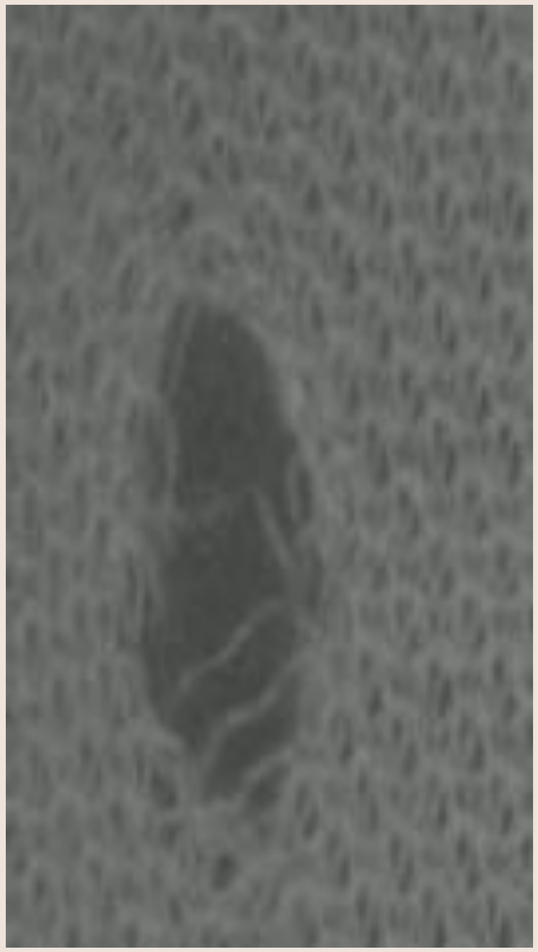
影像處理專題

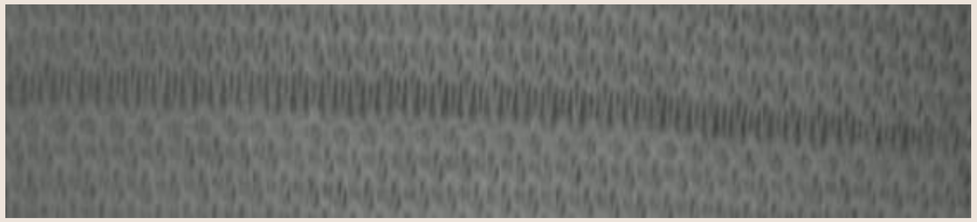
Texture Analysis
使用statistics方法

區分線條/圓形瑕疵

區分線條/圓形瑕疵

從影片 中抓取到的一些瑕疵，要想辦法區分線條或圓形，因為線條和圓形的瑕疵所決定的門檻值不同，所以要先區分出來





區分線條/圓形瑕疵

因為線條和圓形的瑕疵其門檻值有些不同，所以要先判斷是哪種類型的瑕疵

使用7張影像(3線條、3圓形、1正常)，分別計算其GLCM的contrast與dissimilarity特徵矩陣，取得該矩陣的平均、標準差、最大值

以正常影像的數值做為判斷線條/圓形瑕疵的門檻值
有5個條件判斷，當超過3個條件判斷都判定為圓形瑕疵，則之後的步驟會以圓形的門檻條件來進行

區分線條/圓形瑕疵

contrast mean	contrast standard	contrast max
線條1 112.8089	線條1 43.0410	線條1 282.7683
線條2 75.9854	線條2 26.9951	線條2 160.9983
線條3 63.3973	線條3 22.5234	線條3 130.2200
圓形1 179.3911	圓形1 73.8458	圓形1 411.6783
圓形2 170.5760	圓形2 66.9033	圓形2 348.2666
圓形3 115.1458	圓形3 44.4110	圓形3 284.2933
正常 125.0335	正常 38.5447	正常 246.2500
dissimilarity mean	**dissimilarity standard**	
線條1 7.9763	線條1 2.2958	
線條2 6.5080	線條2 1.8203	
線條3 5.9278	線條3 1.6360	
圓形1 10.0218	圓形1 3.0492	
圓形2 9.7933	圓形2 2.8624	
圓形3 8.0274	圓形3 2.3581	
正常 8.4595	正常 2.2448	

區分線條/圓形瑕疵

因為線條和圓形的瑕疵其門檻值有些不同，所以要先判斷是哪種類型的瑕疵

```
#門檻值設定
contrast_mean_threshold = 125          #circle大於 125.033578
contrast_std_threshold = 39           #circle大於 38.544783
contrast_max_threshold = 246           #circle大於 246.250000
dissimilarity_mean_threshold = 8.46   #circle大於 8.459510
dissimilarity_std_threshold = 2.3     #circle大於 2.244871
circle_count = 0
is_circle = True

#過關，判斷線性or圓形
if contrast_mean > contrast_mean_threshold:circle_count+=1
if contrast_std > contrast_std_threshold:circle_count+=1
if contrast_max > contrast_max_threshold:circle_count+=1
if dissimilarity_mean > dissimilarity_mean_threshold:circle_count+=1
if dissimilarity_std > dissimilarity_std_threshold:circle_count+=1
if circle_count >= 3:
    print('image is circle')
else:
    print('image is linear')
    is_circle = False
```

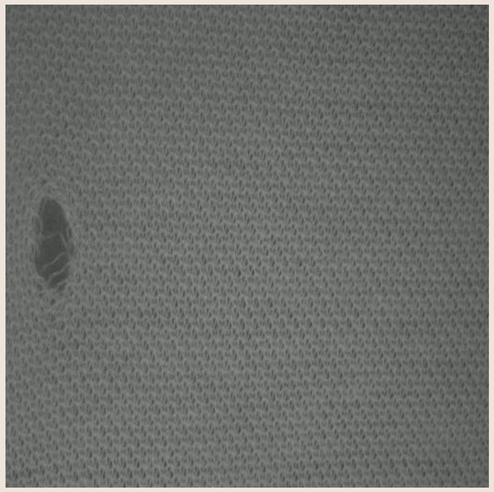
2

影像前處理

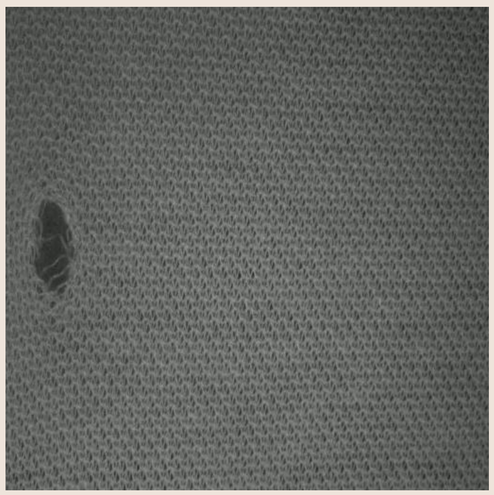
影像前處理

使用beta_correction函式，調整輸入影像的對比度，讓瑕疵破洞處變清楚且明顯，以利於之後的偵測能更容易區分出來。

原影像



對比調整後



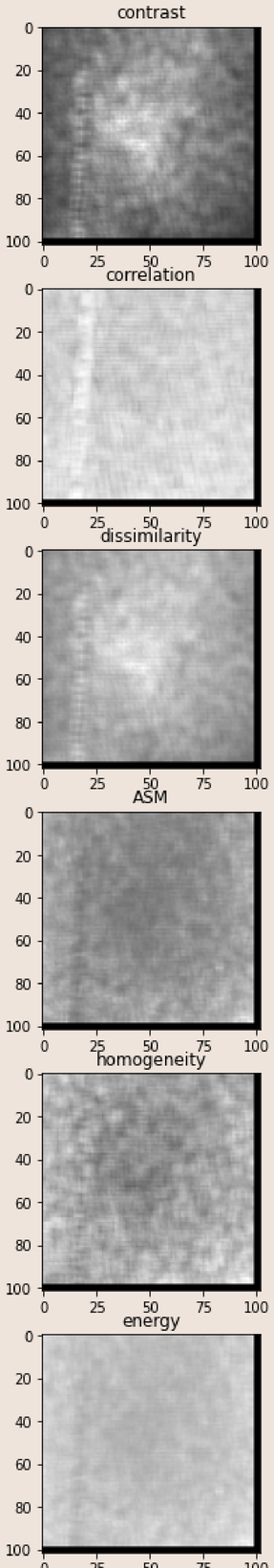
3

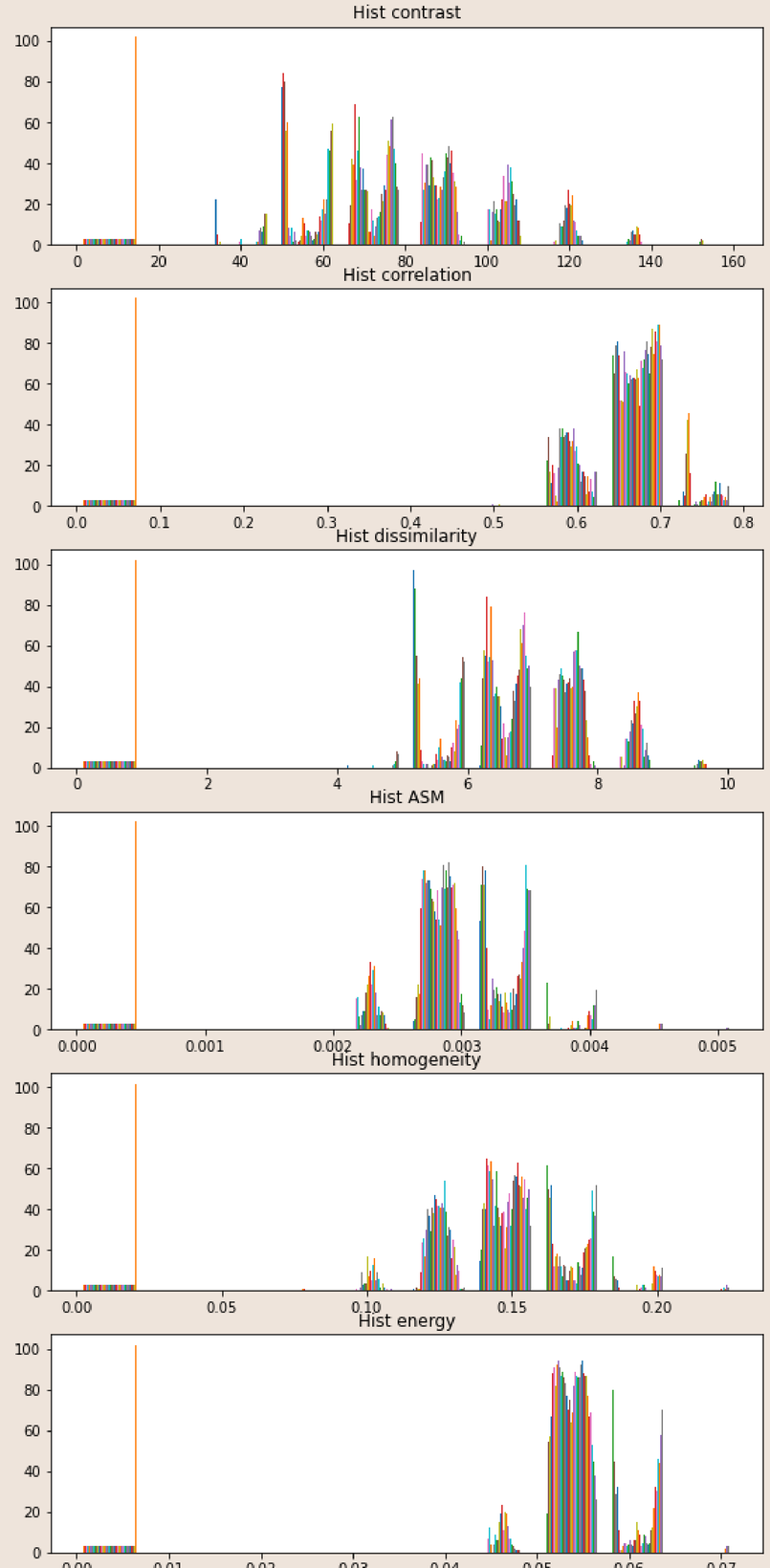
線條瑕疵
特徵選擇與門檻

線條瑕疵特徵選擇與門檻

經過實驗以及比對，使用correlation作為主要的門檻值，會比起使用contrast以及其他方法更有效果，且瑕疵也較能顯著的發現

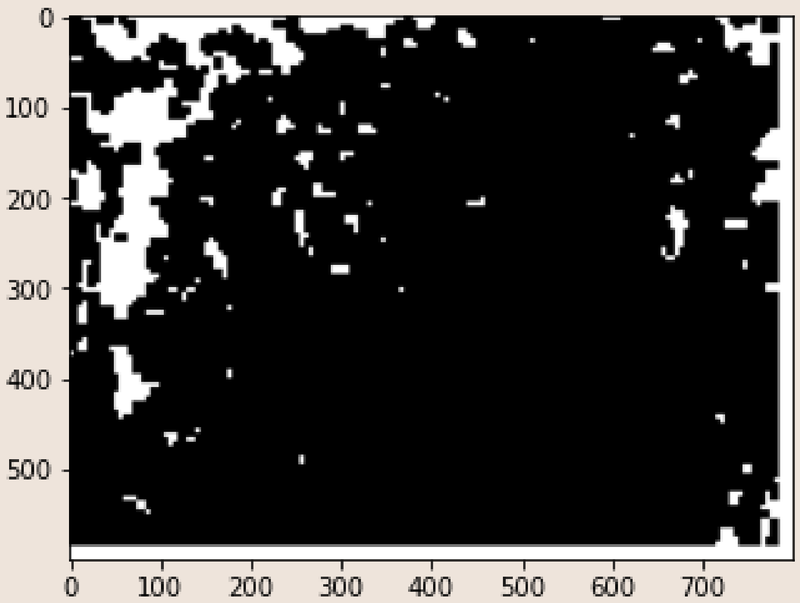
右圖為其它GLCM特徵矩陣的影像和直方圖

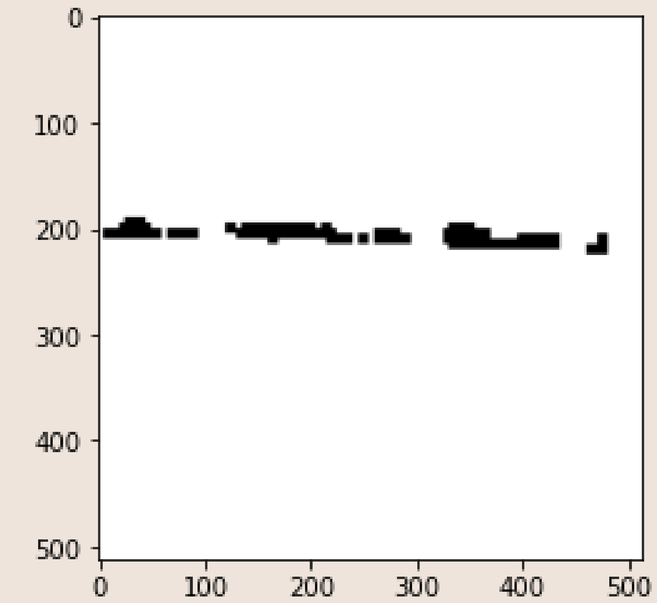




線條瑕疵特徵選擇與門檻

原本是以固定門檻值的方式來判斷，以co_correlation[i,j] > 0.72 舉例，
在某些瑕疵中會無法區分正常和瑕疵的範圍

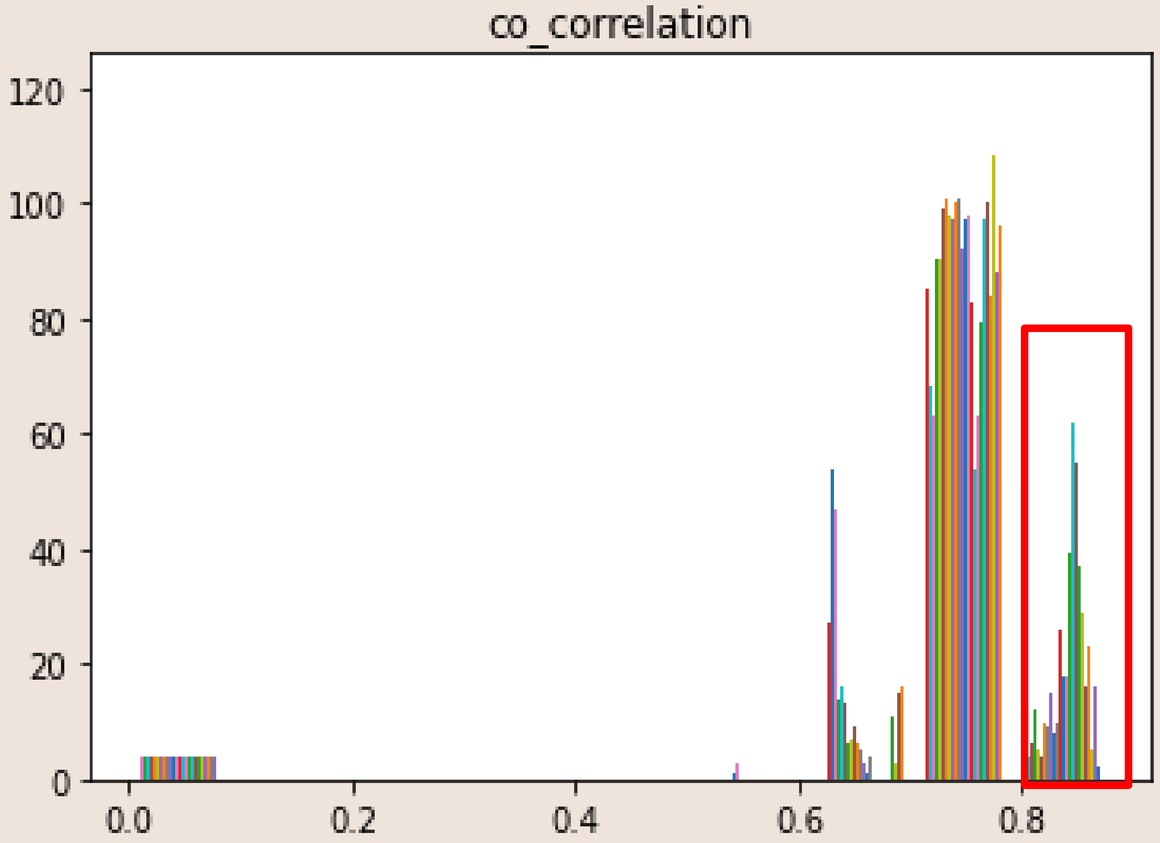




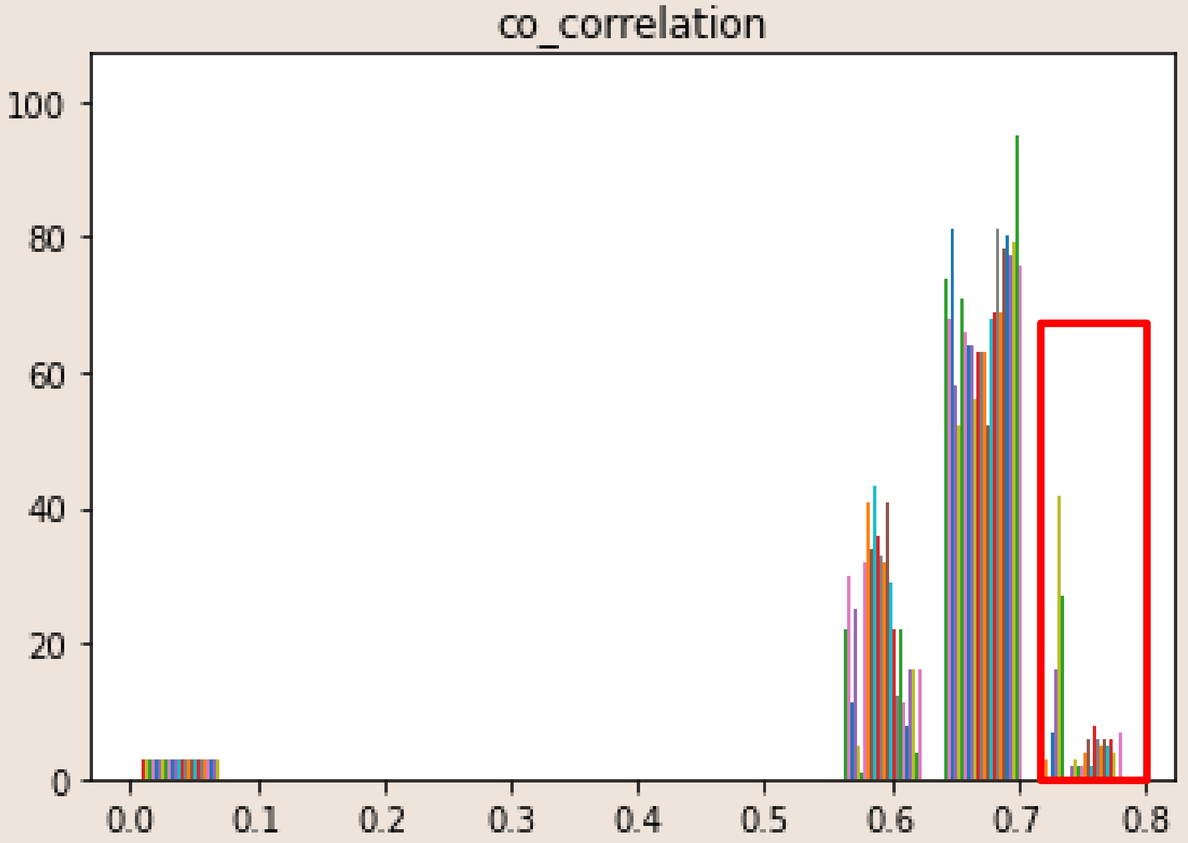
線條瑕疵特徵選擇與門檻

原本是以固定門檻值的方式來判斷，但發現圖片出現雜訊的部分，其數值會有高低落差，若都以固定值判斷會無法找出瑕疵

correlation mean: 0.7123
correlation standard: 0.1483



correlation mean: 0.6217
correlation standard: 0.1578



線條瑕疵特徵選擇與門檻

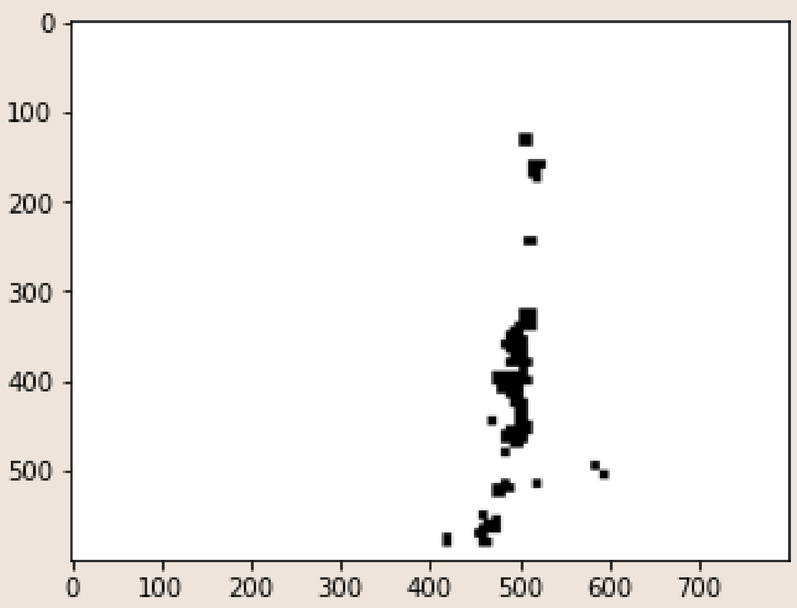
經過不斷的嘗試，最終決定當標準差大於0.15時，門檻值設為correlation的平均加上標準差x0.75，小於則直接將門檻值設為correlation平均加上標準差

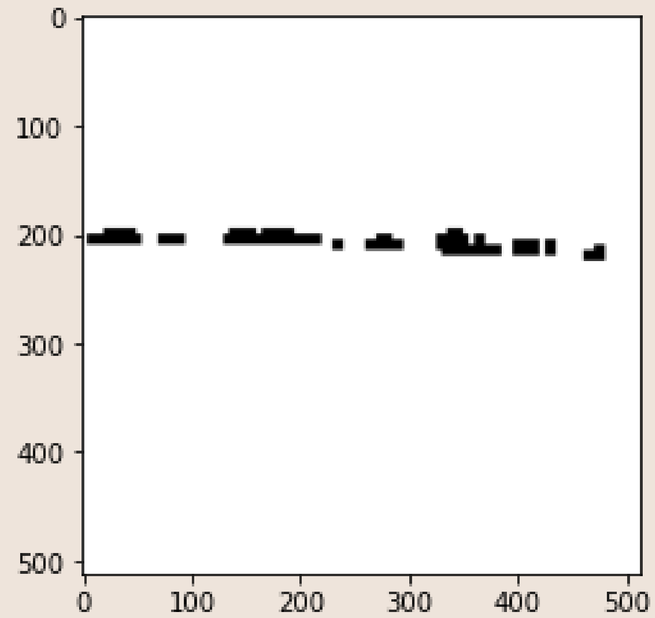
由於線條的瑕疵都是細長型的，所以和圓形破洞相比，使用較小的方格標記即可。

```
for m in range(j*offset,j*offset+10):
    for n in range(i*offset,i*offset+10):
        try:
            detection[n][m] = 0
        except:
            pass
```

線條瑕疵特徵選擇與門檻

經過不斷的嘗試後，雖然沒有辦法將完整的線條都辨識到，但改善了之前辨識出全黑的結果還要好了，所以最終線條的辨識門檻值就改用correlation的平均加上標準差的做法





4

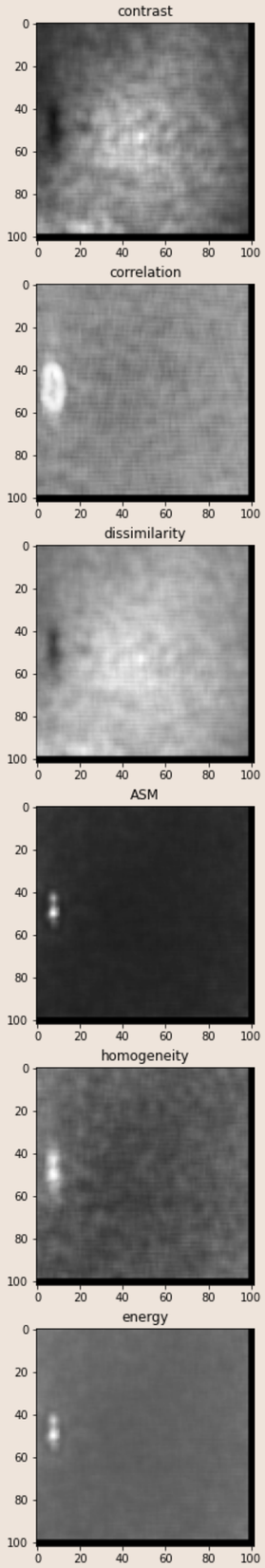
圓形瑕疵
特徵選擇與門檻

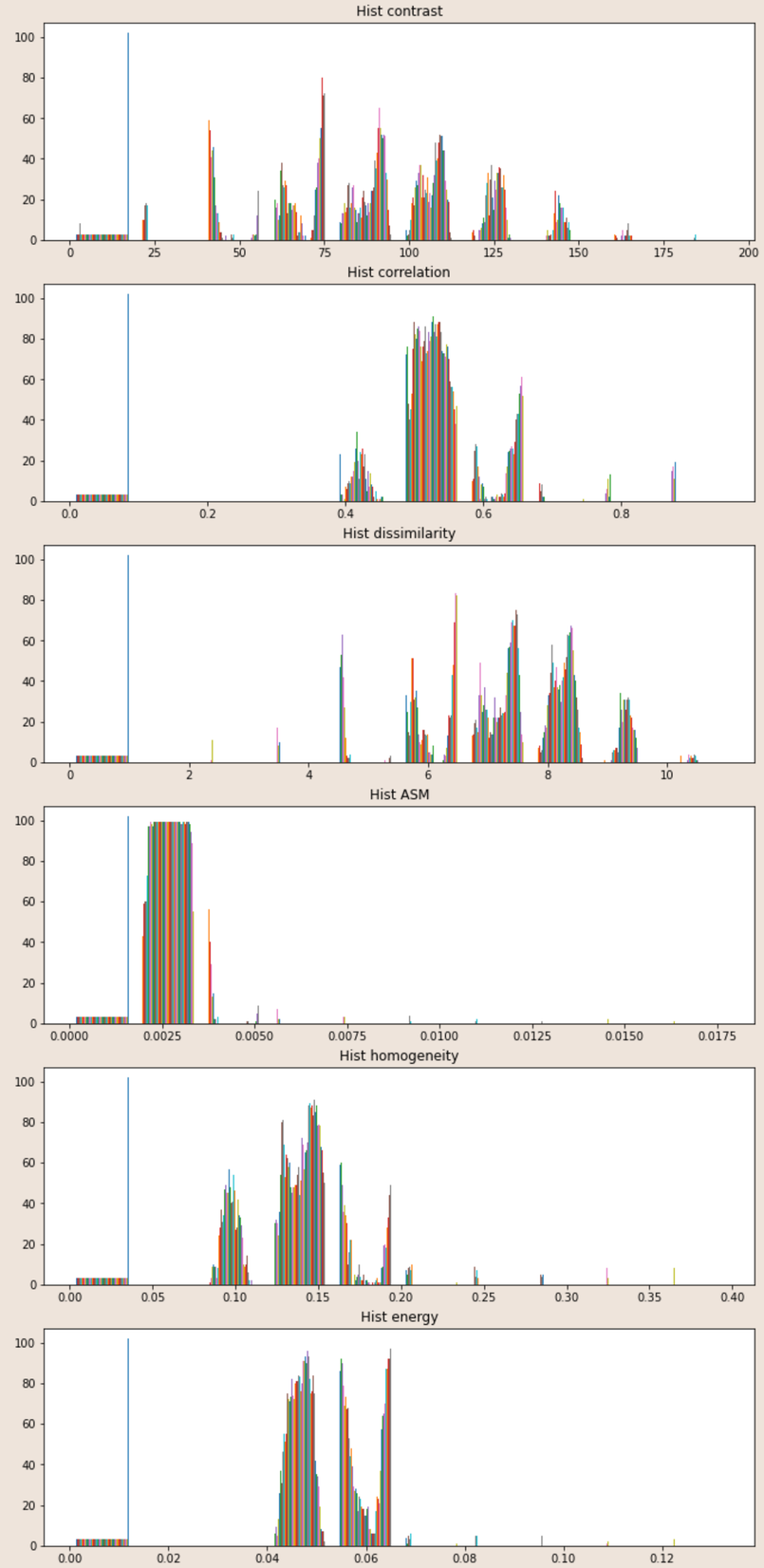
圓形瑕疵特徵選擇與門檻

根據右圖，使用correlation作為主要的門檻值，會比起使用其他方法更有效果，且瑕疵也較能顯著的發現

選用其它的特徵來統計，容易將正常的區塊也混淆，且門檻值也不明顯

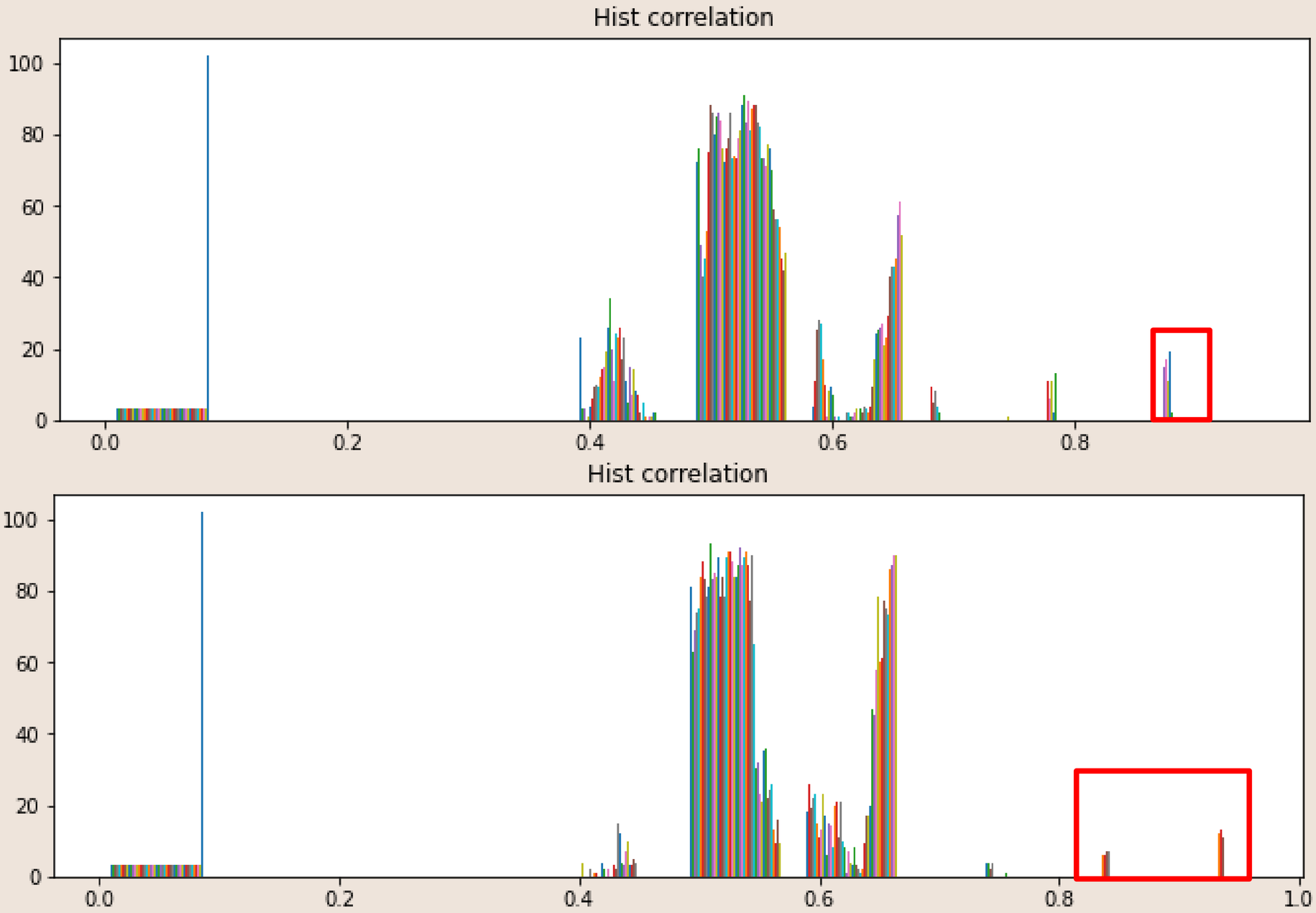
右圖為其它GLCM特徵矩陣的影像和直方圖





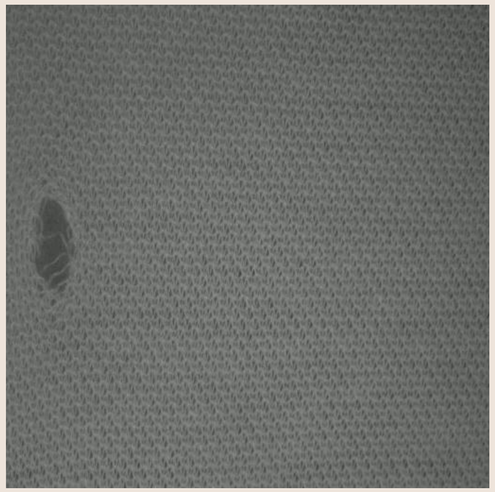
圓形瑕疵特徵選擇與門檻

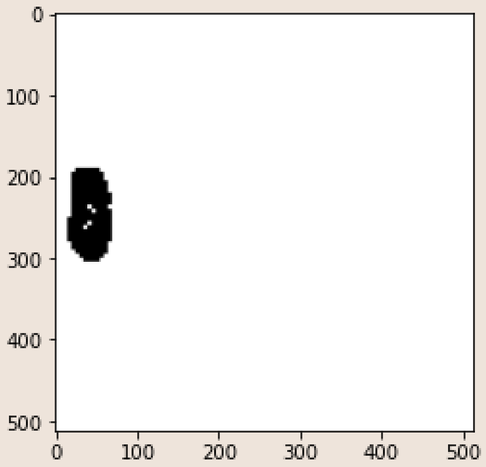
經過不斷的嘗試，門檻值位於0.822上下皆有很好的效果，最終決定將門檻值設為固定的0.822，對於圓形皆能很好的框出瑕疵的部分

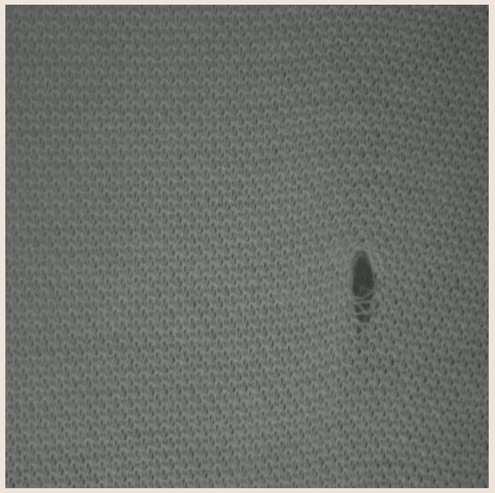


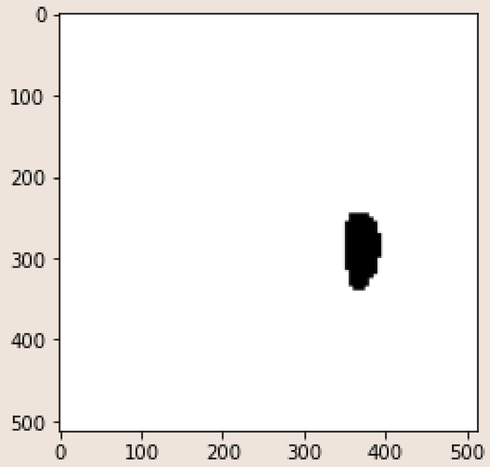
圓形瑕疵特徵選擇與門檻

經過不斷的嘗試，對於圓形皆能很好的框出瑕疵的部分，但有些會有中心無法辨識到的問題發生









5

影片處理

影片處理

原影片為30fps
將影片的每5個fps的圖像抓取出來，以便後續進行影像的瑕疵檢測

```python
#Capture the video
def get_images_from_video(video_name, time_F):
    video_images = []
    vc = cv2.VideoCapture(video_name)
    c = 1

    if vc.isOpened():
        rval, video_frame = vc.read()
    else:
        rval = False

    while rval:
        rval, video_frame = vc.read()

        if(c % time_F == 0):
            video_images.append(video_frame)
        c = c + 1
    vc.release()

    return video_images

time_F = 5
video_name = 'texture_video.avi'
video_images = get_images_from_video(video_name, time_F)
```

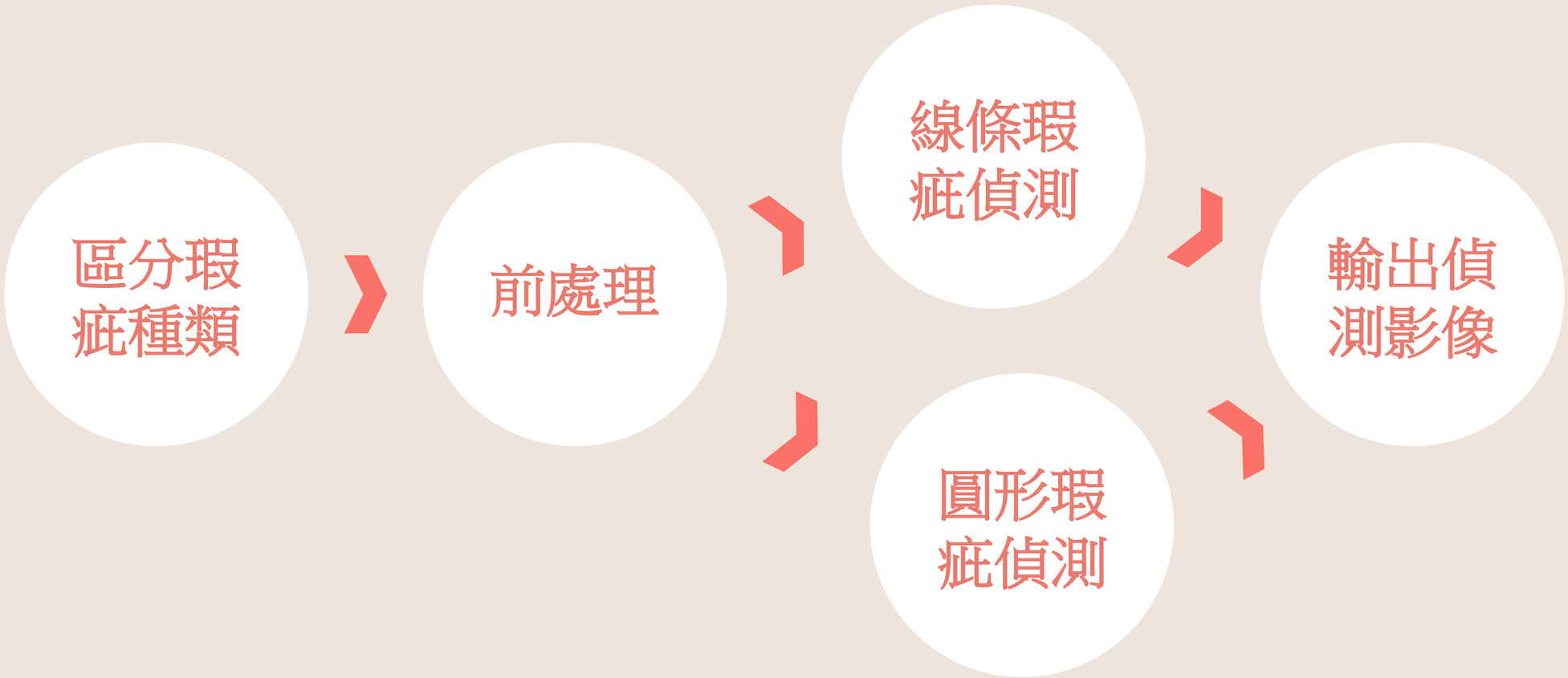
影片處理

將抓取圖片數字前方補0，使圖片能按照順序排列，避免出現順序為 1,10,100 的情形發生

```
for i in range(0, len(video_images)):
    if i < 10:
        num = "000" + str(i+1)
    elif i < 100:
        num = "00" + str(i+1)
    elif i < 1000:
        num = "0" + str(i+1)
    else:
        num = str(i+1)
    cv2.imwrite( "./video/img"+num+".jpg", video_images[i])
```

影片處理

影像瑕疵偵測，方法步驟和前面使用相同



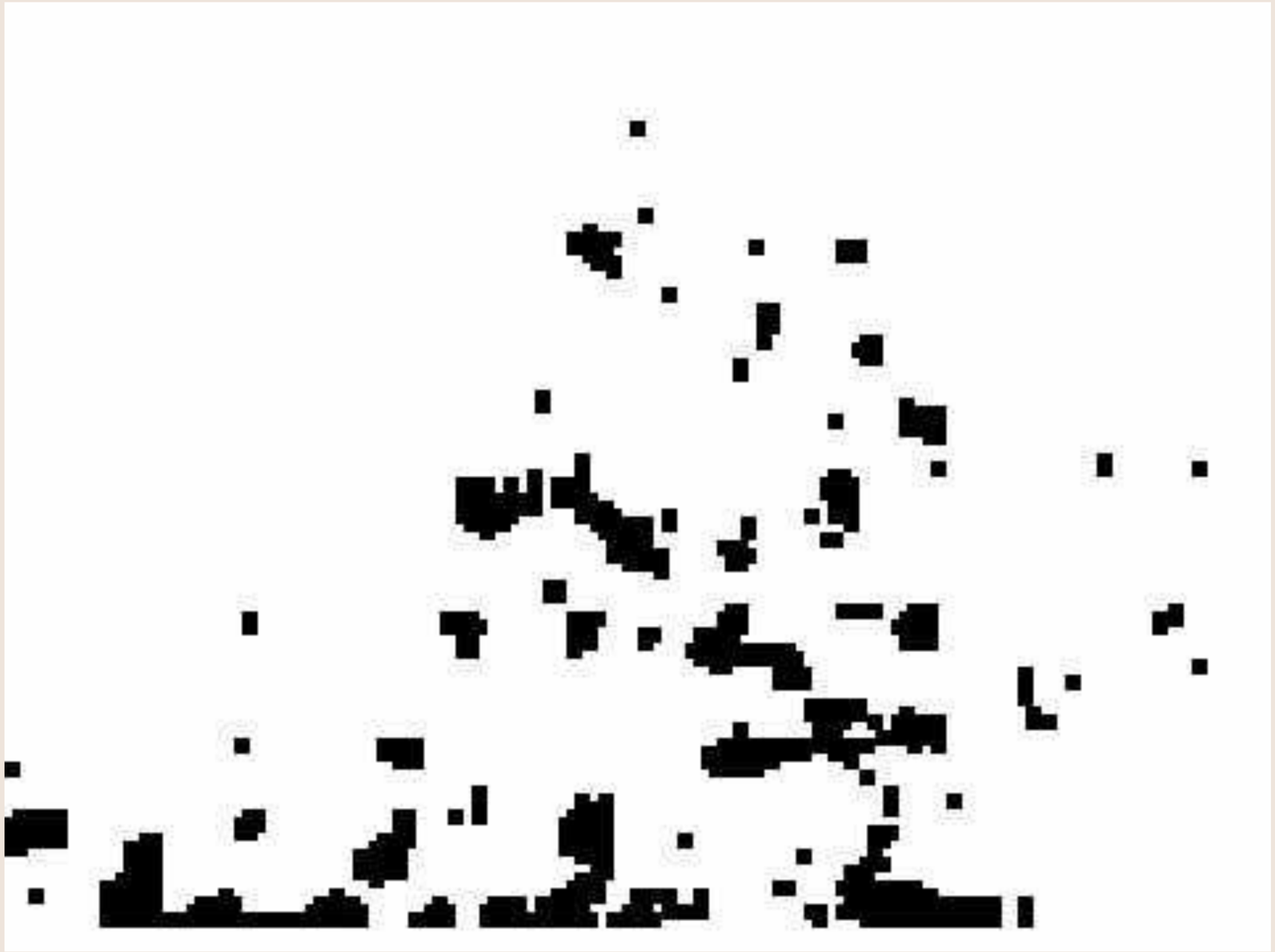
影片處理

將圖片影像轉換成影片

```
#image to video
path = "./result/"
filelist = os.listdir(path)
fps = 2
size = (800, 600)

#使用 XVID 編碼
video = cv2.VideoWriter("VideoTest1.avi",
        cv2.VideoWriter_fourcc('X', 'V', 'I', 'D'), fps, size)

for item in filelist:
    item = path + item
    img = cv2.imread(item)
    video.write(img)

video.release()
cv2.destroyAllWindows()
```

影片處理

範例輸出影片 擷取第300-360張的影像為例



Thank You