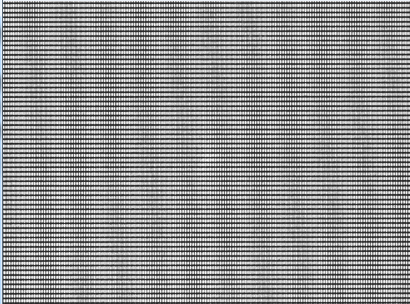
想法（演算法）

構思是以人眼視覺判定，再轉為程式，此做法較為直觀、好理解，且除錯較簡單。

原始圖片有許多的線條，剛開始是對不消去的原圖做偵測，直接做偵測，但每條線條的灰階值不一，甚至有些圖片的線條不清楚，或根本沒有，常常偵測時，過度敏感，使大部分的偵測產生奇怪的結果。



▲同屬瑕疵圖片，左邊線條不明顯，右邊則明顯

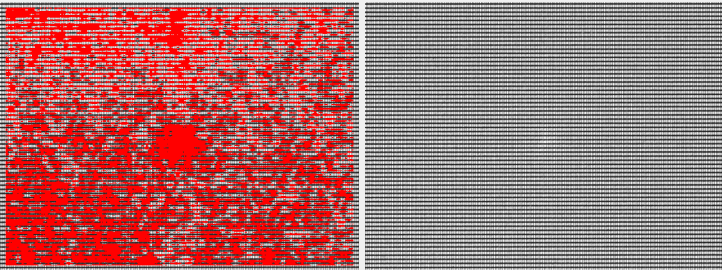
若不將線條去除，演算法會變的比不做邊緣模糊來的複雜許多，為模糊邊緣以達到消除原圖線條，使用了高斯消去法最佳，以5做核心，因只有一次，線條還是很明顯，所以重複做十次



▲同屬瑕疵圖片，兩張圖片變的相似，不像上面差異大

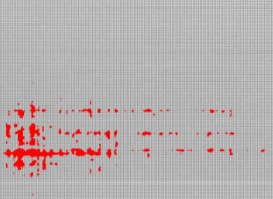
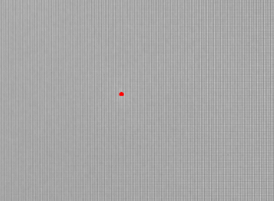
原先使用的方法是以第一個值做參考就將該值列為參考，再與下值做比較，同樣使用Value做變化容許值。若在可容許範圍，改為該值做參考，在往下比較，以此類推。

這方法會被瑕疵周邊的變化量影響，也使變化容許值的範圍減少，無法彈性使用其他演算法，且每張圖的容許變化量無法統一，使偵測的演算法變複雜。



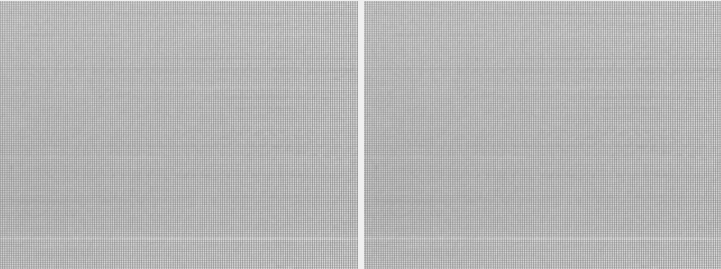
▲瑕疵位於正中間，但是非常敏感，連非瑕疵也偵測到

後改良為只取第一點做參考，此方法無法使"下方白線"與汙3指出正確的瑕疵點，反而偵測到非瑕疵的地方。也會使偵測的演算法變的複雜。



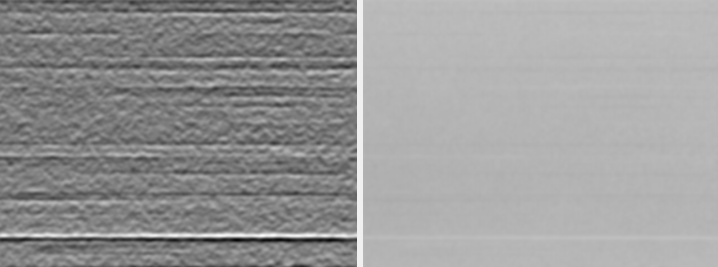
▲取第一點對其他圖OK ，在下方白線無法使用

最後使用現在的方法，將每行平均取值，單用此方法，效果已上述一樣，下方白線無法偵測，但此法可擴大變化容許值，可搭配使用其他運算函數，達到偵測功能。

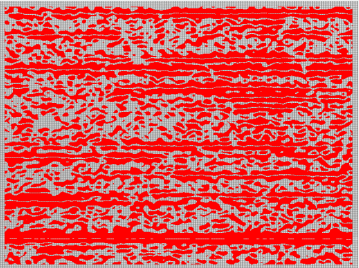


▲單單使用此方法，連紅點都沒出現

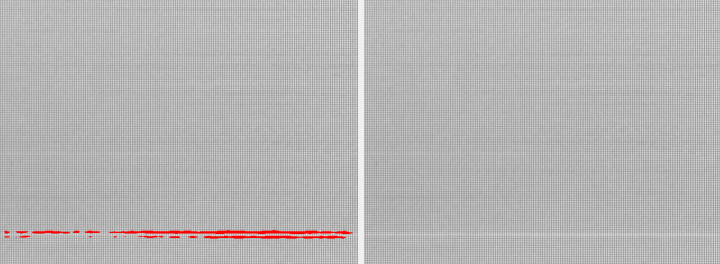
下方白線會無法偵測到正常的瑕疵點，因為此瑕疵點與其他的圖片顏色相近，想以求邊緣的演算法進行過濾，為使附近的顏色有明顯的區別，使用Sobel 運算子，明顯的加大變化量，在使用該方法後，因Sobel運算子會使圖片顏色變化極大，原本預設的變化容許值會無法正常運作，所以將容許值放大8倍，方可找到正確的瑕疵點。



▲左Sobel(0, 1, 25)，右未做

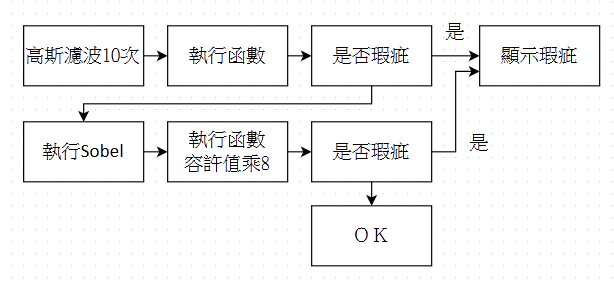


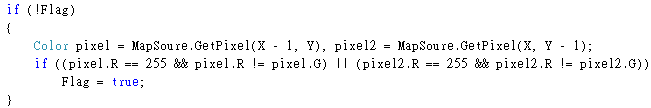
▲Sobel後，直接使用方法



▲放大容許值，再使用方法

過程





因每張圖片一定會有紅點，但瑕疵的紅點是相黏的，以此條件式決定是否有這些紅點是瑕疵。