

我將 HW1 主程式放在 HW1.py，我自己寫的 function 寫在 function.py 供主程式匯入使用。執行 HW1.py 後會依序出現各題的答案畫面，執行後會先出現問題 1 的畫面按下 enter 後再顯示問題 2 的畫面，直到第 5 題(b)結束。

以下是我 python 環境安裝的模組版本

matplotlib==3.7.1
numpy==1.24.2
opencv-python==4.7.0.72

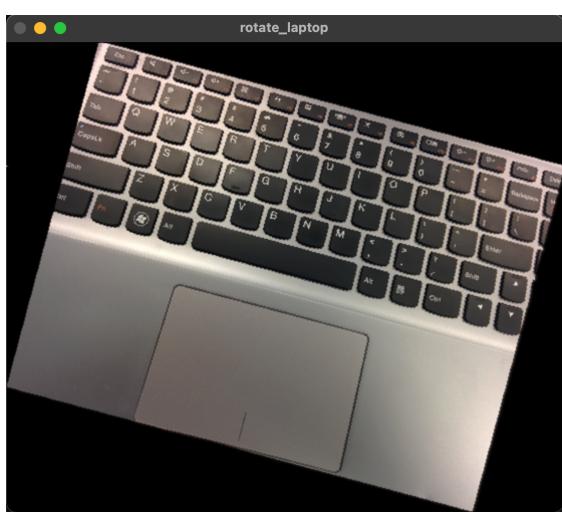
第一題

將 laptop_left.bmp 與 laptop_right.bmp 合併後圖片如下。



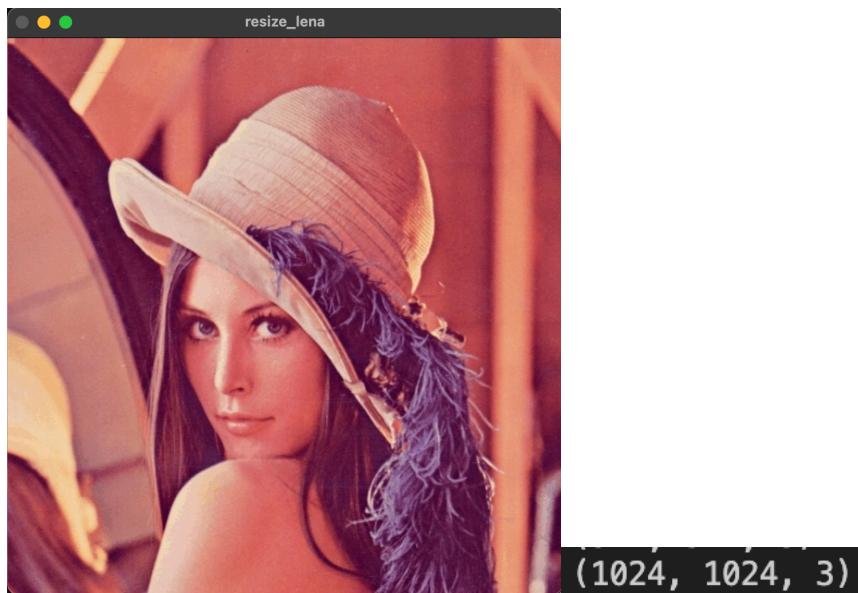
第二題

將第一題結果旋轉 15 度。我自己寫的 rotate function 在 function.py 中。我先算出旋轉後所需顯示完整圖像的畫面大小創建零矩陣。利用公式算出旋轉後像素點所在位置並貼入新的畫面中，之後利用像素四周的平均值填補旋轉後出現的空洞。
輸出畫面如下。



第三題

利用 bilinear interpolation 放大 lena.bmp 到 1024x1024。我自己寫的 bilinear_resize function 在 function.py 中。因為我的電腦畫面無法顯示 1024x1024 大小的圖像所以我將放大後的圖像大小 print 出來。輸出畫面如下圖，左圖是放大後的圖片，右圖是 python 列印出的圖片尺寸。



第四題

將 graveler.bmp 圖像去背並貼在放大後的 lena 圖上。我自己寫的 overlay function 在 function.py 中。輸出畫面如下圖。

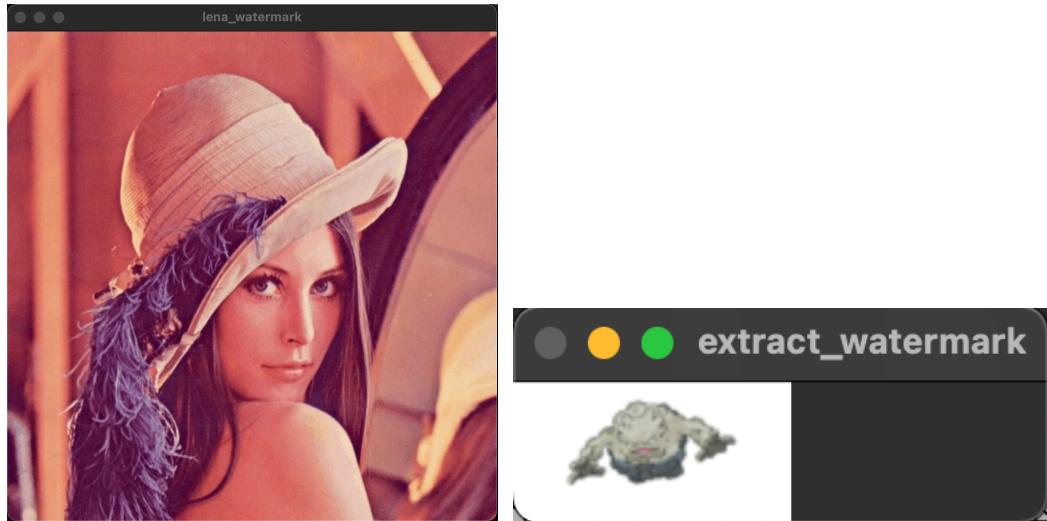


第五題

(a)

使用水印技術將 graveler.bmp 嵌入翻轉的 lena 圖中。我利用 LSB 方法將 graveler.bmp 嵌入翻轉的 lena 圖中。為了保存完整的浮水印圖像我將 graveler.bmp 各個通道的像素點數值轉換成 8bit 二進制數，值並將所有 8bit 數值依序替換 lena 圖 8bit 最低有效位元中，直到 graveler 圖片所有數值插入後停止。為了在提取圖片時知道圖像大小我將 graveler 圖片的高和寬也轉換成 8bit 在最開始時優先藏入

lena 圖最低有效位元中，這可以讓我在提取浮水印時優先知道浮水印大小的資訊，使我知道要提取多少資料。LSB_encode function 是我自己寫的將浮水印插入的函數，LSB_Decode 是我自己寫的將浮水印提出的函數，這兩個函數在 function.py 中。輸出畫面如下，左圖是將 lena 圖翻轉後 LSB 插入浮水印的圖片，右圖是浮水印提出來的畫面。



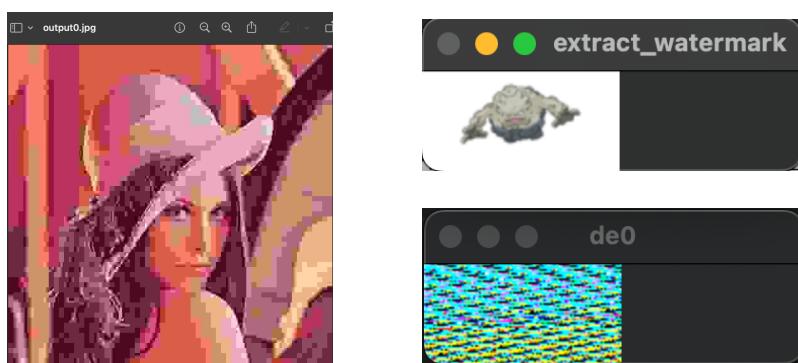
(b)

用 JPEG 標準使用 3 個不同的比率對帶有水印的影象進行編碼，然後進行解碼。檢查是否可以使用客觀品質指標 PSNR 從解碼影象中檢索水印。

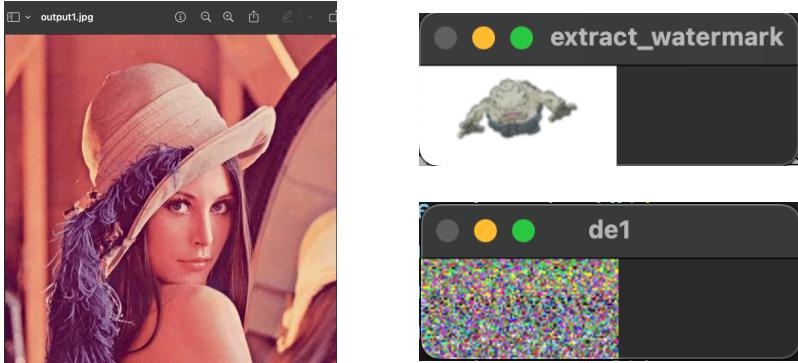
我將嵌入浮水印的圖片以 jpeg 壓縮，壓縮質量分別為 1、50、100，分別存為 output0、output1、output2.jpg 檔。

由於 JPEG 是有損壓縮的關係，經過壓縮後我藏入圖片的浮水印高和寬的資訊也被破壞因此無法使用原來的方式提出浮水印。為了可以提出浮水印我重寫一個提出浮水印的解碼器 LSB_Decode_jpeg，這個解碼器與先前不同的是我直接給定浮水印大小使他可以按照原來邏輯的方式抽出浮水印。抽出後的圖片與浮水印原圖帶入 PSNR 對比計算客觀品值。結果如下。

左圖為 output0.jpg 經過 jpeg 壓縮質量為 1 壓縮後的圖片，右圖上是原浮水印圖片，右圖下是 de0 為從 output0.jpg 抽取出的浮水印。PSNR 為 31.75629，可以看出右圖上及右圖下從視覺方面已經無法看出是相同的圖片。



左圖為 output1.jpg 經過 jpeg 壓縮質量為 50 壓縮後的圖片，右圖上是原浮水印圖片，右圖下是 de1 為從 output1.jpg 抽取出的浮水印。PSNR 為 28.44679，可以看出右圖上及右圖下從視覺方面已經無法看出是相同的圖片。



左圖為 output2.jpg 經過 jpeg 壓縮質量為 100 壓縮後的圖片。因為 JPEG 屬於有損壓縮，即使壓縮質量設為 100 還是會對圖片進行破壞，對比 lena.bmp 及 output2.jpg 檔案大小也從 791KB 將為 131KB。右圖上是原浮水印圖片，右圖下是 de2 為從 output2.jpg 抽取出的浮水印。PSNR 為 27.93592，可以看出右圖上及右圖下從視覺方面已經無法看出是相同的圖片。

