Format

教授：連震杰
助教：
機器人實驗室

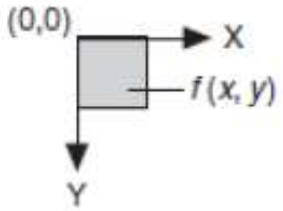
1.0 影像檔格式(Image Format)(1/3)

影像是以二維陣列來表示影像的光影強度大小。影像光強度方程式為

Intensity = grayvalue

$$f(x,y) = I(x,y)$$

x，y 分別代表二維陣列的行(column)與列(row)。



影像光強度方程式

Pixel: Picture element
Voxel: Volume element
Dexel: Depth element

數位影像處理式利用感測器，將影像轉換成離散的像素(pixels)，就是f(x,y)所代表影像感測器(sensor)陣列的一個圖素(pixel)的強度。

1.0 影像檔格式(Image Format)(2/3)

影像與資料型態列表	
影像資料型態	單位 pixel 的 byte 數
1 bit , Line art	1/8 byte per pixel
1/8 byte per pixel	1/8 byte per pixel
8 bit，Grayscale	1 bytes per pixel
24 bit，RGB	3 bytes per pixel
32 Bit，CMYK	4 bytes per pixel
48 bit，RGB	6 bytes per pixel

8 bits = 0~255

CMYK:
C（Cyan）青色 、M（Magenta）洋紅色 、Y（Yellow）黃色、 K（Black）黑色

1.0 影像檔格式(Image Format)(3/3)

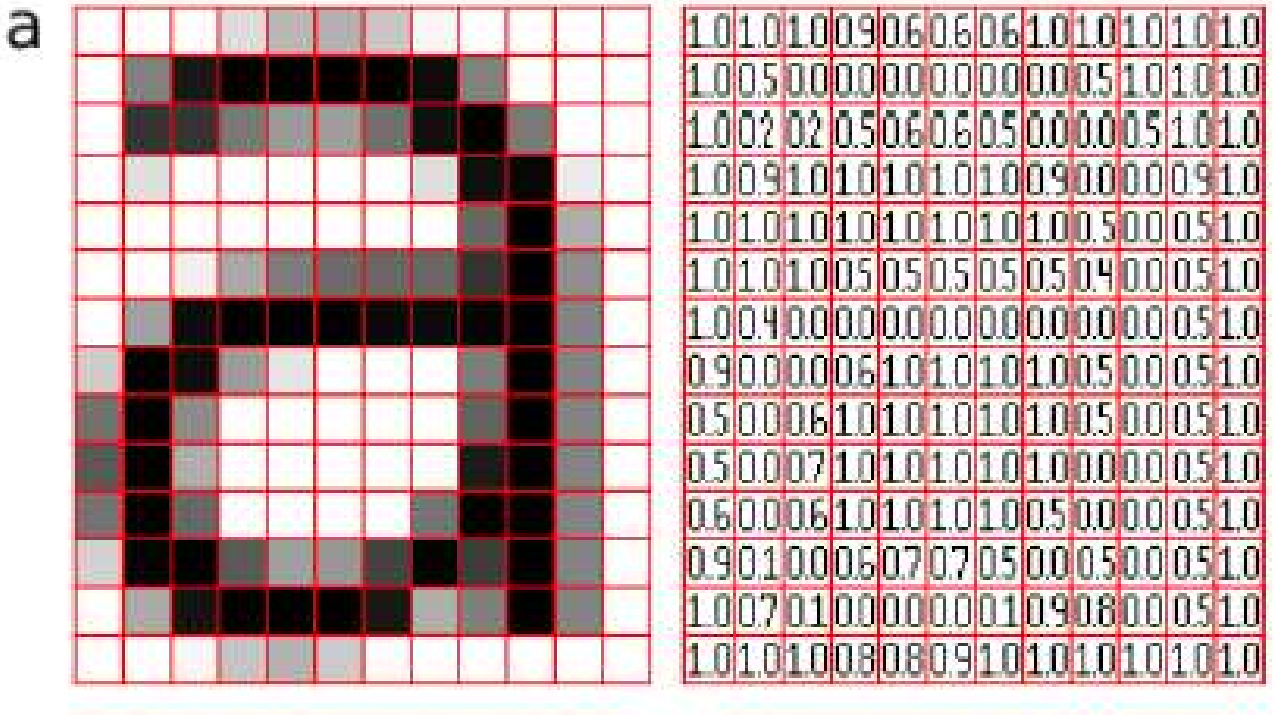
影像檔格式	優點	缺點
1. BMP (Windows Bitmap)	專門用在Windows作業系統，支援RGB全彩	無法壓縮全彩影像，不適用於網路
2. GIF	網頁上最常用，可製作透明圖，交錯圖，動畫，提供非破壞性壓縮(Lossless Compression)，適用遠距傳遞，網頁設計	最多只能存256色 (8 bit, gray value??)
3. JPG/JPEG	高壓縮率 支援RGB全彩，灰階影像	破壞性壓縮，易失真 不支援16色、256色、黑白圖
4. PNG (Portable Network Graphic)	非破壞性壓縮，結合GIF優點，JPEG支援RGB全彩	無法儲存動畫
5. TGA	非破壞性壓縮，支援24位元(RGB)全彩、32位元全彩，用於影像合成，視訊合成，3D動畫製作的影像合成	需藉由專業影像處理軟體解讀
6. TIF (Tagged Image File Format)	跨平台，非破壞性壓縮，支援RGB全彩、16色、256色、灰階、黑白影像	無
7. PCD	洗相片	柯達公司制定的相片光碟格式

1.1 點陣資料格式與取樣 (Raster Image and Sampling)

Raster Image (網格圖檔):
-由掃描機、影像光碟、數位元元像機等設備或影像編輯軟體所產生之圖檔，為印前組頁檔案格式之一種。
-其內容可分為灰階及彩色光點兩種，影像色彩輸出較真實；惟放大後會產生鋸齒狀現象。

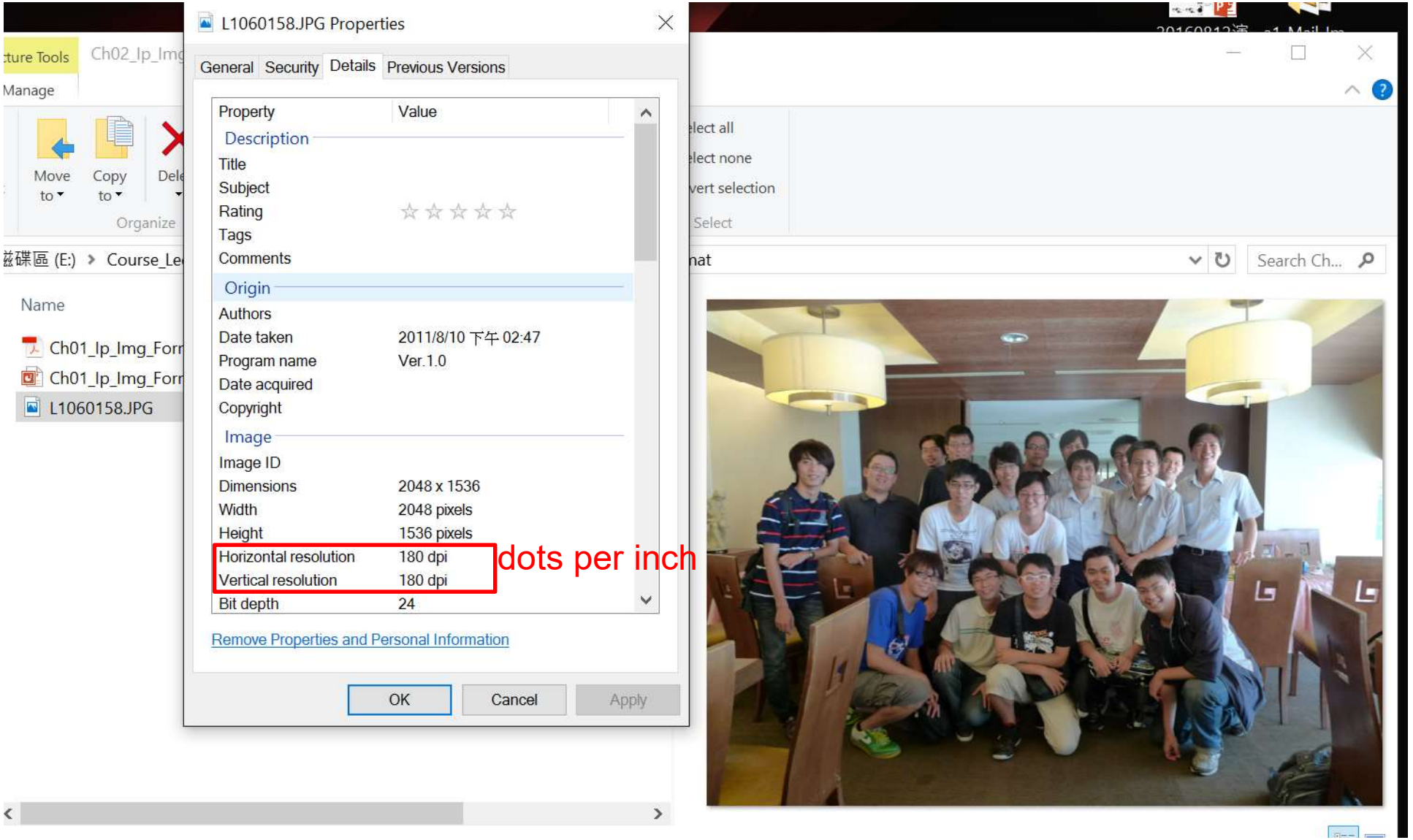
Raster Graphics (點陣圖形):
-與位元影像圖形相同，是利用小點距來顯示出影象圖形，當點距越小，代表它的品質越高。

Gray value:
1.0 = 255 white,
0.0 = 0 black



記憶體影像光強度

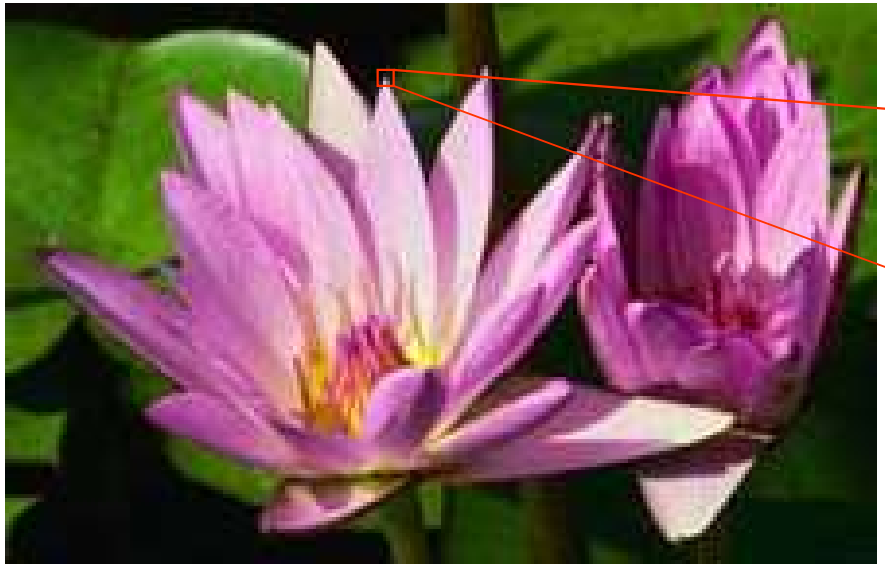
1.1 Image Properity

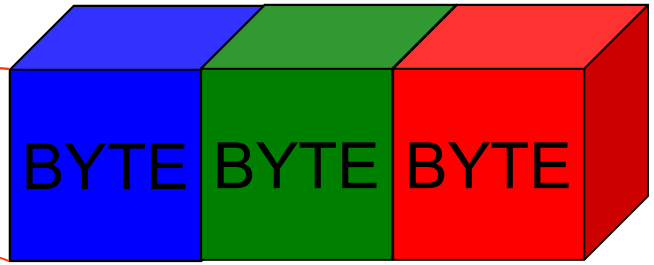


2.0 What is BMP?

- The structure of BMP file was defined by Microsoft (BMP = Bitmap)
- Including black-and-white (1-bit per pixel), gray value (2 , 4 or 8 bits), colorful (8*3=24 or 8*4=32 bits).
- Gray and black-and-white types have palette.
- We only deal with 24-bits BMP.

2.0 24-bit BMP file





1 BYTE=8 bits
=> 0 ~255

0 255

- Check ppt file color map

2.0 BMP file

ASCII Code

```
00000000h: 42 4D B6 49 03 00 00 00 00 00 36 04 00 00 28 00 ; BM貂......6...(.
00000010h: 00 00 15 02 00 00 90 01 00 00 01 00 08 00 00 00 ; ......?........
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 ; ................
00000030h: 00 00 00 01 00 00 00 00 00 00 01 01 01 00 02 02 ; ................
00000040h: 02 00 03 03 03 00 04 04 04 00 05 05 05 00 06 06 ; ................
00000050h: 06 00 07 07 07 00 08 08 08 00 09 09 09 00 0A 0A ; ................
00000060h: 0A 00 0B 0B 0B 00 0C 0C 0C 00 0D 0D 0D 00 0E 0E ; ................
00000070h: 0E 00 0F 0F 0F 00 10 10 10 00 11 11 11 00 12 12 ; ................
00000080h: 12 00 13 13 13 00 14 14 14 00 15 15 15 00 16 16 ; ................
00000090h: 16 00 17 17 17 00 18 18 18 00 19 19 19 00 1A 1A ; ................
000000a0h: 1A 00 1B 1B 1B 00 1C 1C 1C 00 1D 1D 1D 00 1E 1E ; ................
000000b0h: 1E 00 1F 1F 1F 00 20 20 20 00 21 21 21 00 22 22 ; ......   .!!!.""
000000c0h: 22 00 23 23 23 00 24 24 24 00 25 25 25 00 26 26 ; ".###.$$$.%%%.&&
000000d0h: 26 00 27 27 27 00 28 28 28 00 29 29 29 00 2A 2A ; &.'''.(((.))).**
000000e0h: 2A 00 2B 2B 2B 00 2C 2C 2C 00 2D 2D 2D 00 2E 2E ; *.+++.,,,.---...
000000f0h: 2E 00 2F 2F 2F 00 30 30 30 00 31 31 31 00 32 32 ; ..////.000.111.22
00000100h: 32 00 33 33 33 00 34 34 34 00 35 35 35 00 36 36 ; 2.333.444.555.66
00000110h: 36 00 37 37 37 00 38 38 38 00 39 39 39 00 3A 3A ; 6.777.888.999.::
00000120h: 3A 00 3B 3B 3B 00 3C 3C 3C 00 3D 3D 3D 00 3E 3E ; :.;;;.<<<.===.>>
00000130h: 3E 00 3F 3F 3F 00 40 40 40 00 41 41 41 00 42 42 ; >.???.@@@.AAA.BB
00000140h: 42 00 43 43 43 00 44 44 44 00 45 45 45 00 46 46 ; B.CCC.DDD.EEE.FF
00000150h: 46 00 47 47 47 00 48 48 48 00 49 49 49 00 4A 4A ; F.GGG.HHH.III.JJ
00000160h: 4A 00 4B 4B 4B 00 4C 4C 4C 00 4D 4D 4D 00 4E 4E ; J.KKK.LLL.MMM.NN
00000170h: 4E 00 4F 4F 4F 00 50 50 50 00 51 51 51 00 52 52 ; N.OOO.PPP.QQQ.RR
00000180h: 52 00 53 53 53 00 54 54 54 00 55 55 55 00 56 56 ; R.SSS.TTT.UUU.VV
```

1) BITMAPFILEHEADER (Bit Map File Header), 14 Bytes

2) BITMAPINFOHEADER (Bit Map Info Header), 40 Bytes

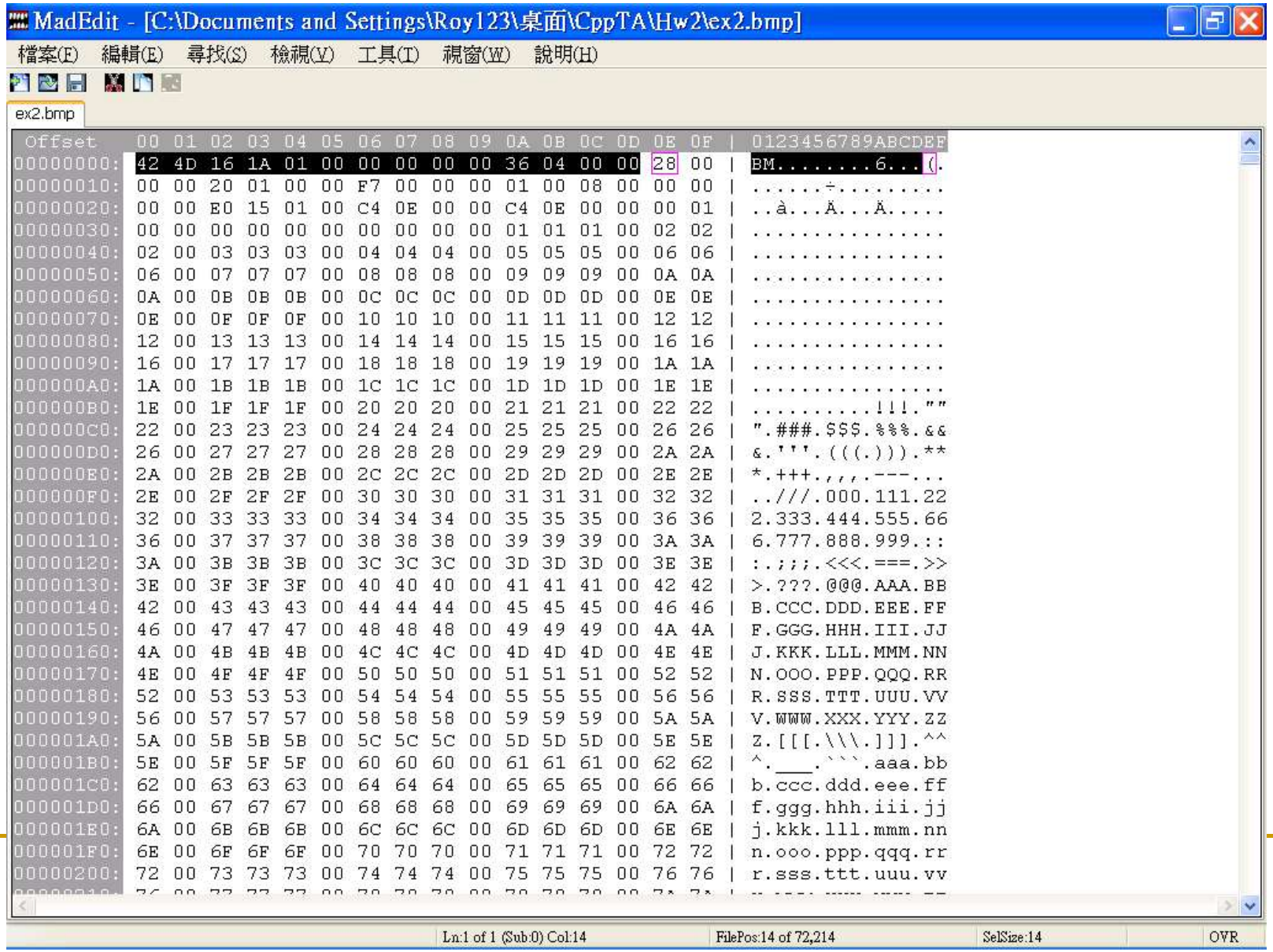
3) RGBQUAD (RGB Quad, palette), 256*(3+1) Bytes

2.0 What's in a BMP file?

1) BITMAPFILEHEADER (Bit Map File Header) //14 bytes
2) BITMAPINFOHEADER (Bit Map Info Header) //40 bytes
3) RGBQUAD * 2^N (RGB Quad) // exp. N=8: 256*(3+1) bytes (in our case)
4) Image Data Array

N (bits): How many bits stand for a pixel (black-and-white (1-bit per pixel), gray value (2 , 4 or 8 bits)).

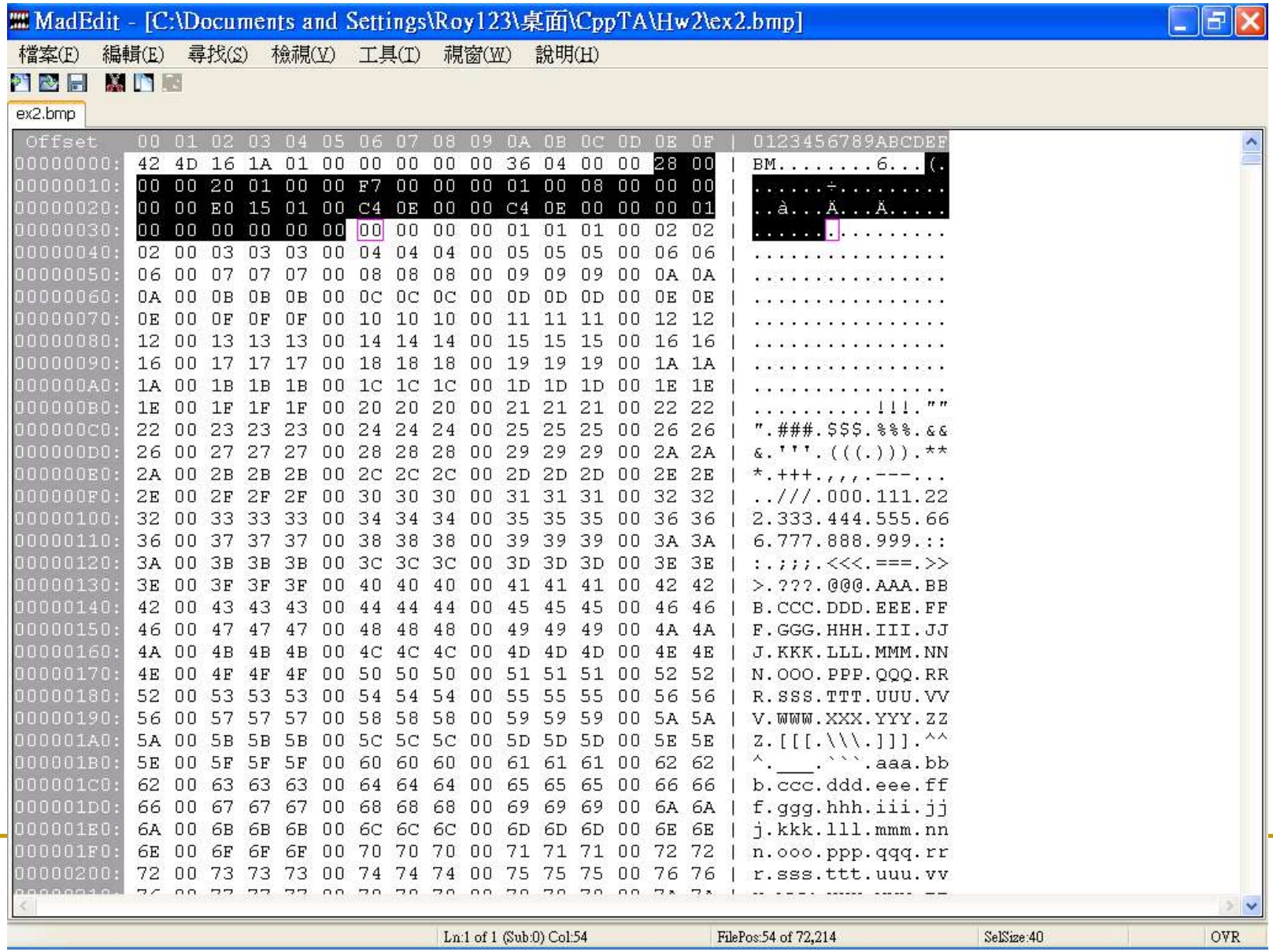
2.1 BITMAPFILEHEADER (1/2)



2.1 BITMAPFILEHEADER (2/2)

```
typedef struct tagBITMAPFILEHEADER {        // (14bytes)
    WORD bfType;                //(2bytes)  File type, in BMP
                                //          case, it'll be
                                //          'BM'(0x424D)
    DWORD bfSize;               //(4bytes)  BMP file size
    WORD bfReserved1;           //(2bytes)  Always 0
    WORD bfReserved2;           //(2bytes)  Always 0
    DWORD bfOffbytes;           //(4bytes)  Size of
                                //          Headers (14+40
                                //          bytes) + Palette,
                                //          14+40 in our case
} BITMAPFILEHEADER;
```

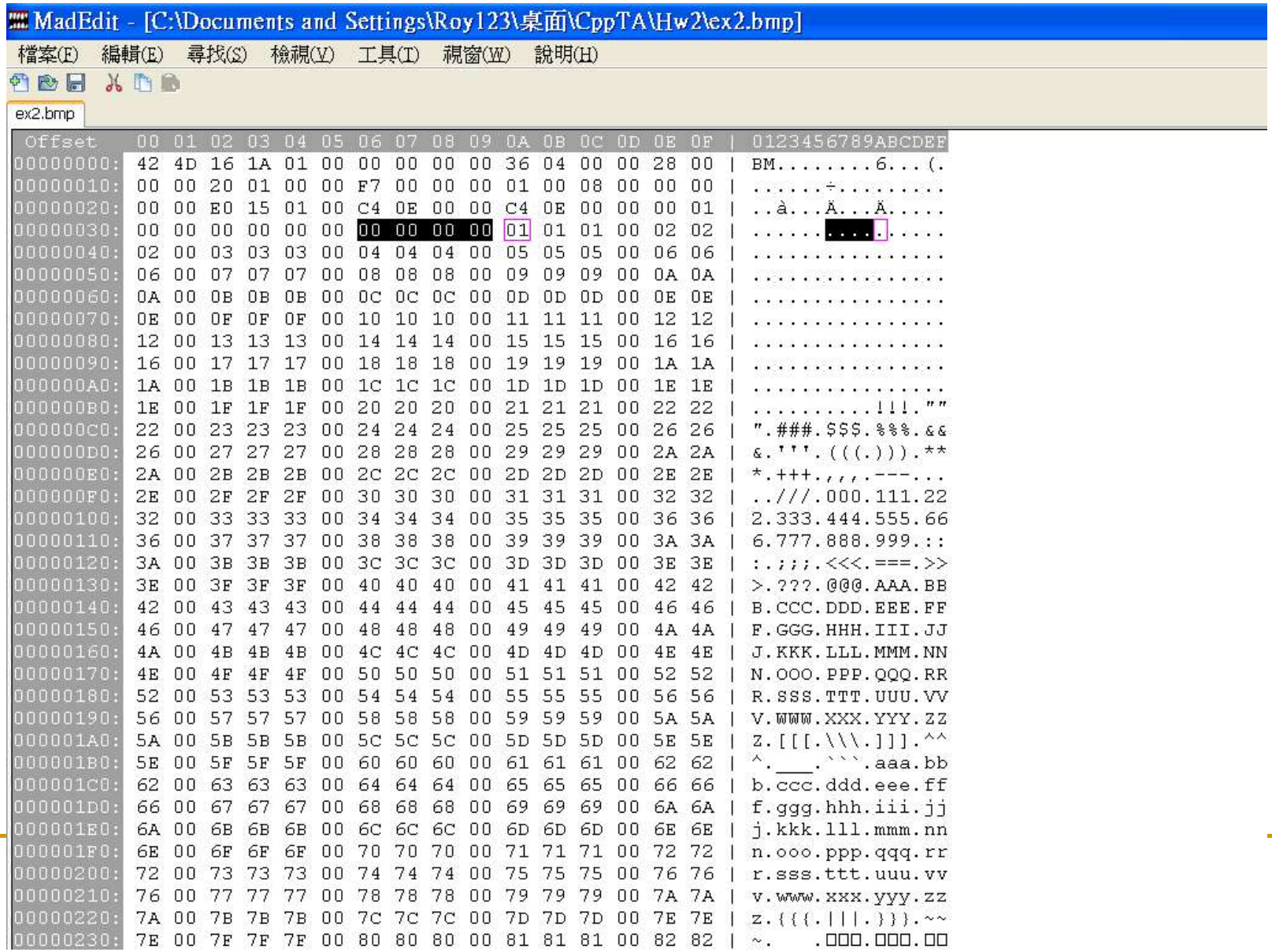
2.2 BITMAPINFOHEADER (1/2)



2.2 BITMAPINFOHEADER (2/2)

```
typedef struct tagBITMAPINFOHEADER{          //(40bytes)
    DWORD biSize;          //(4bytes) After Windows 3.X, it's always 40 (x28), which is
                           //          the structure size of BITMAPINFOHEADER
    LONG biWidth;          //(4bytes) The width of image
    LONG biHeight;         //(4bytes) The height of image
    WORD biPlanes;         //(2bytes) How many images in this file. For BMP, it's
                           //          always 1
    WORD biBitCount;       //(2bytes) How many bits stand for a pixel, 24 in our case
    DWORD biCompression;   //(4bytes) 0 is no compression, 1 is 8-bitRLE
                           //          compression, 2 is 4-bitRLE compression.
                           //          We only deal with no compression image.
    DWORD biSizeImage;     //(4bytes) The image size after compress. If no compression,
                           //          it could be 0 or image size
    LONG biXPelsPerMeter;  //(4bytes) horizontal dots per meter
    LONG biYPelsPerMeter;  //(4bytes) vertical dots per meter
    DWORD biClrUsed;       //(4bytes) How many colors used in palette, 0 for all colors
    DWORD biClrImportant;  //(4bytes) How many colors are important, 0 for all
} BITMAPINFOHEADER;
```

2.3 RGBQUAD (1/2)



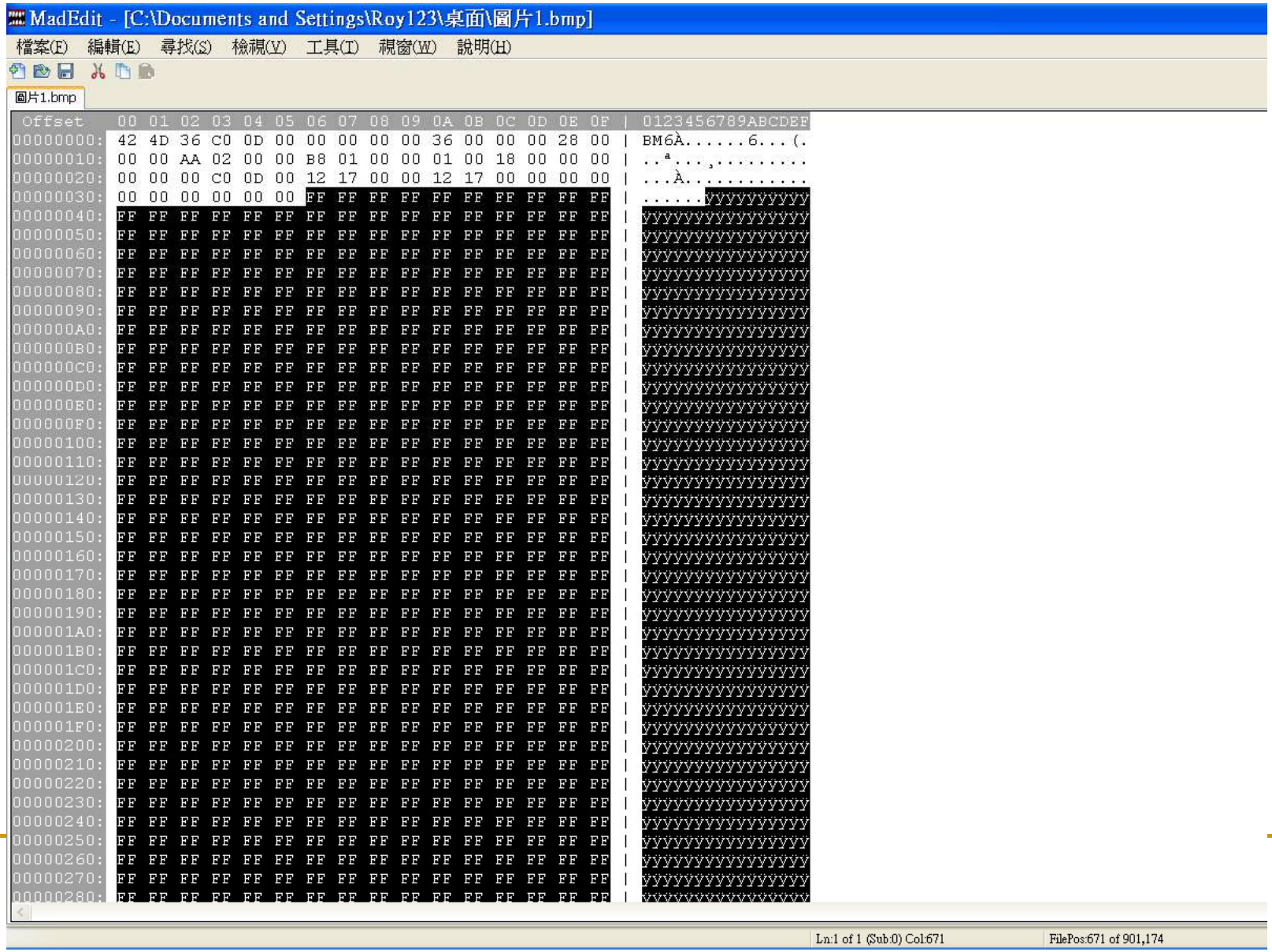
Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000:	42	4D	16	1A	01	00	00	00	00	00	36	04	00	00	28	00	BM.....6...(.
00000010:	00	00	20	01	00	00	F7	00	00	00	01	00	08	00	00	00	.....÷.....
00000020:	00	00	E0	15	01	00	C4	0E	00	00	C4	0E	00	00	00	01	..à...Ä...Ä.....
00000030:	00	00	00	00	00	00	00	00	00	00	01	01	01	00	02	02	.....
00000040:	02	00	03	03	03	00	04	04	04	00	05	05	05	00	06	06	.....
00000050:	06	00	07	07	07	00	08	08	08	00	09	09	09	00	0A	0A	.....
00000060:	0A	00	0B	0B	0B	00	0C	0C	0C	00	0D	0D	0D	00	0E	0E	.....
00000070:	0E	00	0F	0F	0F	00	10	10	10	00	11	11	11	00	12	12	.....
00000080:	12	00	13	13	13	00	14	14	14	00	15	15	15	00	16	16	.....
00000090:	16	00	17	17	17	00	18	18	18	00	19	19	19	00	1A	1A	.....
000000A0:	1A	00	1B	1B	1B	00	1C	1C	1C	00	1D	1D	1D	00	1E	1E	.....
000000B0:	1E	00	1F	1F	1F	00	20	20	20	00	21	21	21	00	22	22	.....!!!.""
000000C0:	22	00	23	23	23	00	24	24	24	00	25	25	25	00	26	26	".###.$$$.%%%.&&
000000D0:	26	00	27	27	27	00	28	28	28	00	29	29	29	00	2A	2A	&.'''.(((.))).**
000000E0:	2A	00	2B	2B	2B	00	2C	2C	2C	00	2D	2D	2D	00	2E	2E	*.+++.,,,.---...
000000F0:	2E	00	2F	2F	2F	00	30	30	30	00	31	31	31	00	32	32	../// .000.111.22
00000100:	32	00	33	33	33	00	34	34	34	00	35	35	35	00	36	36	2.333.444.555.66
00000110:	36	00	37	37	37	00	38	38	38	00	39	39	39	00	3A	3A	6.777.888.999.::
00000120:	3A	00	3B	3B	3B	00	3C	3C	3C	00	3D	3D	3D	00	3E	3E	:.;;;.<<<.===.>>
00000130:	3E	00	3F	3F	3F	00	40	40	40	00	41	41	41	00	42	42	>.???.@@@.AAA.BB
00000140:	42	00	43	43	43	00	44	44	44	00	45	45	45	00	46	46	B.CCC.DDD.EEE.FF
00000150:	46	00	47	47	47	00	48	48	48	00	49	49	49	00	4A	4A	F.GGG.HHH.III.JJ
00000160:	4A	00	4B	4B	4B	00	4C	4C	4C	00	4D	4D	4D	00	4E	4E	J.KKK.LLL.MMM.NN
00000170:	4E	00	4F	4F	4F	00	50	50	50	00	51	51	51	00	52	52	N.OOO.PPP.QQQ.RR
00000180:	52	00	53	53	53	00	54	54	54	00	55	55	55	00	56	56	R.SSS.TTT.UUU.VV
00000190:	56	00	57	57	57	00	58	58	58	00	59	59	59	00	5A	5A	V.WWW.XXX.YYY.ZZ
000001A0:	5A	00	5B	5B	5B	00	5C	5C	5C	00	5D	5D	5D	00	5E	5E	Z.[[[.\\\.]]].^^
000001B0:	5E	00	5F	5F	5F	00	60	60	60	00	61	61	61	00	62	62	^.___.```.aaa.bb
000001C0:	62	00	63	63	63	00	64	64	64	00	65	65	65	00	66	66	b.ccc.ddd.eee.ff
000001D0:	66	00	67	67	67	00	68	68	68	00	69	69	69	00	6A	6A	f.ggg.hhh.iii.jj
000001E0:	6A	00	6B	6B	6B	00	6C	6C	6C	00	6D	6D	6D	00	6E	6E	j.kkk.lll.mmm.nn
000001F0:	6E	00	6F	6F	6F	00	70	70	70	00	71	71	71	00	72	72	n.ooo.ppp.qqq.rr
00000200:	72	00	73	73	73	00	74	74	74	00	75	75	75	00	76	76	r.sss.ttt.uuu.vv
00000210:	76	00	77	77	77	00	78	78	78	00	79	79	79	00	7A	7A	v.www.xxx.yyy.zz
00000220:	7A	00	7B	7B	7B	00	7C	7C	7C	00	7D	7D	7D	00	7E	7E	z.{{{.|||.}}}.~~
00000230:	7E	00	7F	7F	7F	00	80	80	80	00	81	81	81	00	82	82	~. .□□□.□□□.□□

2.3 RGBQUAD (2/2)

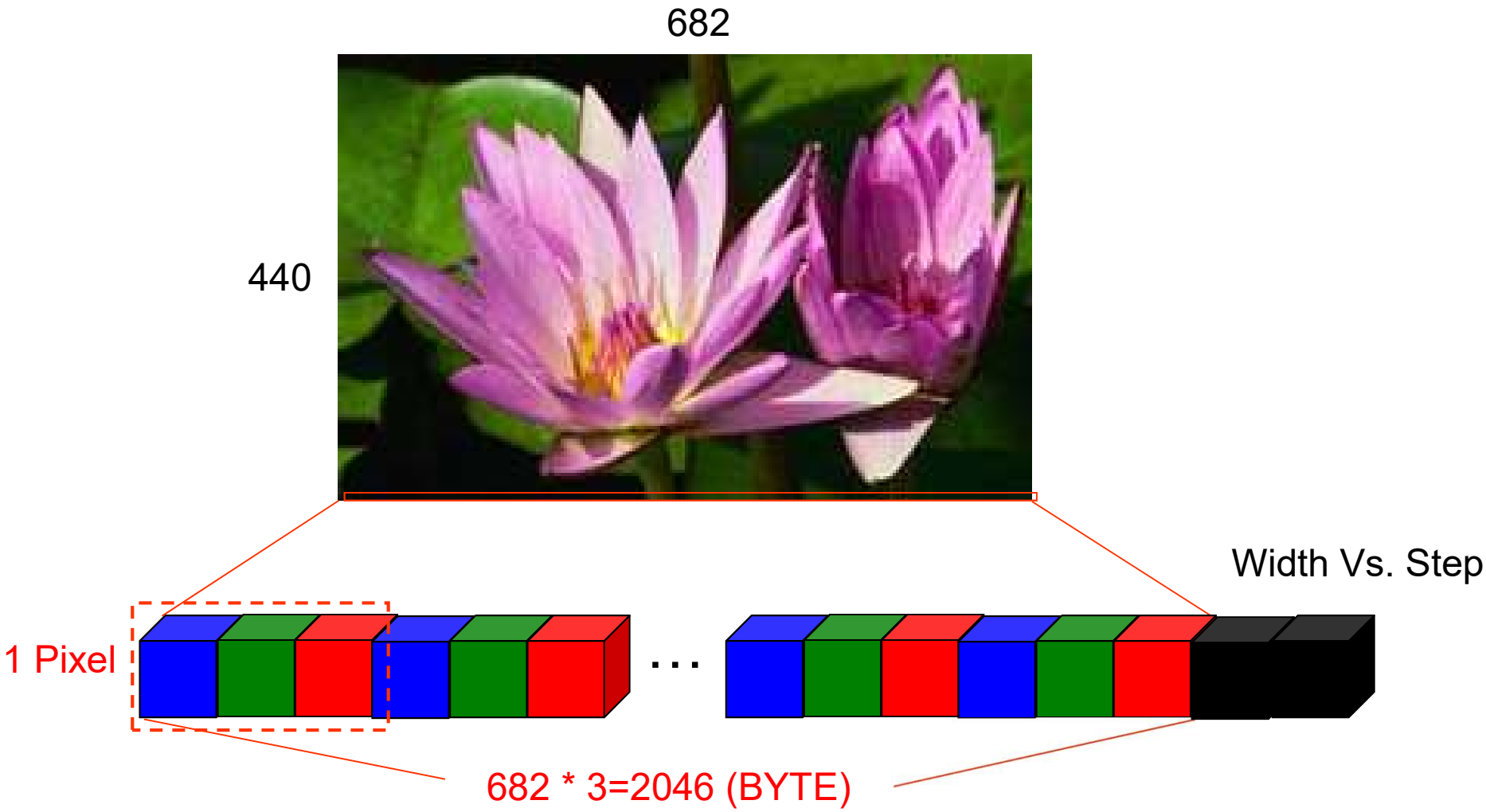
```
typedef struct tagRGBQUAD{              //(4bytes)
    BYTE rgbBlue;           //(1bytes)  blue channel
    BYTE rgbGreen;          //(1bytes)  green channel
    BYTE rgbRed;            //(1bytes)  red channel
    BYTE rgbReserved;       //(1bytes)  must be 0
} RGBQUAD;

// Palette - 256*(3+1) bytes
```

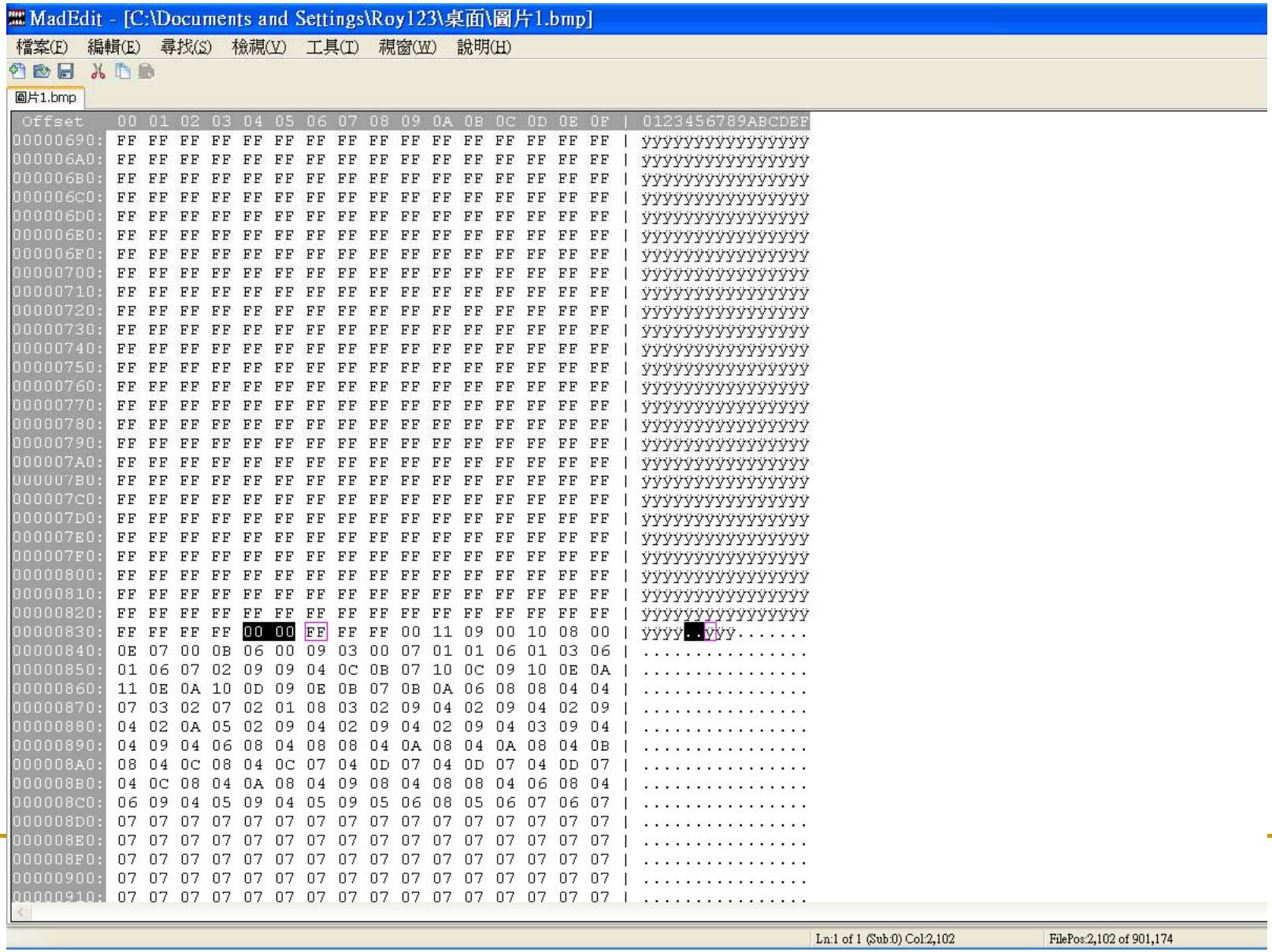
2.4 Image Data Array (1/4)



2.4 Image Data Array (2/4)



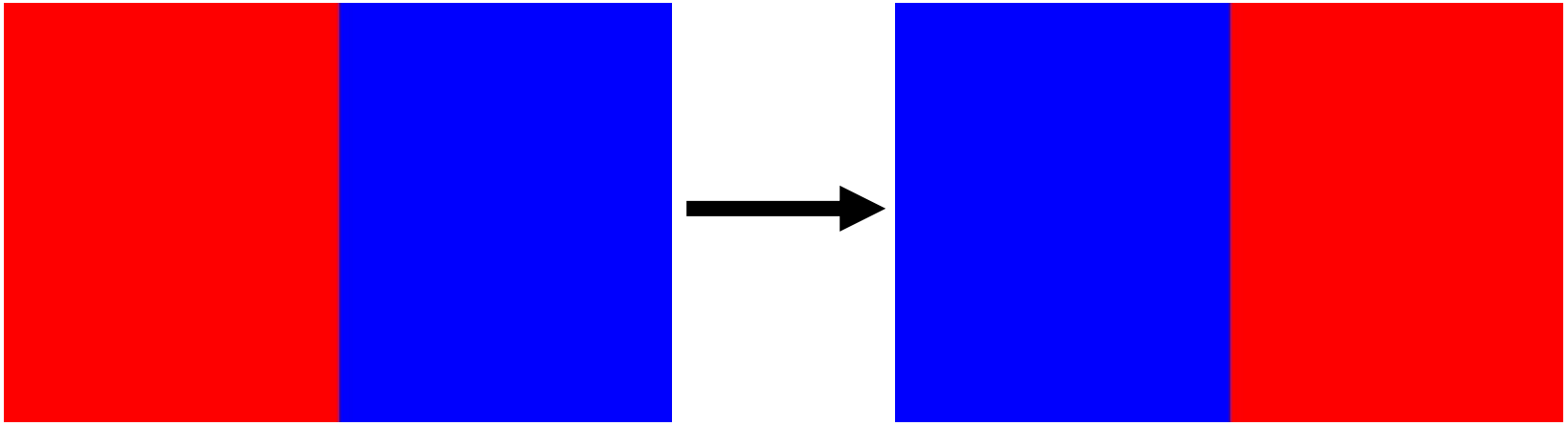
2.4 Image Data Array (3/4)



2.4 Image Data Array (4/4)

- Image data array is a huge array that record all pixel values of image.
- Each row should be a multiple of 4.
- If the width of image wasn't a multiple of 4, we should fill up with zeros.
  - ex: width = 13 → fill up with 3 zeros, then the new width = 16
- The order of image is upper-side-down, the last row in image is the first row in file. It is because of displayer - Scanning.

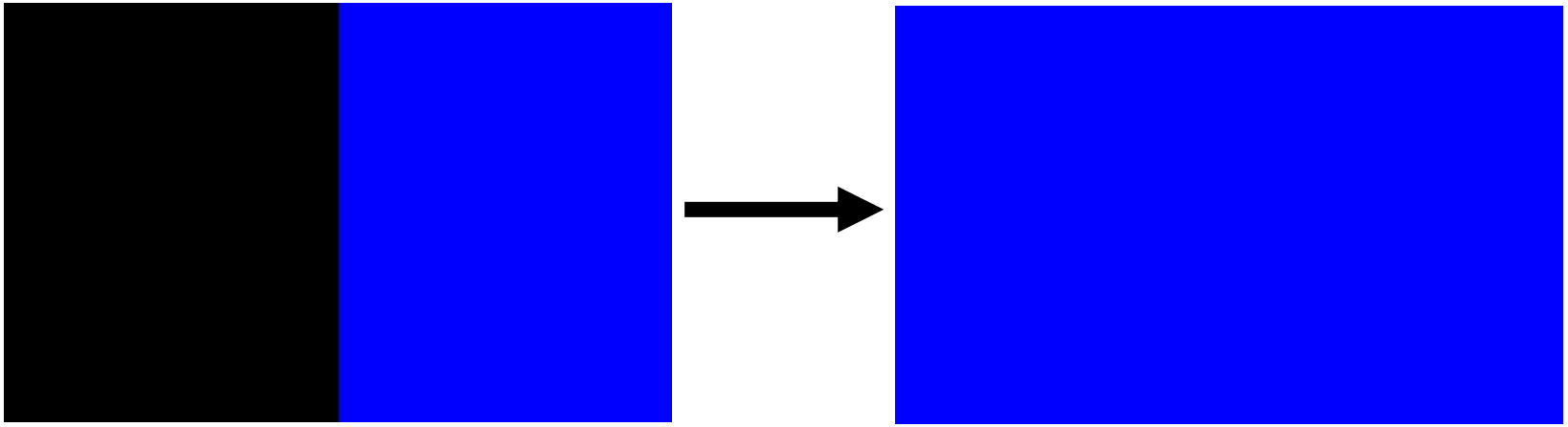
2.5 Others: Mirror



255,0,0	255,0,0	0,0,255	0,0,255
255,0,0	255,0,0	0,0,255	0,0,255
255,0,0	255,0,0	0,0,255	0,0,255

0,0,255	0,0,255	255,0,0	255,0,0
0,0,255	0,0,255	255,0,0	255,0,0
0,0,255	0,0,255	255,0,0	255,0,0

2.5 Others: Color change



0,0,0	0,0,0	0,0,255	0,0,255
0,0,0	0,0,0	0,0,255	0,0,255
0,0,0	0,0,0	0,0,255	0,0,255

0,0,255	0,0,255	0,0,255	0,0,255
0,0,255	0,0,255	0,0,255	0,0,255
0,0,255	0,0,255	0,0,255	0,0,255

2.5 Others: Embossment (1/3)

- Converting the input image to the gray level image.
- Convolute with 2 sobel mask, vertical and horizontal
- Then normalize to [0, 255]

2.5 Others: Embossment (2/3)

Vertical

-1	0	1
-2	0	2
-1	0	1

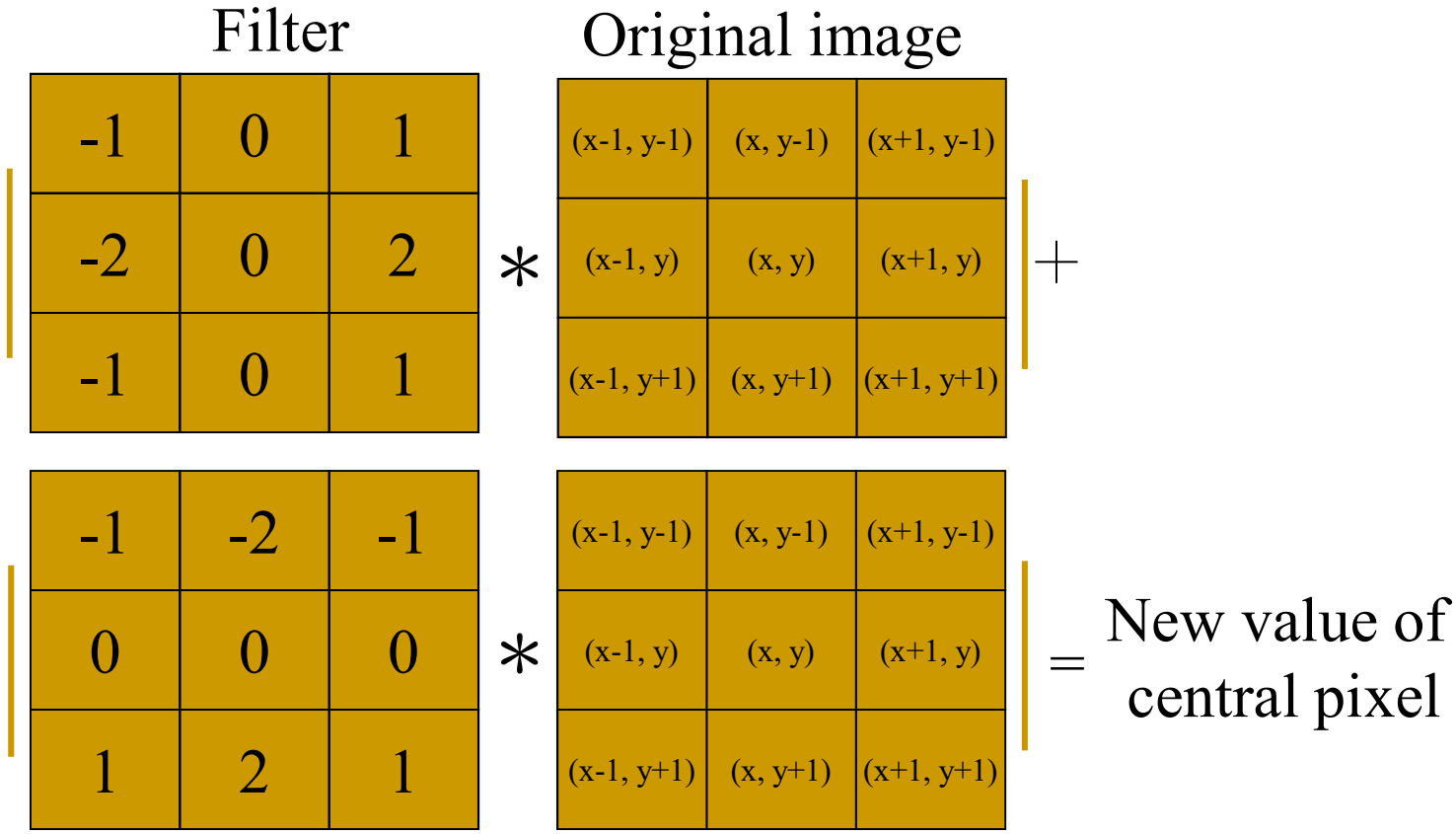
Horizontal

-1	-2	-1
0	0	0
1	2	1

Sobel Mask or Filter

- You may leave the boundary of image without dealing with it.
- New pixel value was calculated by neighboring pixels.

2.5 Others: Embossment (3/3)



- New value → [0, 255]