

# Introduction to Image Processing

## Lab1 Report

105060016 謝承儒

### 壹.Proj 02-02 :

#### Reducing the Number of Intensity Levels in an Image

##### 一.目標

將原本亮暗程度為 256 個 Level 的圖片，轉換成  $2^n$  個 Level 亮暗程度。  
註.  $n$  為整數， $2^n$  在 2~256 之間。

##### 二.函式

**quantizedImage = reduceIntensityLevel(originalImage, intensityLevel)**

originalImage : 還沒 quantize 的原影像

intensityLevel : reduce 的 intensity level 數量 (2 到 256)

quantizedImage : intensity level quantized 後的影像

##### 三.作法說明

因為先將讀圖片得到的 unit8 數值轉成 double，再接著進行轉換的動作，所以接下來數值的範圍會從 0(黑)~255(白)轉為 0(黑)~1(白)。

(一)把 0~1 分成  $2^n$  個 Level 並找出各 Level 的值

$$\text{LevelValue}(x) = \frac{x}{(n-1)}, x = 0, 1, 2, \dots, n-1$$

按照上面的公式，若分成 4 個 Level，則代表：

表 1 Level 和其對應的值

LevelValue(0)	LevelValue(1)	LevelValue(2)	LevelValue(3)
0	0.3333	0.6667	1









把這些值做成一個  $n \times 1$  的矩陣。

(二)找出 originalImage 的值(簡稱 o 值)轉換後應為多少

用一個 for 迴圈，將 o 值和上面做出的 LevelValue 的表格一個個去比較，找出 o 值和哪個 LevelValue 最相近，該 LevelValue 便是轉換後的值。

四.結果

表 2 各 Level 的圖片

	
Level 2	Level 4
	
Level 8	Level 16
	
Level 32	Level 64
	
Level 128	Level 256

## 五.另外一種做法

(一)把 0~1 分成  $2^n$  個 Level 並找出各 Level 的值

$$\text{LevelValue}(x) = \frac{x}{(n-1)}, x = 0, 1, 2, \dots, n-1$$

按照上面的公式，若分成 4 個 Level，則代表：

表 3 Level 和其對應的值

LevelValue 0	LevelValue 1	LevelValue 2	LevelValue 3
0	0.3333	0.6667	1

把這些值做成一個  $n \times 1$  的矩陣。

(二)找出各 Level 的邊界 LevelEdge

$$\text{LevelEdge}(x) = \frac{x}{n}, x = 1, 2, \dots, n-1$$

按照上面的公式，若分成 4 個 Level，則代表：

表 4 Level 和其對應的邊界

LevelEdge(0)	LevelEdge(1)	LevelEdge(2)
0.25	0.5	0.75

(三)找出 originalImage 的值(簡稱 o 值)轉換後應為多少

用一個 for 迴圈，將 o 值和上面做出的 LevelEdge 的表格一個個去比較，找出 o 值落在哪兩個 edge 之間，若比 LevelEdge(0)，其新值為 LevelValue(0)；若在 LevelEdge(0)~LevelEdge(1)，新值為 LevelValue(1)，依此類推下去給予新值。

六.兩做法結果的差異

表 5 4-Level 比較圖

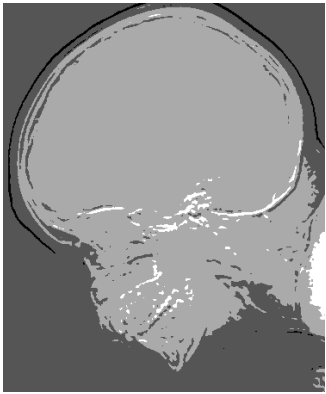
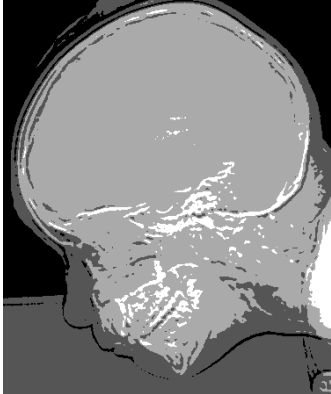
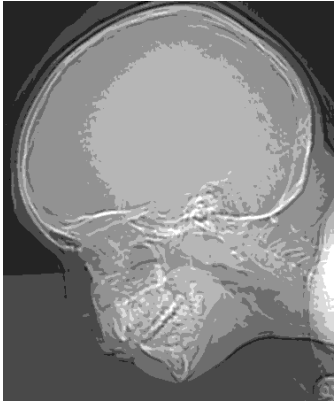
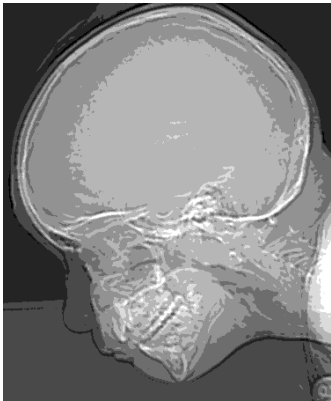
	
前者的做法	後者的做法

表 6 8-Level 比較圖

	
前者的做法	後者的做法

基本上 2-Level 沒有差別，在 4-Level 時差異最為明顯，而到 8-Level 時差異只剩少部分，再更上去的 Level 基本上已經沒有差別。  
前者的做法可以說是一種四捨五入，而後者的做法相當於無條件捨去，所以後者的圖片會比前者有更強烈的對比。

## 貳.Proj 02-03 :

### Zooming and Shrinking Images by Pixel Replication

#### 一.目標

將目標圖片縮小  $n$  倍後，再放大  $n$  倍回復原本大小，方法使用 Pixel Replication。

#### 二.函式

**resizedImage = resizeImage\_replication(originalImage, scalingFactor)**

originalImage : 還沒 resize 的原影像

scalingFactor : resize 的 scale (例如要放大為兩倍: 2 ; 縮小為一半: 1/2)

resizedImage : resize 後的影像

#### 三.做法說明

為先將讀圖片得到的 unit8 數值轉成 double，再接著進行轉換的動作，所以接下來數值的範圍會從 0(黑)~255(白)轉為 0(黑)~1(白)。

(一)找出新圖片的 row、col 數量 => 得到 newR、newC

$$\text{newR} = \text{round}(r * \text{scalingFactor})$$

$$\text{newC} = \text{round}(c * \text{scalingFactor})$$

註.r、c 為原圖的 row、col 數量

round(x) 代表對 x 做四捨五入

(二)利用 newR 找出每個新的 row，往下一 row 相當於在原圖的多少

row，col 也做相同處理 => 得到 rLevel、cLevel

$$rLevel = \frac{(r - 1)}{(\text{newR} - 1)}$$

$$cLevel = \frac{(c - 1)}{(\text{newC} - 1)}$$

(三)用雙層迴圈去跑過新圖的每個點，利用 rLevel、cLevel 去判斷新圖的

第  $i$  個 row、 $j$  個 col，相當於原圖的第  $R$  個 row、第  $C$  個 col。

$$R = (i - 1) * rLevel + 1$$

$$C = (j - 1) * cLevel + 1$$

(四)利用  $R$  找出在原圖中，最接近  $R$  的原圖整數。同理，利用  $C$  去找最

接近的原圖整數 col => replaceR、replaceC

(五)新圖的( $i, j$ )便是原圖的(replaceR, replaceC)

#### 四.結果

表 7 原圖和新圖之比較(使用 Pixel Replication)

 A black and white photograph of a pocket watch. The watch has a round case with a ring at the top. The dial is white with black markings, including Roman numerals for the hours and several sub-dials. The watch is shown against a plain, light background.	 A black and white photograph of the same pocket watch as in the original image. It is a pixel replication, showing some digital artifacts or 'jagged' edges, particularly around the dial and the ring. The watch is shown against a plain, light background.
原圖	新圖 (縮小 10 再放大 10 倍)

可以很明顯看出有些鋸齒的痕跡。

## 參.Proj 02-04 :

### Zooming and Shrinking Images by Bilinear Interpolation

#### 一.目標

將目標圖片縮小  $n$  倍後，再放大  $n$  倍回復原本大小，方法使用 Bilinear Interpolation。

#### 二.函式

**resizedImage = resizeImage\_bilinear(originalImage, scalingFactor)**

originalImage: 還沒 resize 的原影像

scalingFactor: resize 的 scale (例如要放大為兩倍: 2; 縮小為一半: 1/2)

resizedImage: resize 後的影像

#### 三.做法說明

(一)找出新圖片的 row、col 數量 => 得到 newR、newC

$$\text{newR} = \text{round}(r * \text{scalingFactor})$$

$$\text{newC} = \text{round}(c * \text{scalingFactor})$$

註.r、c 為原圖的 row、col 數量

round(x) 代表對 x 做四捨五入

(二)利用 newR 找出每個新的 row，往下一 row 相當於在原圖的多少

row，col 也做相同處理 => 得到 rLevel、cLevel

$$\text{rLevel} = \frac{(r - 1)}{(\text{newR} - 1)}$$

$$\text{cLevel} = \frac{(c - 1)}{(\text{newC} - 1)}$$

(三)用雙層迴圈去跑過新圖的每個點，利用 rLevel、cLevel 去判斷新圖的

第  $i$  個 row、 $j$  個 col，相當於原圖的第  $R$  個 row、第  $C$  個 col。

$$R = (i - 1) * \text{rLevel} + 1$$

$$C = (j - 1) * \text{cLevel} + 1$$

(四)利用 Bilinear Interpolation 做雙內插法做線性變化，線性函式如下：

$$f(x,y) = a_0 + a_1x + a_2y + a_3xy$$

(五)利用 R、C 找出最接近的原圖 row( $R_{\min}$ 、 $R_{\max}$ )、col( $C_{\min}$ 、 $C_{\max}$ )，把

( $R_{\min}$ 、 $C_{\min}$ )、( $R_{\min}$ 、 $C_{\max}$ )、( $R_{\max}$ 、 $C_{\min}$ )、( $R_{\max}$ 、 $C_{\max}$ )4 點代入

$f(x,y)$ ，解聯立得出  $a_0$ 、 $a_1$ 、 $a_2$ 、 $a_3$ 。

(六) $f(R,C)$ 即為舊圖( $i,j$ )的值。

(七)若 R 或 C 其中之一為整數，只需做非整數的那軸的線性變化。

#### 四.結果



表 8 原圖和新圖之比較(使用 Bilinear Interpolation)

	
原圖	新圖 (縮小 10 再放大 10 倍)

可以很明顯看出有鋸齒的痕跡，但比 Proj02-03 的新圖還要少一點。

下面將兩種方法後的新圖擺在一起比較，為了讓差異更佳明顯，把倍數從 10 倍調升至 20 倍。

表 9 Pixel Replication 和之 Bilinear Interpolation 比較  
(縮小 20 倍再放大 20 倍)

	
Pixel Replication	Bilinear Interpolation

可以明顯感覺出 Pixel Replication 會有明顯的顆粒感，而 Bilinear Interpolation 因為是線性變化，比較會有顏色連續的感覺。