2022 NYCU Digital Image Processing - Homework 1 Report

IEE 陳冠瑋 310510221 / Oct. 3, 2022

1 BMP Format

Bitmap image files (.bmp) that are used to store bitmap digital images. BMP files contain four sections: a file header (14 bytes), a bitmap information header (40 bytes), a color table (4*NumColors bytes), and the pixel data (ImageSize bytes).

The file header contains information about the type, size, and layout of a device-independent bitmap file. The information header specifies the dimensions, compression type, and color format for the bitmap. The color table contains as many as many elements as there are colors in the bitmap. The pixel data is an array of bytes that defines the bitmap bits. These are the actual image data, represented by scan lines of the bitmap, in left-to-right order.

Of these four sections, only the color table information may be optional, depending on the bit depth of the bitmap data.

Name	Size	Offset	Description
Header	**14 bytes**		**BITMAPFILEHEADER**
FileType	2 bytes	0000h	'BM' = 0x4d42
FileSize	4 bytes	0002h	File size in bytes
Reversed	4 bytes	0006h	Unused (=0)
DataOffset	4 bytes	000Ah	Offset from the beginning of the file to the beginning of the bitmap data
InfoHeader	**40 bytes**		**BITMAPINFOHEADER**
Size	4 bytes	000Eh	Size of InfoHeader = 40
Width	4 bytes	0012h	The horizontal width of the bitmap in pixels
Height	4 bytes	0016h	The vertical height of the bitmap in pixels
Planes	2 bytes	001Ah	Number of Planes (=1)
Bits Per Pixel	2 bytes	001Ch	Store color table entry information. 1: 1-bit image (NumColors = 1) 4: 4-bit image (NumColors = 16) 8: 8-bit image (NumColors = 256) 16: 16-bit image (NumColors = 65536) 24: 24-bit image (NumColors = 16M) 32: 32-bit image (NumColors = 4G)
Compression	4 bytes	001Eh	Type of Compression 0: no compression 1: 8bit RLE encoding 2: 4bit RLE encoding 3: bit fields

ImageSize	4 bytes	0022h	(compressed) Size of image Valid to set this =0 if Compression = 0
XpixelsPerM	4 bytes	0026h	Horizontal resolution: Pixel/meter
YpixelsPerM	4 bytes	002Ah	Vertical resolution: Pixel/meter
Color Used	4 bytes	002Eh	Number of actually used colors
Important Colors	4 bytes	0032h	Number of important colors (0 = all)
ColorTable (optional)	**4*NumColors bytes**	**0036h**	**Present only if BitsPerPixel less than 8 colors should be ordered by importance**
Red	1 bytes		Red intensity
Green	1 bytes		Green intensity
Blue	1 bytes		Blue intensity
Reserved	1 bytes		Unused (= 0)
Pixel Data	**ImageSize bytes**		**The image data**

Table 1: Contents of BMP file format

2 Horizontal Flip

To flip the image horizontally means that each row of the image is reversed. So, just scan the source image from the right to left order, and write it to the destination image from the left to the right order. In this way, a horizontal flip can be achieved.

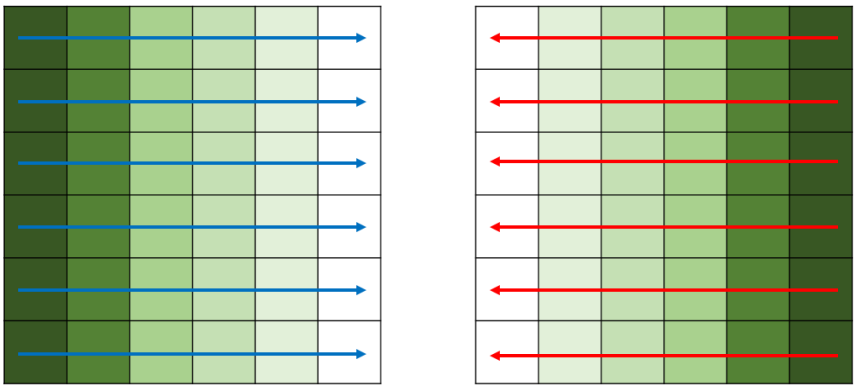


Figure 1: Horizontal flip

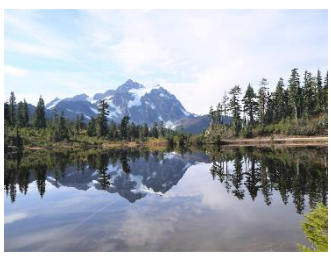
	input1	input2	lena
Source Image			
Horizontal Flip			

Table 2: Result of horizontal flip

3 Resolution Quantization

The intensity range is encoded in 256 levels (8 bits) from 0 to 255 in this work. Resolution quantization reduces the distinct colors used in an image. For the implementation, I define a *quantizer Q(quanBit, src)* **(Figure 2)** which can map the input pixel (*src*) to the quantized output pixel (*dst*) to the specific bits (*quanBit*).

$$dst = Q(quanBit, src) = floor\left(\frac{src}{k}\right) \times k, \qquad where \;\; k = floor\left(\frac{2^8}{2^{quanBit}}\right)$$

For example, if I want to quantize the intensity of pixel = 125 (0111_1101_2) into 2 bits (i.e. 4 levels: **0**~63, **64**~127, **128**~191, **192**~255):

$$k = floor\left(\frac{2^8}{2^2}\right) = 64, \;\; Q(2, 125) = floor\left(\frac{125}{64}\right) \times 64 = 64 = 0010\_0000_2$$

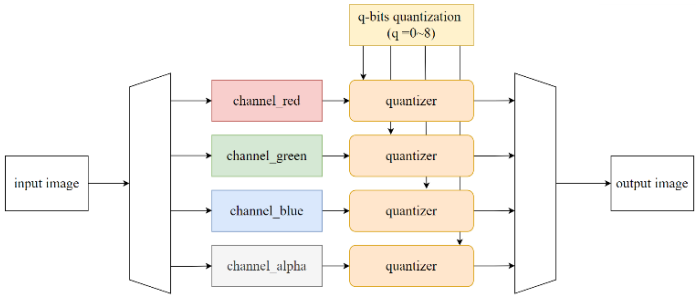


Figure 2: Quantizer

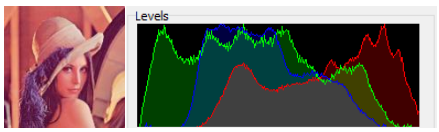
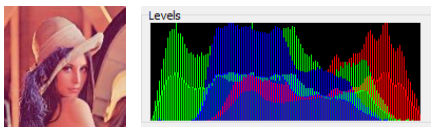
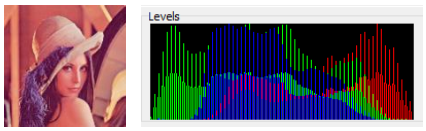
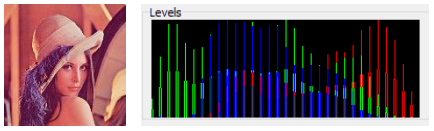
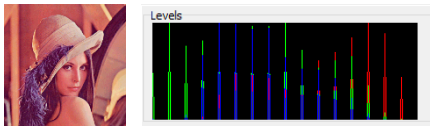
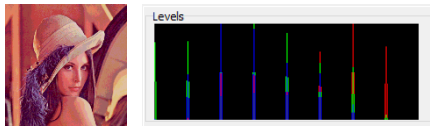
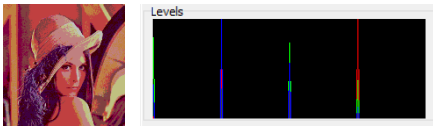
Resolution	input1	input2	lena
quanBit: 8 bits (Source Image)			
quanBit: 7 bits			
quanBit: 6 bits			
quanBit: 5 bits			
quanBit: 4 bits			
quanBit: 3 bits			
quanBit: 2 bits			
quanBit: 1 bit			

Table 3: Result of resolution quantization & intensity distribution

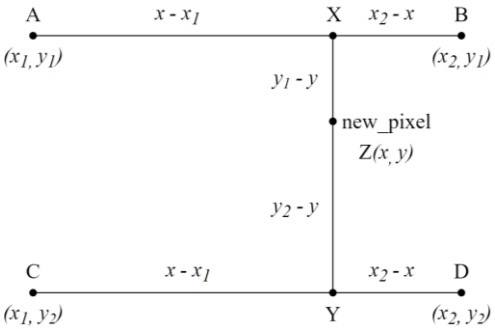
4 Scaling

Scaling an image to a desired spatial dimension is a common operation. There are three common types of scaling algorithms used with digital images.

- *Nearest-Neighbor Interpolation*
- *Bilinear Interpolation*
- *Bicubic Interpolation*

In this practice, we will accomplish up-scaling and down-scaling by bilinear interpolation. Bilinear interpolation is an intuitive algorithm for image scaling. It is a generalization of linear interpolation between two points. The schematic diagram and formula are as follows **(Figure 3)**.

A(x_1, y_1), B(x_2, y_1), C(x_1, y_2) , and D(x_2, y_2) are the points of the original image pixels. The point of the new pixel are new_pixel(x, y). First, I compute the interpolated value of X and Y in the width dimension and then do a linear interpolation between the two interpolated values X and Y in the height dimension. To calculate the interpolation, I need to calculate the weights between the line segments. Weight x (w_x) represents the ratio of the new pixel between the two points on the x-axis, and weight y (w_y) represents the ratio of the new pixel between the two points on the y-axis. Finaly, to select points A, B, C, and D on the original image, we need to map the new_pixel Z(x, y) to the original image. The position of A, B, C, and D are determined by the *ratio* of the interval between the new image and the original image. After getting the scaling ratio, we can calculate the higher bound of x and y and the lower bound of x and y by multiplying the new_pixel Z of the new image by scaling ratio to get a higher bound (x_h, y_h) and lower bound (x_l, y_l).



$$X = A(1 - w_x) + Bw_x$$
$$Y = C(1 - w_x) + Dw_x$$
$$Z = X(1 - w_y) + Yw_y$$
$$= A(1 - w_x)(1 - w_y) + Bw_x(1 - w_y) + C(1 - w_x)w_y + Dw_xw_y$$

$$where\ \ w_x = \frac{x - x_1}{x_2 - x_1}, w_y = \frac{y - y_1}{y_2 - y_1}$$

$$(x_h, y_h) = \left(ceil(ratio_x \times x), ceil(ratio_y \times y)\right)$$
$$(x_l, y_l) = \left(floor(ratio_x \times x), floor(ratio_y \times y)\right)$$

Figure 3: Bilinear interpolation

	Down-Scaling		Original	Up-Scaling	
ratio	**÷ 5**	**÷ 1.5**	**× 1**	**× 1.5**	**× 2**
input1					
input2					

Table 4: Result of down-scaling & up-scaling