

日期:2024/9/01 至 9/10

討論內容:

在察看科展相關類組後，我們選擇了以電腦科學做為我們的方向，而
這是我在之前構思出來的可用題材:

1. 深度學習圖像辨識：CNN 應用實作
2. 演算法設計與實作：效率優化與應用開發
3. Arduino 物聯網應用：感測器與控制系統開發
4. AI 互動與提示工程：自然語言處理應用開發

日期:2024/9/12 至 9/15

討論內容:

和老師討過後，我們替上次延伸出更明確的應用方向，並提出幾個粗略的可行方案

1. 深度學習圖像辨識：CNN 應用實作

甲、手語辨識

乙、貝茲曲線研究

2. Arduino 物聯網應用：感測器與控制系統開發

甲、空品觀測

3. AI 互動與提示工程：自然語言處理應用開發

甲、提示詞與圖靈測試的關係

日期:2024/9/20 至 9/23

討論內容:

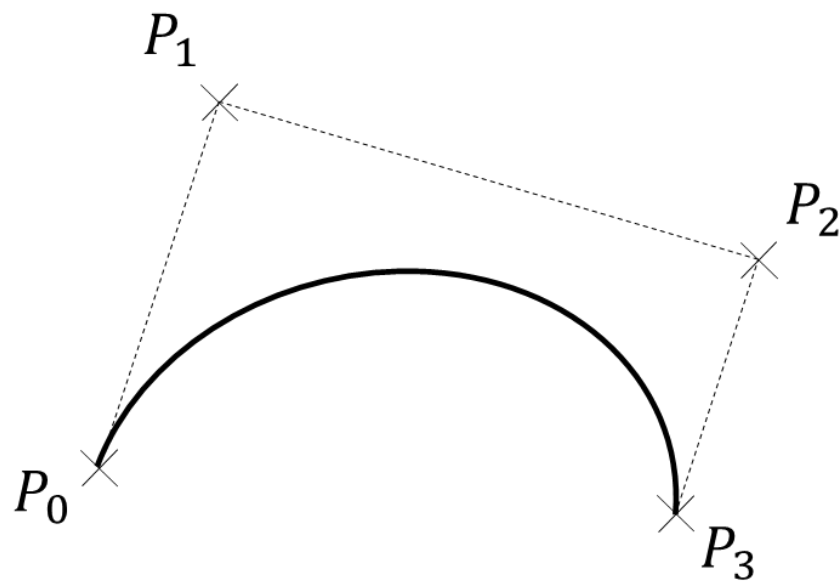
確認了使用 CNN，以下是相關研究方向

1. 曲線擬合:使用向量曲線擬合手繪曲線
2. 手語辨識:使用 CNN 及時間序列 AI 辨識手語影像
3. 圖片修改判斷

日期:2024/9/25 至 9/26

討論內容:

最終選擇「使用向量曲線擬合手繪曲線」作為研究主題，並選擇使用貝茲曲線作為像量化主軸。貝茲曲線是由一組稱為「控制點」的離散點定義，而這些控制點決定了曲線的形狀。而它具有良好的平滑性，並適合應用於擬合手繪曲線。而主流使用為三次貝茲曲線，太高次計算量會過大，對計算造成負擔。



貝茲節點與控制點繪製貝茲曲線（圖片來源：維基百科
網址: https://en.wikipedia.org/wiki/B%C3%A9zier_curve）

日期:2024/10/3 至 2024/10/4

討論內容:

閱讀了幾篇論文

文獻：《Defining a curve as a Bézier curve》

- 主要內容摘要：作者提出了一種判斷任意曲線是否為貝茲曲線的新方法，核心在於創建一個特定的矩陣。該論文證明了此矩陣的反矩陣可用於精確計算貝茲曲線的控制點，強調其精確性而非近似性，適用於計算機輔助幾何設計等多個領域。
- 優點：提供精確的貝茲曲線控制點求解方法。
- 缺點：依賴矩陣反推，可能需要較高的計算資源，且在高階多項式處理上較為複雜。對於已損壞的圖像，難以反推其數學函數，限制了其應用。
- 個人初步想法：該方法在理論上提供了精確的解決方案，但在實際應用中可能需要考慮計算成本和數據的完整性。

文獻：《Bézier Curve Fitting》

- 主要內容摘要：本文研究了使用總體最小二乘法（**TLS**）擬合貝茲曲線的方法，旨在同時最小化水平和垂直方向的殘差。作者深入探討了伯恩施坦多項式與貝茲曲線的數學基礎，並提出了一種基於高斯-牛頓法的優化演算法來尋找最佳控制點和節點。
- 優點：提供更精確的曲線擬合方法，適用於數據分佈不均的情境，並具有較高的幾何穩定性。
- 缺點：計算複雜度較高，尤其是在處理高維數據或實時應用時。論文側重於數學方法，工程應用討論較少，實作可能存在困難。
- 個人初步想法：**TLS** 方法在擬合精度上有所提升，但其計算成本是實際應用中需要仔細評估的因素。

日期: 2024/10/5 至 2024/10/6

討論內容:

閱讀了幾篇論文

- 文獻：《Curve Fitting Using Generalized Fractional Bézier Curve》
- 主要內容摘要：該研究探討了利用廣義分數貝茲曲線進行曲線擬合。作者指出傳統貝茲曲線在形狀調整上的局限性，並引入了帶有形狀參數和分數參數的廣義分數貝茲曲線，允許在不改變控制點的情況下調整曲線形狀，提高了擬合的靈活性和效率。此外，該方法利用分數連續性增強了曲線連接的靈活性。
- 優點：提高了曲線擬合的靈活性和效率，允許在不改變控制點的情況下調整曲線形狀。
- 缺點：主要集中在數學模型構建和理論分析，實際應用中的性能和效果需要進一步驗證。
- 個人初步想法：廣義分數貝茲曲線在理論上提供了更強大的曲線控制能力，但其實際應用效果需要實驗驗證。

文獻：《基於擴散曲線之點陣圖自動向量化》

- 主要內容摘要：作者提出了一種利用擴散曲線技術將點陣圖自動轉換為向量圖的方法。該方法從點陣圖中提取輪廓、顏色和模糊像素點，並轉換為擴散曲線的幾何元素，生成易於編輯和動畫製作的向量圖。
- 優點：生成的向量圖能較好地接近原始圖像，且易於編輯和動畫製作。
- 缺點：在處理高細節圖像時可能面臨精度下降的問題，需要進一步優化。
- 個人初步想法：擴散曲線在向量化方面具有潛力，但對於細節豐富的圖像可能需要額外的處理步驟。

日期: 2024/10/7

討論內容:

閱讀了幾篇論文

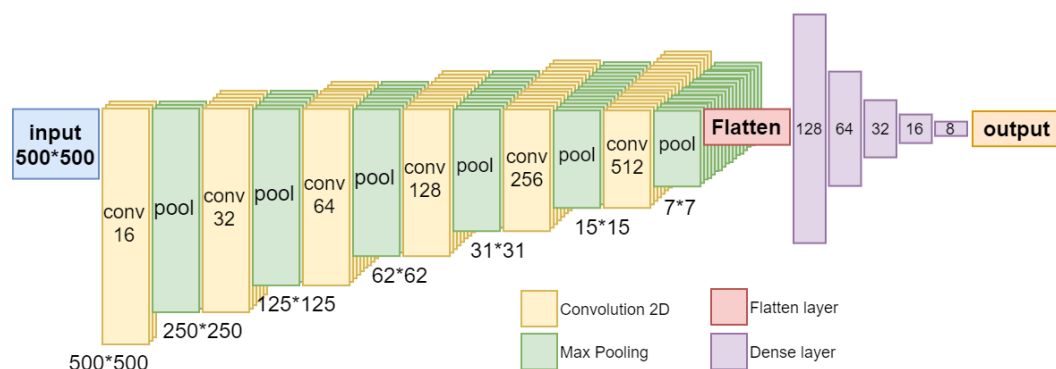
文獻：《Systematic Comparison of Vectorization Methods in Classification Context》

- 主要內容摘要：作者對多種向量化方法在分類任務中的性能進行了系統性比較，評估了不同技術在處理各類數據集時的準確性和效率，旨在為選擇合適的向量化方法提供依據。研究結果強調了根據具體應用場景選擇向量化技術的重要性。
- 優點：為選擇合適的向量化方法提供了實證依據。
- 缺點：可能受限於所選數據集的多樣性，未來需要更多元的数据集驗證結果的普遍性。
- 個人初步想法：不同的向量化方法適用於不同的場景，需要根據具體任務進行選擇。

日期:2024/10/9 至 10/30

討論內容:

本研究參考了經典的 VGG-16 [13] 模型架構，並根據手繪線條切割點預測的特定任務進行了修改和精簡。最終得到的模型命名為 VGG-16(ours)

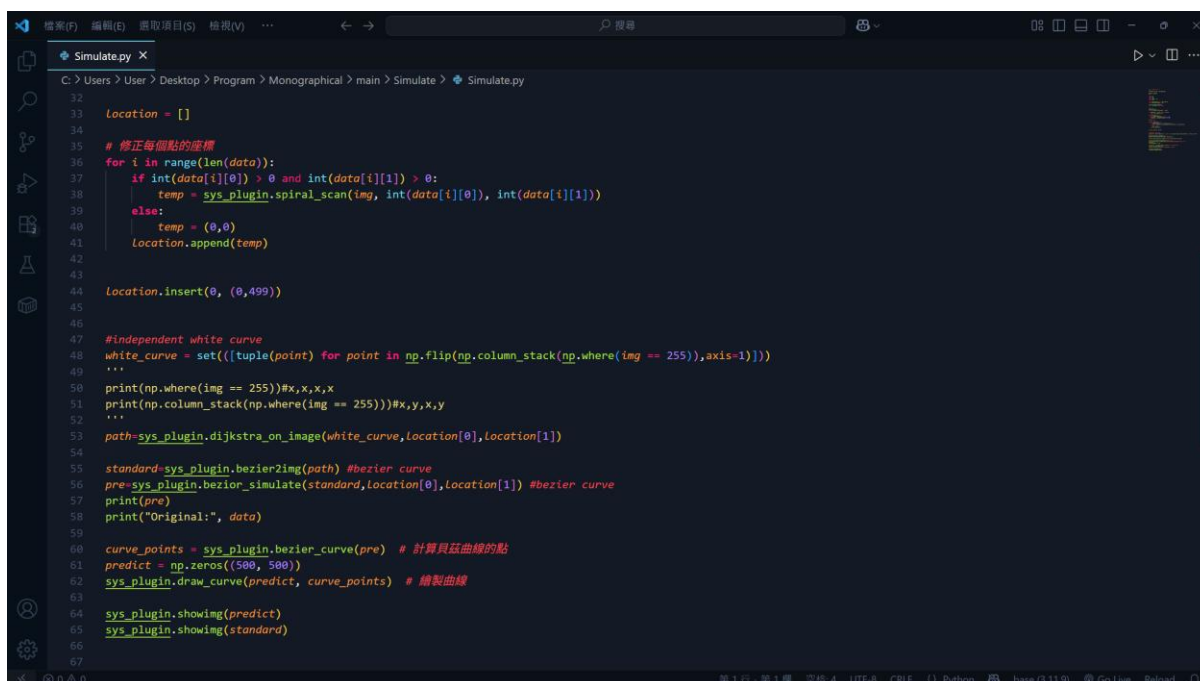


模型說明：

VGG-16(ours) 模型以 [輸入尺寸] 的手繪曲線圖像作為輸入。模型主要由以下幾種層組成：

- 卷積層 (Convolution 2D)：圖中以淺黃色方塊表示，用於提取輸入圖像的局部特徵。每個卷積層後通常會指定卷積核的數量（例如，16, 32, 64, 128, 256, 512）。
- 最大池化層 (Max Pooling)：圖中以淺綠色方塊表示，用於降低特徵圖的空間維度，並提高模型的平移不變性。
- 展平層 (Flatten layer)：圖中以淺紅色方塊表示，將多維的特徵圖展平成一維向量，以便輸入到全連接層。

- 全連接層 (Dense layer)：圖中以淺紫色長條表示，用於進行最終的預測。模型中包含多個全連接層，其輸出維度分別為 [請在此填寫]，最終輸出層的維度為 8。
- 輸出層 (Output)：模型最終的輸出是一個 8 維的向量。由於本模型的目的是預測線段的切割點，這 8 個輸出值預計代表了 [請在此詳細說明這 8 個輸出值的具體含義，例如：4 個切割點的 x 和 y 座標]。



```

32
33 Location = []
34
35 # 修正每個點的座標
36 for i in range(len(data)):
37     if int(data[i][0]) > 0 and int(data[i][1]) > 0:
38         temp = sys_plugin.spiral_scan(img, int(data[i][0]), int(data[i][1]))
39     else:
40         temp = (0,0)
41     Location.append(temp)
42
43
44 Location.insert(0, (0,499))
45
46
47 #independent white curve
48 white_curve = set([(tuple(point) for point in np.flip(np.column_stack(np.where(img == 255)),axis=1))])
49 ...
50 print(np.where(img == 255))#x,x,x,x
51 print(np.column_stack(np.where(img == 255)))#x,y,x,y
52 ...
53 path=sys_plugin.dijkstra_on_image(white_curve,Location[0],Location[1])
54
55 standard=sys_plugin.bezier2img(path) #bezier curve
56 pre=sys_plugin.bezier_simulate(standard,Location[0],Location[1]) #bezier curve
57 print(pre)
58 print("Original:", data)
59
60 curve_points = sys_plugin.bezier_curve(pre) # 計算貝茲曲線的點
61 predict = np.zeros((500, 500))
62 sys_plugin.draw_curve(predict, curve_points) # 繪製曲線
63
64 sys_plugin.showing(predict)
65 sys_plugin.showing(standard)
66
67

```

程式部分截圖(作者自行拍攝)

日期:2024/11/1 至 11/6

討論內容:

簡單研究了 A*和 Dijkstra 演算法，並做了簡單對比表格

特性	Dijkstra 演算法	A* 演算法
核心理念	尋找從起點到所有其他節點的最短路徑，基於已知的實際成本。	在 Dijkstra 的基礎上，引入啟發式函數引導搜尋，預估剩餘成本。
評估函數	$f(n) = g(n)$ (僅考慮從起點到當前節點的實際成本)	$f(n) = g(n) + h(n)$ (考慮實際成本和到目標的估計成本)
搜尋策略	$f(n) = g(n) + h(n)$ (考慮實際成本和到目標的估計成本)	優先擴展總估計成本 $f(n)$ 最低的節點。
啟發式函數	不使用啟發式函數。	使用啟發式函數 $h(n)$ 估計從當前節點到目標節點的成本。
最優性保證	在邊權重非負的情況下，保證找到從起點到所有節點的最短路徑。	如果啟發式函數是可接受的（不 Overestimate ），則保證找到從起點到目標節點的最優路徑。
效率	在沒有目標節點的情況下，需要探索較多節點。	通過好的啟發式函數，可以更快速地找到目標，效率通常更高。
應用場景	尋找單源最短路徑，例如地圖導航中計算到所有地點的最短距離。	尋找單一起點到單一目標的最短路徑，例如遊戲 AI 路徑規劃、更高效的地圖導航。
在線段切割中潛力	可用於基於成本累積尋找切割點，但可能不夠目標導向。	可通過設計啟發式函數，更有效地尋找符合特定切割目標的點。

日期:2024/11/7 至 11/15

討論內容:

初步研究嘗試採用暴力演算法，旨在窮舉所有可能的貝茲曲線控制點組合，並與原始手繪圖像進行比較以尋找最佳擬合。然而，針對本次實驗設定的 500×500 像素畫布，以及一次輸入包含 5 段、每段需要 2 個控制點的三次貝茲曲線的複雜性，計算量分析顯示，可能的控制點組合數量級高達 $3.125 * 10^{11}$ ，呈現指數級增長，遠超出現有計算資源在合理時間內可處理的範圍。因此，實驗結論明確指出，暴力演算法在本次研究情境下是不可行的解決方案，必須轉向探索更高效的優化演算法，例如遺傳演算法或其他基於梯度或啟發式的方法，以應對高維度的控制點搜索空間。

日期:2024/11/16 至 11/20

討論內容:

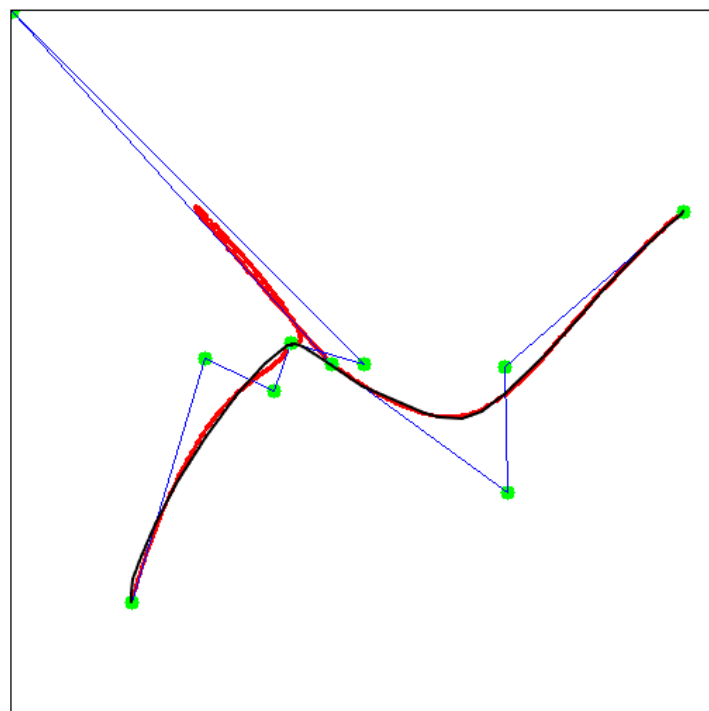
本研究曾嘗試利用 Python 的 OpenCV 套件提供的陣列差異比對技術作為貝茲曲線擬合的損失函數。該方法通過計算手繪線段與擬合線段重疊後不重疊的像素數量，並將其歸一化為損失值。然而，實驗分析發現此損失函數對像素級的微小移動極其敏感，即使僅一個像素的偏差也可能導致損失值劇烈變化，無法有效反映曲線間的真實相似度，並可能誤導優化過程。因此，基於 OpenCV 陣列差異比對的損失函數被判定不適合作為本研究的損失函數使用。

日期:2024/11/21 至 11/29

實作內容:

此方法同樣使用 Python 之 Opencv 套件提供之方法，使用 `contour` 輪廓計算方法為主體，將手繪曲線切割出的線段 A 及擬合出的線段 B 進行重疊，並進行輪廓計算，得到的輪廓面積經標準化過後代表兩線段 AB 的損失，在這裡使用輪廓面積除以整張圖的總像素作為損失，此方法計算效果準確，但當兩條線完全沒交集時會導致無封閉空間，輪廓面積過小而視為異常準確而輸出，且可能會出現面積極小，但線條極長的情況，故不適合使用。

畫布繪圖



日期:2024/11/30 至 2025/1/20

討論內容:

最後我們找到了遺傳演算法，所以在本研究採用遺傳演算法 (Genetic Algorithm, GA) 作為貝茲曲線控制點的優化方法。GA 模擬自然演化過程，通過選擇、交配和突變等機制，從隨機種群中逐步尋找最優解，以期使擬合曲線最大程度地接近手繪曲線。相較於易受限於局部最佳解的傳統數學模型，GA 能夠通過全局搜索逼近最優解。具體的演化流程包括：

- (1) 初始化種群，隨機生成控制點，並固定起點與終點以聚焦搜索，使用 Hausdorff 距離作為初始評估損失
- (2) 選擇操作，基於 Hausdorff 距離選擇損失較低的個體進入下一代
- (3) 交配操作，採用單點交叉交換中間控制點 P_2 和 P_3 的部分信息，確保後代繼承優勢並增加變異
- (4) 突變操作，以一定機率隨機擾動 P_2 和 P_3 的位置，避免陷入局部最佳解
- (5) 迭代演化，重複上述步驟直至滿足收斂條件。遺傳演算法的最大優勢在於其跳脫局部最佳解的能力，且通過調整種群大小和演化次數可兼顧計算速度與準確度。

使用了上面這套演算法，展現出了相當好的，擬合結果超乎預期，故把此演算法作為擬合主軸。

日期:2025/1/21 至 1/30

討論內容:

本研究採用 **Hausdorff** 距離作為衡量手繪曲線線段 **A** 與擬合線段 **B** 座標相似度的度量。**Hausdorff** 距離計算兩條曲線之間的最長距離，並將其標準化後作為損失函數。經過多次測試與嘗試，實驗結果表明 **Hausdorff** 距離在作為貝茲曲線擬合的損失函數時，能夠同時兼顧計算速度與擬合準確度。因此，後續實驗將主要以 **Hausdorff** 距離作為主要的評估指標。

日期: 2025/1/31 至 2/10

討論內容:

經過幾天的查詢，我們最終採用 **Python** 的 **Flask** 套件搭建後端運算伺服器，並利用 **HTML** 建立網站架構以供使用者操作。在使用者端，整合 **Canva** 套件追蹤滑鼠動作，即時獲取使用者的滑鼠位置及狀態，實現網頁畫布的繪圖效果。使用者繪製的路徑數據將被儲存並傳輸至伺服器端進行後續的擬合運算。



日期:2025/2/11 至 2/22

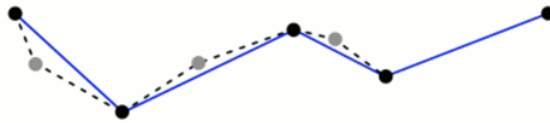
實作內容:

報告製作與學校初期發表

日期:2025/2/23 至 2/27

實作內容:

為了有效加速運算並精確提取曲線特徵，我們找了幾個演算法，最後我們在本研究中整合了 **RDP** 演算法。**RDP** 演算法通過設定容忍誤差值，遞迴地簡化線段，移除對曲線形狀影響不顯著的點，從而在保持主要特徵的同時，顯著減少路徑點數，為後續有序線段的判斷提供了關鍵資訊。



利用 **RDP** 演算法簡化路徑（圖片來源：維基百科 網址:

https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm)

日期:2025/2/28 至 3/15

實作內容:

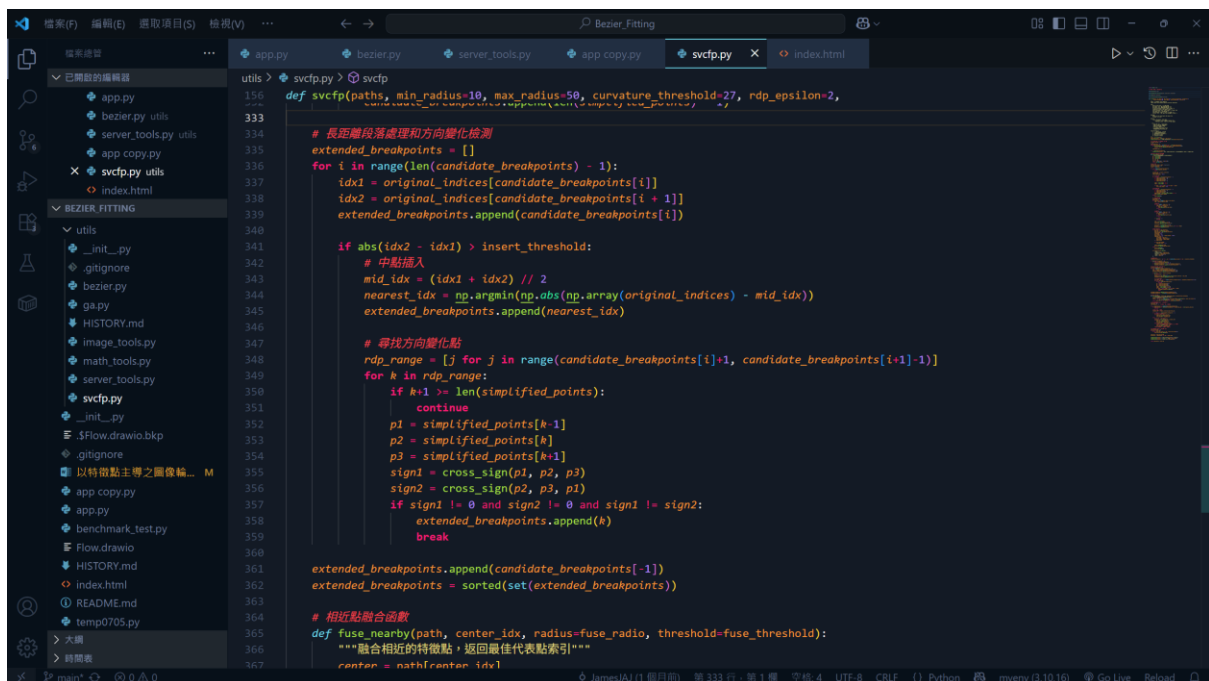
我們增進了我們的演算法，針對接收到的 **RDP** 陣列，採用一種基於多尺度向量預測的簡化點分析方法。該方法通過分析簡化點在不同尺度下的向量變化，結合統計分析和閾值判斷，以更準確地識別簡化點的顯著性。具體做法是對每個簡化點，在其鄰域內（由小到大擴展）進行向量預測，記錄每個尺度下的最大值，其中向量預測通過計算簡化點左右兩側線段向量的平均值評估局部曲線走向。這種多尺度策略有助於避免如 ω 形曲線等特殊曲線的干擾。完成向量計算後，對每個簡化點記錄的最大值進行統計分析（計算平均值和標準差），並結合預定義的權重進行評分，量化其顯著程度。

日期:2025/2/28 至 3/15

實作內容:

我們嘗試針對 RDP 陣列中的每個點進行角度變化分析，以識別顯著轉折。

方法是計算每個點與其前後相鄰點單位向量的外積並轉換為角度，若角度變化量超過閾值，則標記為轉折點並增加其評分權重；反之，若偵測到直線狀態則降低權重。在完成角度變化分析和權重調整後，提取評分超過預設閾值的點位作為特徵點，用於後續計算。

A screenshot of a code editor window with a dark theme. The editor shows a Python file named 'svcf.py' with a function 'def svcfp' that processes RDP points. The code includes comments in Chinese and Python code for calculating angles, identifying breakpoints, and fusing nearby points. The left sidebar shows a file explorer with various project files. The bottom status bar indicates the file is 'svcf.py' and the editor is using Python 3.10.16.

```
def svcfp(paths, min_radius=10, max_radius=50, curvature_threshold=27, rdp_epsilon=2,
          min_angle=0, max_angle=180, simplify=True, simplify_ratio=0.5):
    """
    最近鄰點處理和方向變化檢測
    """
    extended_breakpoints = []
    for i in range(len(candidate_breakpoints) - 1):
        idx1 = original_indices[candidate_breakpoints[i]]
        idx2 = original_indices[candidate_breakpoints[i + 1]]
        extended_breakpoints.append(candidate_breakpoints[i])

        if abs(idx2 - idx1) > insert_threshold:
            # 中點插入
            mid_idx = (idx1 + idx2) // 2
            nearest_idx = np.argmin(np.abs(np.array(original_indices) - mid_idx))
            extended_breakpoints.append(nearest_idx)

        # 尋找方向變化點
        rdp_range = [j for j in range(candidate_breakpoints[i + 1], candidate_breakpoints[i + 1] - 1)]
        for k in rdp_range:
            if k + 1 >= len(simplified_points):
                continue
            p1 = simplified_points[k]
            p2 = simplified_points[k + 1]
            sign1 = cross_sign(p1, p2, p2)
            sign2 = cross_sign(p2, p3, p1)
            if sign1 != 0 and sign2 != 0 and sign1 != sign2:
                extended_breakpoints.append(k)
                break

    extended_breakpoints.append(candidate_breakpoints[-1])
    extended_breakpoints = sorted(set(extended_breakpoints))

    # 相近點融合函數
    def fuse_nearby(path, center_idx, radius=fuse_radius, threshold=fuse_threshold):
        """
        融合相近的特徵點，返回最佳代表點索引
        """
        center = path[center_idx]
```

程式部分截圖(作者自行拍攝)

日期:2025/3/31 至 4/11

實作內容:

報告製作

日期:2025/4/12 至 4/20

實作內容:

整理資料與教授重點。

1. 時間太久 導致其使用不夠順暢
2. 泛用性太差 精度太低
3. 無法使用圖片進行擬合

針對以上問題 我們訂定了以下目標做修改

1. 尋找更好的演算法進行替代
2. 修正 **SVCFP** 並增加限制
3. 增加前處理使其可以進行擬合

日期:2025/4/20 至 5/1

實作內容:

我們在嘗試著改進 **GA**，但經過我們多天的研究，我們發現了他在先天上的劣勢，因是從所有可能中暴力逼近最近的可能，所以耗時必定較慢，而我們轉而研究三種方法:

1. 高斯牛頓插值法

優點為精度高，如果初始猜測接近真實解，高斯牛頓法可以快速收斂到一個高精度的解。但對起始猜測極為要求，如果期使猜測較為不佳，可能會陷入局部最優解甚至發散，並且計算量極大，違背了我們想要加速的初衷，故放棄此方法。

2. B-Spline

優點為局部控制性佳，移動一個控制點只會影響曲線的一小部分，有利於精確調整曲線形狀，避免蝴蝶效應。而且其連續性也好，可以很容易地實現指定階次的連續性，使得曲線更加平滑。但在定義 **B-spline** 時，曲線的起始和結束方式通常需要通過特別設定節點向量來實現。如果節點設置不當，曲線可能不會從預期的點開始或結束，與我們想要固定首尾點的想法相互矛盾，故放棄此方法。

3. LSM 最小平方法

優點視相對於插值方法，**LSM** 在一定程度上能夠更好地處理數據中的噪聲或異常值。相較於高斯牛頓法，**LSM** 對初始值的依賴性較小，尤其是在線性最小平方問題中。但如果曲線的自由度（控制點數量）不足，可能會導致曲線過於平滑，無法捕捉數據的細節，但這個方法可以被我們的 **SVCFP** 解決，故最後使用 **LSM** 作為我們的擬合方法。

但在實作中，我們遇到了當點位數過少時，會發生無法擬合的問題，我們最後是使用融合相近點的方式來避免此種問題。

日期:2025/5/2 至 5/15

實作內容:

在經過我們幾天的設計與嘗試過後，我們將其擬合的過程與推導整理如下：

首先在固定首尾點 P_0 、 P_3 的條件下，推估中間控制點 P_1 、 P_2 ，利用最小平方方法對三次貝茲曲線進行擬合，三次貝茲曲線的一般數學表達形式為

$B(t) = (1-t)^3P_0 + 3(1-t)^2tP_1 + 3(1-t)t^2P_2 + t^3P_3, t \in [0,1]$ 已知一組

離散點 $\{Q_i\}_{i=1}^n$ ，擬使一條貝茲曲線 $B(t_i)$ 通過這些點。首先先使用弦長參

數化 (Chord-length Parameterization) 將每個點對應到一個參數值 t_i ，以

保留曲線幾何的分布性，接著將貝茲曲線的表達重新改寫為 $B(t_i) =$

$A_1(t_i)P_1 + A_2(t_i)P_2 + R(t_i)$ ，其中 $A_1(t_i) = 3(1-t_i)^2t_i$ 為 P_1 的基底項、

$A_2(t_i) = 3(1-t_i)t_i^2$ 為 P_2 的基底項、 $R(t_i) = (1-t_i)^3P_0 + t_i^3P_3$ 為已知端

點貢獻。最後將下式的誤差最小化，整理如下：

$$\min_{P_1, P_2} \sum_{i=1}^n \|Q_i - B(t_i)\|^2 = \sum_{i=1}^n \|Q_i - A_1(t_i)P_1 - A_2(t_i)P_2 - R(t_i)\|^2$$

可將上式轉為矩陣形式如下：

$$b = Q - R \cdot A$$

則有

$$A \cdot \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \approx b, \text{ 其中 } A = \begin{bmatrix} A_1(t_1) & A_2(t_1) \\ \vdots & \vdots \\ A_1(t_n) & A_2(t_n) \end{bmatrix}$$

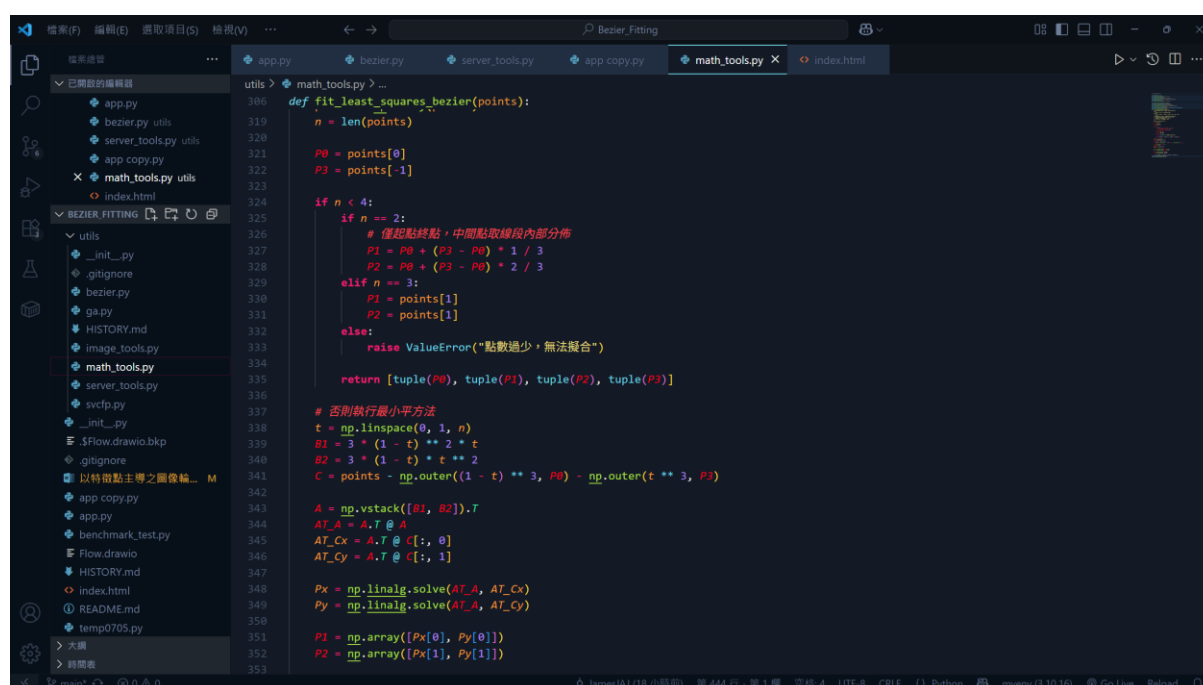
最後分別對 x 與 y ，使用最小平方法，可分別得到最佳解為：

$$P_x = (A^T A)^{-1} A^T b_x, P_y = (A^T A)^{-1} A^T b_y$$

即可得到

$$P_1 = (P_{x1}, P_{y1}), P_2 = (P_{x2}, P_{y2})$$

最後就完成對控制點 P_1, P_2 達到擬合的動作。



```
def fit_least_squares_bezier(points):  
    n = len(points)  
    P0 = points[0]  
    P3 = points[-1]  
  
    if n < 4:  
        if n == 2:  
            # 僅起點終點，中間點取線段內部分佈  
            P1 = P0 + (P3 - P0) * 1 / 3  
            P2 = P0 + (P3 - P0) * 2 / 3  
        elif n == 3:  
            P1 = points[1]  
            P2 = points[1]  
        else:  
            raise ValueError("點數過少，無法擬合")  
  
    return [tuple(P0), tuple(P1), tuple(P2), tuple(P3)]  
  
# 否則執行最小平方法  
t = np.linspace(0, 1, n)  
B1 = 3 * (1 - t) ** 2 * t  
B2 = 3 * (1 - t) * t ** 2  
C = points - np.outer((1 - t) ** 3, P0) - np.outer(t ** 3, P3)  
  
A = np.vstack([B1, B2]).T  
AT_A = A.T @ A  
AT_Cx = A.T @ C[:, 0]  
AT_Cy = A.T @ C[:, 1]  
  
Px = np.linalg.solve(AT_A, AT_Cx)  
Py = np.linalg.solve(AT_A, AT_Cy)  
  
P1 = np.array([Px[0], Py[0]])  
P2 = np.array([Px[1], Py[1]])
```

程式部分截圖(作者自行拍攝)

日期:2025/5/16 至 5/28

實作內容

為增加其實用性，我們預計增加圖片擬合功能，首先我們選擇使用輪廓作為擬合的依據，避免遇到填充圖形中需要骨骼化等問題，但在實際編程過後發現兩個技術難點:抓取輪廓和如何替輪廓線排序，而在研究相關資料過後，我發現 **Opencv** 的 **findcontour** 套件可以完美的解決我的問題，**findcontour** 函數適用於找尋一圖片的套件，並依序取出輪廓點位數，而為了更精確的提取圖片重點，我們使用高斯模糊與二值化去噪，最後完成提取輪廓作業。

日期:2025/5/29 至 6/4

實作內容

在查詢相關文獻過後，我們發現市面上沒有比較適合我們的指標來比較兩個手繪圖像的相似度，所以我們最後選擇使用自行設計的指標: **BMND** (雙向平均最近鄰距離相似度: **Bidirectional Mean Nearest Distance Similarity**) 作為主要的評估指標，目的是為了更客觀地衡量擬合曲線與原始輪廓之間的幾何相似程度。以下是 **BMND** 公式，令兩個點集為 $A = \{a_1, a_2, \dots, a_n\}$ ， $B = \{b_1, b_2, \dots, b_m\}$ ，則其相似度定義為：

$$\text{Similarity}(A, B) = \frac{1}{1 + \frac{1}{2} \left(\frac{1}{m} \sum_{j=1}^m \min_i \|b_j - a_i\| + \frac{1}{n} \sum_{i=1}^n \min_j \|a_i - b_j\| \right)}$$

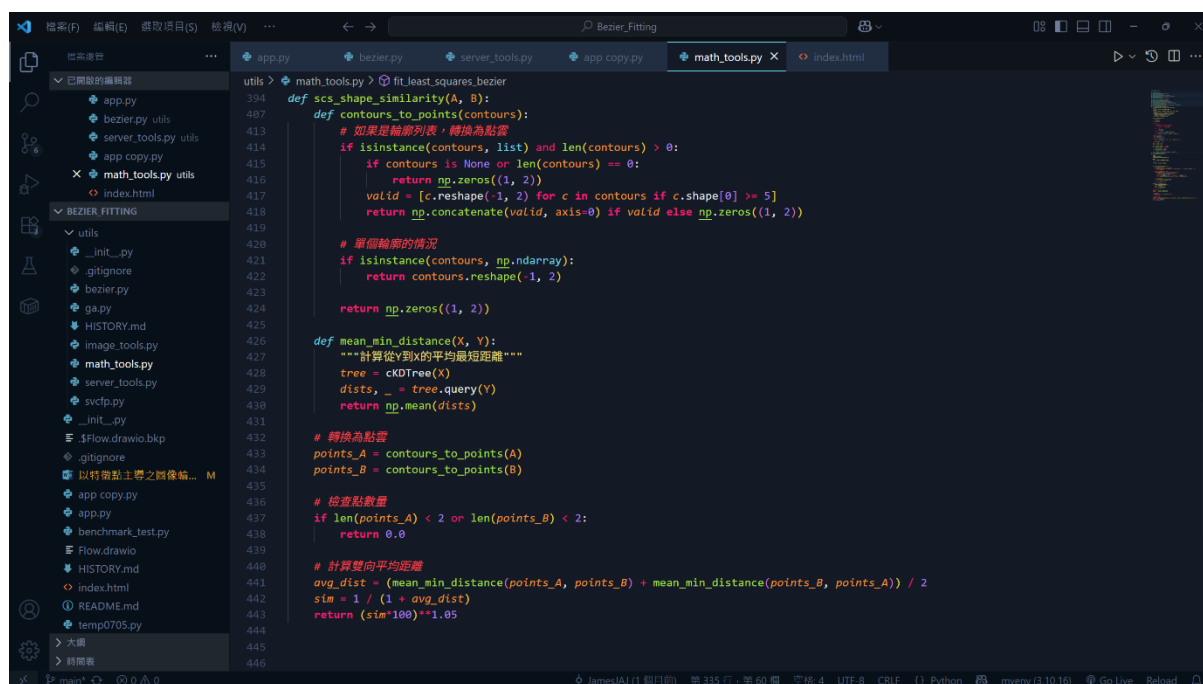
而對於其他傳統常用的 **RMSE** (均方根誤差) 或僅以像素差異為基礎的方法，雙向平均最近鄰距離相似度能從點集分布的幾何結構出發，雙向考量擬合與原圖之間的最近距離，並轉換為標準化相似度指數 (0 ~ 1)，因此更適合用於本研究中向量曲線與手繪線條之間的形狀重構品質評估

指標名稱	全名	數值範圍	優點	缺點	適用情境
MSE	Mean Squared Error 平均平方誤差	≥ 0 (越小越好)	計算簡單、實作容易	對人眼不敏感，無法反映結構性差異	初步誤差分析，需搭配其他指標

RMSE	Root Mean Squared Error 均方根誤差	≥ 0 (越 小越好)	與 MSE 相 似，具單位意 義 (與像素範 圍同)	同樣無法反 映結構，易 受極端值影 響	圖像壓 縮、數值 誤差統計
PSNR	Peak Signal-to-Noise Ratio 峰值信噪比	通常介於 20~50 dB (越大越 好)	常用於圖像壓 縮評估，對數 表現使變化更 平滑	仍無法捕捉 視覺結構、 邊緣資訊	視訊壓 縮、影像 傳輸品質 評估
SSIM	Structural Similarity Index 結構相似性 指標	-1 ~ 1 (通常介 於 0~1， 越接近 1 越好)	反映人眼感 知、可評估亮 度、對比與結 構	計算較複 雜，對部分 圖像類型表 現有限	視覺相似 度分析、 圖像品質 主觀評估
BMND	Bidirectional Mean Nearest Distance Similarity 雙向平均最 近鄰距離相 似度	0 ~ 1 (越接近 1 越好)	對輪廓、點雲 結構敏感，計 算對稱、具幾 何意義，可量 化形狀重構品 質	不考慮亮度 或對比，僅 針對幾何形 狀	輪廓擬合 品質評 估、向量 圖形相似 度量化、 手繪線條 重構分析

BMND (Bidirectional Mean Nearest Distance Similarity) 是一種專門用來評估兩組幾何點集在空間分布與形狀結構上是否相似的指標。它透過計算雙向的平均最近鄰距離 (**A** 到 **B**，**B** 到 **A**)，並進一步轉換為 0 ~ 1 之間的標準化相似度值，藉此量化兩者在幾何外型上的重合程

度。這種方法能有效捕捉輪廓走向與局部結構變化，比傳統僅考慮像素誤差的方法更具形狀敏感性與對稱性。



```
utils > math_tools.py > fit_least_squares_bezier
394 def ses_shape_similarity(A, B):
395     def contours_to_points(contours):
396         # 如果是輪廓列表，轉換為點雲
397         if isinstance(contours, list) and len(contours) > 0:
398             if contours is None or len(contours) == 0:
399                 return np.zeros((1, 2))
400             valid = [c.reshape(-1, 2) for c in contours if c.shape[0] >= 5]
401             return np.concatenate(valid, axis=0) if valid else np.zeros((1, 2))
402
403         # 單個輪廓的情況
404         if isinstance(contours, np.ndarray):
405             return contours.reshape(-1, 2)
406
407         return np.zeros((1, 2))
408
409     def mean_min_distance(X, Y):
410         """計算從Y到X的平均最短距離"""
411         tree = cKDTree(X)
412         dists, _ = tree.query(Y)
413         return np.mean(dists)
414
415     # 轉換為點雲
416     points_A = contours_to_points(A)
417     points_B = contours_to_points(B)
418
419     # 檢查點數量
420     if len(points_A) < 2 or len(points_B) < 2:
421         return 0.0
422
423     # 計算雙向平均距離
424     avg_dist = (mean_min_distance(points_A, points_B) + mean_min_distance(points_B, points_A)) / 2
425     sim = 1 / (1 + avg_dist)
426     return (sim*100)**1.05
```

程式部分截圖(作者自行拍攝)

日期:2025/6/5 至 6/20

實作內容

為驗證 BMND 精確度，我設計了以下幾個實驗驗證 BMND 準確性：

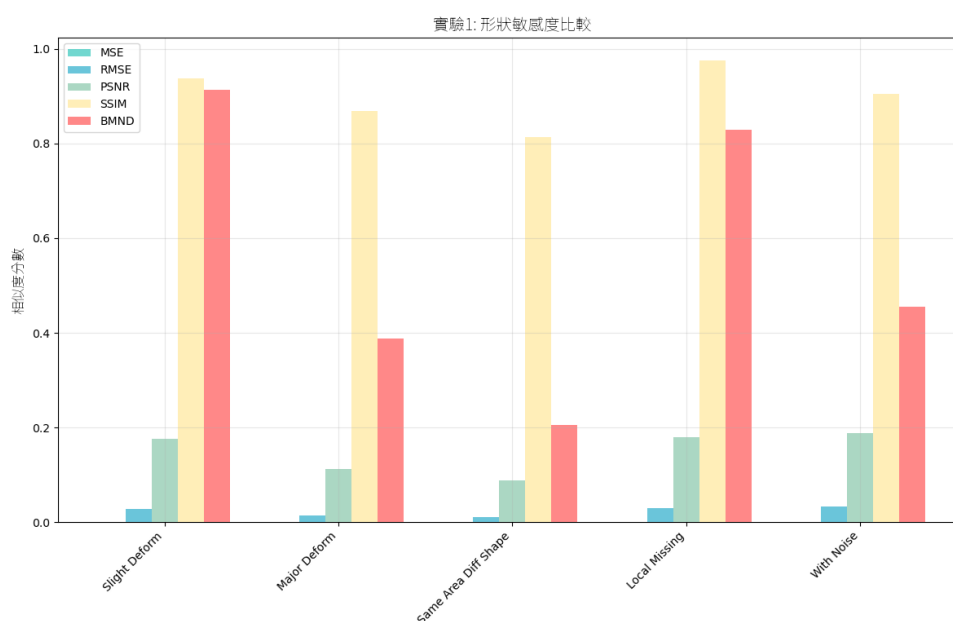
(一) 形狀敏感度比較

為了評估各相似度指標對於形狀微變化的辨識能力，本實驗進行了一組以形狀變異為主軸的測試。起始圖像為標準圓形，並依序變化為：

1. 輕微變形橢圓 (**Slight Deform**，圓形稍拉長)。
2. 嚴重變形橢圓 (**Major Deform**，長短軸差異明顯)。
3. 面積相近的正方形 (**Same Area Different Shape**，圓形稍拉長)。
4. 局部缺失的圓形 (**Local Missing**，切掉一角)。
5. 加入隨機噪點的圓形 (**With Noise**)。

此測試設計的核心目的在於觀察各指標是否能準確偵測形狀幾何上的細微改變，並避免被非幾何性擾動（如亮度、雜訊等）所干擾。實驗結果如圖 4-1 所示，傳統像素誤差型指標（如 **MSE**、**PSNR**）與結構性指標 **SSIM**，對於輕微形狀變化的反應有限，多數情況下分數變化極小（常維持在 0.9 以上），無法有效區分輕微與嚴重的形變差異。相較之下，本研究提出的 **BMND**（基於對稱輪廓點雲距離的相似度指標）展現出更強的幾何辨識能力。

在嚴重變形橢圓與正方形替代時，分數有明顯下降，有效區分幾何結構改變的程度；對於面積相近的正方形，**BMND** 能準確反映其視覺與幾何上的不同。此外，在異常點測試中，**BMND** 展現出卓越的穩健性：分數下降趨勢相對緩慢，即使異常點數量達 50 個，仍維持在約 0.59 分上下；傳統指標如 **PSNR**、**MSE** 在加入少量噪點後即降至接近 0；**BMND** 對局部雜訊具備高度容忍性，顯示其評估核心聚焦於輪廓整體幾何形狀，而非局部像素干擾。

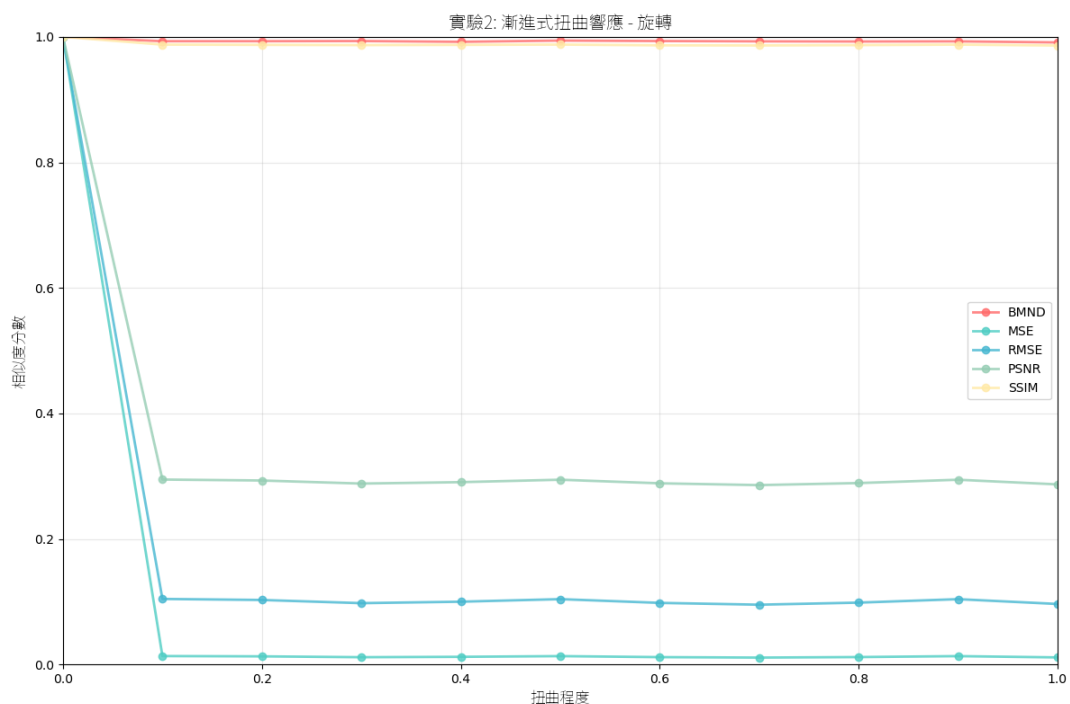


形狀敏感度比較 (圖片來源：作者自行繪製)

(二) 旋轉扭曲反應

為了分析不同相似度指標對幾何變形的穩定性，本實驗針對圖像施加旋轉與縮放變換，以觀察各指標的相似度分數響應曲線隨變形程度的變化趨勢，並量化其對幾何變化的敏感程度與穩健性。

在針對圖像旋轉的測試中，將測試圖像以 5° 為間距依序旋轉，角度範圍涵蓋 0° 至 45° ，模擬常見手繪偏差或圖形錯位的情境。針對每個旋轉角度，分別計算四種相似度指標: SSIM、MSE、PSNR、RMSE 以及本研究提出的 BMND (Bezier-based Mean Normalized Distance)，以評估其對旋轉變異的敏感度，如圖 4-2 所示，BMND 的響應曲線趨近水平線，即使在最大旋轉角度 (45°) 下仍維持穩定的高相似度分數，顯示其對旋轉具備高度不變性。而相較之下，傳統像素誤差指標 (MSE、PSNR、RMSE) 在旋轉角度僅 5° 時即出現顯著下降，RMSE 更是迅速趨近於 0，表現出對像素位置變動的高度敏感。SSIM 雖在初期旋轉中維持一定穩定性，但整體分數仍隨角度逐步下降，顯示其對結構位置的變動仍具有限度。



漸進式扭曲響應、旋轉（圖片來源：作者自行繪製）

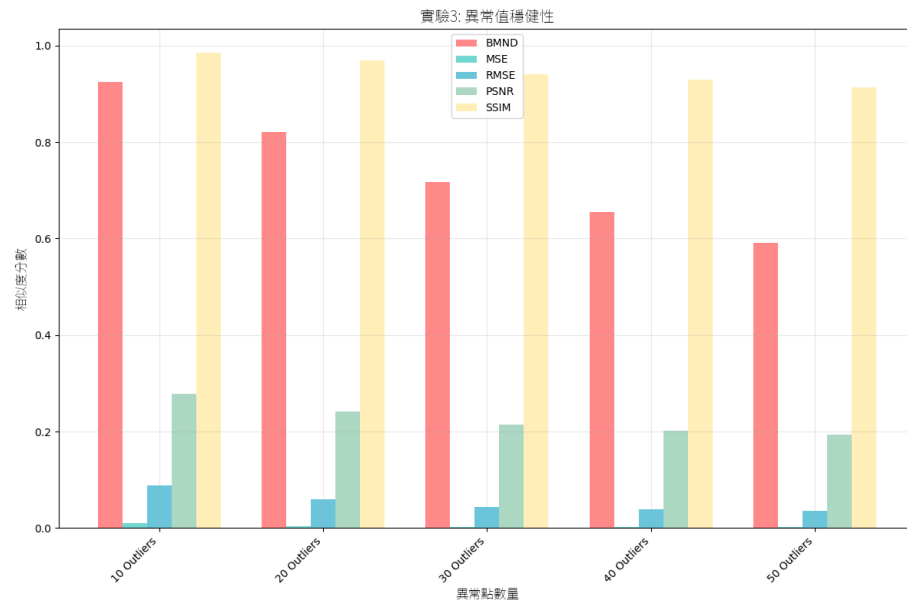
（三）異常值穩健性

為了評估各相似度指標在雜訊干擾條件下的穩健性，本研究設計一組異常值穩健性測試（**Outlier Robustness Test**）。於簡單封閉輪廓的基礎圖像上，人為隨機添加不同數量的異常點（**Outliers**），模擬手繪圖形中常見的局部雜訊、筆誤或描繪偏差情形。異常點分別設定為 10、20、30、40、50 點，並均勻分佈於圖像空間中，以測試各相似度指標在圖形完整性受干擾時的表現穩健性。

如圖 4-3 所示，傳統相似度指標（如 **MSE**、**PSNR**、**RMSE**）對異常點極度敏感，在加入少量雜訊後即出現劇烈下降，失去對形狀整體輪廓的辨識能力。而 **SSIM** 雖對異常點具一定容忍度，分數仍維持於 0.9 以上，但卻無法有效反映圖形幾何結構實際受損的程度，導致其在形狀異常檢測上準確性不足。相較之下，本研究所提出的 **BMND**（**Bezier-based Mean Normalized Distance**）指標展現出以下顯著優勢：

1. 能夠清晰區分不同程度的形狀變異，如對 **slight deform** 給予高分（0.91），對 明顯變形則下降至約 0.39。
2. 在異常點數量達 50 點的情況下，**BMND** 仍維持約 0.59 的穩定分數，顯示其能有效忽略局部雜訊干擾。
3. 相較於 **SSIM**，**BMND** 對幾何結構的變化具更高靈敏度與判別

力，特別適用於需要捕捉筆劃輪廓異常或誤差分析的應用場景。



異常值穩健性（圖片來源：作者自行繪製）

整體而言，**BMND** 透過點雲輪廓結構進行相似度衡量，能較準確地反映圖形整體的幾何一致性與破壞程度，相較於傳統方法基於像素的指標更適合應用於草圖擬合、手寫圖形分析與幾何結構比對等任務中。總結來說，**BMND** 更符合本研究對結構精準還原與幾何相似度量化的核心需求，能夠在有效簡化節點數的同時，提供一個具備數學穩健性與直觀解釋力的評估指標，因此被選為本研究中最具代表性的形狀品質評估方法。

日期:2025/5/29 至 6/1

實作內容

修正 **SVCFP**:新增角度與外積變化分析

針對接收到的 **RDP** 陣列，本研究對每個點進行角度變化分析，以識別曲線中的顯著轉折。具體而言，先計算每個點與其前後相鄰點之間的單位向量，然後在簡化後貝茲節點上進行角度變化與方向分析，接著選取簡化點並以其前後兩個點為對象，分別形成兩條向量，經由內積計算成角度變化，同時也進行外積偵測，以判斷前後向量是否存在方向改變。當外積的符號發生由正變負，或由負變正的方向變化，即認定為該點發生了重大轉向，在完成上述角度變化分析和權重調整後，對每個點的評分進行閾值判斷。具體而言，提取評分超過預先設定閾值的點位，並將其標記為特徵點進行下個步驟的計算。

日期:2025/6/2 至 6/15

實作內容

修正 **SVCFP**:新增中點插入機制、相近點融合機制

為了彌補 **RDP** 簡化或特徵值閾值篩選可能遺漏的關鍵曲線形態，系統會精確評估相鄰候選特徵點在原始曲線中的索引距離。當兩點間的距離超過預設閾值時，系統將啟動進一步的幾何分析。

首先，會計算該區段內積所對應的夾角變化。若偵測到該角度超過特定門檻，表示曲線在此處可能存在彎折或方向轉變，系統便會在區段中央插入一個新的特徵點，以補足簡化策略中未能涵蓋的幾何特徵。

最後為了避免新插入節點造成過度密集或冗餘，在節點確立前，系統亦會進行相近點檢查，若發現與周邊節點過於接近，則會執行消融策略（**fusing**），自動剔除冗餘點，確保最終輸出的特徵點集合兼具資訊密度與分布均衡性，此機制有助於提高整體特徵提取的完整性與準確性。

日期:2025/6/15 至 7/1

實作內容

製作海報與修改報告

日期:2025/7/2 至 7/10

實作內容

練習報告