Graphics and Image Data Representation

Lecturer：Jun Xiao
（肖俊）
College of Software and Technology

Main content

- Basic Data Types
  - 1-Bit Images
  - 8-Bit Grey-Level Images
  - 24-Bit Color Images
  - 8-Bit Color Images
  - Color Lookup Table (LUTs)

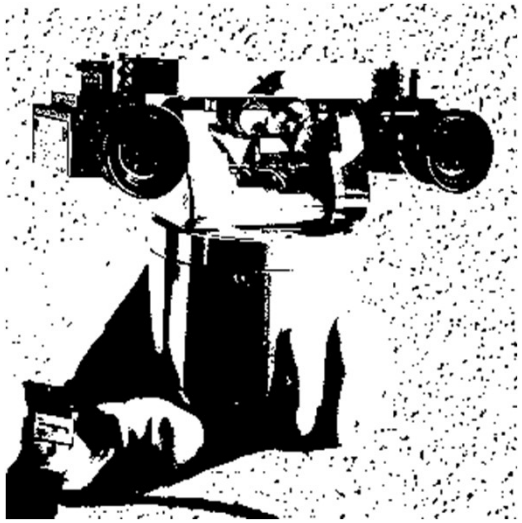
- Popular File Formats
  - JPEG ,GIF, BMP, others

1. Basic Graphics/Image Types

- 1-Bit Image
- 8-Bit Grey-Level Image
- 24-Bit Color Image
- 8-Bit Color Image
- Color Lookup Tables
- How to Devise a Color Lookup Table

1.1 1-Bit Image：Case

Alao called Binary Image or Monochrome Image

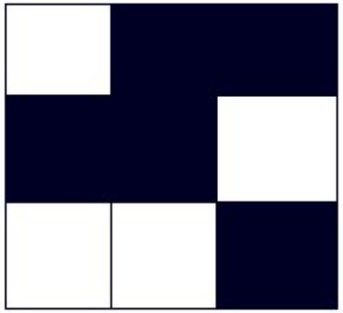


1-Bit Image Examples

1.1 1-Bit Image: Features

- Consist of on and off pixels (pixel--picture elements in digital images)
- Each pixel is stored as a single bit (0 or 1),

  0--black, 1--white



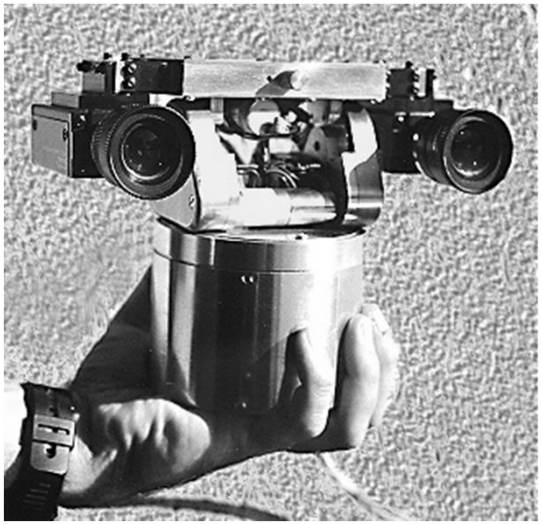
$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

1.1 1-Bit Image：Size and Usage

- Storage
  - Monochrome image with resolution: 640×480
  - $640 \times 480/8$ bytes
  - Storing space needed: 38.4KB

- Usage
  - Pictures containing only simple graphics and text

1.2 8-Bit Gray-level Image：Case



8-Bit Gray-Level Image Examples

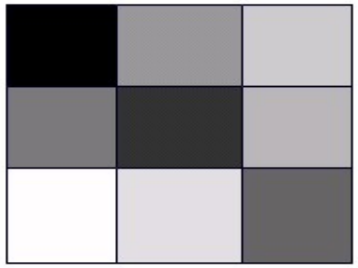
1.2 8-Bit Gray-level Image：Case

8-Bit Gray-Level Image VS 1-Bit Image

1.2 8-Bit Gray-level Image：Features

- Each pixel is represented by a single byte
  - A gray value between 0 and 255

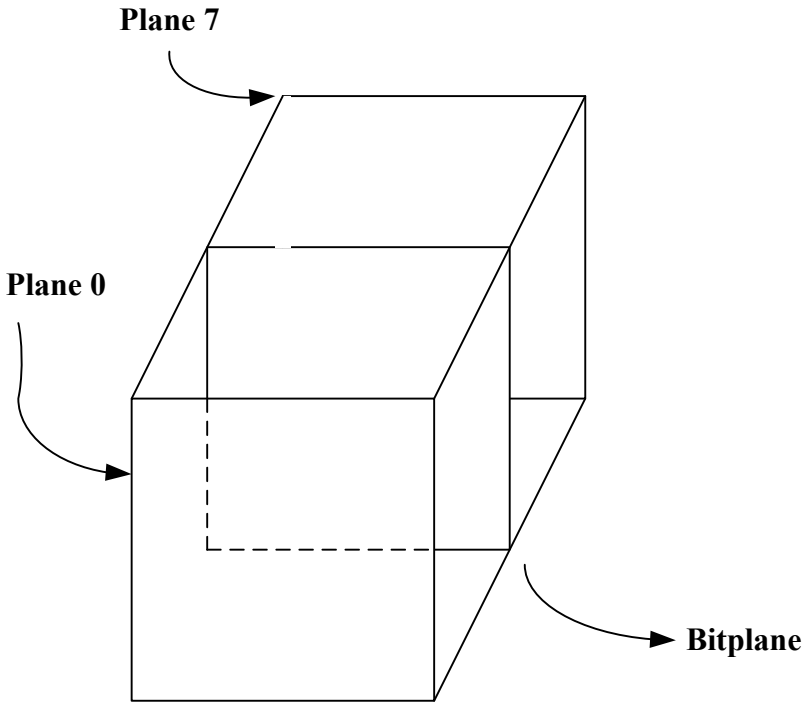
- The entire image can be thought of as a two-dimensional array of pixel values
  - Called bitmap



$$I = \begin{bmatrix} 0 & 150 & 200 \\ 120 & 50 & 180 \\ 250 & 220 & 100 \end{bmatrix}$$

1.2 8-Bit Gray-level Image：Features

- 8-Bit image as a set of 1-bit bitplanes
  - Each plane consists of a 1-bit representation of the image at one level
  - All the bitplanes make up a single byte that stores the value between 0 ~ 255



1.2 8-Bit Gray-level Image：Size

- Resolution
  - High： 1600×1200
  - Low： 640×480
  - Aspect Ratio : 4:3

- The space needed by a 640×480 grey image
  - $640 \times 480 = 307{,}200$ bytes

- Hardware storing Image Array
  - frame buffer / ”Video” card

1.2 8-Bit Gray-level Image：Print

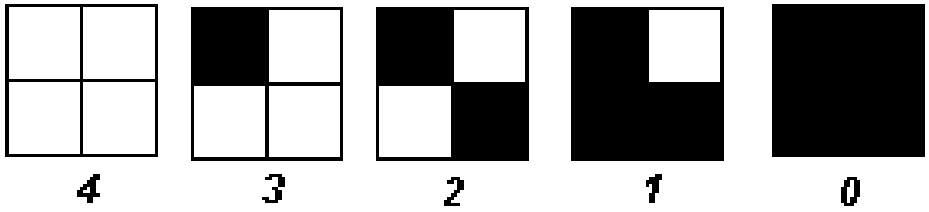
- How to print an 8-bit gray-level image on 2-level (1-bit) printer?

- DPI
  - Dot per inch

- Printing such image is complex
  - Use Dithering
  - Convert intensity resolution into spatial resolution

1.2 8-Bit Gray-level Image：Print

- Dithering
  - The main strategy is to replace a pixel value by a larger pattern, say 2 x 2 or 4 x 4, such that the number of printed dots approximates the varying-sized disks of ink used in analog, in **halftone printing** (e.g., for newspaper photos).
  - Convert the color resolution into the spatial resolution.
- An $N \times N$ matrix represents N^2+1 levels of intensity
  - 2×2 pattern can represent five level：



1.2 8-Bit Gray-level Image：print

- we can first re-map image values in 0..255 into the new range 0..4 by (integer) dividing by 256/5. Then, e.g., if the pixel value is 0 we print nothing, in a 2×2 area of printer output. But if the pixel value is 4 we print all four dots.

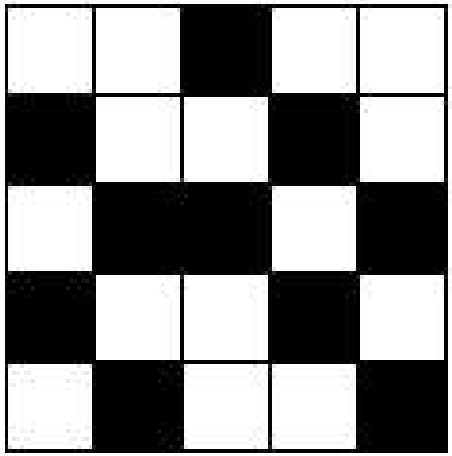
- If the intensity is > the dither matrix entry then print an `on` dot at that entry location: replace each pixel by an n x n matrix of dots.

- The above method increasing the size of the output image
  - If one pixel uses 4×4 pattern, the size of an $N \times N$ image becomes $4N \times 4N$, makes an image 16 times as large!

1.2 8-Bit Gray-level Image：print

- One better method：Avoid enlarging the output image
  - Store an integer matrix (Standard Pattern), each value from 0 to 255
  - Comparing the grey image matrix with pattern, print the dot when the value greater than the grey

0	14	22	5	8
18	9	1	19	13
6	24	16	7	23
21	2	12	20	3
10	15	4	11	17



One 25-grey level case: left is standard, the right with grey=15

1.2 8-Bit Gray-level Image：print

- An algorithm for ordered dither, with n x n dither matrix, is as follows:

BEGIN

for x = 0 to x_{max} // columns

for y = 0 to y_{max} // rows

$i = x \bmod n$

$j = y \bmod n$

// $I(x, y)$ is the input, $O(x, y)$ is the output,

//D is the dither matrix.

if $I(x, y) > D(i, j)$

$O(x, y) = 1$;

else

$O(x, y) = 0$;

END

1.2 8-Bit Gray-level Image：Print

- Example
  - Print an image (240*180*8bit) on a paper (12.8*9.6 inch) by a printer with 300*300 DPI, what's the size of each pixel (dots)?

  - (300*12.8)*(300*9.6) = 3480*2880 dots
  - (3840/240)*(2880/180) = 16*16=256

1.2 8-Bit Gray-level Image：Print

- Generate the output image using standard matrix

(a) (b) (c)

Fig. 3.4: Dithering of grayscale images.

(a): 8-bit grey image “lenagray.bmp”. (b): Dithered version of the image. (c): Detail of dithered version.

Question?

- Print an image (600*450*8bit) on a paper (8*6 inch) by a printer with 300*300 DPI, what's the size of each pixel (dots)?

- (300*8)*(300*6) = 2400*1800 dots
- (2400/600)*(1800/450) = 4*4 only 17 levels

- Reduce the image size to 150*113?
- Reduce the gray-level from 256 to 16?

1.3 24-Bit Color Image：Case

嫦娥

李商隐

云母屏风烛影深，
长河渐落晓星沉。
嫦娥应悔偷灵药，
碧海青天夜夜心！

ChangE flying to the Moon

1.3 24-Bit Color Image：Feature

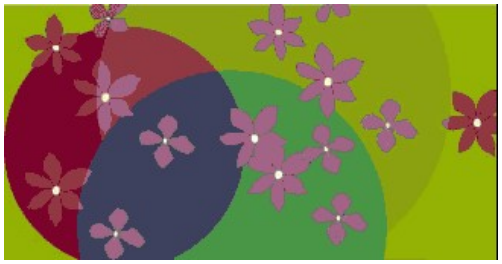
- Each pixel using three bytes: representing RGB
  - Value from 0 to 255；
  - Supports $256 \times 256 \times 256$ colors， 16,777,216
- Each pixel described by different grey values of RGB

$$R = \begin{bmatrix} 255 & 240 & 240 \\ 255 & 0 & 80 \\ 255 & 0 & 0 \end{bmatrix} \quad G = \begin{bmatrix} 0 & 160 & 80 \\ 255 & 255 & 160 \\ 0 & 255 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 80 & 160 \\ 0 & 0 & 240 \\ 255 & 255 & 255 \end{bmatrix}$$

1.3 24-Bit Color Image：Size

- 640 × 480 24-Bit Color image，921.6KB
  - 640 × 480 × 3 bytes

- Many 24-bit color image actually stored as 32-Bit image
  - Extra data of each pixel used to store αvalue，indicate the special effect information（such as, transparency flag）



  - Semi-transparency image color = Source image color ×（100% - transparency）+Background image color × transparency

1.3 24-Bit Color Image

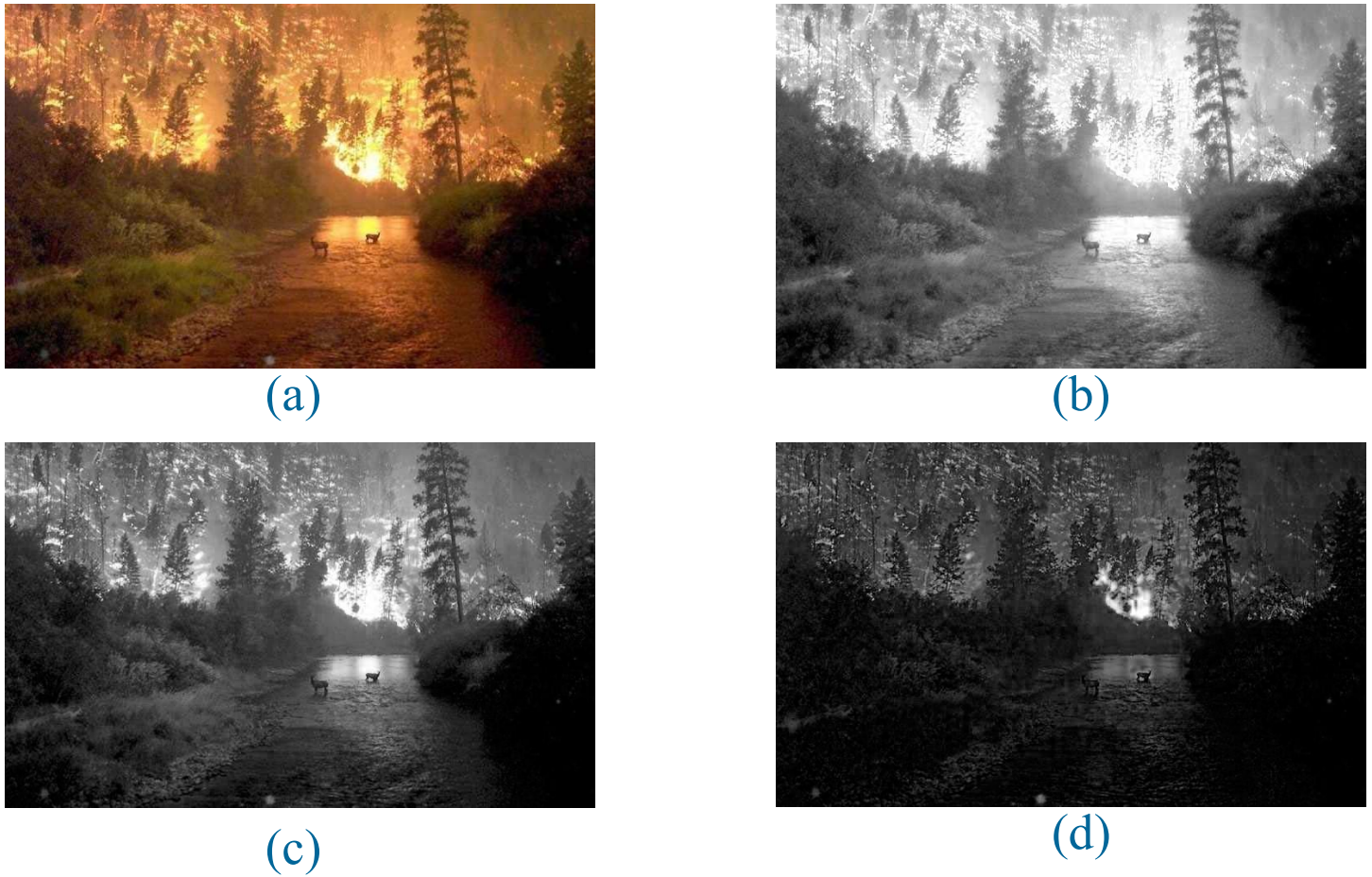


Fig. 3.5: High-resolution color and separate R, G, B color channel images. (a): Example of 24-bit color image "forestfire.bmp". (b, c, d): R, G, and B color channels for this image

1.4 8-Bit Color Image：Case

- Also called 256-colors image

1.4 8-Bit Color Image：Case

8-Bit Color Image VS 24-Bit Color Image

1.4 8-Bit Color Image：Features

- The idea of using Lookup table(palette)
  - An image store a set of bytes，not the real color
  - Bytes value is the index to a 3-bytes color table
  - Choosing what colors to put in table is important

- Choose the most important 256 colors
  - Generated by clustering the $256 \times 256 \times 256$ colors
  - Median-cut Algorithm
  - More accurate version of the Median-cut Algorithm

1.4 8-Bit Color Image

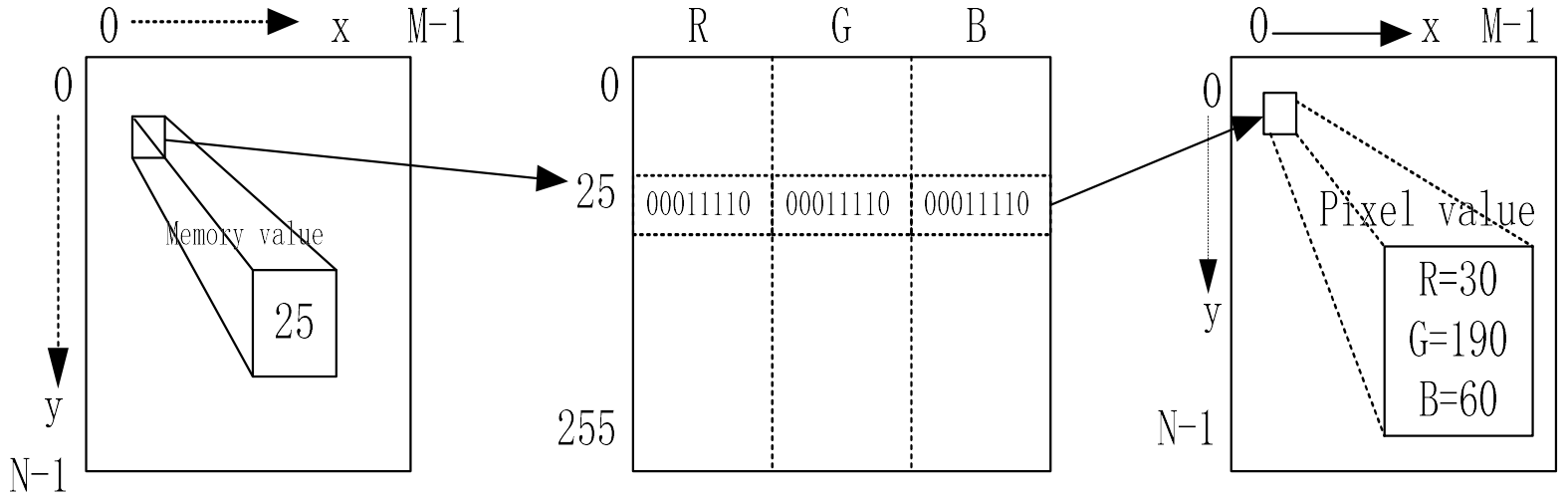
24-bit Color Image

8-bit Color Image

•Note the great savings in space for 8-bit images, over 24-bit ones: a 640 x 480 8-bit color image only requires 300 kB of storage, compared to 921.6 kB for a color image (again, without any compression applied).

1.5 Color Lookup Tables：Case

•The idea used in 8-bit color images is to store only the index, or code value, for each pixel. Then, e.g., if a pixel stores the value 25, the meaning is to go to row 25 in a color look-up table (LUT).



Value as the Index to Table

Get the color values by Searching

The RGB value of the pixel

1.5 Color Lookup Tables：How to apply

- Change color by adjusting the LUT
  - LUT less than image, with the advantage of speed

Example： change LUT

Index	R	G	B
1	255	0	0

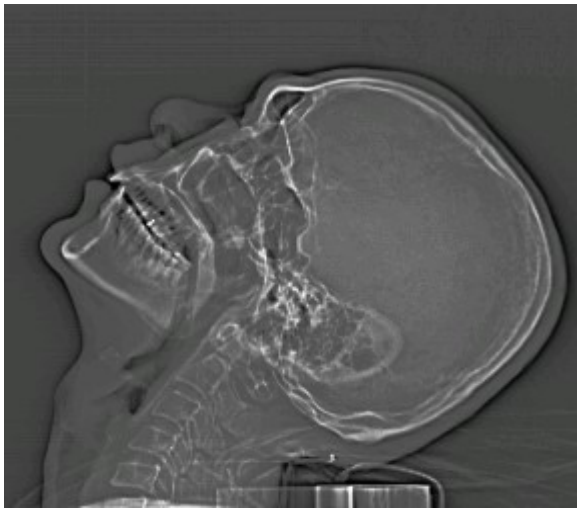
into

Index	R	G	B
1	0	255	0

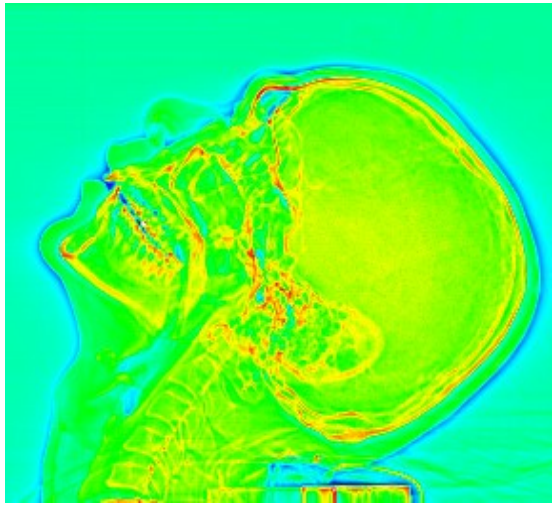
For the color index by 1, that is to convert red to green

- An important application：Medical Image
  - Convert the grey image into color image

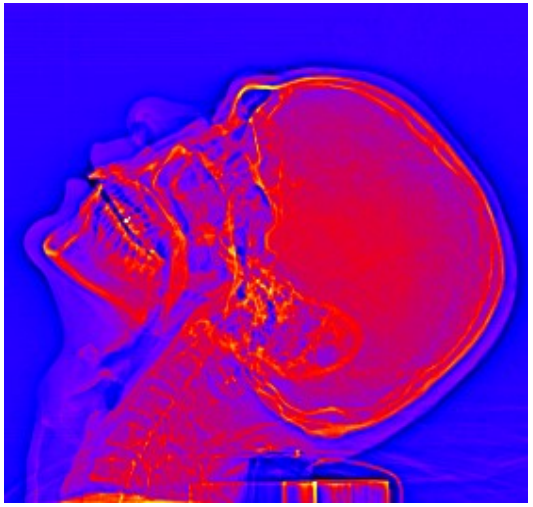
1.5 Color Lookup Tables：medical image



Grey Image



Color Image-1



Color Image-2

By modifying the LUT to convert grey image into color image

1.5 Color Lookup Tables：Medical image

Index	R	G	B
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
. . .	. . .	. . .	. . .
64	64	64	64
65	65	65	65
66	66	66	66
67	67	67	67
68	68	68	68
69	69	69	69
. . .	. . .	. . .	. . .
254	254	254	254
255	255	255	255

正常灰度LUT

Index	R	G	B
0	0	0	0
1	0	0	7
2	0	0	15
3	0	0	23
. . .	. . .	. . .	. . .
64	0	255	255
65	0	255	247
66	0	255	239
67	0	255	231
68	0	255	223
69	0	255	215
. . .	. . .	. . .	. . .
254	255	248	248
255	255	252	252

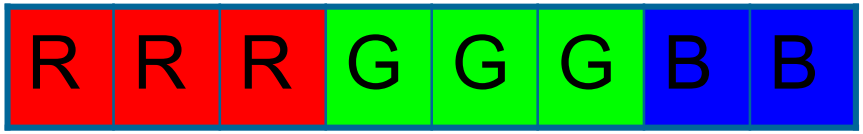
彩虹编码LUT

Index	R	G	B
0	0	0	0
1	0	0	4
2	0	0	8
3	0	0	12
. . .	. . .	. . .	. . .
64	0	0	255
65	4	0	255
66	8	0	255
67	12	0	255
68	16	0	255
69	20	0	255
. . .	. . .	. . .	. . .
254	255	255	248
255	255	255	252

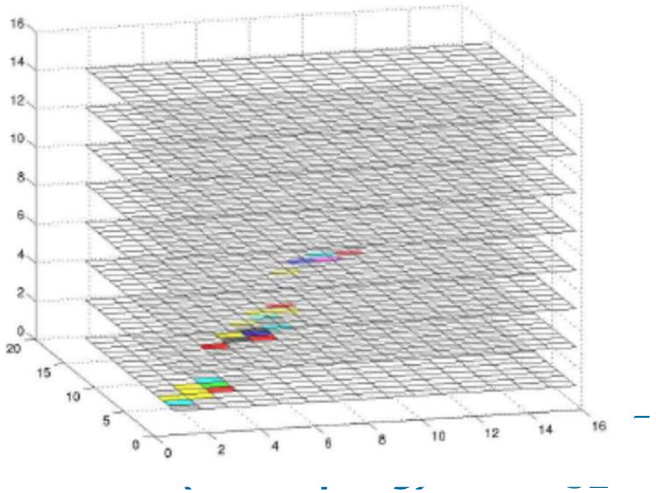
热金属编码LUT

1.6 How to Devise a Color Lookup Table

- Example: 8-Bit Color Image
  - Humans are more sensitive to R and G than to B
  - So R=3, G=3 and B=2



- Basic Idea —— Clustering
  - Analyzing the three-dimensional histogram of RGB colors
  - Clustering is an expensive and slow process



1.6 How to Devise a Color Lookup Table

- The most straightforward way to make 8-bit look-up color out of 24-bit color would be to divide the RGB cube into equal slices in each dimension.

  (a) The centers of each of the resulting cubes would serve as the entries in the color LUT, while simply scaling the RGB ranges 0..255 into the appropriate ranges would generate the 8-bit codes.

  (b) Since humans are more sensitive to R and G than to B, we could shrink the R range and G range 0..255 into the 3-bit range 0..7 and shrink the B range down to the 2-bit range 0..3, thus making up a total of 8 bits.

  (c) To shrink R and G, we could simply divide the R or G byte value by (256/8)=32 and then truncate. Then each pixel in the image gets replaced by its 8-bit index and the color LUT serves to generate 24-bit color.

1.6 How to Devise a Color Lookup Table

- Then each pixel in the image gets replaced by its 8-bit index.

Eg.

– R: 16, 48, 80, 112, 144, 176, 208, 240

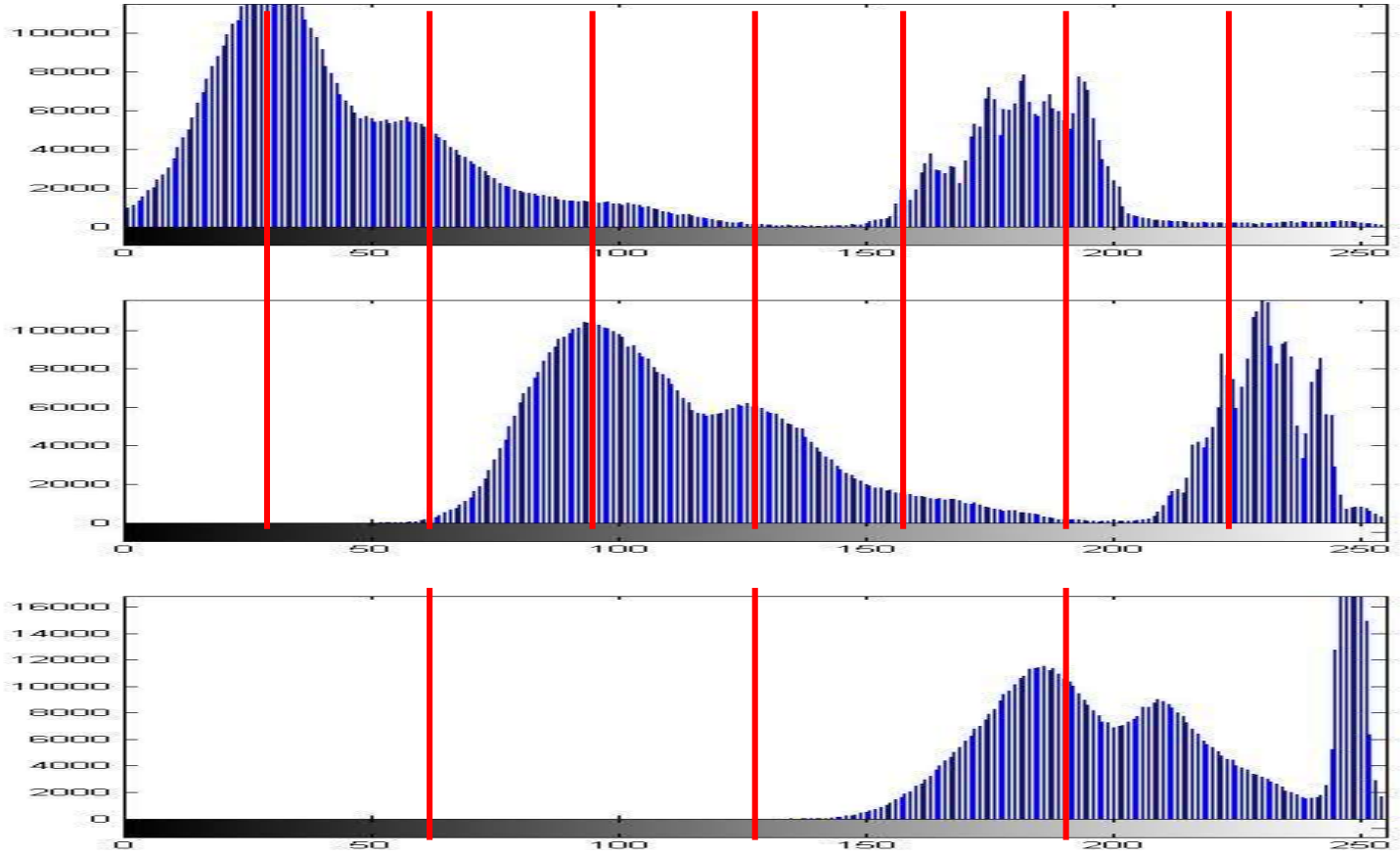
– G: 16, 48, 80, 112, 144, 176, 208, 240

– B: 32, 96, 160, 224

A pixel with color [30, 129, 80] should be converted into: [16, 112, 96]

1.6 How to Devise a Color Lookup Table

- Can we achieve better result?



1.6 How to Devise a Color Lookup Table

- **Median-cut algorithm**: A simple alternate solution that does a better job for this color reduction problem.

(a) The idea is to sort the R byte values and find their median; then values smaller than the median are labelled with a "0" bit and values larger than the median are labelled with a "1" bit.

(b) This type of scheme will indeed concentrate bits where they most need to differentiate between high populations of close colors.

(c) One can most easily visualize finding the median by using a histogram showing counts at position 0..255.

(d) Fig. 3.11 shows a histogram of the R byte values for the `forestfire.bmp` image along with the median of these values, shown as a vertical line.

1.6 How to Devise a Color Lookup Table

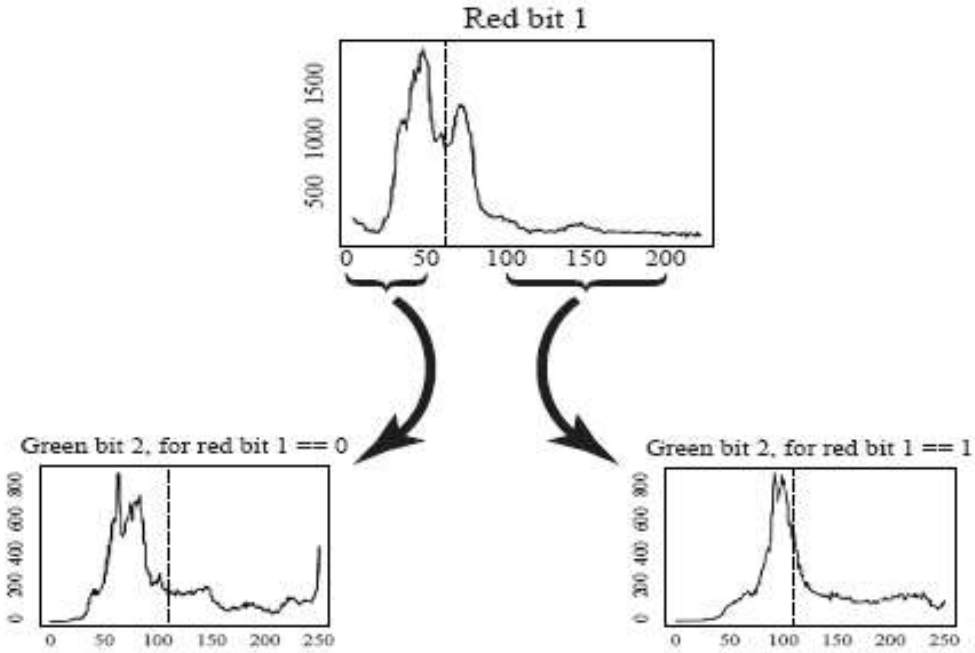
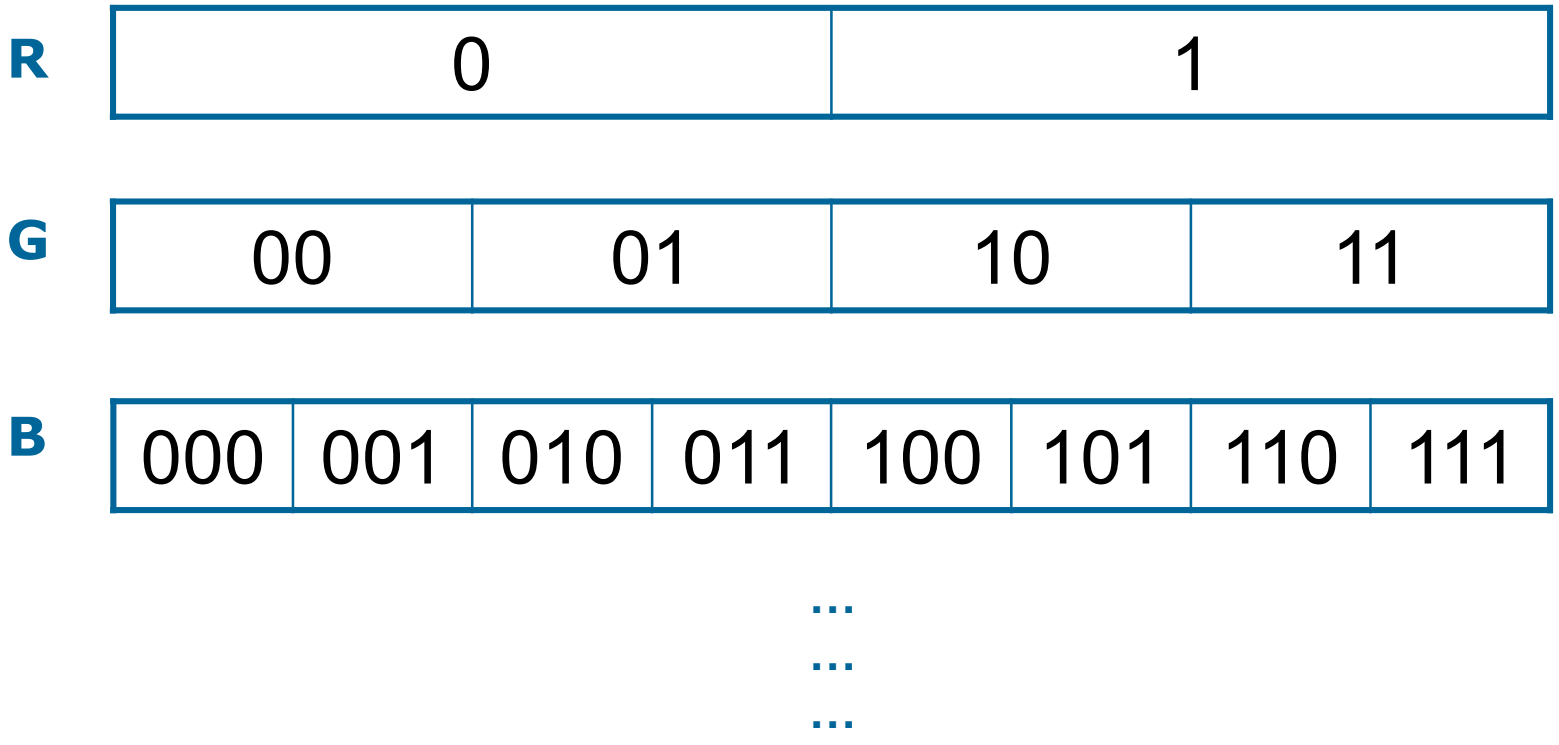


Fig. 3.11 Histogram of R bytes for the 24-bit color image "forestfire.bmp" results in a "0" bit or "1" bit label for every pixel. For the second bit of the color table index being built, we take R values less than the R median and label just those pixels as "0" or "1" according as their G value is less than or greater than the median of the G value, just for the "0" Red bit pixels. Continuing over R, G, B for 8 bits gives a color LUT 8-bit index

1.6 How to Devise a Color Lookup Table

- Median-cut Algorithm



2、Popular image file format

- GIF
- JPEG
- BMP
- PNG
- TIFF
- EXIF
- others

2.1 GIF Image：Case

2.1 GIF Image：features

- GIF (Graphics Interchange Format)
  - Invented by UNISYS Corporation and Compuserve in 1987
  - Initially transmit graphical image through telephone line
  - Not belong to any application program, presently supported by almost all relevant software

2.1 GIF Image：features

- Using LZW（Lempel-Ziv-Welch）Compression Algorithm
  - LZW algorithm is lossless format with continuous color, compression rate about 50%

- Limited to 8-bit（256）color image
  - GIF image depth from 1bit to 8bit
  - GIF image supports 256 colors

2.1 GIF Image：features（Cont.）

- Interlacing
  - Decode speed fast
  - Store in interlacing method
  - Can Gradually Display by four passes
- The GIF89a supporting animation
  - Storing multiple color images in one image file

2.1 GIF Image : Case Analysis

- One 120*160 gif image

Offset	Length	Contents
0	3 bytes	"GIF"
3	3 bytes	"87a" or "89a"
6	2 bytes	<Logical Screen Width>
8	2 bytes	<Logical Screen Height>
10	1 byte	bit 0: Global Color Table Flag (GCTF) bit 1..3: Color Resolution bit 4: Sort Flag to Global Color Table bit 5..7: Size of Global Color Table: 2^(1+n)
11	1 byte	<Background Color Index>
……		

Gif: file head information

offset, length, contents

Gif file format

2.1 GIF Image : Case Analysis

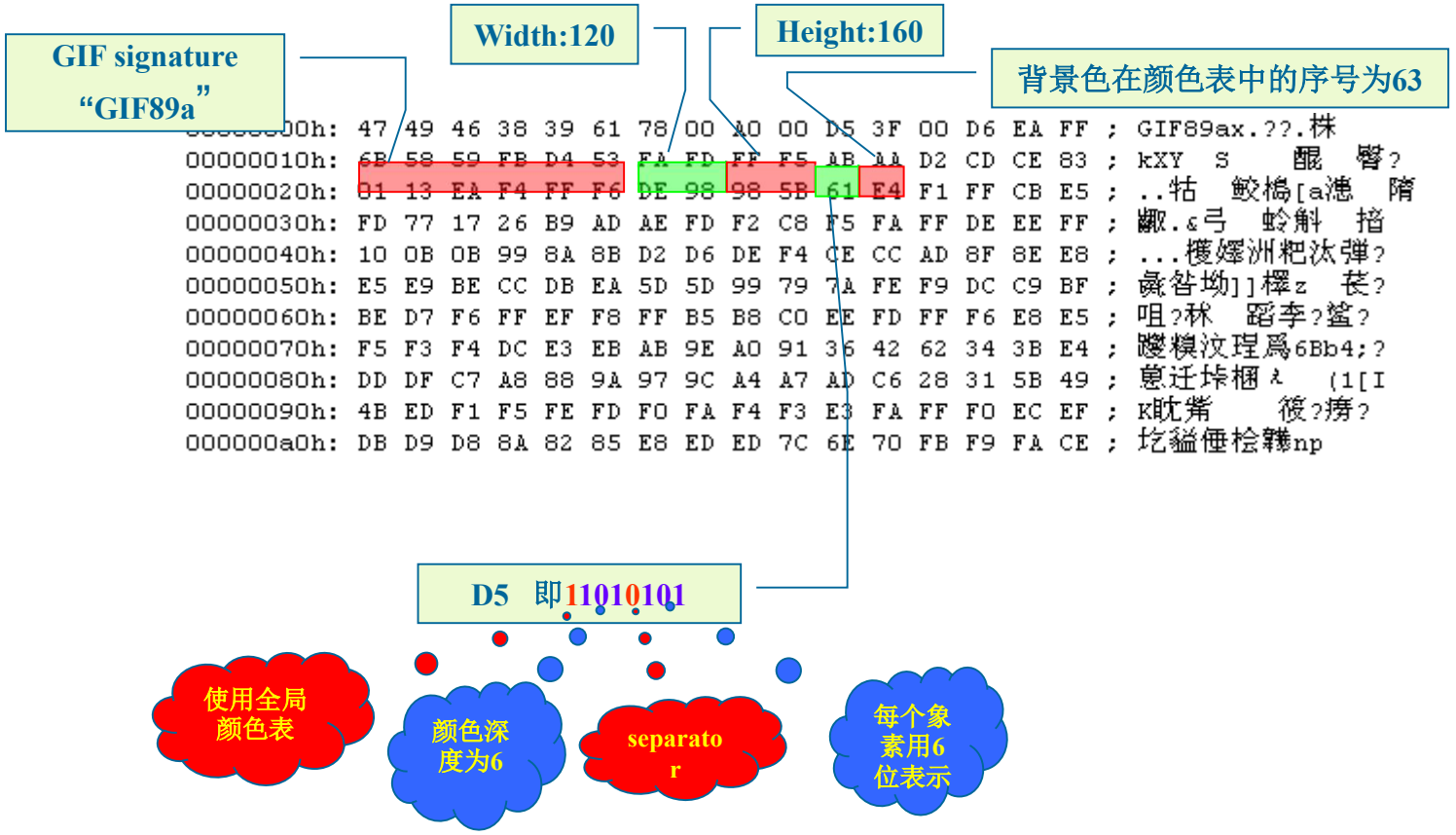


Image file analysis opened by Ultra-edit

2.2 JPEG Image ： Case

2.2 JPEG Image：Features

- JPEG（Joint Photographic Experts Group）
  - Created by the Task Group of the International Standard Organization（ISO）
- Take advantage of some limitations of human vision system
    - JPEG achieve high rates of compression
- A lossy compression method
    - Allow user to set a desired level of quality, or compression ratio (input divided by output)

2.2 JPEG Image：Example1

JPEG Image (1)：252kB

2.2 JPEG Image：Example2

JPEG Image(2): 45.2kB

2.2 JPEG Image：Example3

JPEG Image (3): 9.21kB

2.3 BMP Image

- Created by Microsoft as Window's main image format, can store 1bit, 4bits, 8bits, as well as real color data

- BMP file has three storage forms:
  - Original data without compression, most popular
  - Run Length Encoding: Used for 8-bits image（256 colors）BI-RLE8
  - RLE: used for 4-bits image (16 colors) BI_RLE4

2.3 BMP Image

- BMP file consists four components:
  - File Head： type and other information
  - Information head of bitmap： length， width， compression algorithms and so on
  - Palette： Color LUT table， 24-bits real color image with no palette
  - Image Data： Real color image stores (R,G,B) three components， image with palette stores the index to the palette

2.3 BMP Image

- BMP file: case analysis
  - 128*128 lena grey image

Offset	Length	Contents
0	2 bytes	"BM"
2	4 bytes	Total size included "BM“
6	2 bytes	Reserved1
8	2 bytes	Reserved2
10	4 bytes	Offset Bytes
14	4 bytes	Header size (n)
18	n-4 bytes	Header (See right)
14+n .. s-1		Image data

BMP file format: offset, length, contents

Bitmap head format

Offset	Length	Contents
18	4 bytes	Width
22	4 bytes	Heiht
26	2 bytes	Planes
28	2 bytes	Bits per Pixel
30	4 bytes	Compression
34	4 bytes	Image size
38	4 bytes	X Pixels per meter
42	4 bytes	Y Pixels per meter
46	4 bytes	Number of Colors
50	4 bytes	Colors Important
54 (n-40) bytes		OS/2 new xtentional fields

2.3 BMP Image

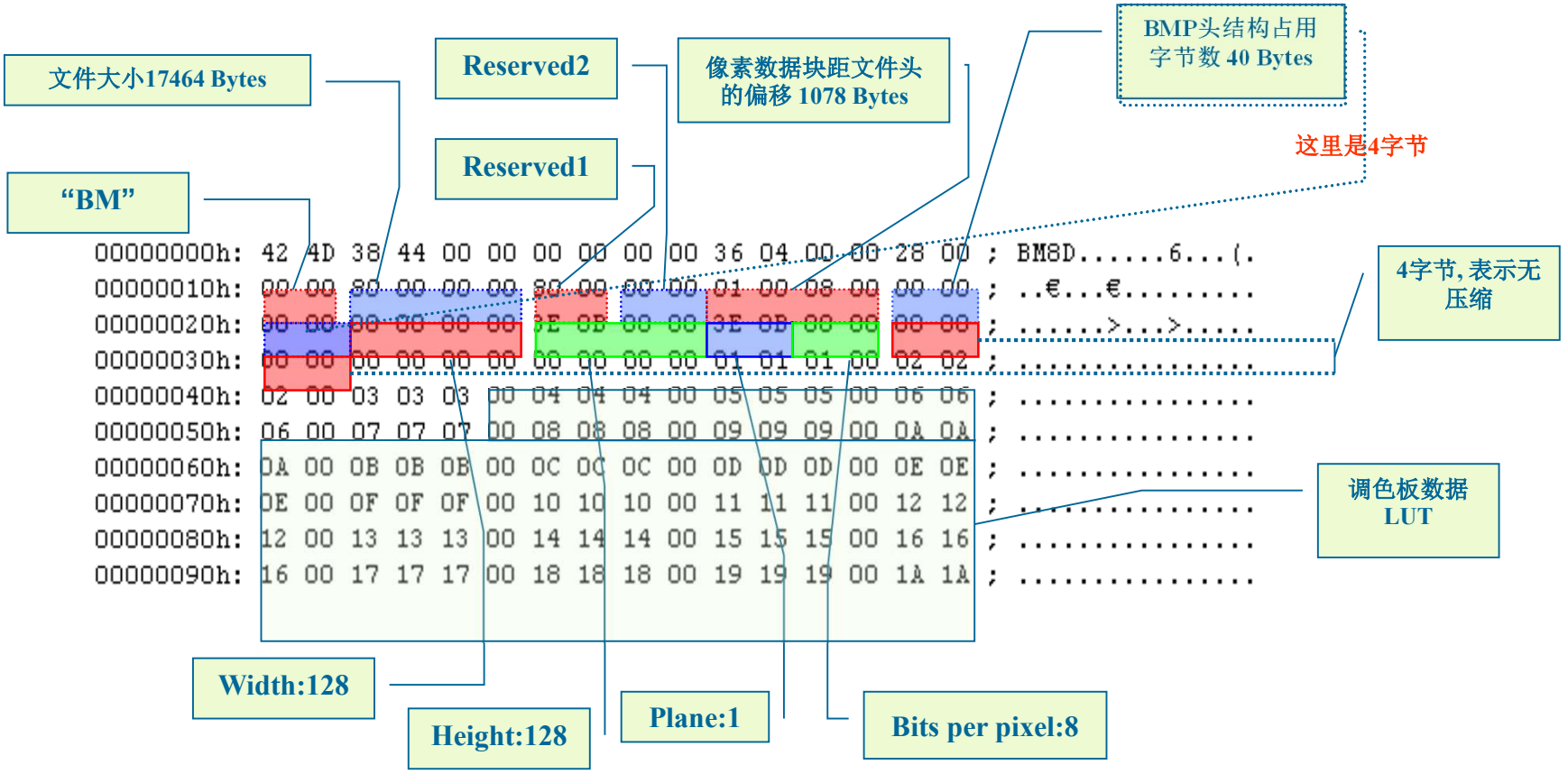


Image file analysis open by Ultra-edit

2.3 Other typical image formats

- PNG（Portable Network Graphics）
- TIFF （Tagged Image File Format）
- EXIF（Exchange Image File）
- Others

The End

Thanks!

Email: junx@cs.zju.edu.cn