣圖文件使用 RGB 彩色模型。在這種模型中，每種顏色都是由不同強度（從 0 到最大強度）的紅色（R）、綠色（G）和藍色（B）組成的，也就是說，每種顏色都可以使用紅色、綠色和藍色的值所定義。

點陣圖數據是逐個像素表示圖像，而且像素是由下而上，由左而右讀取紀錄保存的，在 BMP 圖片檔案中最開始的起始點是圖片的左下角。(參考資料：[4])

2.2 Midi

2.2.1 midi 簡介

樂器數位介面（Musical Instrument Digital Interface，簡稱 MIDI）是一個工業標準的電子通訊協定，為電子樂器等演奏裝置（如合成器）定義各種音符或彈奏碼，容許電子樂器、電腦或其它的演奏配備彼此連接，調節和同步，得即時交換演奏資料，MIDI 不傳送聲音，只傳送像是音調和音樂強度的數位數據， 音量，抖音和方位等參數的控制訊號，還有設定節奏的時鐘信號。在不同的電腦上，輸出的聲音也有所不同。

MIDI 播映控制協議(MSC Protocol)是為 MIDI 而設的工業標準，由 MIDI 設備生產商協會在 1991 年制定。它允許不同種類的媒體控制裝置在相互之間的通訊，藉助電腦可以表現現場顯示控制的功能與娛樂應用。與音樂 MIDI 相同，MSC 並不傳輸實際顯示的媒體 — 它只是簡單地傳輸有關多媒體性能的數位訊號。現在，幾乎所有的音樂錄音將 MIDI 作為一項關鍵開放技術來記錄音樂。

MIDI 使得電腦、合成器、音效卡以及電子鼓樂器能 互相控制、交換資訊。雖然現在的電腦的音效卡;都是 MIDI-相容的，並能逼真地模擬樂器的聲音，事實上，音效卡的 MIDI 合成器在歷史上導致了半信半疑 的聲音的質量損害了一臺作為 MIDI 樂器的通用電腦的形象。這是 MIDI 規格本身和根據

音效卡聲音使用的質量無關導致。

2.2.2 Midi 音軌訊息運作簡介

當 MIDI 樂器演奏了一個音符的時候，它隨之將音符轉換成 MIDI 資訊。一個典型的由鍵盤獲取的音符的 MIDI 資訊的過程包括：

1. 使用者以特定速率（此速率通常轉變成音符的音量，但也可以用合成器設定音符的音色）演奏中央 C 音符
2. 使用者改變按壓鍵盤按鍵的力度-這個技術稱為鍵後觸感
3. 使用者釋放並停止演奏中央 C 音符

MIDI 資訊傳輸速率達到每秒 31250 位。其它的相關參數同時也被一同轉換。例如，例如當變調輪有所變化的時候，這個資訊也將在 MIDI 資訊中有所呈現。只要演奏者演奏音符，樂器就可以自主的完成這樣的資料採集工作。

樂器所演奏的所有音符根據其音名和音程的不同都有特定的 MIDI 資訊。例如，任何樂器演奏的中央 C 音符，它的 MIDI 資訊都是一致的。這樣使得它所生成的二進制資訊也保持一致，這種規範化的宣告方法是 MIDI 標準的核心部分。

所有的 MIDI 樂器都遵循 MIDI 規範說明，這樣使得其生成的 MIDI 資訊能夠明確的指明具體的音符。藉助這樣的標準和協定，所有的 MIDI 樂器可以相互交換資訊，同時也可以和具有 MIDI 識別或者 MIDI 軟體的電腦進行資訊交換。MIDI 介面用於將當前 MIDI 樂器生成的 MIDI 資訊轉換成二進制 代碼，以讓接收端的 MIDI 樂器或電腦識別處理。所有的 MIDI 樂器都有內建介面。另外，電腦的音效卡通常也具有這種內建的 MIDI 介面。(參考資料:[5])

2.3 辨識方式

有關於圖片的辨識方面的各種研究，在目前其實有著不少此方面的研究，其中一部分的研究是有關於車牌辨識方面(參考資料：[1])，而本專題也是由著這一方面去做一個參考，在其中在這邊針對在此專題中沒用到比較特殊的方式來進行討論。

邊緣遮罩：

邊緣遮罩是一個可以將圖片邊緣地區找出的方式，邊緣的定義為，當兩色灰階度差距時，便稱他是一個邊緣，灰階差距越大，則是一個越明顯的邊緣，灰階差距越小，則是一個不明顯的邊緣。

邊緣遮罩主要知名的方式為『Sobel 遮罩』以及『拉普拉斯遮罩』，遮罩的計算方式是，將一座標本身和身旁的八個方向座標點，總共九個座標點，分別乘上遮罩的係數，將所得到的答案取代回原本座標的像素值。

但是在樂譜的圖片中，因為邊緣基本都是非常明顯，並且在經過簡化的過程，也就是灰階和二值化，基本可以判斷邊緣，而沒有特殊模糊地帶，因此沒有特意在去做尋找邊緣的必要，所以此方式就沒有使用到了。

第三章 系統架構與實作

3.1 系統架構

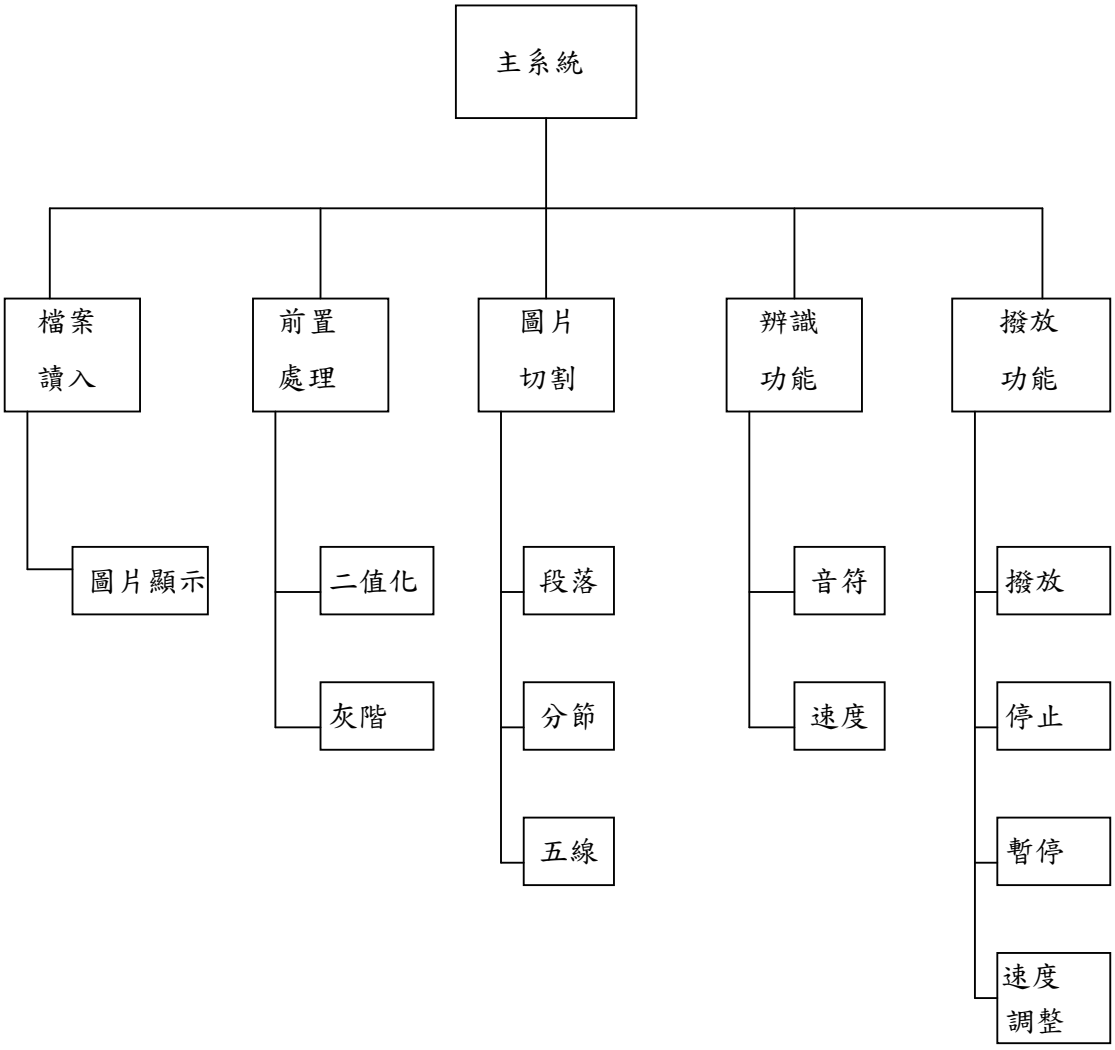

圖 3.1　系統架構圖

3.2 系統流程

研究的過程最主要有三個步驟，第一步是開啟樂譜，並將樂譜做一個初步的處理，然後第二步是切割圖片，將不同區域的位置判斷出來，第三步則是辨識音符以及速度，然後將辨識結果撥放出來。

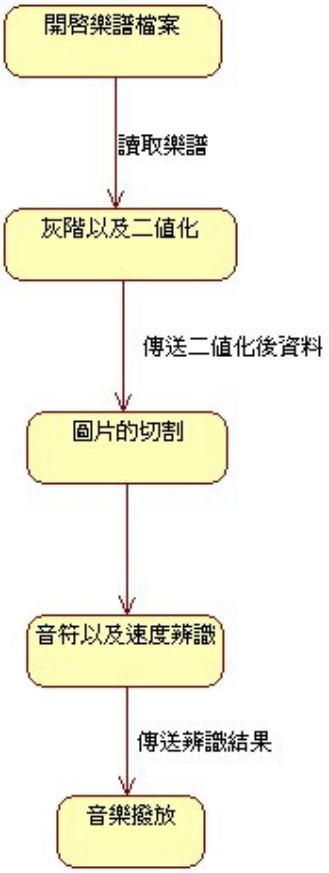

圖 3.2　系統流程圖

3.3 圖片限制

本專題中的圖片來源為圖片經過掃描機取得，在系統上判斷所使用的圖片為 BMP 檔案的格式，因此如果掃描結果必須為 BMP 格式檔，如果掃描結果非 BMP 檔案則必須自行轉換成 256 色的 BMP 格式檔案。

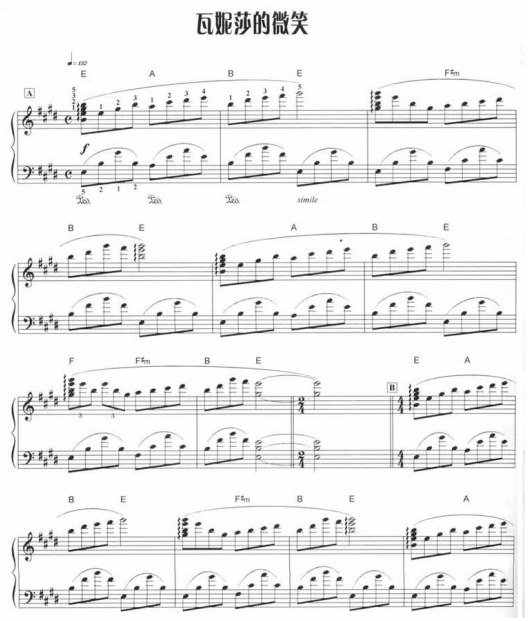

圖 3.3　複雜度高的鋼琴譜

在以上的鋼琴圖譜中，可以看到在除了音符之外還有許多其他的符號，而那一些符號在本次的專題是不需要使用的，並且因為太多的符號造成圖譜的複雜性，因此考慮過後選擇簡單性的圖譜，因此在選擇圖片判斷時，並不接受過於複雜的圖譜，並且在這次的判斷中，也不接受有升降記號的樂譜。

3.4 前置處理

3.4.1 BMP 檔案格式讀取

因為在讀取圖片的過程中，無法事先知道讀取的圖片大小為多少，因此必須要從 BMP 圖片中直接讀取資訊，來設定其寬(X)與高(Y)，在藉由讀取到的寬和高來儲存讀入的圖片檔案中的 RGB 的值，將檔案儲存為(Y,X)格式。

在 BMP 中前 54 位元的檔頭中，我們會使用到的是 18~25 位元的資訊，18~21 是保存 BMP 圖片的寬度，而 22~25 則是保存 BMP 圖片的高度。

```
switch(count)
{
  case 18:
    w = getin;
    break;
  case 19:
    w = getin*256 + w;
    break;
  case 20:
    w = getin*65536 + w;
    break;
  case 21:
    w = getin*16777216 + w;
    break;
  case 22:
    h = getin;
    break;
  case 23:
    h = getin*256 + h;
    break;
  case 24:
    h = getin*65536 + h;
    break;
  case 25:
    h = getin*16777216 + h;
    break;
  default : break;
}
```

圖 3.4　寬和高計算方式程式碼

3.4.2灰階

雖然鋼琴譜乍看之下是屬於灰階值的圖，但是我們無法百分之百的肯定經過掃描的圖，是否會出現非灰階值的值，因此須先將樂譜進行灰階的轉換，而經過灰階的轉換過後的圖，RGB 三色會是相同的，如此一來便有利之後進行二值化以及判斷的過程，相反，如果出現非灰階值，則可能影響到二值化的結果進而影響後面的判斷。

灰階公式如下：

Gray 值 = R 值 * 0.299 + G 值 * 0.587 + B 值 * 0.114

3.4.3 二值化

因為希望在辨識的過程中，可以讓圖片複雜度盡量簡單，以便於降低便識的困難度，因此希望圖片可以讓像素呈現純粹的黑白兩色，但是一般來說圖片不會一開始就是以純粹的黑白兩色像素值呈現，所以需要經過一個二值話的過程。

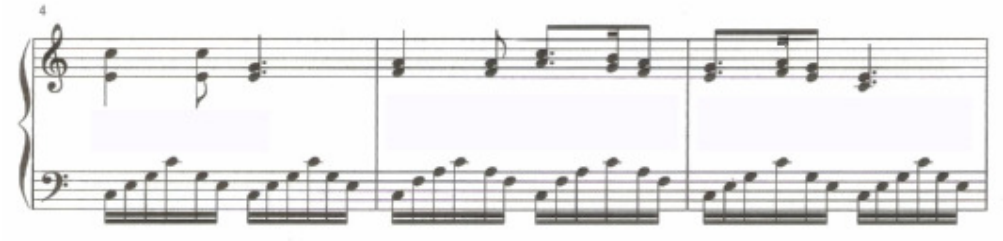

圖 3.5 未經過二值化的圖片

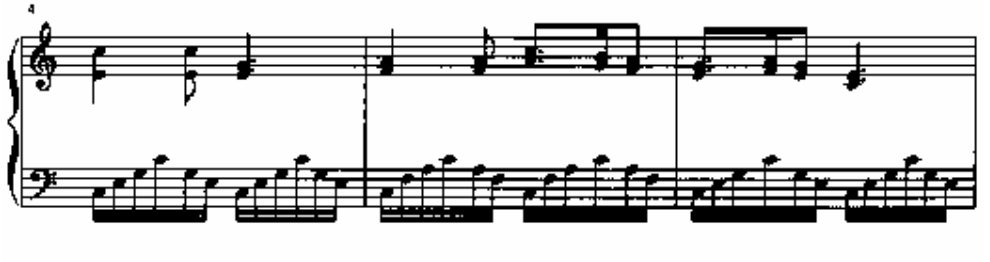

圖 3.6 經過二值化的圖片

二值化的概念是找出一個門檻值，將大於門檻值得像素值轉換成 255，而相反的則是將低於門檻值的像素轉換成 0，以下為門檻值選取程式碼：

```
do{
   rem = avg;
   big = 0;
   small = 0;
   bc = 0;
   sc = 0;

   for(int a = 0; a < h; a++){
      for(int b = 0; b < w; b++){
         if(putin[a][b] > avg){
            big  += putin[a][b];
            bc++;
         }
         else{
            small += putin[a][b];
            sc++;
         }
      }
   }

   big  = big / bc;
   small = small /sc;

   avg = (big + small) / 2;
}while(avg != rem);
```

圖 3.7　二值化門檻程式碼

此門檻值程式的概念是，先將圖片中所有的像素累加找出並找出累加過後各像素的平均值 Tn，然後以此平均值作區隔，將高於平均值 Tn 的像素值累加，並算出此累加的平均值，以及低於平均值 Tn 的像素值累加，並也算出此累加平均值，將這兩個平均值再做一次平均，並設定為新的平均值，直到平均值不改變，則將此平均值設定為此二值化的門檻值。

3.5 圖片切割

3.5.1　段落切割

首先我希望可以將樂譜中以高低音為基本的每一個區域做一個段落的分割，而進行此一段落判斷的概念為，先由樂譜中的 Y 方向足一找尋其像素值，然後再延著 X 座標方向往下的 Y 方向再重新進行一次足一像素值的尋找，重複此動作直到出現第一個黑色的像素值，然後便將這個找到的第一個黑色像素值的 Y 座標位置定為此一段落的起始值，之後在繼續沿著 X 座標往下尋找，直到 Y 座標的方向不再出現黑色像素值，便將此 Y 座標定為這一個段落的終點值。

```
for(a =0; a < h; a++){
   rcut1 = 0;
   count = 0;

   for(b = 0; b < w; b++ ){
      if(putin[a][b] == 0){
         count++;
         rcut1 = 1;
      }// if black
   }//end for

   if(count > w/3)fpost = 1;

   if(rcut2 == 0){
      if(rcut1 == 1)savestar = a;
   }
   else if(rcut2 == 1){
      if(rcut1 == 0){
         saveend = a;
         if(fpost == 1){
            set_se(savestar,saveend,c);
            c++;
         }
         fpost = 0;
      }
   }

   rcut2 = rcut1;
}//end for
```

圖 3.8　段落分割程式碼

在判斷過各個段落的起點值以及終點值後，雖然在理論上應該會將圖片剛剛好切割成進行其他判斷所需要的段落數量，但是在實際操作上，則會出現一些細微雜亂的段落，而這一些段落則非此專題中所需要的段落，因此在判斷段落的當中，會加入另一個判斷方式，利用這一個方式來判斷此段落是否為需要的段落，這個判斷的方式為將段落中 X 方向的黑色像素值逐一累加，如果此其中一行 X 方向的黑色像素值的累加數目大於寬度的 1/3，則將此判斷斷定為需要的段落，其餘如果完全不出現大於寬度 1/3 的數目，則將此段落捨棄不使用。

此一判斷的概念原由於，再樂譜上的有高低音五條線所經過的地方其黑色像素會相當接近寬度，但是避免其中有線段不完整的情況所以將判斷定為寬度 1/3，而在五線譜中所有的符號會與線段相連接所以將判斷定為只需出現一次大於寬度 1/3 的數目即可。

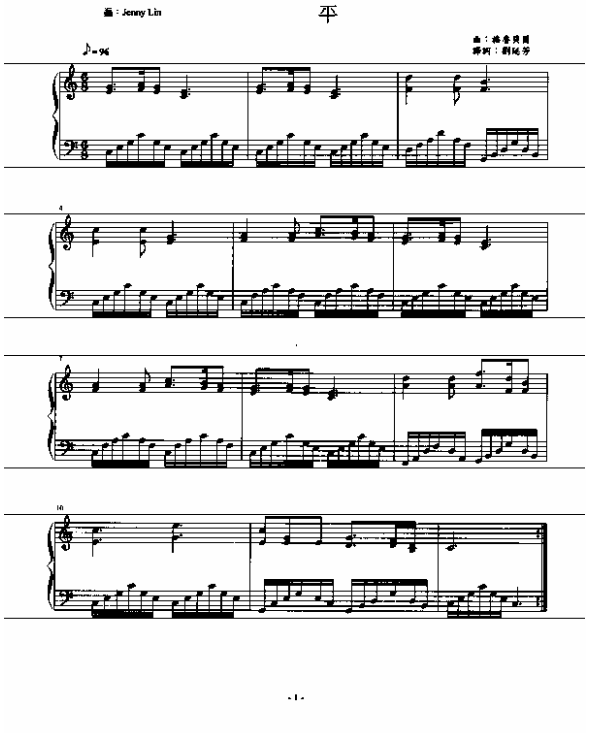

圖 3.9　段落分割示意圖

3.5.2　分節切割

在進行完段落的判斷之後，再來的步驟是希望可以將每一段落中的每一個小節進行分割，因此首先會先選擇一個段落，然後沿著 Y 座標方向進行黑色像素值判斷，將 Y 方向的黑色像素值累加起來，然後如果累積的數值超過 2/3 的段落高度則將其判斷為第一小節的第一條線，並且紀錄這一條線的起始點和終止點，然後以這個起始點與終止點為準，再次進行 Y 座標方向的黑色像素值搜尋，並且進行累加，如果累加的數值大於此起始點與終止點的 7/10 高度，則紀錄為小節的分隔線。

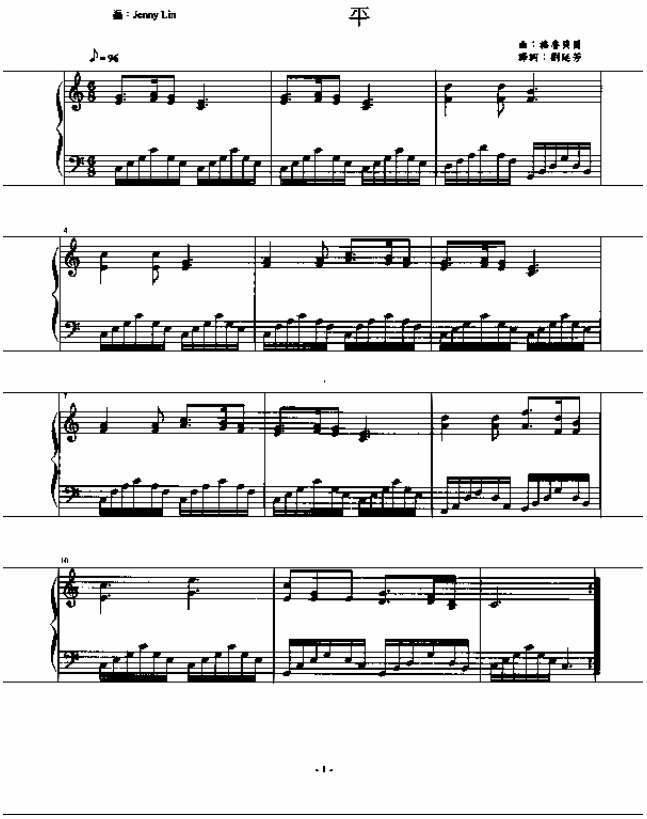

圖 3.10　小節分割是意圖

在一開始的想法是直接將 Y 方向的黑色像素進行累加，然後如果進行累加的數目超過一定數值時，就將他設定為小節的分隔線，但是在實際的操作上面，發現如果在樂譜之中高低音區同時出現音符的線段較長者，就可能會造成一個誤判的狀況存在，因此才會改變做法，先取第一條線段位置，因為第一條線出現的位置，前後沒有其他的音符存在，避免了誤判的可能性，之後才由第一條線的位置做為判斷準則，是因為在高低音之間基本會有一段很大的空白處存在，在區域間出現黑色像素值的機會不高，藉此大幅的降低了誤判的可能性。

而在尋找小節線分隔的過程中，因為可能遇到重複性的小節線，但是實際上非所需要、重複的小節，因此在判斷中，加上了一個條件，也就是將找尋到的小節線記錄起來，並且根據已經紀錄的各個小節線作為根據，如果有出現距離過近的小節線時，則此小節線便不會紀錄，因為我們可以從圖 3.3 中發現其時每一個小節線都有距離一段的距離，因此過近的分隔線是不會出現的，因此憑藉這一點來做為條件。

3.5.3 高低音的五線位置

在高低音五線切割最初的想法是將 X 方向的黑色像素值做累加，然後大於一定寬度的比例，將其判斷為一條線的存在，這個原理的概念是因為從樂譜圖片中，可以發現，五條線的黑色像素在寬度上佔據了極大的空間，但是因為在二值化過程中，為了降低圖片的複雜度，因此線段可能會產生消失不完整的情況，進而判斷錯誤導致線段的消失，並且在其中如果出現許多的連續性的音符存在，如圖 3.4，那麼底下的線段也可能會被系統判斷成線段，但實際上卻不是，因此考慮到這些問題的存在後，便放棄這樣的判斷方式。

圖 3.11 連續音符

五線的分割方式改成先選則在一個段落中最前面線段的位置，如圖 3.5 所表示區域，從圖 3.5 中可以發現，在一開始分節線的後面會存在有一小段的位置是沒有任何符號的存在，因此直接使用那一小段的位置來進行判斷，由 Y 座標方向往下搜尋，只要遇到黑色像素值，就將其判斷為五線譜中一條線，而高低音部分由上往下則各會有著五條線的存在。

圖 3.12 樂譜前段位置

因此在紀錄的方面，是先紀錄前五條線的存在，如果判斷出並且紀錄了五條線之後，其這五條線則為高音部分的線段，而在之後的五條線也就知道了是低音部分的線段。

圖 3.6 中，填滿 X 方向寬度黑線表示出經過判斷為五線的位置。

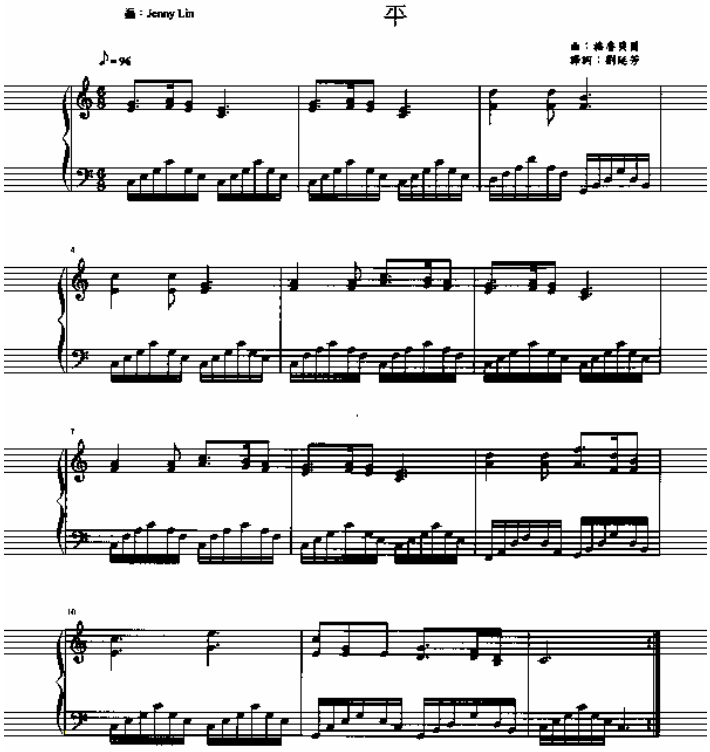

圖 3.13 五線分割示意圖

第四章 音符與速度辨識

4.1 音符切割

在判斷音符的過程，首先最需要的就是先區別出音符的位置，因為在樂譜中，音符出現的位置並不是固定的，因此該怎麼判斷一個音符位置的開始到結束，就是首先遇到的一個問題。

在這邊我採用將高低音部分分開判斷的方式，分隔方式是利用判斷的段落高度的中間值做一個高低音的區隔，切割結果如下圖，中間那條線就代表著切割位置。

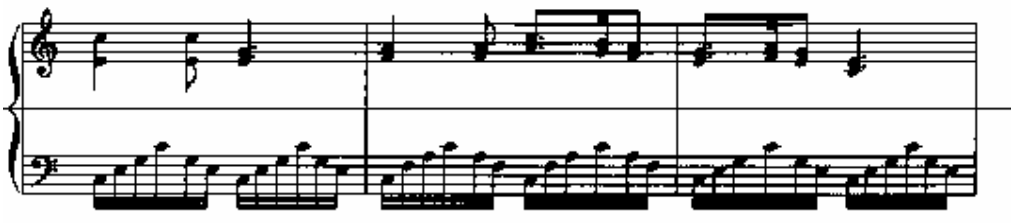

圖 4.1　高低音分割示意圖

然後會先將前面也就是有著表示高低音符位置的地方先切除，然後由起始點，高音部分為段落起始點，低音部分為高低音分割的中間值，搜尋到終止點，高音部分為中間值，低音部分則為段落的終止點，如果遇到黑色像素出現，則先判斷是否為高低五線位置，如果為是，則將此黑色像素視為白色像素，如果為非，則將此點視為音符的開始位置，如果起點至終止點中沒有出現黑色像素，則持續往右移動並重複以上動作，直到找到起始點黑色像素，定義出音符的開始位置後，依舊往右尋找黑色像素，但是如果期間有出現白色像素時，則終止尋找，並將出現白色像素此位置定義為音符的終點位

址。

根據以上方式，根據樂譜可能會尋找到兩種不同的切割，一種是只有一個音符存在的切割，在此先將其稱之為不連續性音符，如圖 4.2。而另外一種則是連接在一起的一整組音符切割，在這將其稱之為連續性音符，如圖 4.3。然後根據不同的切割方式，在進行音符辨識。

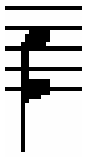

圖 4.2　不連性音符

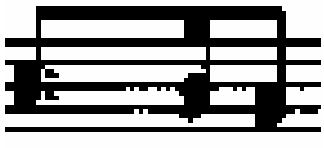

圖 4.3　連續性音符

4.1 不連續音符辨識

在經過分割確認後的不連續音符中，可以看到可能會出現的兩種狀況，如下圖。

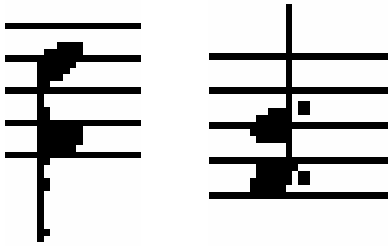

圖 4.4　不連續音符的兩種狀況

在這兩種狀況中我們可以看到，音符會以一條線為主，分開了左右狀況，在線左邊的音符特徵碼會在下方，而在右邊的音符特徵碼則在上方。

因此在判斷的時候，必須要先找出最主要的那條線，從圖中我們可以看到，由上到下，那條線的黑色像素值為最多，因此只要根據音符切割中，黑色像素最多的地方找出，就可以知道線的位置了，然後再由圖中可以知道，音符位於右邊的時候，線的位置基本為此音符的開始點，因此利用線的位置與音符開始的位置比較，如果不超過 3 個像素，則判斷音符為右邊，在以線的位置往右一小段做基本，就可以由上往下找出特徵碼所在位置，而超過三個像素的則判斷音符在線的左邊，因此以線的位置往左一小段做基本，就可以由上往下找出特徵碼的所在位置。

4.2 連續音符辨識

類似於不連續音符，連續音符經過切割後，也會出現兩種狀況

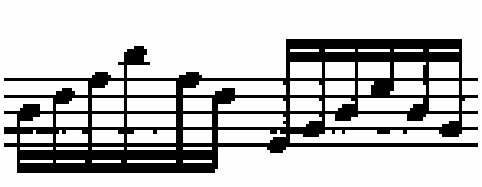

圖 4.4　連續音符的各種狀況

而且在連續性音符中，比起不連續性的音符而言，複雜度更為高，因為在連續性的音符中，同樣無法判斷其中有多少個音符，以及音符在的位置，只知道此連續性音符的開始點與結束點。

不過觀察圖中可以發現到，起始點的位置也基本就是第一個音符出現位置的開始點，而在經過測試後發現在開始點往後 9 個像素便可以將一個音符區隔開來，因此距離起始點 9 個像素的位置就先將其設置為第一個音符的終止點，然後仿照非連續性音符的方式，判斷出線的位置，以及音符特徵碼左右邊，之後就可以判斷出第一個音符的位置，所以最大問題在於下一個音符的起始點位置。

在圖 4.4 中，可以發現從一個音符到下一個音符之間在連續性的橫槓上黑色像素值會是一樣的，因此我們設定將一個音符結束點往後一個像素的位置，紀錄其黑色像素總值，然後往右邊開始搜尋時，紀錄每一行累加的黑色像素，如果有大於紀錄的黑色像素總值，則將其定為下一個音符的開始，之後便在依照著出現第一個音符位置的方式進行判斷特徵碼，然後重複以上步驟，直到這一段連續性音符切割的終止點為止。

4.3 速度辨識

在速度的辨識中，從樂譜經過觀察之後，可以發現，在不連續性的音符中所有關於速度的特徵碼判斷皆位於線的右邊，如下圖所示。

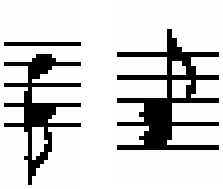

圖 4.5　音符速度特徵碼示意圖

而且如果音符在右邊則特徵碼位於線的下端，如果音符在左邊則特徵碼位於線的上端，而從音符的判斷中以知音符與線的位置關係，因此如果音符位於右邊，則從線所在的位置，由下而上搜尋出第一個黑色像素值出現位置，就可以知道線下端位置在何處，然後在往上幾個像素點後往右尋找，就可以判斷出是否有特徵碼出現，而如果音符位於左邊，則從線所在位置，由上而下搜尋出第一個黑色像素值出現位置，則可知道開始位置，然後在略為往下幾個像素點後往右尋找，就可以找出是否有特徵碼的出現。

而在連續性音符中，特徵碼往往會連接在一起，由圖 4.3 可知，因此所需要判斷的就是連結在一起的部分黑色像素的寬度，因此在判斷出音符位置為左或者右之後，可以知道連續性線段位置是出現在線段的上方或者下方，然後藉此判斷此黑色像素總合的寬度，如果連結的線段只有一條，那寬度應該只有約 3 個像素上下，但如果連結的線段為兩條的話，則寬度會有 7 個像素上下，因此從其寬度便可知道這一段連續性音符的速度特徵碼為何。

4.5 播放方式

判斷完速度以及音符的位置之後，會將判斷的結果存入一個陣列之中，然後再利用 java api 中所含有的 Javax.sound.midi 來進行播放。開始播放音樂時，會將所有在陣列中的數字依序取出，然後對應到 midi 中所規定的各個音階中的各個音符，之後再將這個數字放入對應的 midiChannel 中進行撥放。

Midi 的各個音階計算方式如下：

i*12 + 24 + [0,2,4,5,7,9,11]

括號中的 0,2,4,5,7,9,11 代表著各個音階中的各音符。

當 i 等於 3 時，則為中央 C 的音階，數字越大則表示的音階越高。

第五章 實作結果

5.1 系統介面

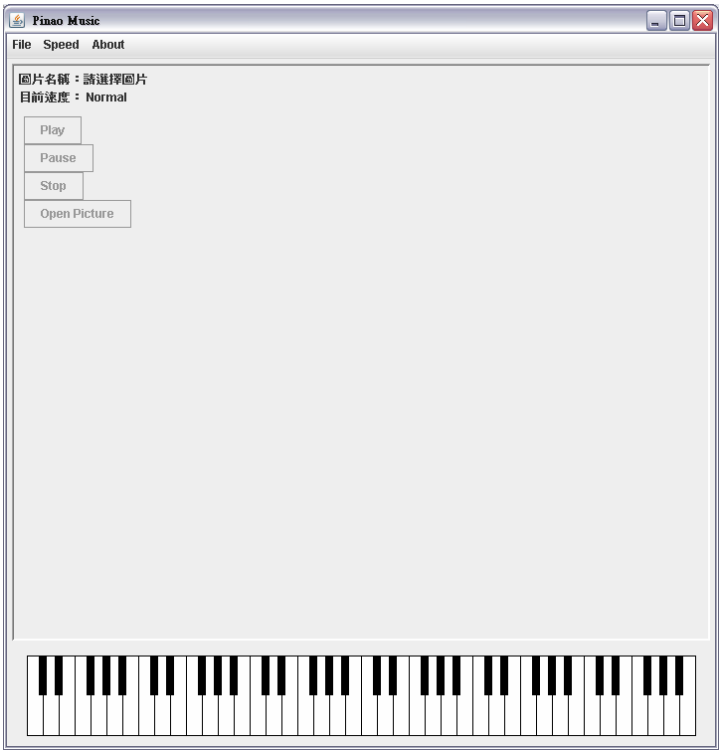

圖 5.1　系統開啟介面

這是系統剛開啟最初始的畫面，因為在目前沒有開啟任何檔案，因此在左邊的功能鍵設定為無法使用，所以呈現灰色，因此在開啟樂譜圖片之前，基本功能是無法使用的，而右邊大塊灰色面積則是開啟檔案後圖片放置的位置，最下面的鋼琴鍵，在播放音符時，會顯示出與播放音符對應的按鍵。

在最上方的“File”是用來開啟檔案、“Speed”是設定樂譜播放速度，目前選擇的速度會顯示在下方的目前速度欄位中，一開始的速度預設為Normal，而“About”則是關於此系統的介紹。

5.2 Menu 功能介紹

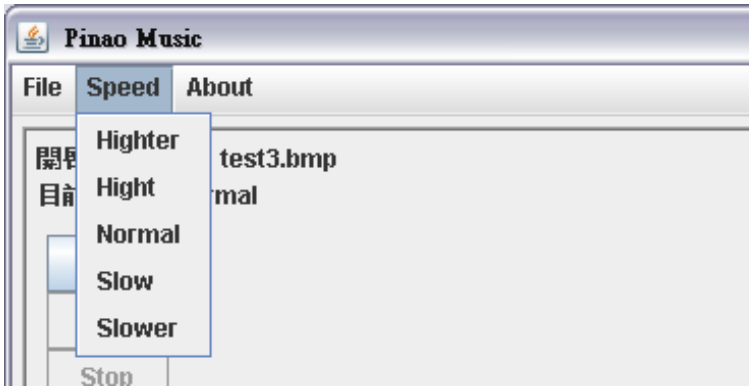

圖 5.2　速度選擇

關於音樂播放的速度，總共分成五個程度，由最快到最慢依序往下，也就是說 Highter 為最快速度而 Slower 為最慢速度，速度這個選項在任何時間皆可選取，未讀圖片時選取也不會影響到後面的功能，如果在音樂播放途中選取，則是會立刻改變音樂的播放速度。

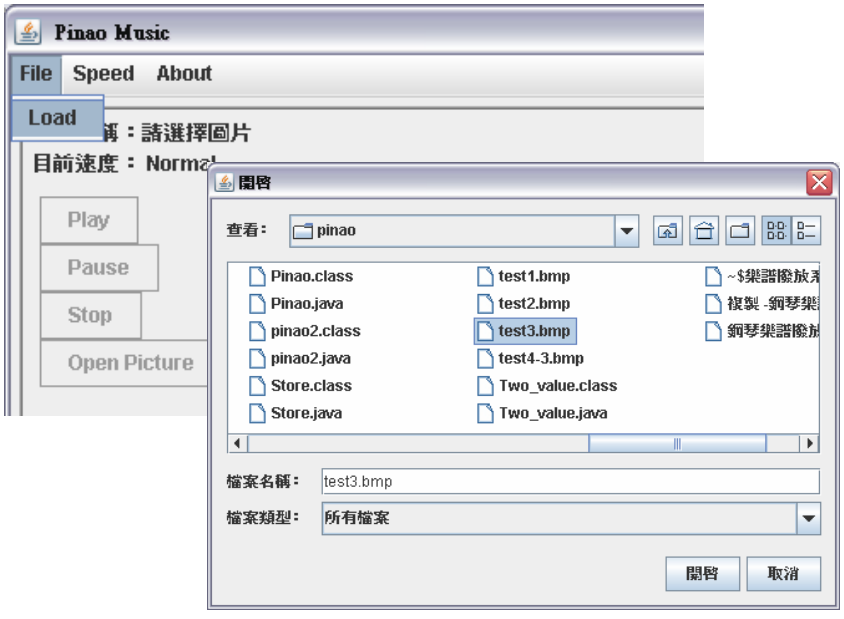

圖 5.3　開啟檔案

按下左上角的 File Load 之後便會開啟一個檔案的選擇器，在選擇完要辨識的 BMP 圖檔並開啟後，便會自動的開始進行樂譜圖片的辨識，等到辨識完成後左邊的功能選單就會出現改變，也就是判斷結束，可以開始進行音符的播放。

從下圖中，可以看到“Play”以及“Open Picture”兩個選項由灰色轉為藍色，也就是可以使用的狀態，而在開啟檔案到按鍵轉換成藍色可使用狀態之前，是樂譜圖片辨識的等待時間。

在開啟樂譜之後，在下方開啟圖片名稱的欄位之中會顯示出目前所選擇開啟的圖片名稱。如果改變檔案的選擇，名稱也會隨之改變。

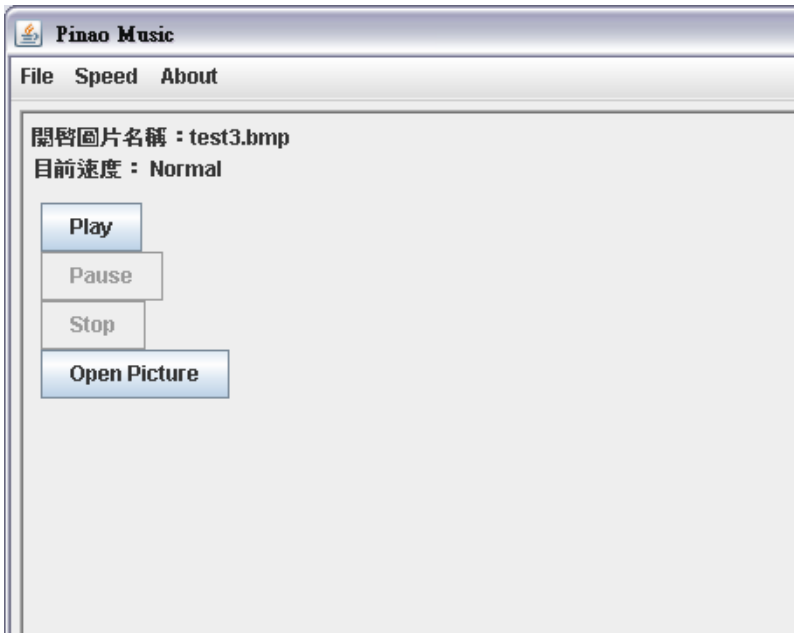

圖 5.4　開啟檔案後改變狀態

5.3 播放以及圖片顯示功能

5.3.1 圖片顯示

在選擇過檔案之後，如果按下 ”Option Picture”便會將選擇的圖片顯示在介面原本右邊的灰色區塊，見圖 5.1，而顯示的圖片會如圖 5.5，顯示出來的圖片大小是固定的，也就是說無論在讀取圖片檔案時的大小為何，在介面中所顯示出來的大小是一樣的。

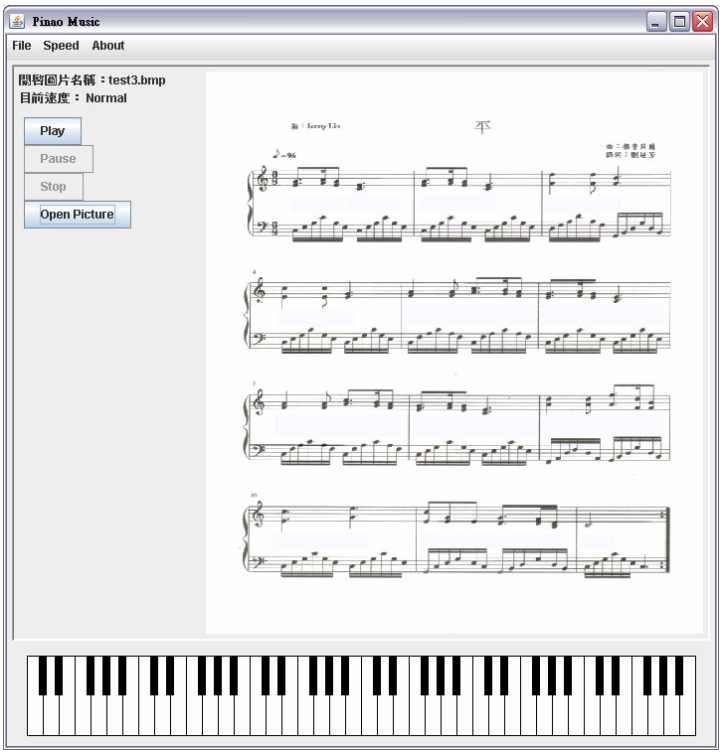

圖 5.5　顯示圖片(圖：平安夜)

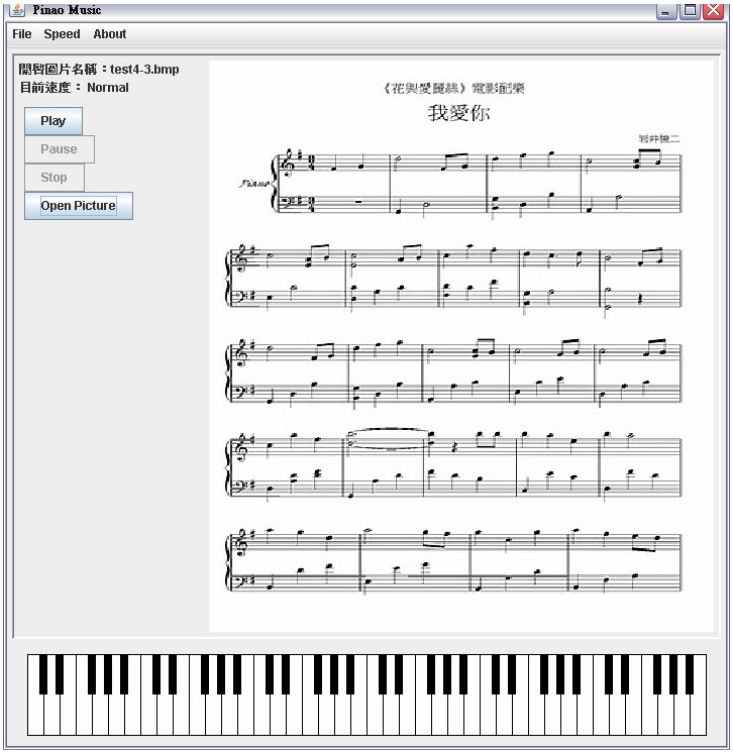

圖 5.6　顯示圖片(圖：我愛你)

5.3.2　播放功能

播放功能，當辨識完成後，左邊功能按鍵中的“Play”鍵便可以使用，但是在此時的“Stop”以及“Pause”鍵都還是呈現灰色無法使用的狀態。而在按下“Play”鍵後，便會立刻播放音樂，並且將音符對應到最下方的按鍵中，當演奏到某個音符時，其對應的按鍵也會變成藍色，相當於在鋼琴中按下的鋼琴建，圖 5.7。

而在按下“Play”鍵後，“Play”會立刻變成灰色，無法使用的狀態，而“Stop”鍵和“Pause”鍵卻會呈現藍色，也就是可以使用的狀態。

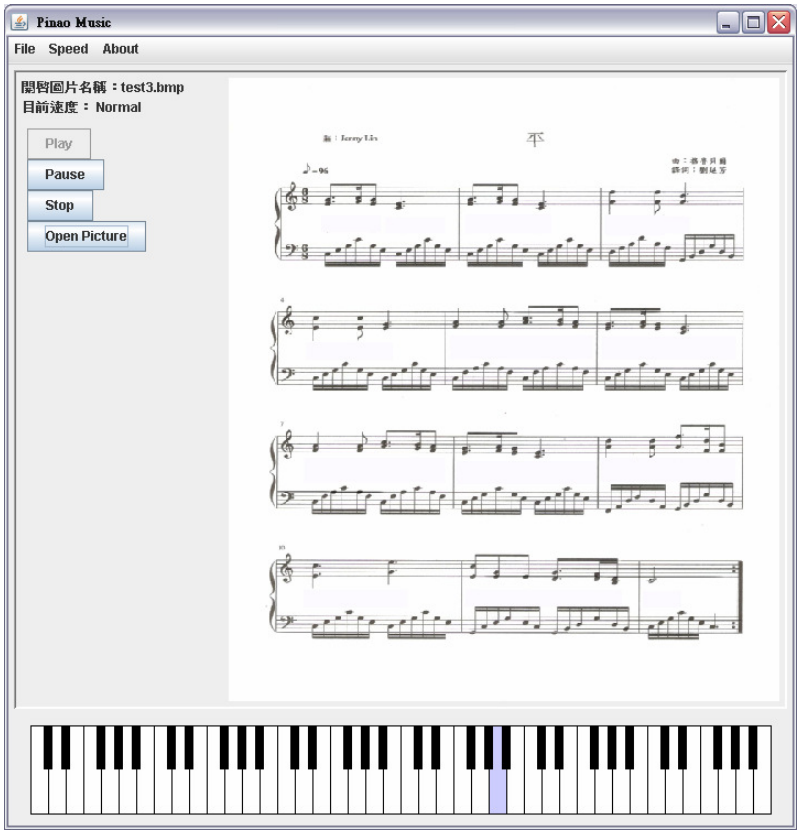

圖 5.7　播放音符以及按鍵對應

如果在音符播放的途中，按下“Stop”鍵，那麼音樂會立刻停止，而下方的按鍵則會停在原本播放到的音符的按鍵上，而且“Play”鍵會重新設置為可以使用，但是同時的“Stop”以及“Pause”則會重新呈現無法使用的狀態，如圖 5.8。

之後在重新按下“Play”鍵的話，會重頭播放起，而鍵盤的顯示也會重新開始。

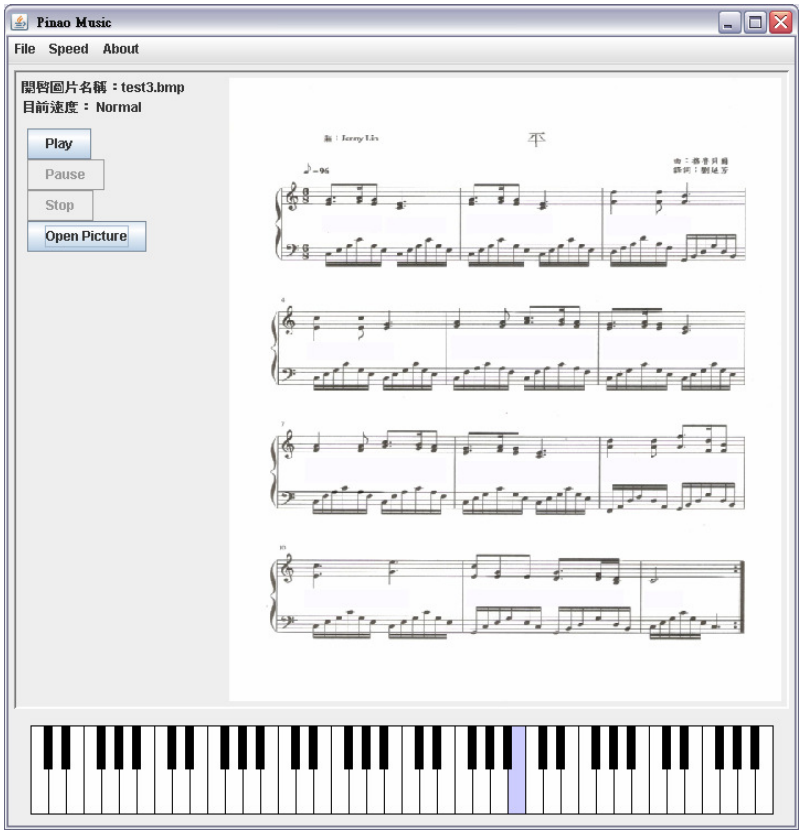

圖 5.8“Stop”狀態

而如果在音樂播放的途中，按下的是“Pause”按鍵的話，那麼音樂也會立刻停止，底下表示音符的按鍵也會停在原本的位置上，並且原本顯示為“Pause”的按鍵會改變成顯示為“Restar”按鍵，如圖 5.9，而之後如果按下“Restar”鍵的話，播放的音符則會從被中斷的地方重新接上，不過在這一個過程中，“Play”鍵都是無法使用狀態的，但是此時的“Stop”案件卻依舊為可使用狀態，如果在按下“Stop”按鍵，則會重新設定為“Stop”狀態下的情況，如圖 5.8，而如果原本被設定為“Restar”的按鍵也會重新設定回“Pause”。

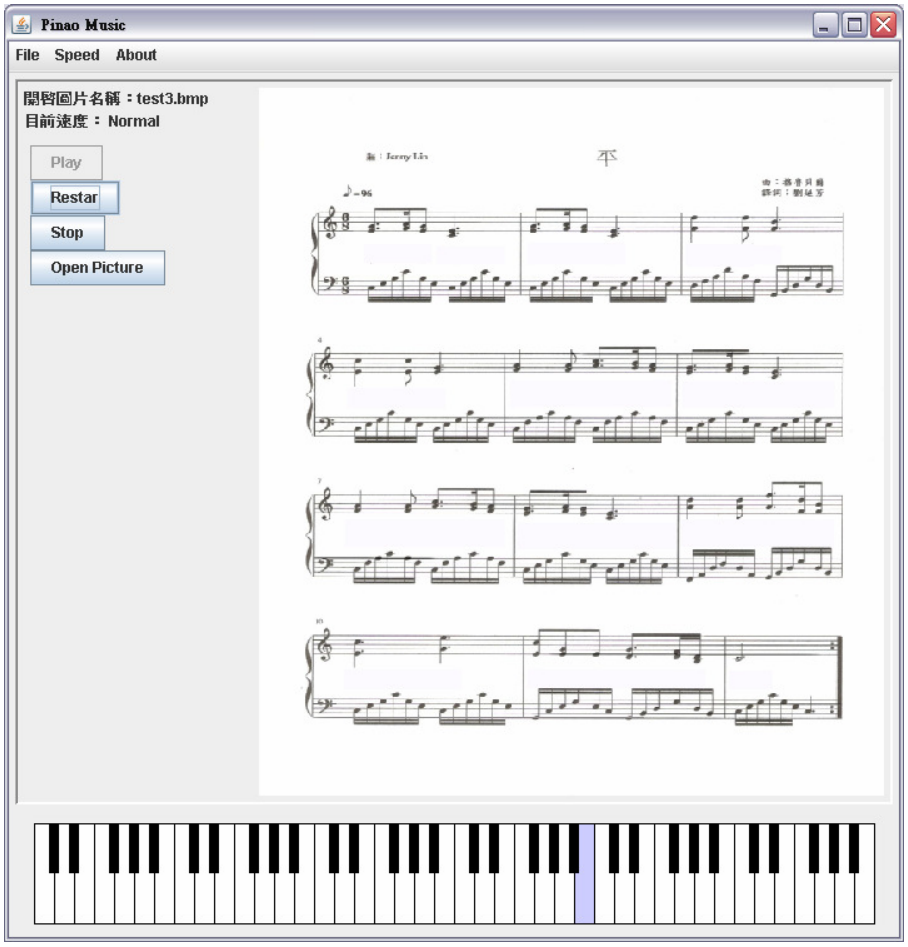
Pinao Music
File Speed About
開啓圖片名稱：test3.bmp
目前速度： Normal
Play
Restar
Stop
Open Picture

圖 5.9 “Pause” 狀態

第六章 心得與未來展望

專題是在大學的四年裡，所碰到的最大的一個程式，從一開始的收集資料、開始對系統做架構設計然後撰寫製作中，往往遇到許多的問題，一直不斷的在修改架構、修改設計的想法，有時就算寫到一半，覺得不對勁，然後就某些部分要整個重新來過，並且在製作的過程中，也一次又一次的了解到自己所學的東西是有多麼的不足，過程中一值是不斷的上網尋找資料、不斷的翻閱書籍，在專題中，真的學習到很多的概念，尤其是撰寫程式方面的一些技術，明白到邏輯在程式中的重要，以及事先的設計對於程式是非常有影響的，因為在撰寫的過程中，往往因為一個思考邏輯的錯誤，就導致程式整個出現奇怪的問題，偏偏城市中往往又是一環接著一環，只有一個地方出現問題，可能就影響到後面，因此有時只要出現設計上的錯誤，就必須一直回頭去觀察到底是哪邊的思考錯了，甚至是整個設計上是不是有問題。

這個專題，或許在現在並不是一個非常完善的系統，其中的問題也還是很多，甚至在辨識上也都還是有著極大的進步空間，但是這畢竟是第一個由自己親自思考、設計出來的程式，因此在當他能夠執行，有了成果後，自己也有著極大的成就。

在未來也希望這一個系統能夠在更加完善，讓音樂不在只能是由 MP3 等音源播放，而是可以由一隻手機，照出一張相片就可以撥放出一組音樂，讓音樂不在只限制於製作好的音效檔。

致謝

感謝在這一個專題的製作過程中，老師張貴忠對我的指導，在我迷惑不知如何進行時，為我提供意見，在專題出現問題時，和我一起討論可能的解決方式。

參考資料

[1] 逢甲大學專題報告 車牌辨識系統 / 學生童子倫 李維斌指導

[2] 影像處理與電腦視覺 / 鍾國亮著

[3] Java 程式設計實例 / 高橋麻奈著 博碩文化編譯

[4] 維基百科 - BMP 簡介 http://zh.wikipedia.org/zh-tw/BMP

[5] 維基百科 - MIDI 簡介 http://zh.wikipedia.org/wiki/MIDI

[6] Java Sound

http://download.oracle.com/javase/1.5.0/docs/guide/sound/index.html