Rendering Techniques (6650)

Assignment 09: Final Term Report

授課教師：王宗銘

系級：資訊工程所

學號：7112056111

姓名：黃照恩 Chao-En Huang

西元 二零二三 年 十二 月 二十八 日

目錄

目錄 i

課程作業簡述 1

最滿意作業評述 2

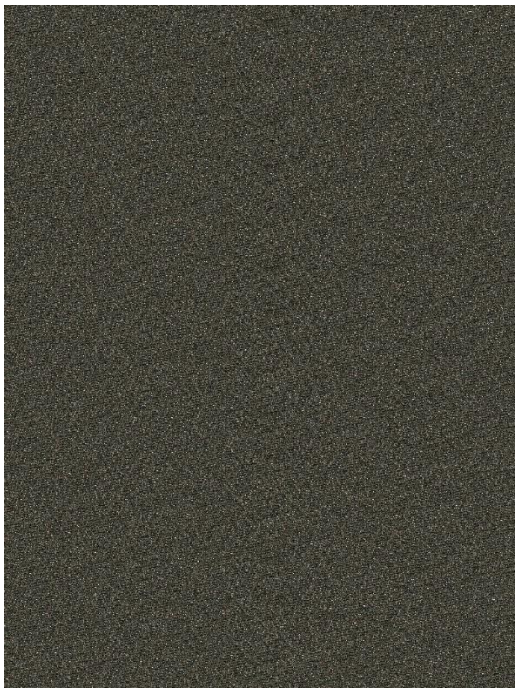
所有作業評述 3

課程整體評述 5

課程作業簡述

- **Assign01: Image Encryption by 2D EAT and RP**
  利用 2D-EAT + Durstenfeld 的 Random Permutation 對影像加密，並使用 2D-EAT 的 Reverse Matrix 以及 Durstenfeld 的 Reverse Random Permutation 對影像作解密。
- **Assign02: Image Encryption Using Enhanced Two-dimensional Sine Logistic Modulation (2D-SLM)**
  加入 pixel scrambling，利用初始值，對第一個 pixel 的 R、G、B，分別一個串一個加密，再利用下一個 pixel 的 R、G、B 將圖像每個 pixel 做加密。
- **Assign03: Determine the Rectangular Transformation Matrix**
  計算 legal transformation Matrix 在給定 M、N 數值，a、b、c、d 範圍內符合結果，並判斷是否符合條件，以及計算幾個 period 會回到原本的圖片。
- **Assign04: Determine the Period from the Matrix Coefficients**
  計算 Rectangular Transformation 在給定 M、N、a、b、c、d 下，判斷條件確認是否是 RT Matrix。
- **Assign05: Inverse Rectangular Transformation**
  利用 Rectangular Transformation 對影像加密，並使用 Inverse Rectangular Transformation 對影像進行解密， RT 的計算參數由作業 4 算出。
- **Assign06: General Weighted Modulus Data Hiding**
  使用 General Weighted Modulus 演算法，在給定 Pixel Alternation Table 下，將秘密訊息嵌入到掩護影像，同時也將訊息從掩護影像中還原。
- **Assign07: General Weighted Modulus Reversible Data Hiding (GMWRDH) Algorithm**
  使用 General Weighted Modulus Reversible Data Hiding 演算法，將秘密訊息嵌入到灰階影像並產生三張灰階影像，並反向取回秘密訊息。
- **Assign08: Integrated Message Embedding and Encryption Algorithm**
  結合 GWMRDH 將秘密訊息嵌入灰階影像，再利用 RT 將 pixel 進行加密，同時也反向解密並取得嵌入的訊息。

最滿意作業評述

Assign05	
作業心得： Rectangular Transformation 的算法比起 EAT，可以處理非正方形的圖像，增強了這個算法的通用性。相比其他計算參數的作業，作業五有圖像輸出，視覺成就感也比較大。此作業也建立在作業 4 之上，需要先計算要用到的參數，讓這份作業有在前一份作業的努力上，更顯出它的成果出來。 在撰寫程式碼中，我了解到每一個 pixel 的移動軌跡，更學習到如何使用 Numpy 的 indices 加速計算每一個位置乘以 Matrix 的算法，讓位置計算速度大幅度增加。 另外原本需要計算 Period 後才能還原圖像的算法，在使用 Inverse Rectangular Transformation 的算法後，可以用相同計算次數就能還原，這種算法讓我覺得很神奇，其中的算法也有很多小數點以及高斯函數的使用，原本預想位置應該是個整數，卻在計算過程中有許多小數，但最後卻能夠還原，最後能實際將論文的方法時做出來，成就感也無比的巨大。	
	
使用 RT 對影像進行 pixel 加密	使用 IRT 將影像還原

所有作業評述

Assign04

收穫心得：

這份作業讓我覺得有趣的點在他是一個純運算的作業，因此完全不需要考慮圖像的輸入和輸出，在撰寫程式碼中，我是以一個 arange 去代表每一個 pixel 的數值，經過每次 RT 運算後，再來判斷是否恢復到原本的 arange 的順序，來判斷圖形是否恢復了。

	A	B	C	D	E	F	G	H
1	No.	M	N	a	b	c	d	periodd
2	1	36	96	1	1	8	1	24
3	2	176	260	1	1	65	2	528
4	3	176	264	1	1	44	1	520
5	4	260	176	1	1	44	1	520
6	5	260	180	1	1	9	2	312
7	6	384	256	1	1	2	1	384
8	7	256	384	1	1	3	2	384
9	8	512	864	1	1	27	2	2304
10	9	864	512	1	1	16	1	1728
11	10	1024	1280	1	1	5	2	768
12	11	1280	1024	1	1	4	1	2560
13	12	2560	3840	1	1	3	2	768

在給定 M、N 大小下，計算符合 RT 的 a、b、c、d 參數，以及他的 period

困難的點 (1/2)：

但這份作業因為是一個純運算的作業，在自己的電腦上跑已經很吃力，當圖像大小來到 2560 * 3840 的時候，基本上已經運算不太動，因此開始優化程式的算法。

首先我觀察到 python 在計算每個 pixel 的時候，我使用 for 迴圈，這樣的效率非常差，因此發現 Numpy 的 indices 算法，可以在一行裡面計算完所有位置的矩陣乘法，這樣的算法效率有大幅度的上升，也加速每一次算 RT 的時間，但仍然遇到計算 RT 需要 1/2 * 2560 * 3840 次，依然太慢。

```python
def do_rectangular_transformations(M: int, N: int, a:
    encrypt_image = np.zeros_like(image)
    MN = np.array([M, N])
    A = np.array([[a, b], [c, d]])
    for y, x in np.ndindex(image.shape[:2]):
        xy = np.array([x, y])
        xy_prime = np.remainder(np.matmul(A, xy), MN)
        x_prime, y_prime = xy_prime[0], xy_prime[1]
        encrypt_image[y_prime][x_prime] = image[y][x]
    return encrypt_image
```

使用 for 迴圈

```python
def do_rectangular_transformations(M: int, N: int, a:
    encrypt_image = np.full((M, N), -1)
    MN = np.array([M, N])
    A = np.array([[a, b], [c, d]])
    row_indices, col_indices = np.indices(image.shape
    coordinates = np.stack((row_indices, col_indices)
    new_coordinates = np.remainder(np.matmul(coordina
    encrypt_image[new_coordinates[ ... , 0], new_coordi
    return encrypt_image
```

使用 Numpy 的 indices

困難的點 (2/2)：

考慮計算 RT 需要 1/2 * 2560 * 3840 次依然太慢，因此從計算 RT 的地方下手，觀察每個參數計算 RT，發現 RT 做完一次後，有些 pixel 會重複選到一樣的位置，也說明有些位置不會被選到，基於這個原因，我首先將計算後的 RT 結果圖片預設每個 pixel 為-1，然後將原圖計算後附值到結果圖片上，再判斷圖片中是否有 pixel 為-1，如果有就不需要在做下一次的 RT，因為圖像已經遭到破壞，經過這樣的修正，終於可以計算 2560 * 3840 的 RT 參數。

```
def do_rectangular_transformations(M: int
    encrypt_image = np.full((M, N), -1)
    MN = np.array([M, N])
    A = np.array([[a, b], [c, d]])
    row_indices, col_indices = np.indices
    coordinates = np.stack((row_indices,
    new_coordinates = np.remainder(np.mat
    encrypt_image[new_coordinates[ ... , 0]
    return encrypt_image
```

預設每像素為-1，做完 RT 後附值

```
for period in range(1, (M * N) // 2):
    encrypt_image = do_rectangular_tr
        M, N, a, b, c, d, encrypt_ima
    )
    if -1 in encrypt_image:
        break
    if is_the_same_picture(image, enc
        parameters_result.append((a,
        break
```

判斷圖像是否有-1，代表 RT 不完全

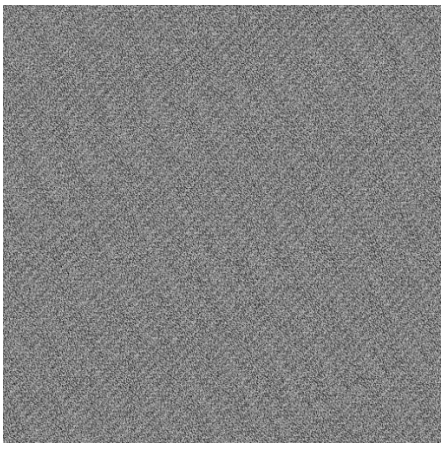
Assign08

收穫心得：

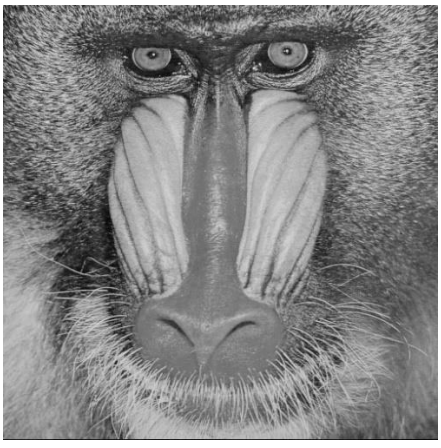
這份作業統整了嵌密演算法和 RT 的加密，加密讓整個嵌密訊息更加安全，另外在實做程式碼上，也結合前後作業做出一個完整的結果，非常有成就感，同時也感嘆原來加密和嵌密的算法其實不難，簡單的運算卻能有很好的保護讓我覺得很意外。

困難的點：

需要將之前作業的程式碼做一個融合，需要考慮每一個運算都要做成函式的形式，同時也考慮到每個功能的泛用性，這點是在前幾個作業沒有想到的，因此在這次作業的實作上，就需要仔細考慮每個運算的過程，也因為運算步驟變多，在除錯上也需要花更多的時間和心力。



經過嵌密訊息和 RT 加密的圖片



還原圖片並取得嵌密訊息

課程整體評述

教師教學之優點與缺點：

老師很細心解釋每個算法的過程，以及來源的論文，並且在上課的時候會詳細的描述整個運算過程，帶著學生走過每個步驟，中途也會確認學生有沒有跟上進度，並且重複針對不懂的地方再解釋一次。此外，也會在上課開始時，討論作業是否有一些問題，會跟同學一起討論有沒有更優的解法，並且在與學生討論時，也能知道老師都有將作業自行跑過一次，真的覺得很用心。

不過也因為上課說明很詳細，有時候聽課的前一小時，無法獲取新的知識，重複複習許多相似的內容，在聽課上要維持專注力會變得很吃力。

課程內容之優點與缺失：

課程安排上我覺得很適合，先從最基礎的方形圖片加密，了解純黑白 pixel 的問題，因此加入 pixel 的雜訊，再到每個 pixel 之間互相串聯加密，並做到位置的加密，並測量混亂程度，最後學習到如何嵌入訊息，這一連串由淺入深的安排我自己非常的喜歡。

在後續的作業中，如果能把測量的方法也一起實做出來，我覺得效果和加密過程會更加有說服力。

建議後續課程改進之事項：

在上課的一開始，可以先上新的課程進度，當天的後半部再來討論之前的作業或做法，比較能維持課程原本的進度，同時也保留學生剛開始上課的專注力。

另外，作業的部分會希望有更多的時間撰寫，有時候上課說會出作業到下次上課，但 ilearning 的時間只有一個禮拜，偶而還會出現新的作業，也不知道還會出多少的量，學生在安排時間上比較困難，因此會希望當天上課完，就能知道到下次上課前要完成多少作業以及他們的截止時間。