



國立臺北科技大學

電機工程系 學士班

一百一十學年度專題製作報告書

影像辨識鋼琴譜生成器

專題生：電機四甲 葉宏洋 107310115

電機四甲 蕭方凱 107310119

電機四甲 吳宥霖 107310120

電機四甲 呂紹愷 107310131

指導教授：郭天穎 老師

中華民國一百一十年十二月十四日



國立臺北科技大學

電機工程系 學士班

一百一十學年度專題製作報告書

影像辨識鋼琴譜生成器

專題生：電機四甲 葉宏洋 107310115

電機四甲 蕭方凱 107310119

電機四甲 吳宥霖 107310120

電機四甲 呂紹愷 107310131

指導教授：\_\_\_\_\_（簽名）

中華民國一百一十年十二月十四日

# 目錄

|    |                               |    |
|----|-------------------------------|----|
| 壹、 | 摘要.....                       | 3  |
| 貳、 | 致謝.....                       | 4  |
| 參、 | 研究動機.....                     | 5  |
| 肆、 | 使用工具及相關知識.....                | 6  |
| 1、 | Opencv 介紹 .....               | 6  |
| 2、 | Lilypond.....                 | 6  |
| 3、 | Perspective Transform 介紹..... | 7  |
| 4、 | 邊緣處理.....                     | 8  |
| 5、 | 形態學(Morphology) .....         | 9  |
| 6、 | 前景/背景分離.....                  | 9  |
| 伍、 | 設計方法.....                     | 10 |
| 1、 | 系統架構.....                     | 10 |
| 2、 | 系統流程圖.....                    | 10 |
| 3、 | 琴鍵偵測.....                     | 11 |
| 4、 | 琴鍵按壓偵測.....                   | 11 |
| 5、 | 數值處理.....                     | 12 |
| 陸、 | 專題成果.....                     | 16 |
| 1、 | 理想情況模擬.....                   | 16 |
| 2、 | 實際情況.....                     | 17 |
| 柒、 | 結論.....                       | 18 |
| 1、 | 光線影響.....                     | 18 |
| 2、 | 未來展望.....                     | 18 |
| 捌、 | 參考文獻.....                     | 19 |

# 壹、摘要

本組專題利用影像處理去辨別及生成鋼琴樂譜，將網路上或手機程式上的鋼琴彈奏影片進行處理，搭配寫譜軟體，實現輸入影片，輸出五線譜的功能。

主要分為三個部分，第一個部分為鋼琴琴鍵的辨識，辨別每個鋼琴琴鍵的座標位置；第二個部分為鋼琴琴鍵的按壓辨識，辨識按壓琴鍵的座標位置；第三個部分為數值處理，處理第一個部分及第二個部分所獲得的數值，並將其整理轉為五線譜輸出。

## 貳、 致謝

一開始著手製作專題，我們連最基本的鍵盤邊緣都找不太出來，甚至一度想改題目，但實驗室的學長及郭教授仍給了我們許多不同的想法，最後我們決定完成它，遇到問題就想辦法解決，善用資源，才有了我們現在的成果，真的十分感謝 212 實驗室的協助。

謝謝郭天穎教授在每次開會時，都能提供非常多實用且關鍵的建議，也很信任本組的專題，像盞明燈，給予我們方向。也十分謝謝實驗室的兩位學長劉榛、昭榮，每周都不厭其煩地與我們討論，若沒有學長的指引，專題製作的難度也會增大，再次感謝兩位學長及郭教授，從安裝環境到專題各項困難點突破，都能給予我們協助，希望有朝一日，我們也能將所學回饋給學弟妹。

## 參、研究動機

隨著串流影音技術的成熟，現在網路上有非常多的鋼琴彈奏影片，但多數都不會附上免費樂譜，多半只能自己聽、自己寫譜，而有些影片是無聲的，或者是走音的鋼琴，導致無法單靠聲音來辨別，於是本組利用影像辨識，只需透過影像就能生成出五線譜，既能省下時間，又能補足聲音辨識的不足，帶給使用者更好的體驗。

# 肆、使用工具及相關知識

## 1、Opencv 介紹

OpenCV (Open Source Computer Vision) 是由 Intel 公司所開發出來的 Open Source 函式庫，可以製作圖片、視訊、矩陣運算、統計、圖論、資料儲存的相關 C 語言程式設計，相關的領域為：影像處理、電腦視覺、圖形識別、電腦圖學、資訊檢索或遊戲設計，比較有名的製作為物體追蹤、人臉辨識、傅立葉轉換、紋理分析、可以整合不同圖檔格式的矩陣運算，應用在靜態圖片 (bmp、jpg、tif、png)，動態 Webcam 的影像處理[\[1\]\[2\]](#)。

## 2、Lilypond

LilyPond 是一款專業樂譜製作的程式，且支援 GNU / Linux，Mac OS X 和 Windows，它擁有十分強大而且容易使用的樂譜編輯器，僅需要輸入數字符號、程式，即可輸出五線譜的程式，我們只要照著 Lilypond 所要求的格式輸入給它，如圖(1)，就能幫我們生產出我們所要的譜[\[10\]](#)。

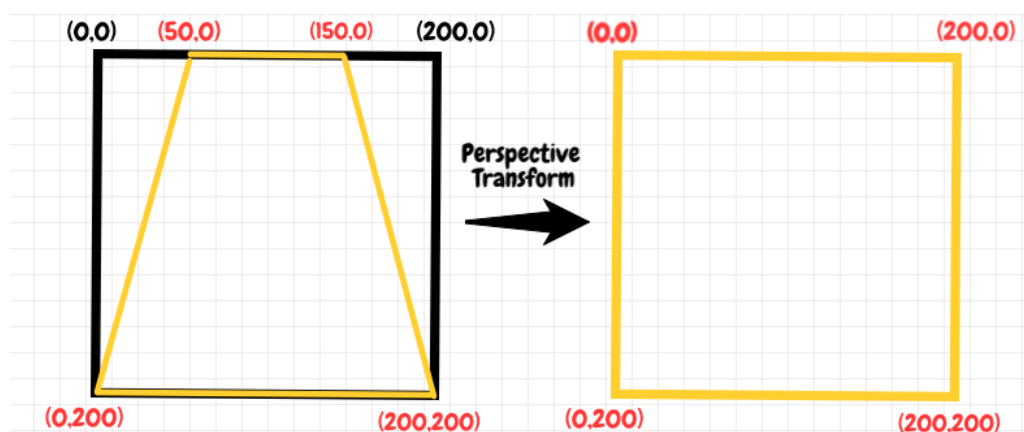
```
lhMusic = \relative {  
  r2 <c' g ees>2 |  
  <d g, d>1 |  
  r2. d,,4 r4 r |  
  r4  
}
```



圖(1) Lilypond 格式及對應之譜

### 3、 Perspective Transform 介紹

透視變換(Perspective Transform)可以將一張圖片投影到一個新的視平面，由於我們需要琴鍵的俯視角，但輸入的影片可能非完全俯視圖，於是使用 Perspective Transform 去將影片拉正為俯視角[3][4]。



圖(2) 透視變換示意圖

如圖(2)的四個點座標[(50,0),(150,0),(0,200),(200,200)]映射到另四個點座標 [(0,0),(200,0),(0,200),(200,200)]，透過一變換矩陣 $\begin{pmatrix} 2 & 0.5 & -100 \\ 0 & 2 & 0 \\ 0 & 0.005 & 1 \end{pmatrix}$ 進行轉換。

圖(3)、圖(4)為實際應用 Perspective Transform 在本專題上。



圖(3) 透視變換前



圖(4) 透視變換後

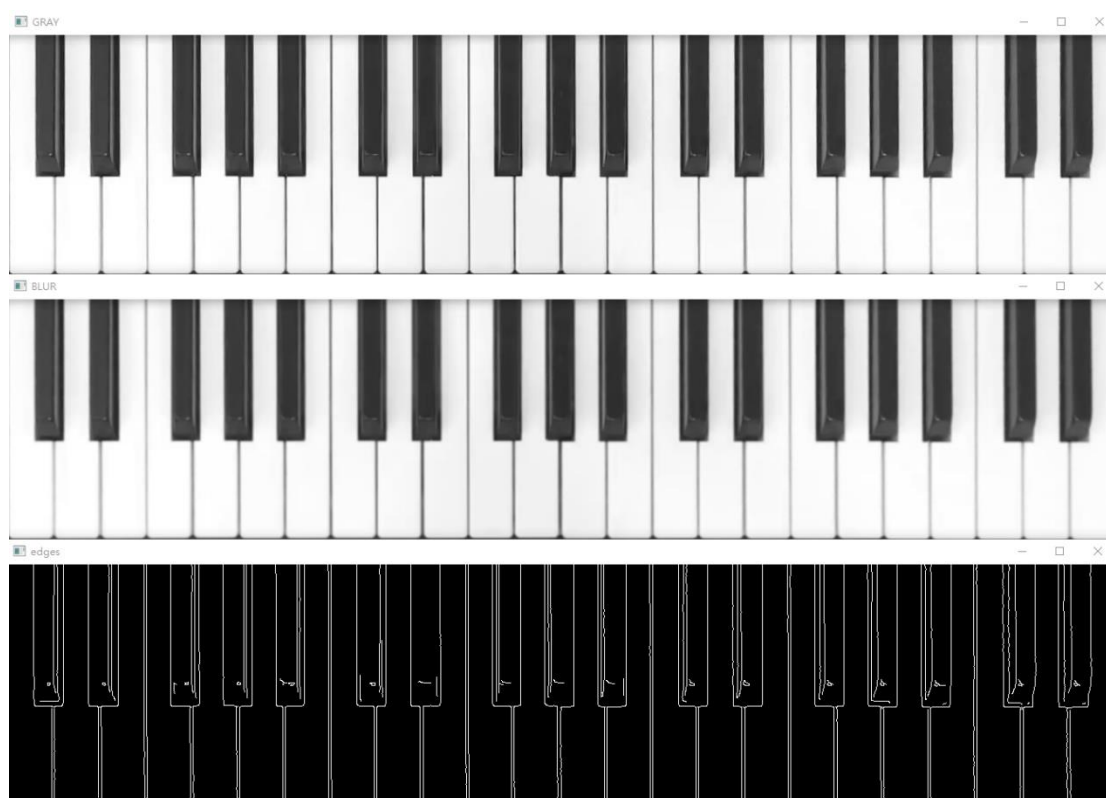


## 4、邊緣處理

邊緣檢測主要目的為標識數位影像中亮度變化明顯的點，是圖像處理和電腦視覺中，尤其是特徵檢測中的一個研究領域。邊緣和物體間的邊界並不相等，邊緣指的是圖像中像素的值有突變的地方，而物體間的邊界指的是現實場景中的存在於物體之間的邊界。有可能有邊緣的地方並非邊界，也有可能邊界的地方並無邊緣。

本專題採用 Canny 方式，使用的是 Gradient methods（梯度原理），它是透過計算像素光度的一階導數差異（detect changes in the first derivative of intensity）來進行邊緣檢測。

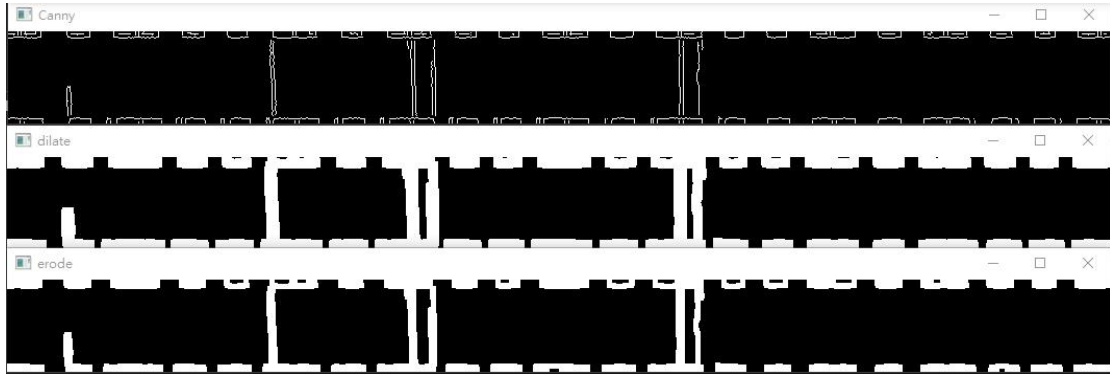
Canny 演算法是一個相當著名的邊緣檢測算法，具有低錯誤率、定位準確且能標識出的邊緣與實際邊緣接近、解析度高使邊緣線條細緻。Canny 演算法是一個複合性的邊緣偵測演算法，結合了 Gaussian Filter、梯度偵測、非最大值抑制、判斷邊界四個演算法去實踐邊緣偵測，在找尋邊緣時相當簡單易用，一口氣把濾雜訊的 Gaussian 跟邊緣偵測的 Sobel 兩個截然不同的濾波器都做進去，直接一個函式找出邊緣，但相對的，Canny 濾波器在實作上也相對沒那麼靈活，無法像 Sobel 跟高斯可以搭配各種不同的影像處理演算法[5][6][7]。



圖(5) 將琴鍵進行邊緣偵測

## 5、 形態學(Morphology)

形態學為在進行影像處理時，經常使用的一種方法，可以用來去除雜訊，也可以用來增強圖片所要傳達的訊息，主要分為四個部分：膨脹(Dilation)、侵蝕(Erosion)、關閉(Closing)、開啟(Opening)，關閉(Closing)為先膨脹再侵蝕；開啟(Opening)為先侵蝕再膨脹[8]。



圖(6) 將琴鍵按壓變化去做 Closing

## 6、 前景/背景分離

前景/背景分離技術可以分離出移動中的前景，有許多種方法可以做到前/後景分離，像是 Background Subtractor MOG、Background Subtractor MOG2、Background Subtractor GMG、Background Subtractor KNN、abs.diff，我們選擇使用 abs.diff 去進行前景/背景分離，它的功能是將兩張照片去做相減後取絕對值，即可找出兩張照片中的相異之處[9]。

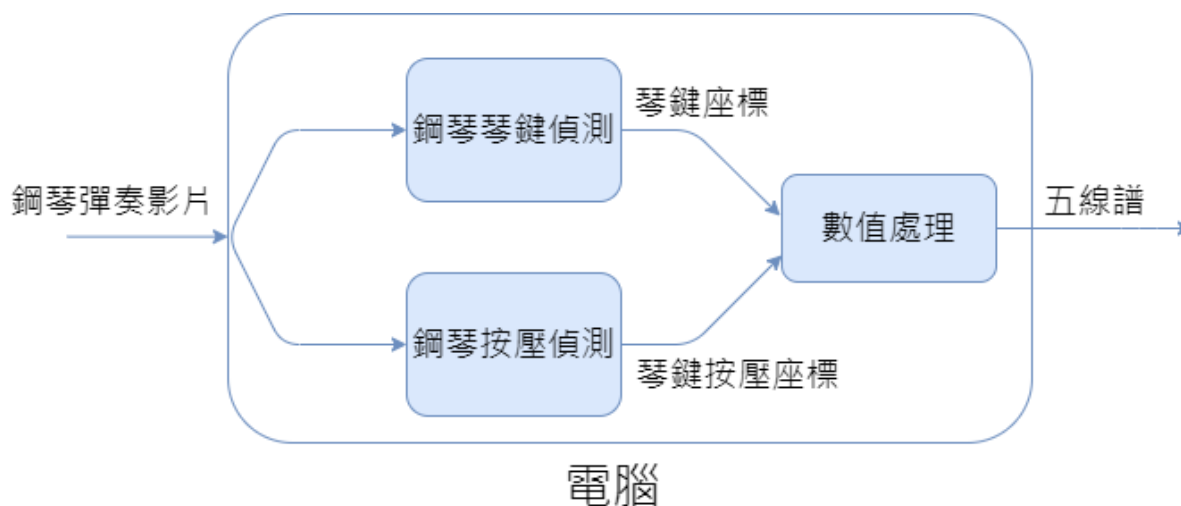


圖(7) 將彈奏影片去做前景/背景分離

## 伍、 設計方法

### 1、 系統架構

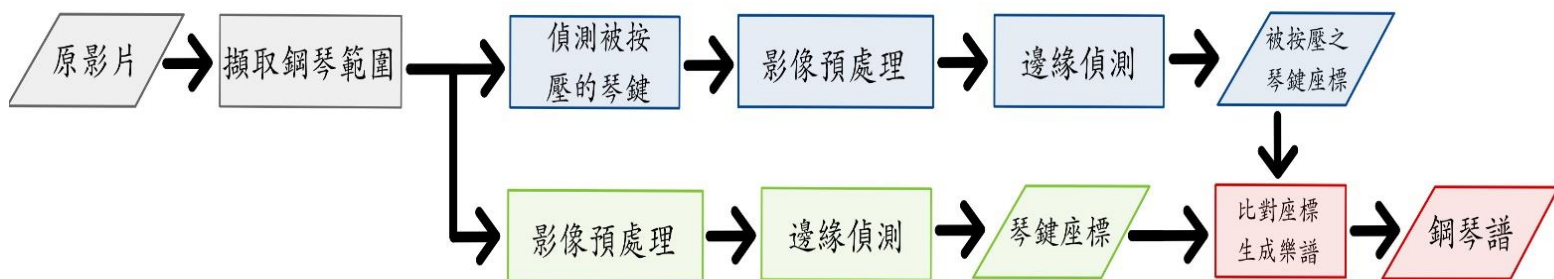
本專題的系統架構如圖(8)所示，輸入一影片到電腦中，即可輸出五線譜。



圖(8) 系統架構圖

### 2、 系統流程圖

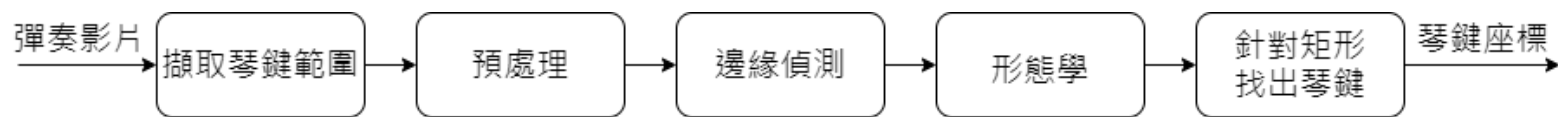
本專題流程分成三大部分，分別為琴鍵偵測、按壓偵測以及將數值處理為樂譜，我們透過影像預處理、邊緣偵測、形態學等，將琴鍵的座標與被按下琴鍵的座標進行比對去生成出鋼琴樂譜。



圖(9) 系統流程圖

### 3、 琴鍵偵測

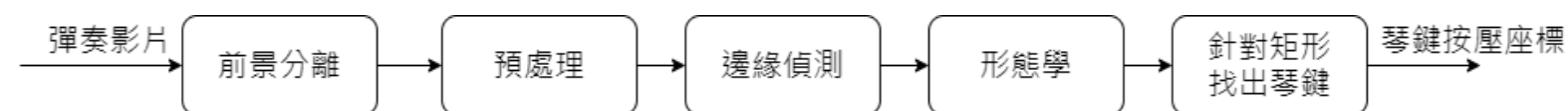
為了將各琴鍵能獨立出來，我們要先獲得每個琴鍵的座標。首先先將影片輸入，接著進行鋼琴範圍擷取，我們只需要琴鍵的部分，所以將多餘的範圍截掉，我們所使用的方法為 Perspective Transform，它除了能做到擷取外，還能將琴鍵拉成我們所要的樣子，接著將擷取後的結果進行預處理，再去產生出邊緣圖，去進行形態學，強化我們所要的資訊，再針對矩形，就能框出每一個琴鍵，最後再將座標數值記錄下來。



圖(10) 琴鍵偵測流程圖

### 4、 琴鍵按壓偵測

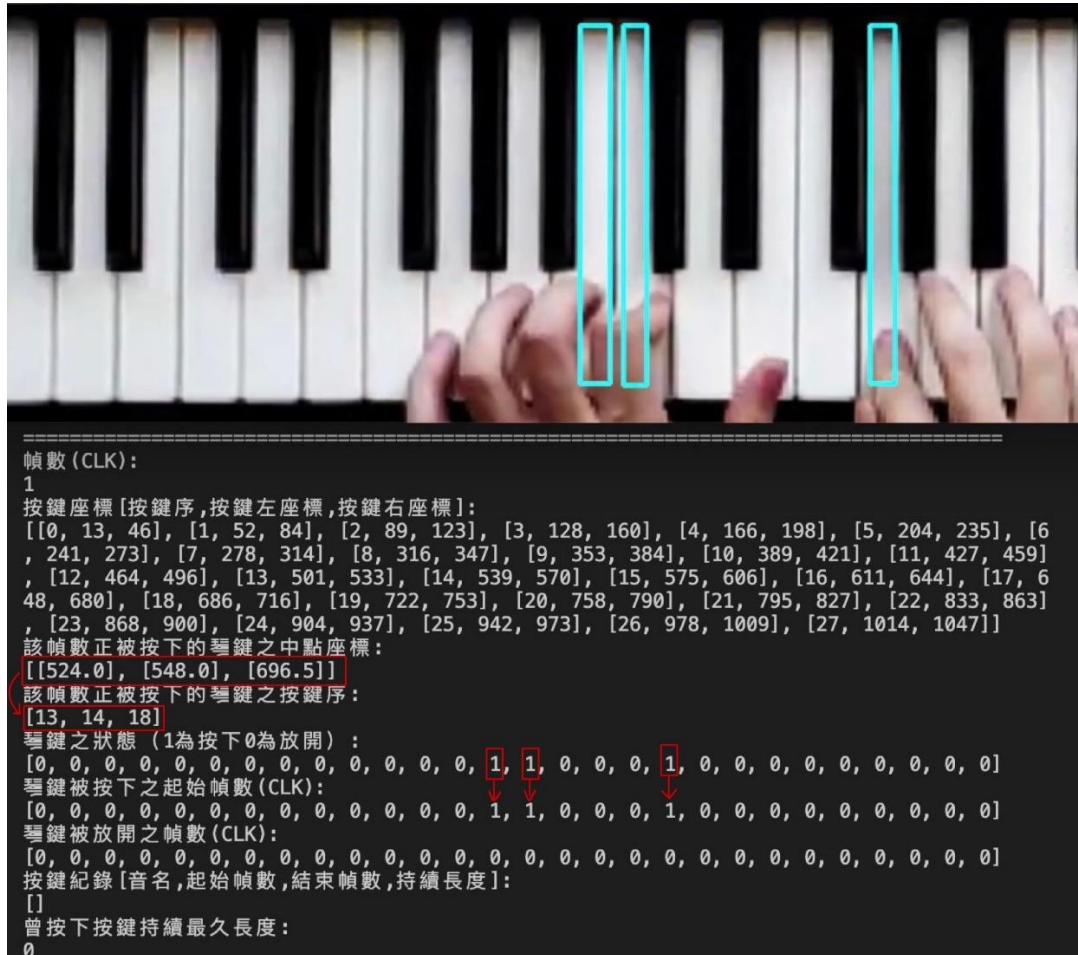
我們利用前景/背景分離，將影片的每一幀去跟純琴鍵的照片去做比較，來辨別變化，有了變化後再進行預處理，預處理完後去產生出邊緣圖，再去進行形態學處理，同樣的去針對矩形，去框出有變化的地方，即可獲得變化的座標，也就是有被按壓的琴鍵位置，並將座標數值記錄下來。




圖(11) 琴鍵按壓偵測流程圖

## 5、 數值處理

有了琴鍵的座標及按壓琴鍵的座標後，我們將這兩組座標進行比對，找出每一幀分別有哪些琴鍵被按下並記錄下來，最後使用 Lilypond 生成樂譜。如圖(12)所示，第一幀紀錄下被按壓之琴鍵座標，並記錄起始幀數。



圖(12)數值處理第一幀

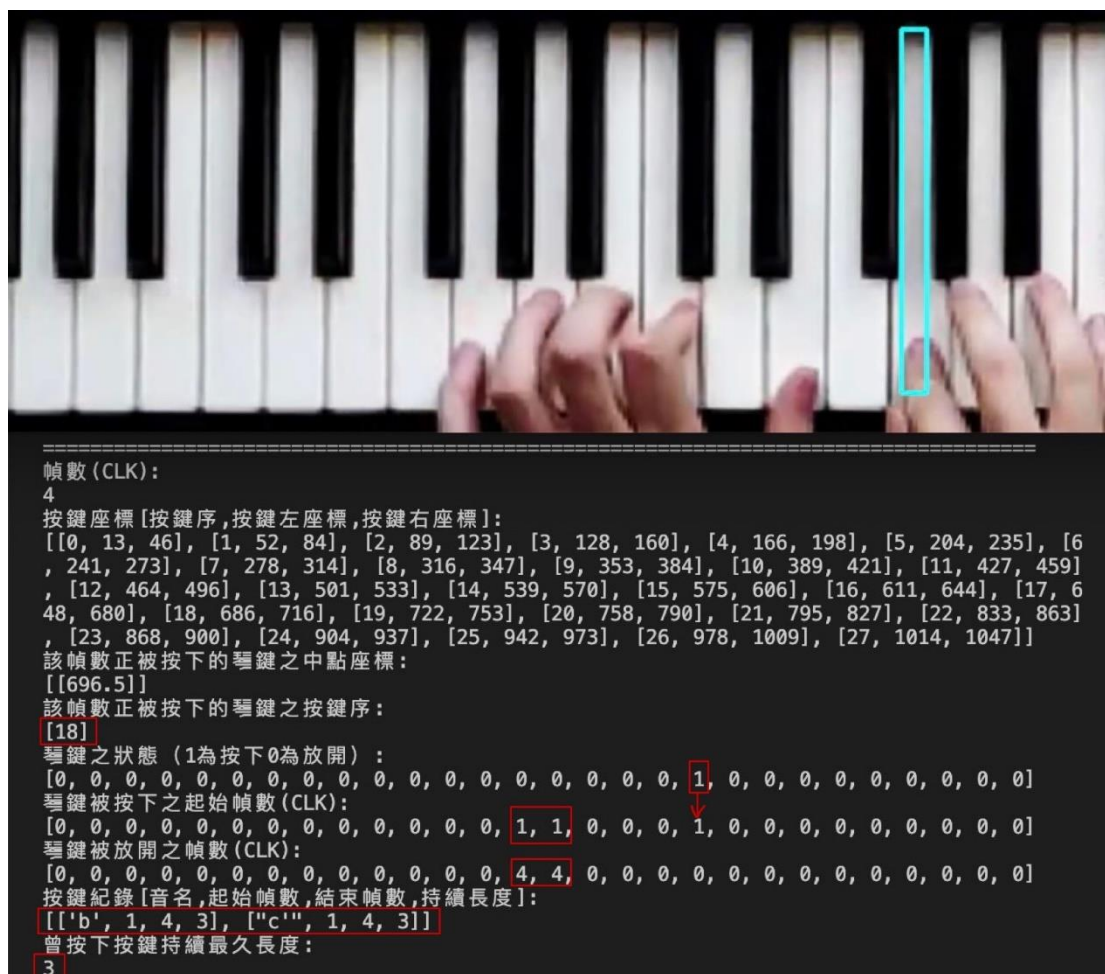


```
=====
幀數 (CLK):
2
按鍵座標 [按鍵序,按鍵左座標,按鍵右座標]:
[[0, 13, 46], [1, 52, 84], [2, 89, 123], [3, 128, 160], [4, 166, 198], [5, 204, 235], [6, 241, 273], [7, 278, 314], [8, 316, 347], [9, 353, 384], [10, 389, 421], [11, 427, 459], [12, 464, 496], [13, 501, 533], [14, 539, 570], [15, 575, 606], [16, 611, 644], [17, 648, 680], [18, 686, 716], [19, 722, 753], [20, 758, 790], [21, 795, 827], [22, 833, 863], [23, 868, 900], [24, 904, 937], [25, 942, 973], [26, 978, 1009], [27, 1014, 1047]]
該幀數正被按下的琴鍵之中點座標:
[[524.0], [547.5], [696.5]]
該幀數正被按下的琴鍵之按鍵序:
[13, 14, 18]
琴鍵之狀態 (1為按下0為放開):
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
琴鍵被按下之起始幀數 (CLK):
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
琴鍵被放開之幀數 (CLK):
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
按鍵紀錄 [音名,起始幀數,結束幀數,持續長度]:
[]
曾按下按鍵持續最久長度:
0
```

13



如圖(14)所示，可以觀察到此時只剩下一個琴鍵仍處於按壓狀態，兩個琴鍵狀態已改變，將此幀數記錄下來，並加以整理為[音名, 起始幀數, 結束幀數, 持續長度]，且紀錄下按鍵持續最久長度。



圖(14) 數值處理第四幀

依照此程序去跑影片，最後將整理出如圖(15)之結果，將其輸入至 Lilypond，即可獲得五線譜。

```
曾按下按鍵持續最久長度：
64
按鍵紀錄整理[音名,幾分音符,起始幀數,同時按鍵數]:
[["b c' g'", 16, '1', 3], ["e'", 16, '5', 1], ["e'", 16, '23', 1], ["c'", 16, '31', 1],
["b'", 16, '46', 1], ['e', 8, '49', 1], ['e', 1, '56', 1], ["e'", 16, '74', 1], ["e'",
16, '76', 1], ["e'", 16, '85', 1], ["e'", 16, '98', 1], ["g'", 8, '111', 1], ["a'", 16,
'124', 1], ["a'", 16, '136', 1], ['e', 8, '166', 1], ['a', 16, '178', 1], ['b', 4, '192',
1], ['b', 16, '204', 1], ["c'", 4, '208', 1], ['b', 16, '217', 1], ["e'", 16, '224',
1], ["e'", 16, '228', 1], ['b', 8, '232', 1], ["c' b", 16, '242', 2], ["c'", 16, '245',
1], ['d', 1, '282', 1], ['a', 16, '306', 1], ["c'", 16, '310', 1], ["b'", 16, '314', 1],
["b'", 16, '316', 1], ['d f', 16, '325', 2], ['d', 4, '330', 1], ["d'", 16, '333', 1],
["a'", 2, '338', 1], ["a'", 2, '366', 1], ['e', 4, '377', 1], ["a'", 2, '391', 1], ['a',
8, '394', 1], ['e', 8, '404', 1], ["c'", 8, '416', 1], ["g'", 16, '417', 1], ["a'", 8,
'421', 1], ["g'", 2, '425', 1], ["b c'", 16, '432', 2], ['e', 16, '442', 1], ["e'", 16,
'467', 1]]
按鍵整理 (lilypond格式):
["<b c' g'>16", "e'16", "e'16", "c'16", "b'16", ',e8', ',e1', "e'16", "e'16", "e'16", "e'
16", "g'8", "a'16", "a'16", 'e8', 'a16', 'b4', 'b16', "c'4", 'b16', "e'16", "e'16", 'b8
', "<c' b>16", "c'16", 'd1', 'a16', "c'16", "b'16", "b'16", '<d f>16', 'd4', "d'16", "a
'2", "a'2", 'e4', "a'2", 'a8', 'e8', "c'8", "g'16", "a'8", "g'2", "<b c'>16", 'e16', "e'
16"]
```

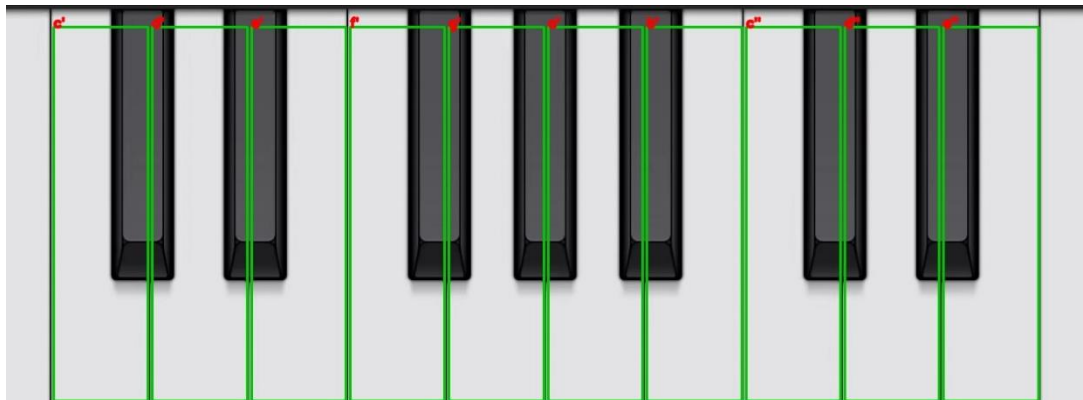
圖(15)整理為 Lilypond 之格式



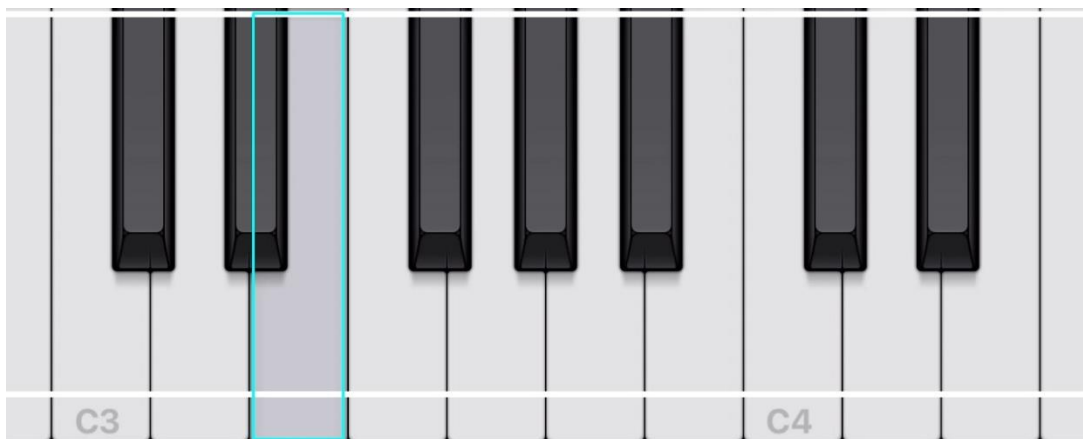
# 陸、 專題成果

## 1、 理想情況模擬

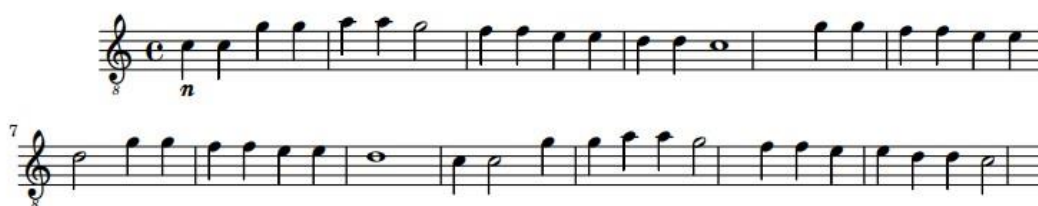
使用線上鋼琴模擬理想情況，也就是無光影影響環境，如圖(16)所示，為琴鍵偵測之結果且完全正確，圖(17)為擷取琴鍵按壓的結果。在理想狀況中，按壓偵測結果完全正確，輸出到 Lilypond 中結果如圖(18)所示，也完全正確。



圖(16) 理想鋼琴琴鍵



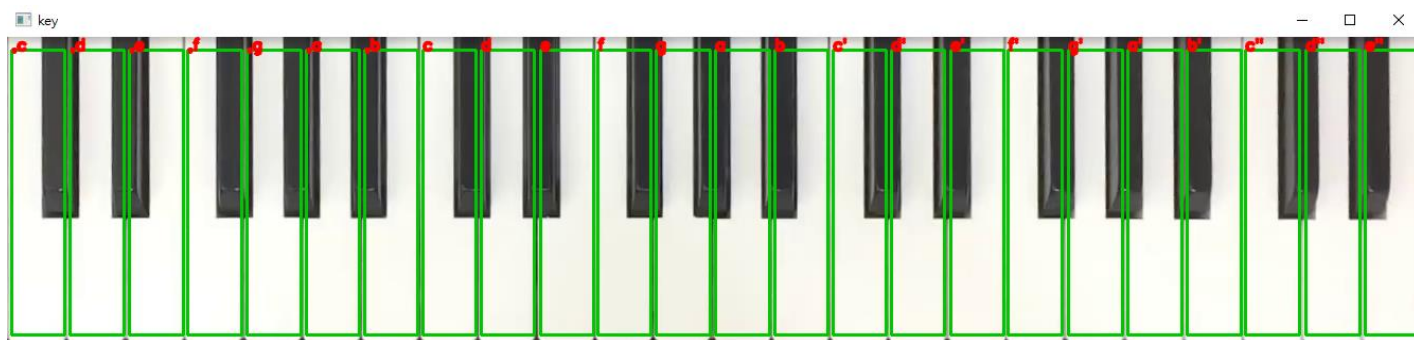
圖(17) 理想鋼琴按壓偵測



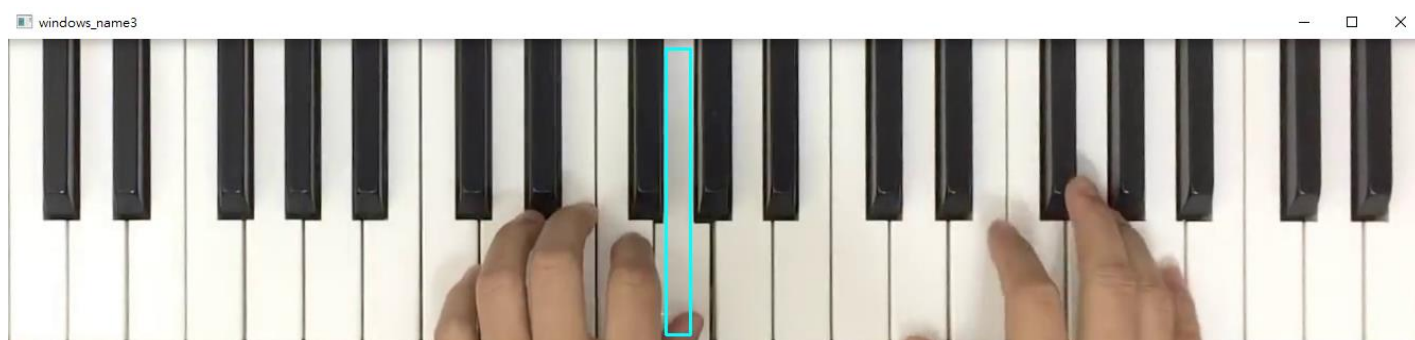
圖(18) 理想情況五線譜

## 2、 實際情況

由於實際的鋼琴彈奏會受到光影變化的影響，這會造成程式對影片的要求極高，琴鍵偵測的部分能完美偵測如圖(19)，但琴鍵按壓偵測就會出現誤判，能夠成功辨認出按壓如圖(20)，但也會誤判沒有按壓的琴鍵，導致五線譜準確率下降。



圖(19) 實際鋼琴琴鍵



圖(20) 實際鋼琴彈奏



圖(21) 實際情況五線譜

# 柒、 結論

## 1、 光線影響

本專題遇到最大的問題是光線部分，在琴鍵按壓偵測中，邊緣檢測是標示亮度變化明顯的點，故影子會干擾到白鍵及黑鍵，邊緣處理成果就會很雜亂，影子的邊緣也會被抓出來，之後就更難去找出矩形的部分。我們做琴鍵按壓偵測的原理是將影片每一幀去跟純琴鍵的照片去比對差異，而純琴鍵的照片我們是去擷取影片最初始的時候，但影片的每一幀的亮度很難與初始的亮度相同，只要影片的亮度有稍微改變，就也會被反應在琴鍵按壓的結果上，進而影響到程式的判斷，所以輸入影片的品質要求很高。

## 2、 未來展望

由於較容易受到光線影響，未來可利用 openpose 手指骨架辨識去輔助按壓偵測來提高鍵盤按壓的判斷準確度(目前只使用 `abs.diff` 觀察按壓前後差異)。Lilypond 的部分則是可以用五線譜變更複雜，更貼近真實的五線譜，目前我們只有右手的高音五線譜，並沒有加入左手低音譜，這導致即使能準確判斷鍵盤按壓位置，並且輸入在五線譜上，只有一個右手五線譜在現實中意味這份鋼琴譜只需要右手即可彈奏完畢，但常常出現兩個同時按壓之音符距離遠遠大於一隻手的大小，所以還是得利用程式打出左手的低音譜號，讓結果更貼近真實、能夠彈奏的五線譜。

在未來，我們希望能夠把程式設計成一個可以直接丟彈奏影片去生成譜的 APP，因為有一些樂譜是需要花錢去購買，所以這將會變成一個很受用的程式。

## 捌、 參考文獻

1. <https://charlottehong.blogspot.com/2017/06/opencv-320-vc14-visual-studio-2017.html>
2. <https://zh.wikipedia.org/wiki/OpenCV>
3. <https://blog.csdn.net/u010925447/article/details/77947398>
4. <https://medium.com/analytics-vidhya/opencv-perspective-transformation-9edffefb2143>
5. <https://zh.wikipedia.org/wiki/%E8%BE%B9%E7%BC%98%E6%A3%80%E6%B5%8>

### B

6. <https://chtseng.wordpress.com/2016/12/05/opencv-edge-detection%E9%82%8A%E7%B7%A3%E5%81%B5%E6%B8%AC/>
7. <https://medium.com/@bob800530/opencv-%E5%AF%A6%E4%BD%9C%E9%82%8A%E7%B7%A3%E5%81%B5%E6%B8%A>  
<C-canny%E6%BC%94%E7%AE%97%E6%B3%95-d6e0b92c0aa3>
8. <https://www.cs.ccu.edu.tw/~damon/photo/proj/2013/601415056/601415056.pptx>
9. <https://makerpro.cc/2018/11/opencv-background-subtractor/>
10. <http://www.kxdw.com/soft/32718.html>