資訊專案開發與實作

氣囊瑕疵檢測

指導老師：蘇文鈺 老師

業師：劉軒志、張睿龍 老師

學生：呂碩晨、吳赫宥

專案功能

傳入一晶圓製程耗材瑕疵圖片

↓

經適當處理

↓

顯示框選結果

目錄

RECAP

1.動機&目標

2.修正先前問題

3.前導概念－名詞解釋

專案動機與目標

1.動機－減少時間成本，增加效率

2.目標－抓取瑕疵，避免失真

修正先前問題

1.Memory leak－ by GC

2.燒焦型態瑕疵(黑色)－優化FFT算法

前導概念

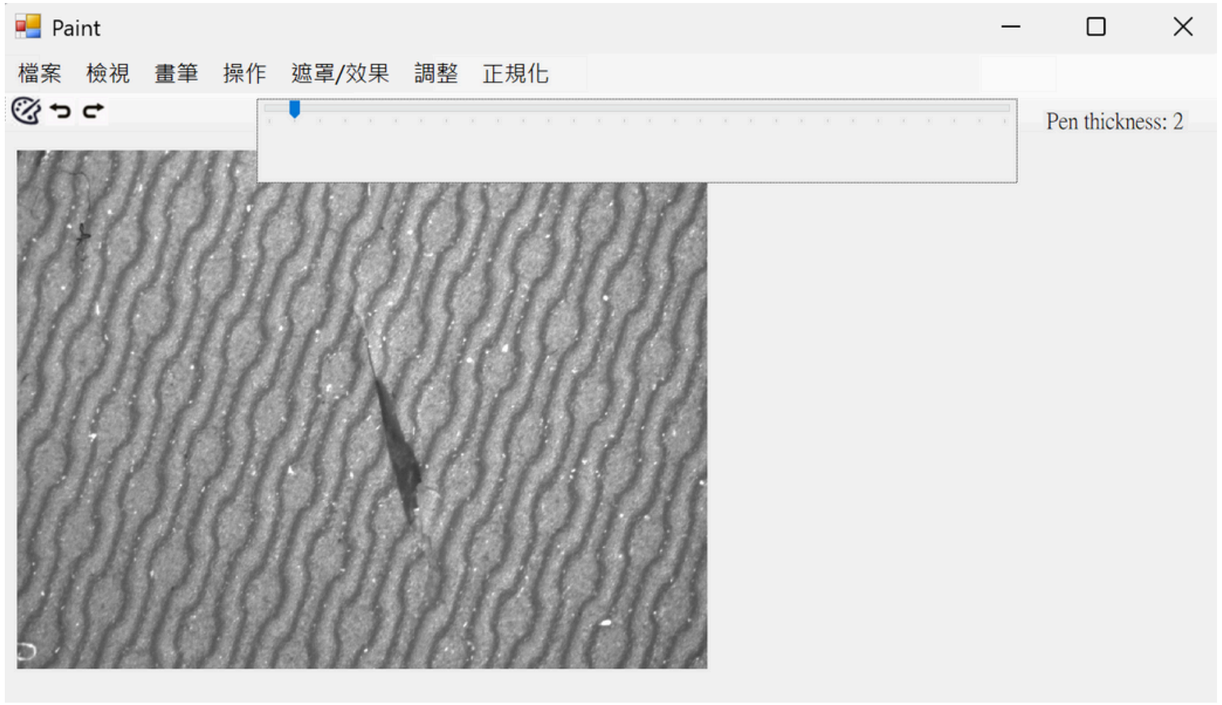
FFT：快速傅立葉變換，可將時域/空間域轉為頻域的變換

頻率：圖片中頻率的概念為「相鄰像素差的灰度變化」

濾波：類似電學中的濾波器

形態學：有多種應用，可用於改善圖片主體的形狀，尤其是用於二值化圖片

功能介紹



檔案

- 開啟圖檔
- 新增畫布
- 儲存檔案
- 結束

檢視

- 放大
- 縮小

畫筆

- 自由
- 直線
- 矩形
- 圓
- 橢圓
- 三角形

遮罩/效果

- 臨界處理
- 型態變化
- LUT

調整

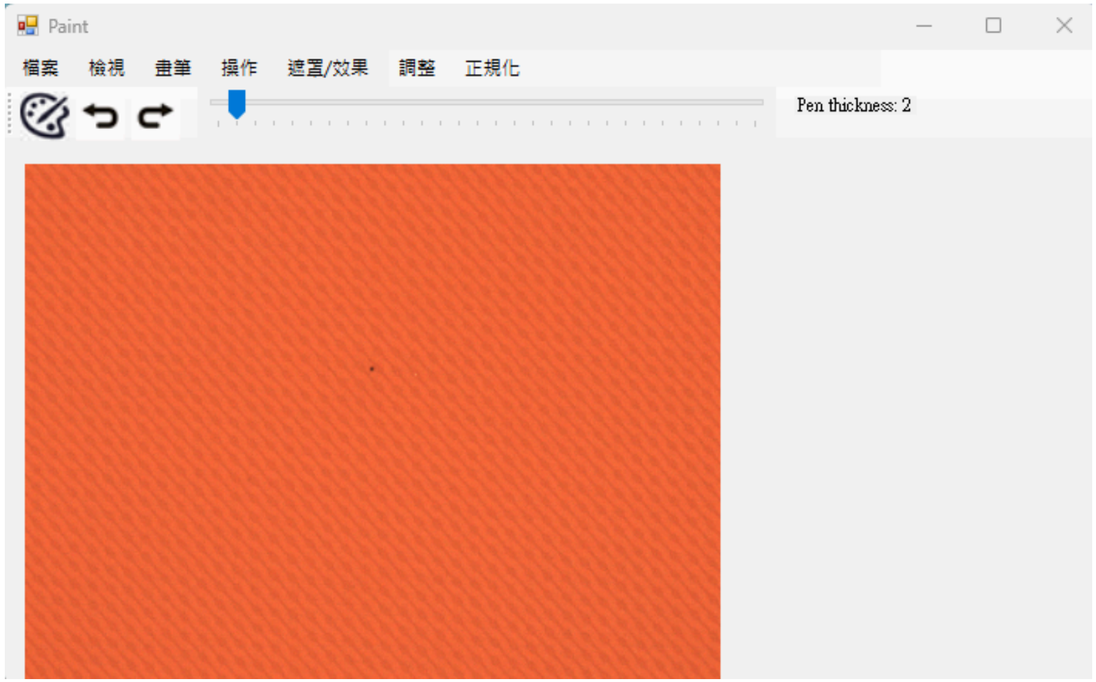
- 強度轉換
- 空間濾波
- 轉換成灰階
- 使用FFT
- IFFT
- 色彩空間
- RGB調整

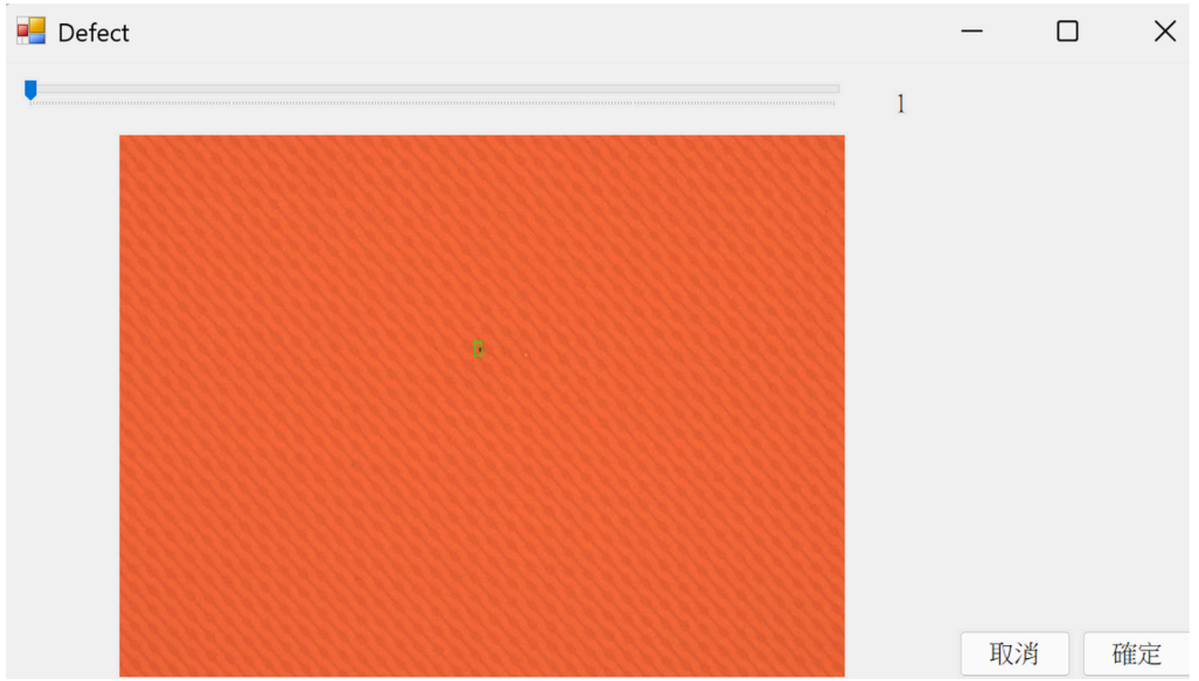
操作

- 復原(Undo)
- 重做(Redo)
- 繪製亮度直方圖
- 描繪輪廓
- FindContours
- 框選瑕疵
- 邊緣

新增功能

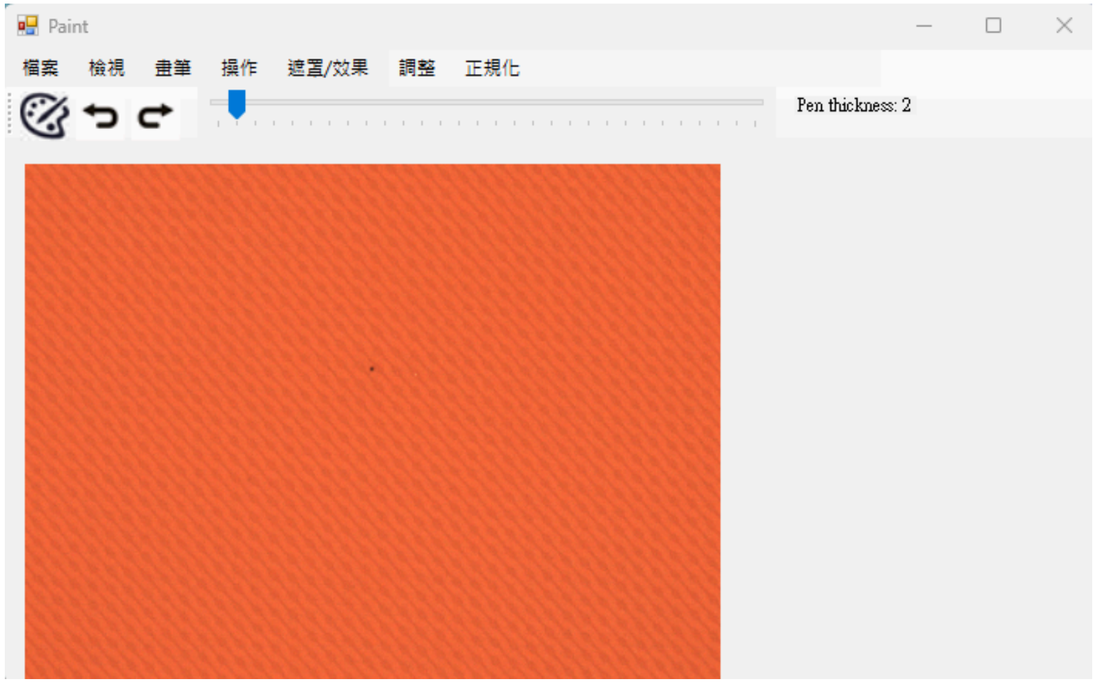
框選瑕疵：標記位置，醒目，有助比對

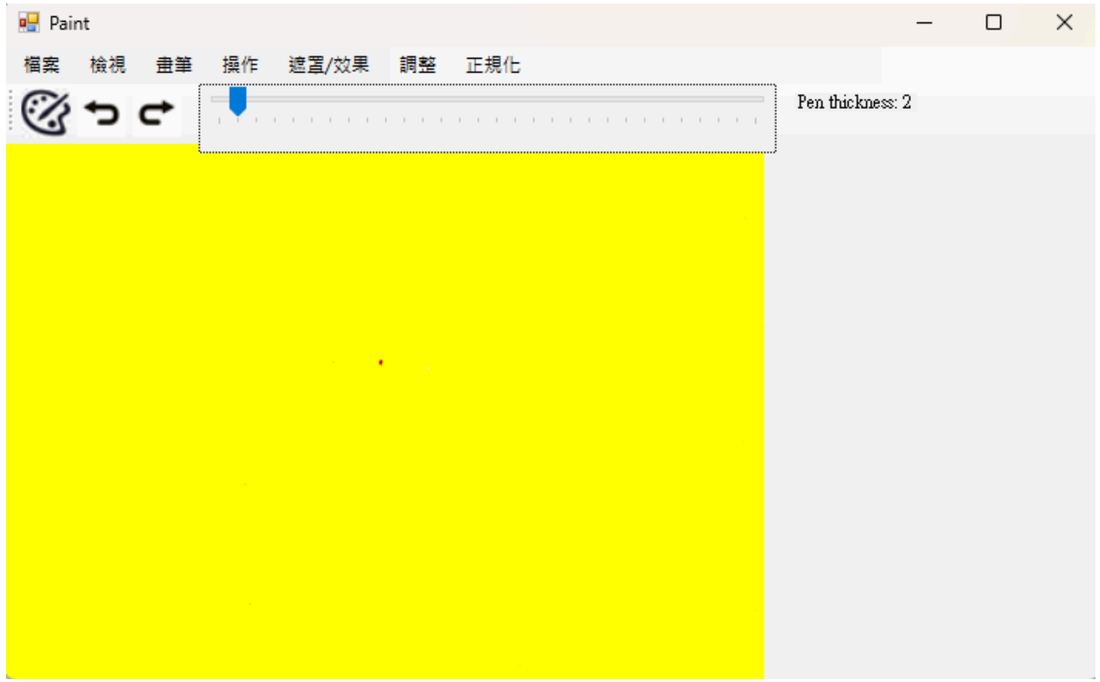




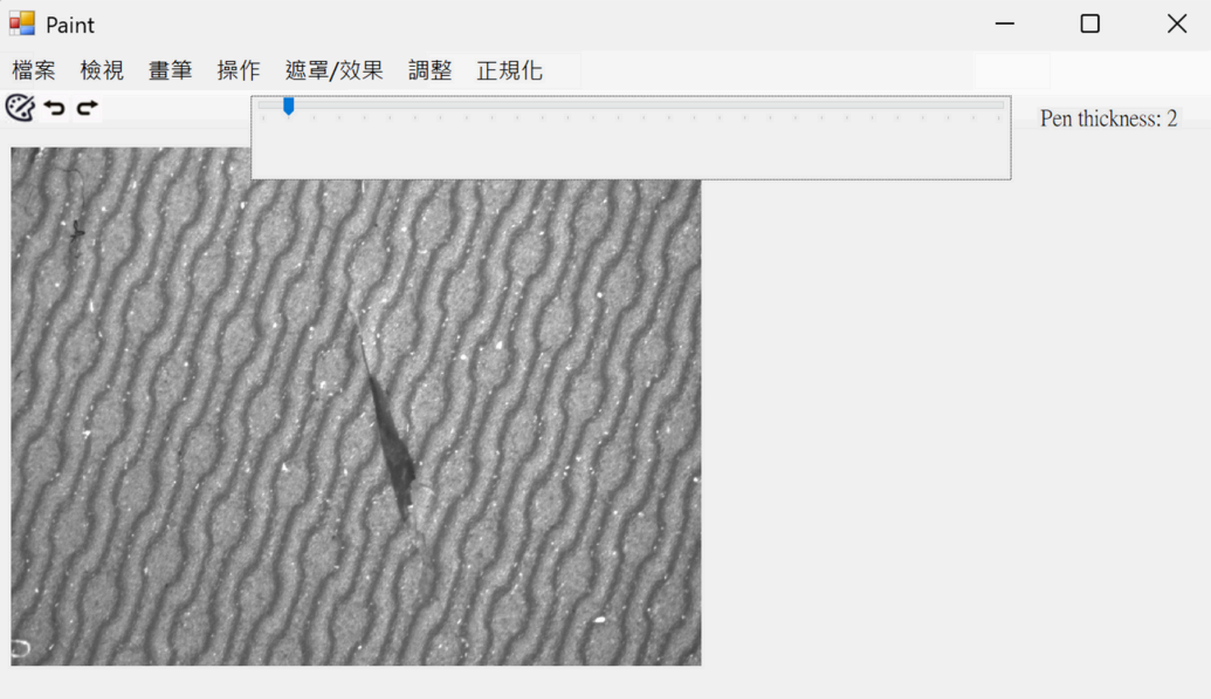
新增功能

LUT(Look Up Table)：色彩轉換表，可提取特定亮度資料





主要功能

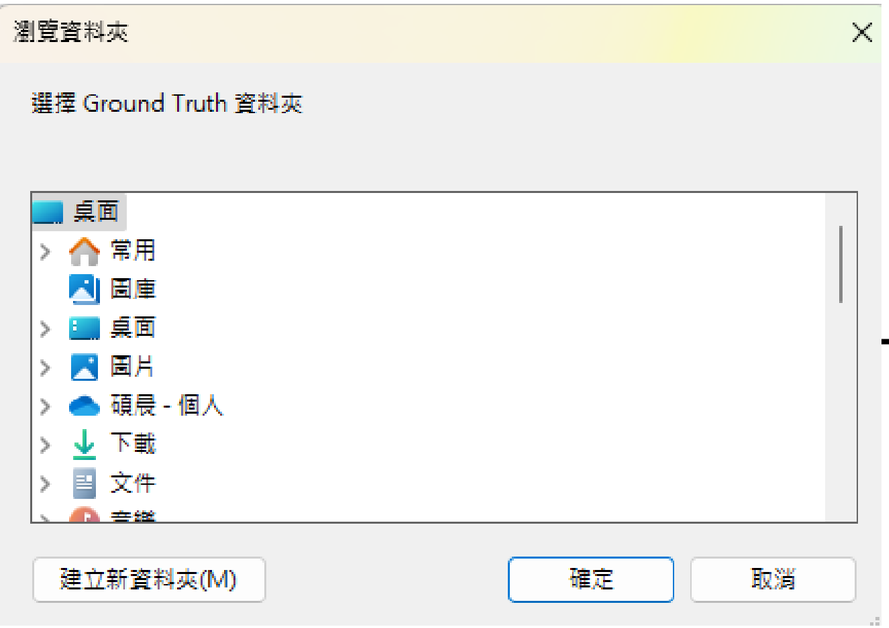


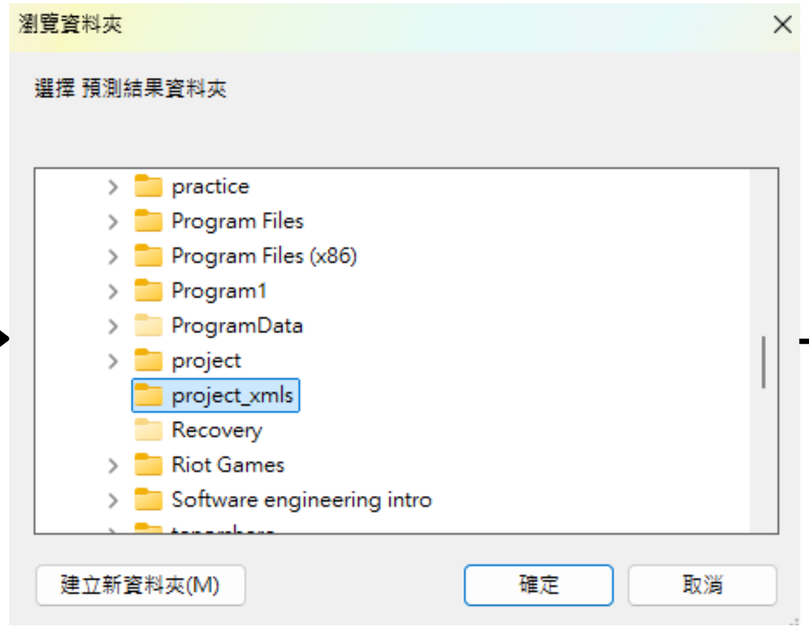
正規化：亮度標準化

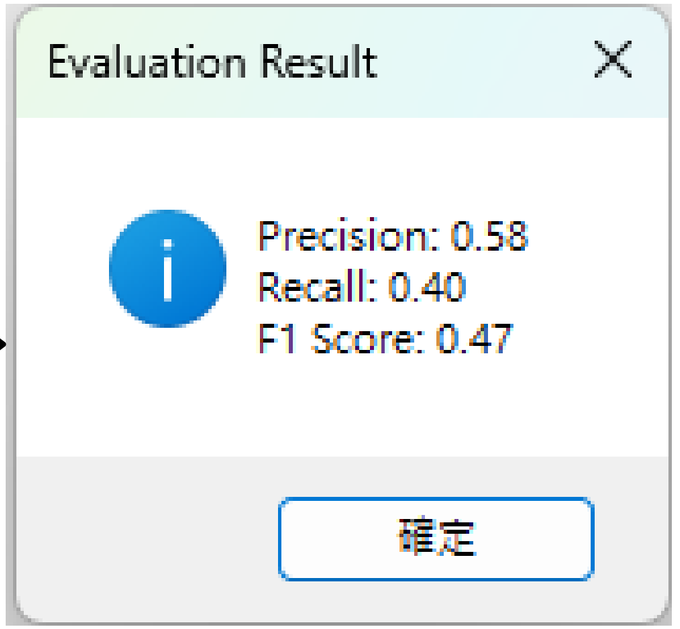
```
Normalize
equalizeHist
```

主要功能

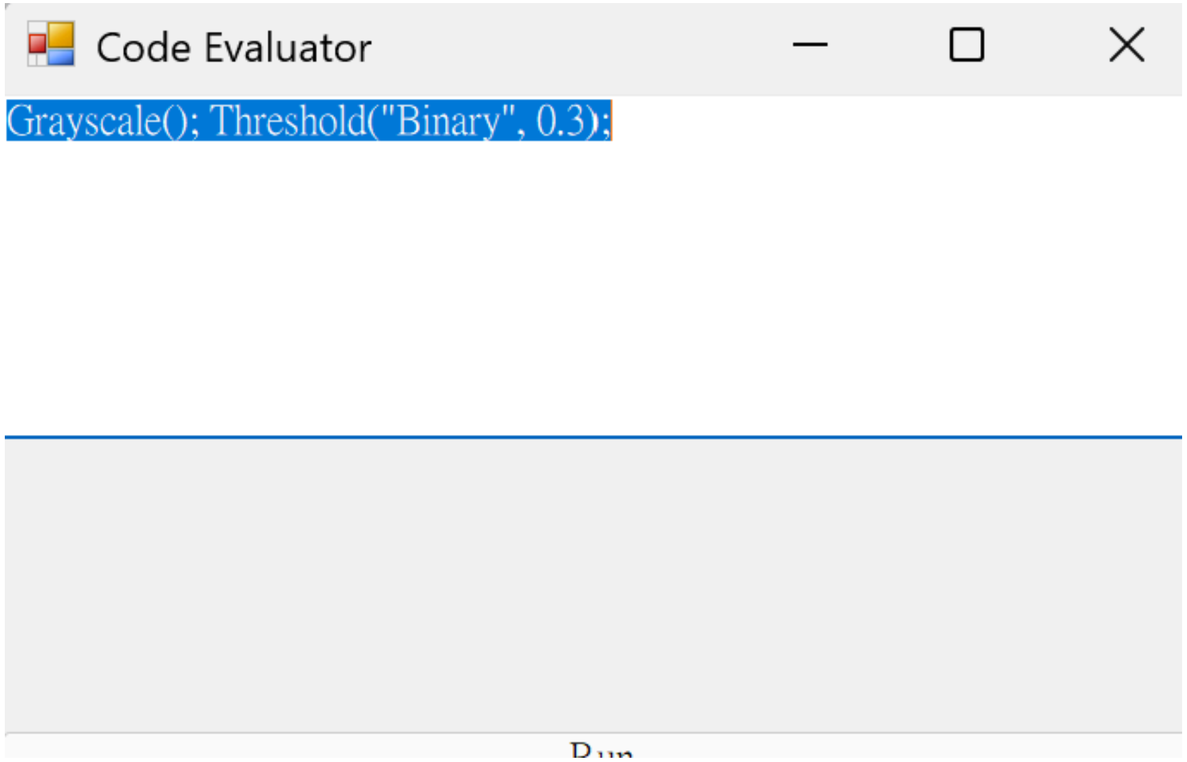
資料比對系統：快速比對正確性







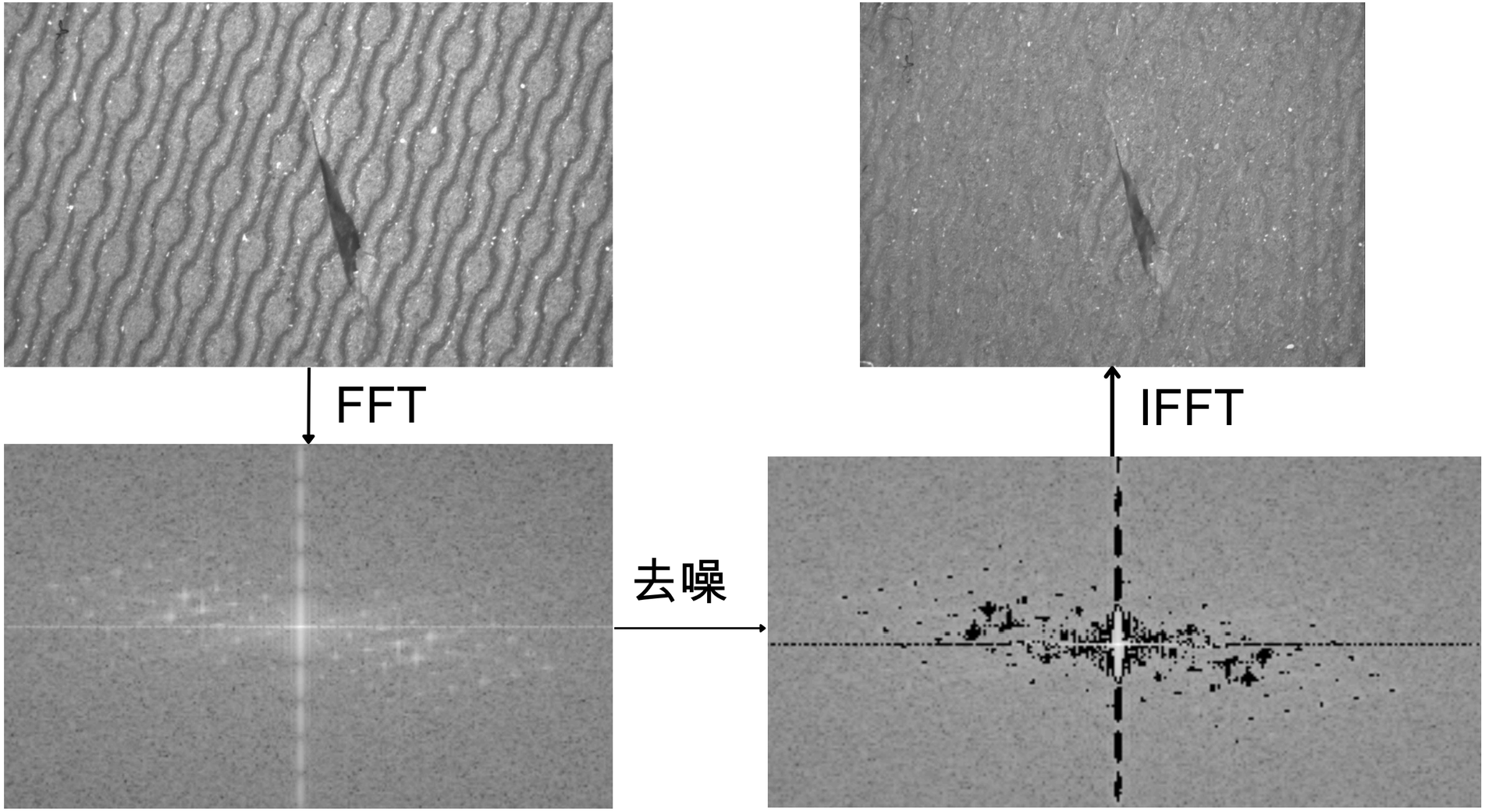
主要功能



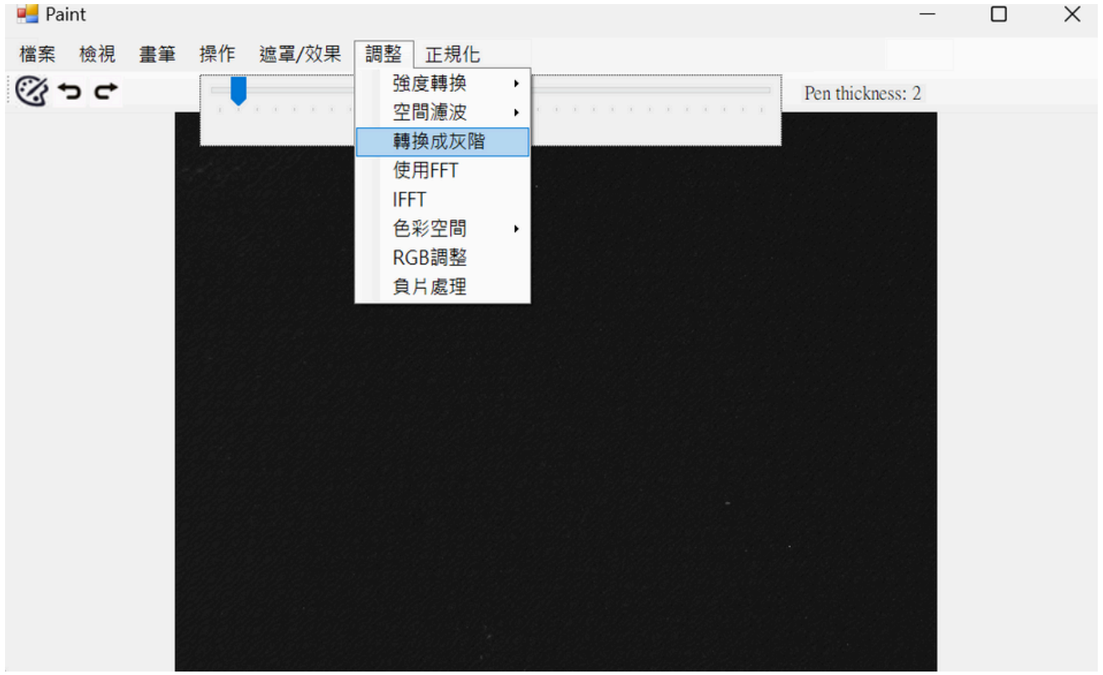
code evaluator

可增加操作效率，整合成一步驟

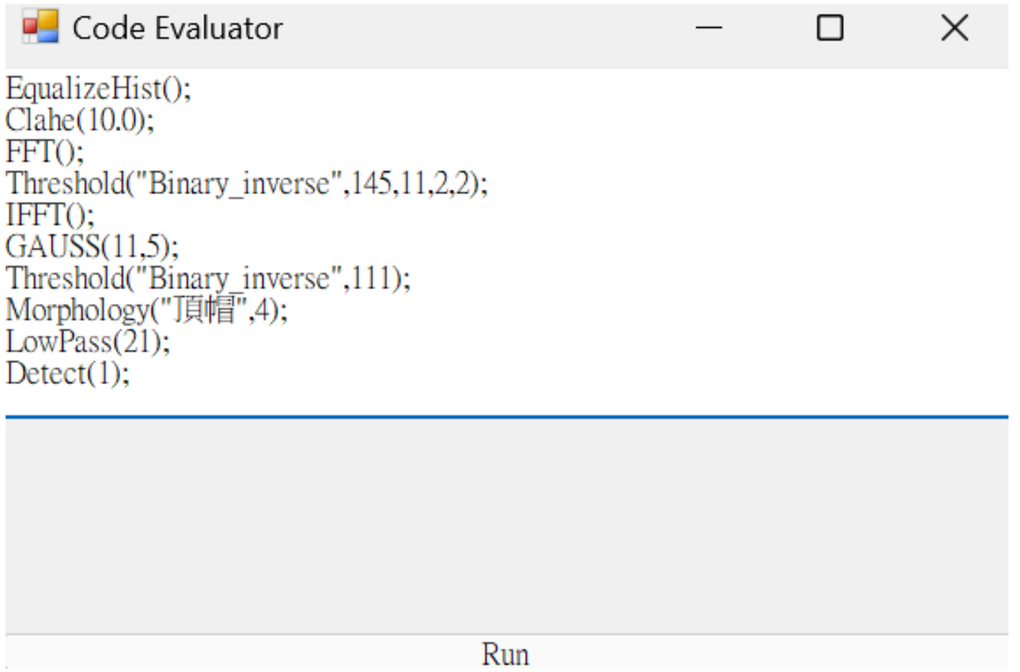
使用方式



使用者自行查找功能按鍵



利用eval執行腳本

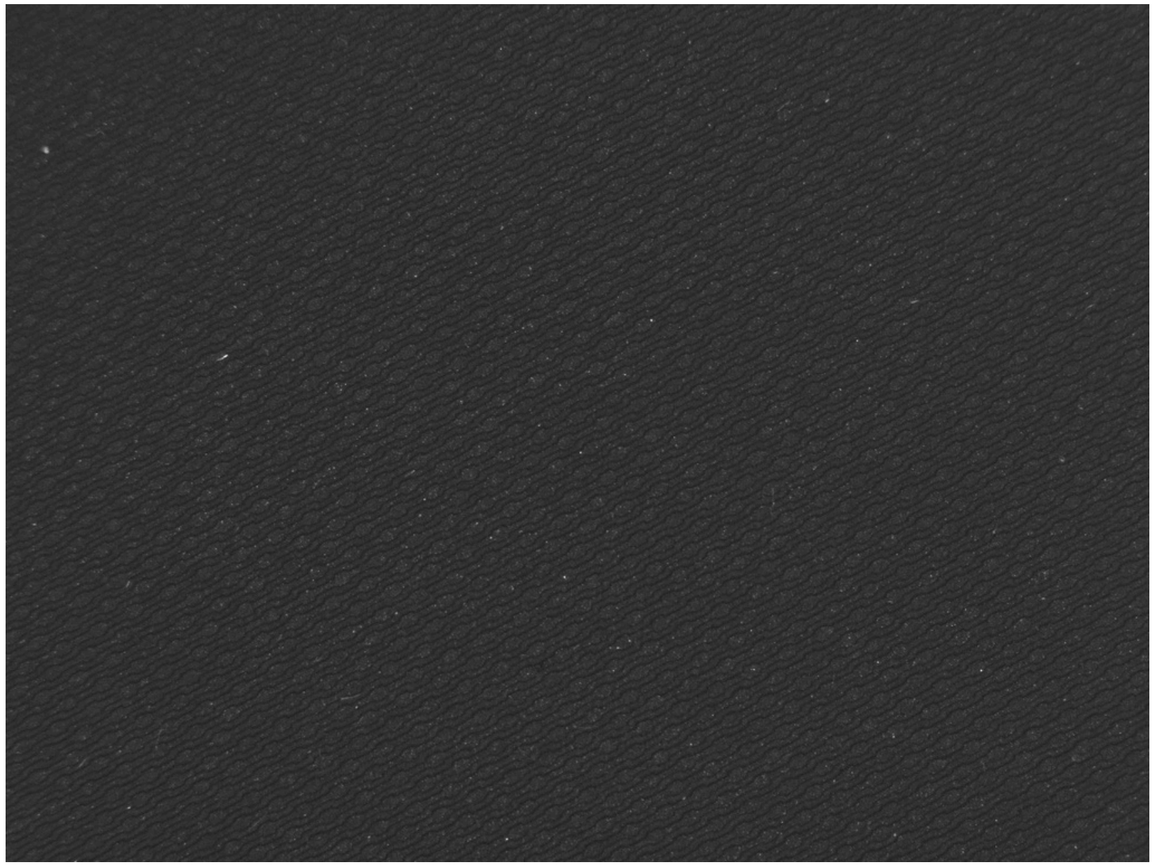


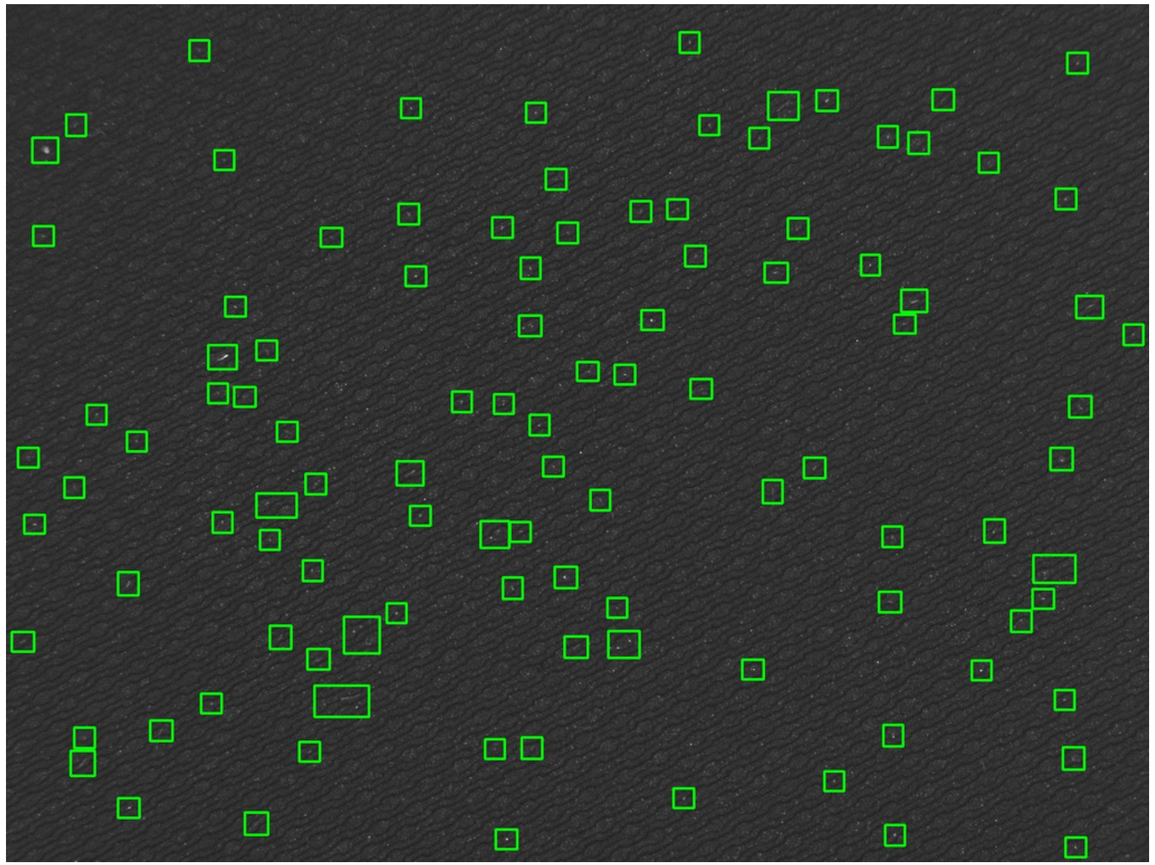
功能名稱	呼叫方式
灰階轉換	Grayscale()
快速傅立葉轉換	FFT()
反傅立葉轉換	IFFT()
閾值處理	Threshold(string modeName, double value, int blockSize, int cValue, int fftSeed)
低通濾波	LowPass(int ksize)
高斯濾波	GAUSS(int kernelSize, double sigma)
直方圖等化	EqualizeHist()
CLAHE自適應直方圖	Clahe(double clipLimit)
RGB通道線性變換	RGBmotify(int redMultiplier, int greenMultiplier, int blueMultiplier, int redBias, int greenBias, int blueBias)
對比與亮度調整	C_B(double a, int b)
Normalize標準化	Normalized(int thres)
LUT映射轉換	LookUpTable(int lt, int ht)
缺陷檢測	Detect(int a)
擷取白色缺陷	CaptureDefect_white()
擷取黑色缺陷	CaptureDefect_black()
合併缺陷遮罩	CombineDefect()
形態學處理	Morphology(string operationType, int iterationCount)
重置畫布	Reset()

黑盒子處理腳本

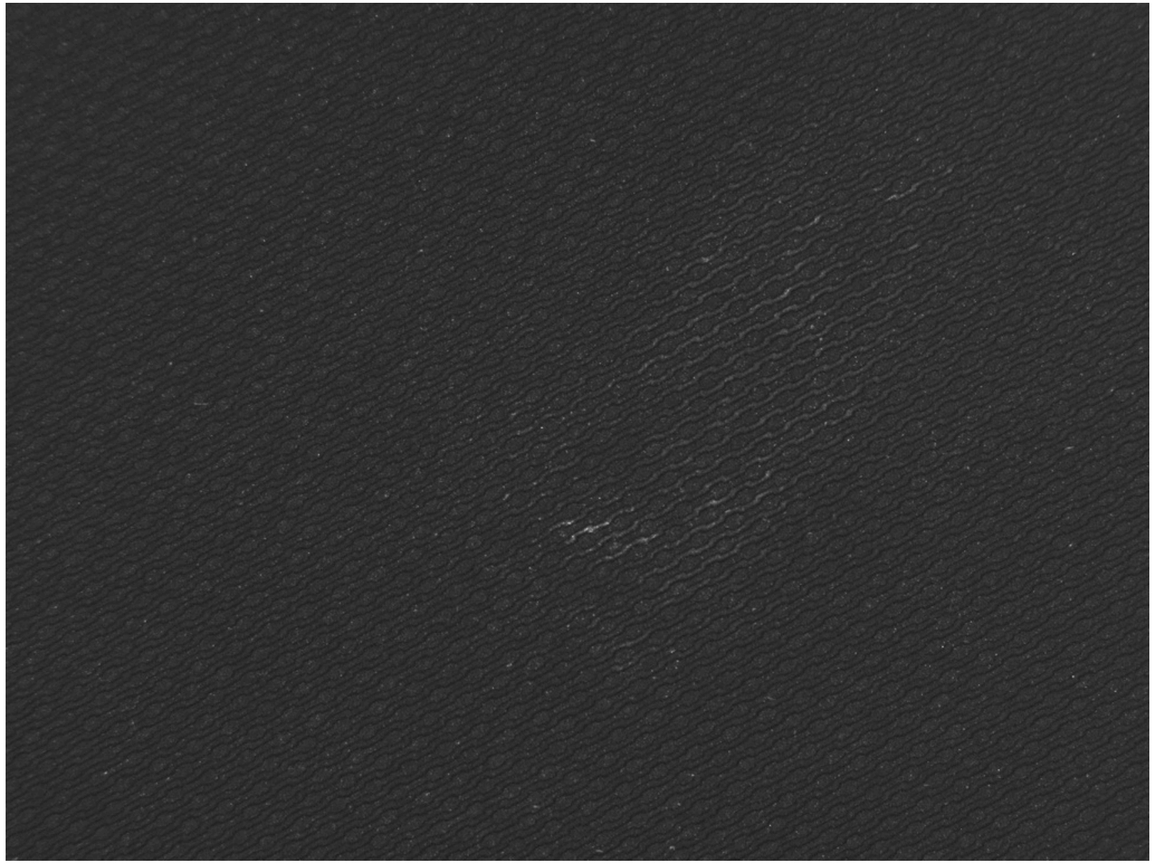
```
Grayscale();
Normalized(225);
LookUpTable(0, 57);
Threshold("Binary",151.34);
Detect(7);
```

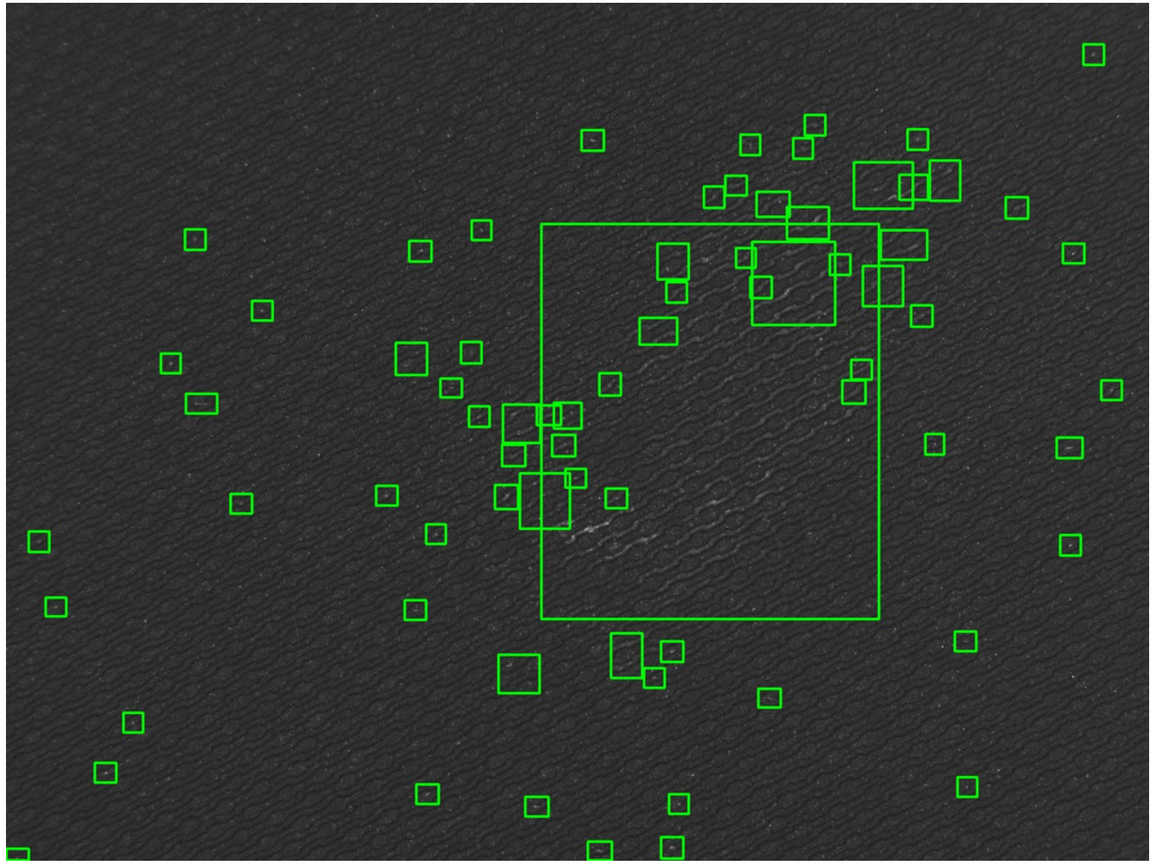
效果展示



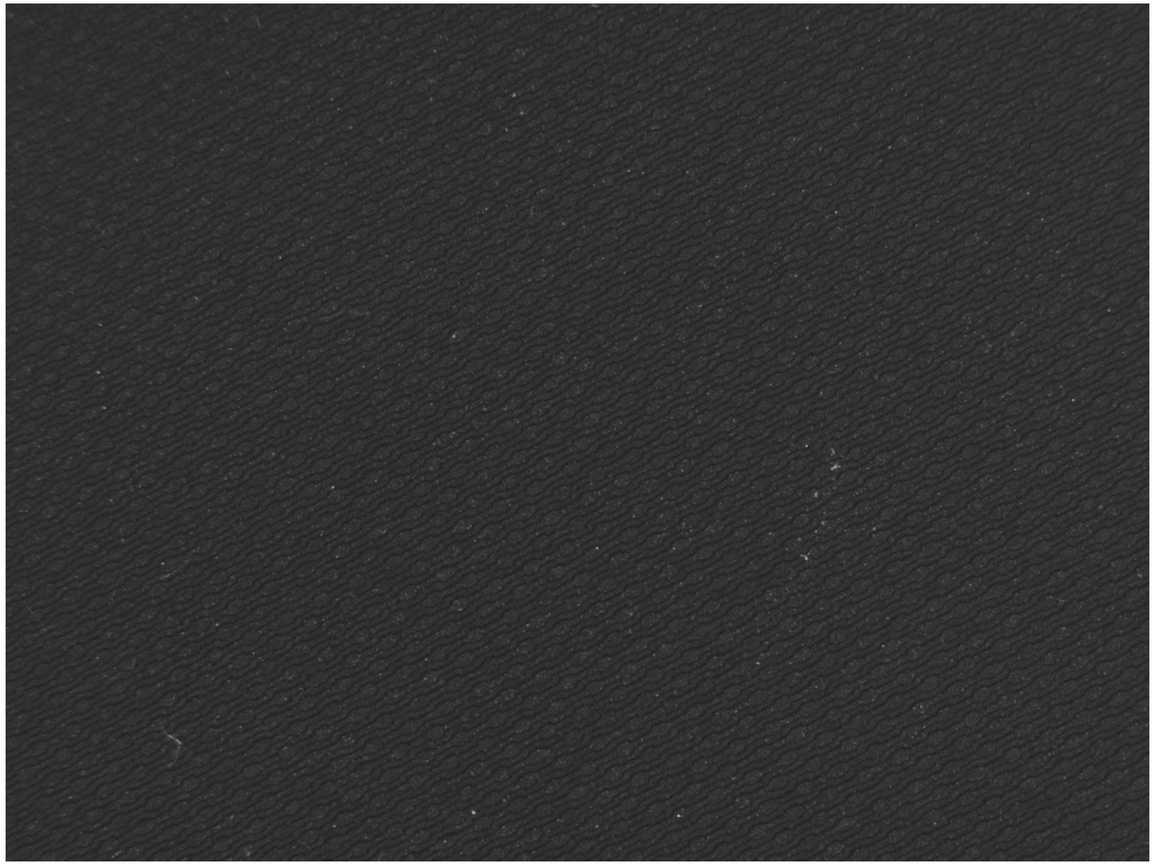


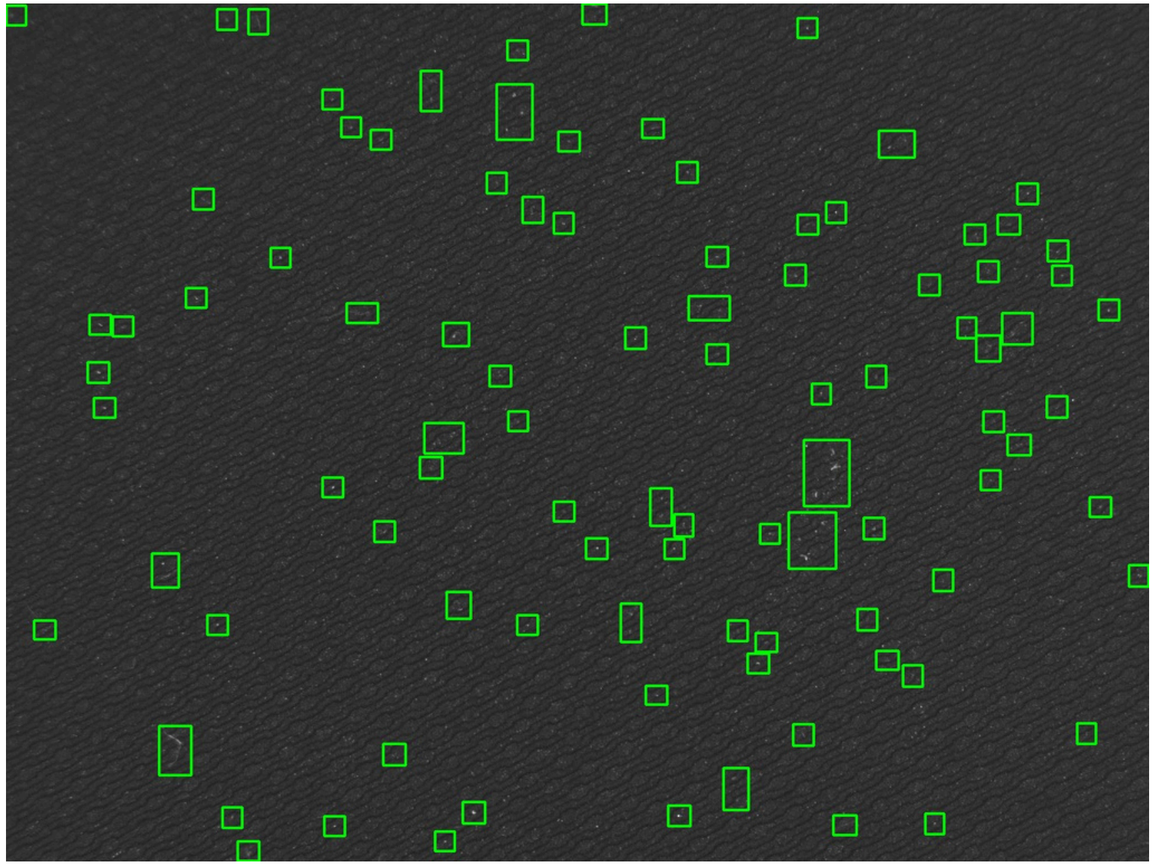
效果展示



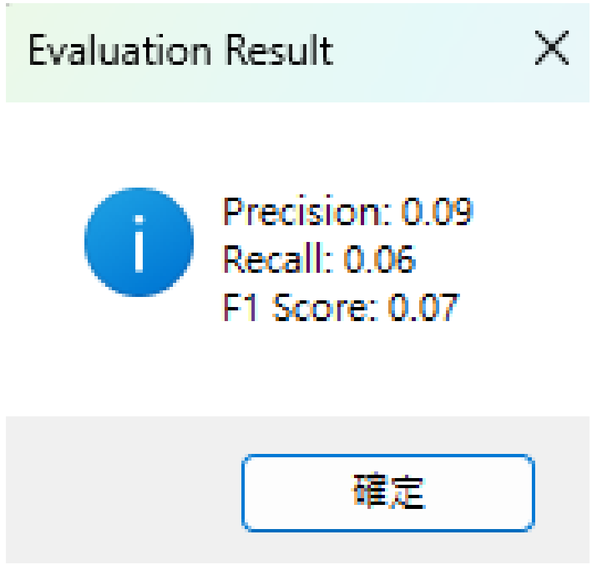


效果展示

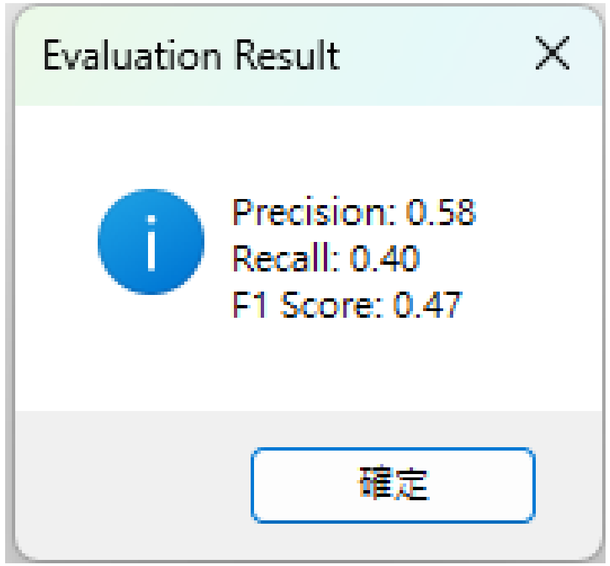




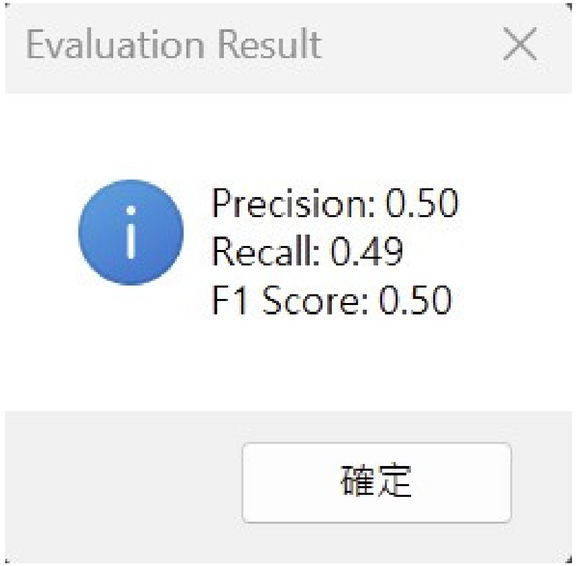
比對結果



原腳本(before)



原腳本(after)



改良腳本

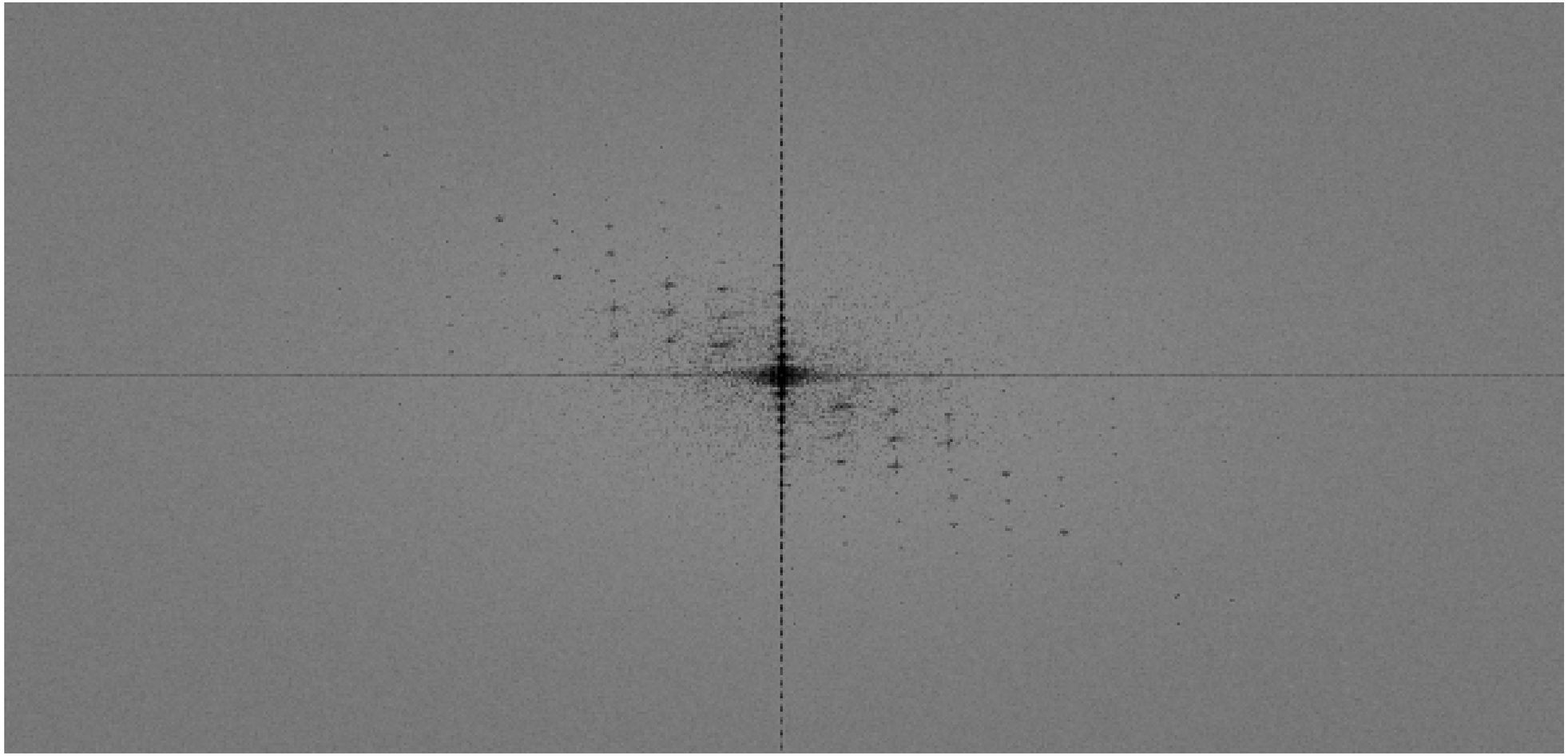
影響因素

1.ground truth 偏差－人工標準、框選大小

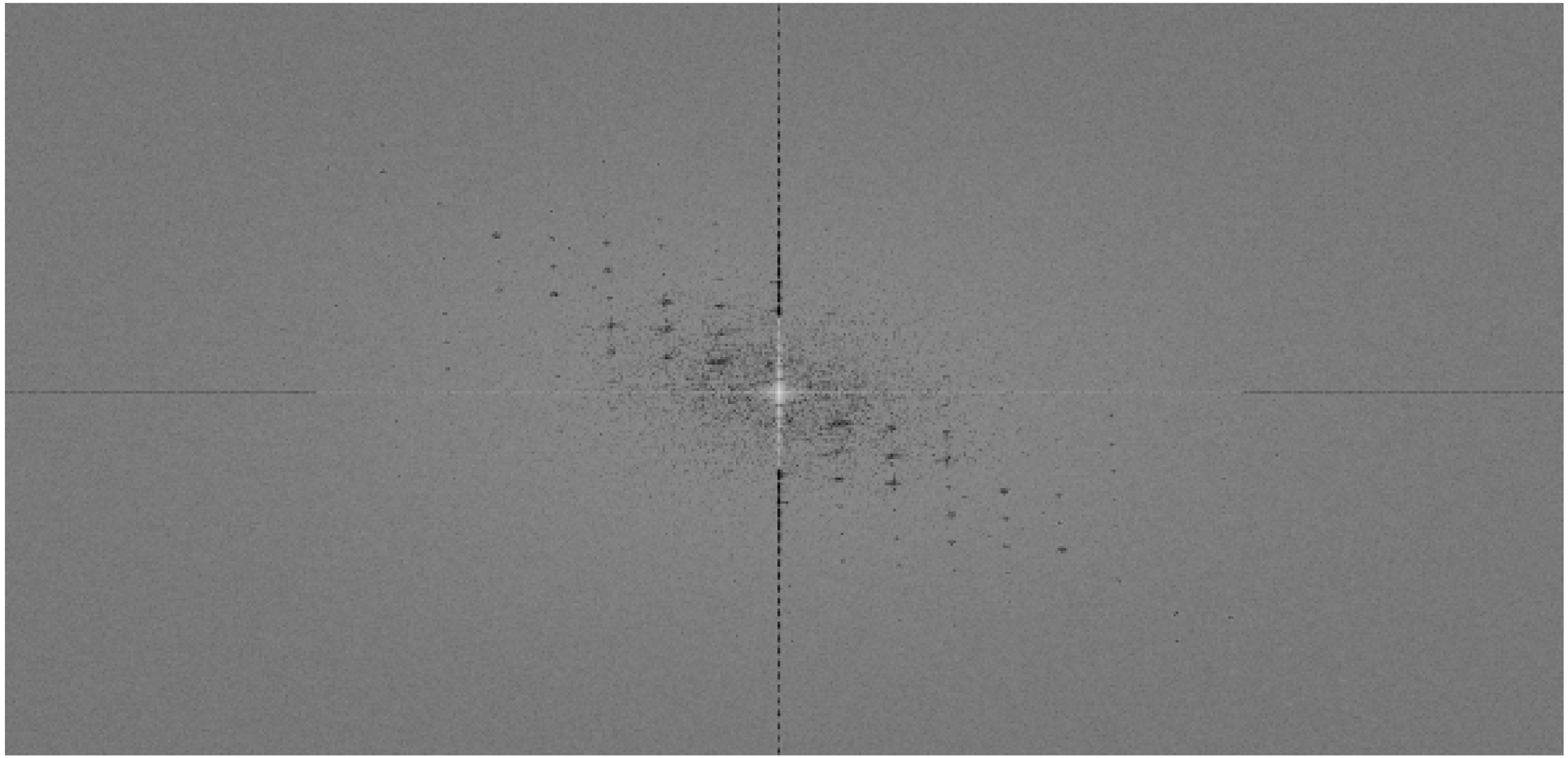
2.比對演算法－IoU未包含coverage

3.專案表現－參數調整

遭遇困難



解決方式



結論與反思

1.在進行比對系統優化前，準確率不到10%，經修正後提升至約50%

2.測試與抓取瑕疵速度藉由eval顯著提升

3.專案code重構