Image Compression

cs4: Computer Science Bootcamp

Çetin Kaya Koç
http://koclab.cs.ucsb.edu/teaching/cs4
cetinkoc@ucsb.edu

Size of an Image

- The size of an image is the number of bits an image file takes
- Additionally there is (generally) a header file, giving (meta)-information about the image
- The meta-information informs the application program about the type of the image, the color coding method (rgb, cmyk), and other relevant data
- The header file is usually very small, a few tens of bytes
- Image files take significantly more space than text files, as expected

Size of an Image

- Consider the 500×500 image file we studied, represented in rgb:

- Since each pixel takes 3 bytes (24 bits), we calculate the (raw) size of this image as $3 \times 500 \times 500 = 750,000$ bytes or $6,000,000$ bits

Size of an Image

- However, the Unix command "ls -l" shows its actual size as 140,461 bytes

```
mac% ls -l
total 280
-rw-r--r--@ 1 koc  staff  140461 Feb 26  2014 leo.gif
mac%
```

- That is because the image file leo.gif is **compressed**
- The purpose of image compression is to have a significantly reduced file size for storage and transmission efficiency
- Most commonly used image compression methods are: gif, jpg, png

Image Compression Methods

- Each one of the file types (gif, jpg, png) refer to a different image compression method
- We will study GIF, which is the acronym of Graphics Interface Format
- GIF was developed by "CompuServe" and published in June 15, 1987: `http://www.w3.org/Graphics/GIF/spec-gif87.txt`
- CompuServe was the first major commercial online service
- CompuServe dominated the field in the 80s, up until mid-90s, along with AOL (which later purchased CompuServe)

GIF File Format

- The Hex-Edit program shows the first few bytes of the file `leo.gif` as

0	47494638 3761F401 F401F700	GIF87aô ô ÷
12	00000000 30020405 040B0A05	0
24	05130505 1A050608 06110A06	
36	0B13060A 07090A0C 0A0C0C0B	
48	14130B0C 060C0513 0C121B0C	
60	0C230C0B 140F1B2F 0F0C1B10	# /

- All gif image files has their first 6 bytes as `47 49 46 38 37 61` which encodes the ASCI text `GIF87a` to denote that this is gif image file, described in a standard document published in 1987
- These 6 bytes are also called "GIF Signature"
- The last three characters `87a` is the version number, and refers to one of the standards of GIF

GIF File Format

- The following 2 bytes shows the image size: First the width (the number of columns) and the height (the number of rows), represented in hexadecimal and the least significant byte first
- Since image file `leo.gif` is of size 500×500, and the hex representation of 500 is $01F4$, we see these two bytes ordered as $\boxed{F401\ F401}$

```
 0 47494638 3761F401 F401F700 GIF87aô ô ÷
12 00000000 30020405 040B0A05     0
24 05130505 1A050608 06110A06
36 0B13060A 07090A0C 0A0C0C0B
48 14130B0C 060C0513 0C121B0C
60 0C230C0B 140F1B2F 0F0C1B10  #     /
```

GIF File Format

- Consider this image, which is of size 500×400 (500 columns and 400 rows, or width is 500 while the height is 400 pixels)

- Since the hex representation of 500 is $01F4$ and 400 is 0190, we see these two bytes ordered as $\boxed{F401\ 9001}$

```
 0 47494638 3761F401 9001F700 GIF87aô . ÷
12 00000000 0605090A 06061306
24 06050A0D 0B0A0D14 0B0C0D0D
36 111B0D0D 0F100F24 100D1312        $
48 0E1C1211 39131014 14132314     9     #
60 1315151A 1A161C2A 16131A1A        *
```

GIF File Format

- The next byte holds the image and color map information
- It contains the following bits from left to right:

7	6	5	4	3	2	1	0
M	cr			0	$pixel$		

- M is a 1-bit; $M = 1$ implies a color map exists
- cr is 3 bits; $cr + 1$ gives the number of bits of color resolution
- $pixel$ is 3 bits; $2^{pixel+1}$ gives the size of color table

GIF File Format

- In image file `leo.gif` we have this byte as $F7 = (1111\ 0111)$

7	6	5	4	3	2	1	0
M	cr			0	$pixel$		
1	111			0	111		

- Thus, we have $M = 1$, $cr = 7$, and $pixel = 7$
- $cr = 7$ implies that the color resolution 8 bits,
- $pixel = 7$ implies that the size of the color table is $2^8 = 256$, **which means there are 256 colors in this image**

GIF File Format

- 256 is the maximum number of colors allowable in a GIF image
- A raw rgb image would have $2^{24} = 16,777,216$ colors, most of which are indistinguishable for human eye
- Also many displays are incapable of displaying such number of colors
- Therefore, GIF standard restricts the number of different colors as 256, and keeps them in a table inside the image, which is called the color table or the color palette
- Since there 256 different colors, and each color has 3 byte values, the maximum color table would have $3 \times 256 = 768$ bytes
- However, the color table will only keep the colors used in that particular image, and therefore, it could be shorter

GIF File Format

- Before getting into color tables, we should mention that the next byte (after `F7`) in `leo.gif` (which is `00`) is the background color index
- This byte represents the color in the global color table to be used for pixels whose value is not specified in the image data
- This byte should be zero, if there is no global color table or there is no need for a background pixel
- Finally the last byte (in `leo.gif`: `00`) is the aspect ratio of the pixel
- This byte is meaningful for pixels that are not square in size, which is applicable to certain printer and display technologies
- If this byte is zero, we have only square pixels, otherwise if it has the value of x (between 1 and 255), then the aspect ratio is $\frac{x+15}{64}$

GIF Color Table

- A randomly selected image (a bird, a flower, a person, sky, etc.) could have any set of colors in them
- It is therefore not wise to select a small set set of colors and force every imaginable image (!) to use these colors
- Instead we should let an image to have its "own palette" of colors
- This is exactly what GIF standard had done

GIF Color Table

- GIF provides a color table to specify up to 256 colors for an image
- The color table has at most 256 rows
- Since each color is 3 bytes, each row takes 3 bytes

index	red	green	blue
0	FF	00	00
1	00	FF	00
2	00	00	FF
...	...	...	...
255	FF	FF	FF

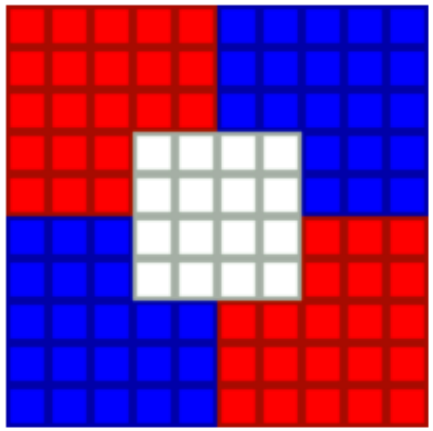
- However, some images may use much fewer than that, and thus, in such cases, the color table will be smaller

GIF Color Table

- The color table in image file `leo.gif` has 256 colors
- It starts with 00 00 00 and proceeds as 30 02 04 05 04 0B ...
- This means the first color used in the image is 00 00 00 which is black
- The next color is 30 02 04 which is a very dark red:
  `http://www.colorhexa.com/300204`
- The next color is 05 04 0B which is a very dark blue:
  `http://www.colorhexa.com/05040b`
- Another web site for displaying RGB colors:
  `http://www.rapidtables.com/web/color/RGB_Color.htm`

GIF Color Table

- Let us take a simple image of size 10×10 pixels



- This is a GIF89a file, whose representation is given as

```
47 49 46 38 39 61 0A 00 0A 00 91 00 00
FF FF FF FF 00 00 00 00 FF 00 00 00 21 F9 04
00 00 00 00 00 2C 00 00 00 00 0A 00 0A 00 00
02 16 8C 2D 99 87 2A 1C DC 33 A0 02 75 EC 95
FA A8 DE 60 8C 04 91 4C 01 00 3B
```

GIF Color Table

- The byte for color map in this file is `91: 1 001 0 001`
- Therefore, $M = 1$, $cr = 1$, and $pixel = 1$
- $M = 1$ implies that we have a global color table
- $cr = 1$ implies that the color resolution is 2 bits
- $pixel = 1$ implies the size of color table is $2^2 = 4$, i.e., we have only 4 colors in this image
- These colors are given in the image file in this order:
  `FF FF FF`: white
  `FF 00 00`: red
  `00 00 FF`: blue
  `00 00 00`: black

GIF File Format

- The next 8 bytes after the color table is called "Graphics Control Extension": `21 F9 04 00 00 00 00 00`
- The first two bytes are `21F9`, and is called the extension introducer, which always starts with `21F9`
- The third byte is the total block size in bytes: `04`
- The fourth byte is a packed field, containing several flag bits: `00`
- The delay time value follows in the next two bytes: `00`
- After that we have the transparent color index byte: `00`
- Finally we have the block terminator which is always `00`

GIF File Format

- The next 10 bytes is the image descriptor:
  `2C 00 00 00 00 0A 00 0A 00 00`
- A single GIF file may contain multiple images
- This is useful for creating animated images
- Each image begins with an image descriptor block
- This block is exactly 10 bytes long
- The first byte is the image separator: `2C`
- Every image descriptor begins with the value `2C`
- The next 8 bytes represent the location and size of the image

GIF File Format

- An image in the stream may not necessarily take up the entire canvas size defined by the logical screen descriptor
- Therefore, the image descriptor specifies the image left position and image top position of where the image should begin on the canvas
- These are $(x, y) = (00\ 00, 00\ 00)$ in this image
- The next 4 bytes specify the image width and image height: $(w, h) = (00\ 0A, 00\ 0A)$
- Our sample image starts at (0,0) and is of size 10×10
- This image does take up the whole canvas size

GIF File Format

- The last byte is another packed byte: 00
- In our sample file this byte is zero, so all bits are zero
- The first (most significant) bit in the byte is the local color table flag
- Setting this flag to 1 implies that the image data is using a different color table than the global color table
- The local color table is similar to the global color table
- If no local color table is specified, the global color table is used

GIF File Format

- Finally, we get to the actual image data:

```
02 16
8C 2D 99 87 2A 1C DC 33
A0 02 75 EC 95 FA A8 DE
60 8C 04 91 4C 01
00
```

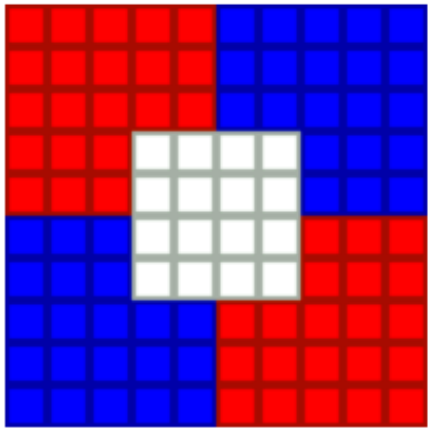
- The image data is composed of a series of output codes which tell the decoder which colors to spit out to the canvas
- These codes are combined into the bytes that make up the block
- GIF files are encoded using an encoding scheme called LZW, which stands for Lempel-Ziv-Welch
- LZW is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch

GIF File Format

- The first byte of this block is the LZW minimum code size
- The next byte indicates that there is 22 (Hex: 16) bytes of data
- These 22 bytes represent the compressed image data
- After we have read those 22 bytes, we see the next value is 00
- This marks the end of the image data
- Finally, all GIF files end with the trailer marker 3B

LZW Compression

- Finally, we need to understand how the 100 pixels in our sample image are to be encoded using only 22 bytes
  `8C 2D 99 87 2A 1C DC 33 A0 02 75`
  `EC 95 FA A8 DE 60 8C 04 91 4C 01`
- The color table had 4 colors, and the colors were placed in the color table in this order: white (0), red (1), blue (2), black (3)



FF0000 FF0000 · · · 0000FF 0000FF	1 1 1 1 1 2 2 2 2 2
FF0000 FF0000 · · · 0000FF 0000FF	1 1 1 1 1 2 2 2 2 2
FF0000 FF0000 · · · 0000FF 0000FF	1 1 1 1 1 2 2 2 2 2
FF0000 FF0000 · · · 0000FF 0000FF	1 1 1 0 0 0 0 2 2 2
FF0000 FF0000 · · · 0000FF 0000FF	1 1 1 0 0 0 0 2 2 2
0000FF 0000FF · · · FF0000 FF0000	2 2 2 0 0 0 0 1 1 1
0000FF 0000FF · · · FF0000 FF0000	2 2 2 0 0 0 0 1 1 1
0000FF 0000FF · · · FF0000 FF0000	2 2 2 2 2 1 1 1 1 1
0000FF 0000FF · · · FF0000 FF0000	2 2 2 2 2 1 1 1 1 1
0000FF 0000FF · · · FF0000 FF0000	2 2 2 2 2 1 1 1 1 1

LZW Compression

- However, even if we choose not to use LZW compression at this stage, we have already accomplished a lot
- Normally, 3-byte representation of colors would have required $100 \times 3 = 300$ bytes for the raw image
- The above matrix requires at most: $100 \times 1 = 100$ bytes, if we reserve one byte for every pixel
- However, the existing color indices are only one of these 3 values: $\{0, 1, 2\}$ (white, red, blue)
- We can reserve only 2 bits per pixel, and thus, we would need $100 \times 2 = 200$ bits!

LZW Compression

- The LZW algorithm is slightly more efficient
- It represents these 100 pixels using 22 bytes or 176 bits
- To understand how LZW works, we need to study the algorithm
- However, in order to understand how data compression methods work, we need to start with simpler methods, such as Huffman encoding
- We will study data compression methods later on in this course

Summary

Bytes	Meaning
`47 49 46 38 39 61`	GIF89a
`0A 00 0A 00`	Size: 10×10
`91`	Packed byte: `1 001 0 001`
	Color Table Flag $= 1$, cr $= 1$, pixel $= 1$
	Color Table Size $= 2^2 = 4$ colors
`00 00`	Background Color and Pixel Aspect Ratio
`FF FF FF`	White
`FF 00 00`	Red
`00 00 FF`	Blue
`00 00 00`	Black

Summary

Bytes	Meaning
21 F9 04 00 00 00 00 00	Graphics Control Extension
2C 00 00 00 00 0A 00 0A 00 00	Image Descriptor
02 16	LZW Code Size and the Number of Bytes
8C 2D 99 87 2A 1C DC 33 A0 02 75	LZW Encoded Image File
EC 95 FA A8 DE 60 8C 04 91 4C 01	
00	End of Image
3B	End of File