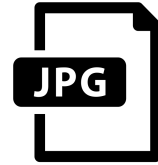
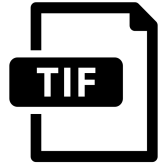
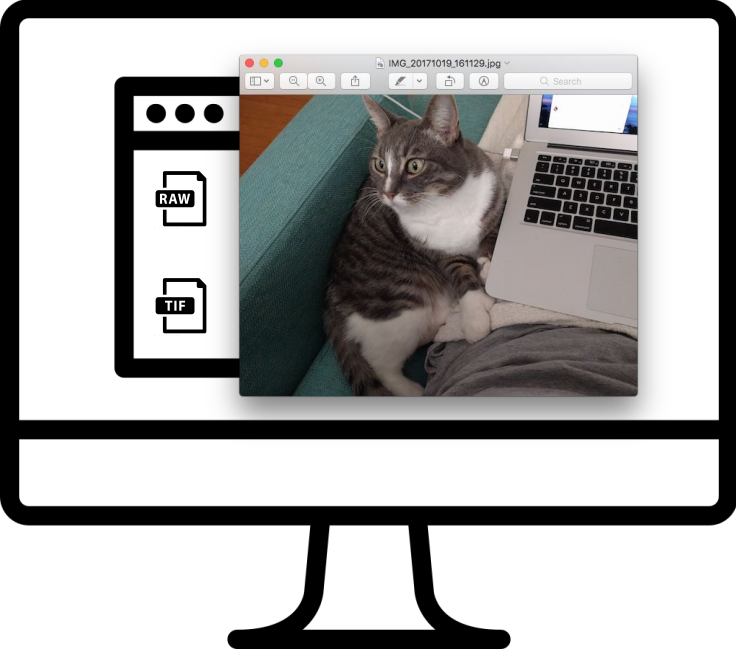


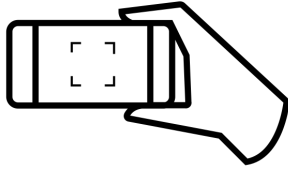
EE 376A Lecture 17: Image compression

From theory to practice

Irena Fischer-Hwang
March 5, 2019





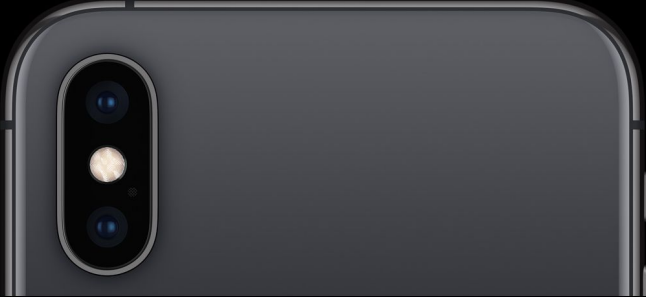


iPhone XS - Cameras - Apple

Apple Inc. [US] | https://www.apple.com/iphone-xs/cameras/

Display Face ID A12 Bionic Cameras Only iPhone Tech Specs Buy

Dual 12MP rear cameras



f/1.8
wide-angle lens

f/2.4
telephoto lens

Dual OIS
optical image stabilization

4K video
up to 60 fps

← Back up & sync

Upload, search, organize, edit & share your photos from any device

Help Center

Hi Irena, how can we help?

[Return to Facebook](#)

[Home](#) [Using Facebook](#) [Managing Your Account](#) [Privacy and Safety](#) [Policies and Reporting](#) [Support Inbox](#)

[Creating an Account](#)
[Friending](#)
[Your Home Page](#)
[Messaging](#)
[Stories](#)
Photos

Upload and Edit Photos

- Manage Albums
- Tagging
- Manage Photos of Your Child
- Using the Camera
- Fix a Problem

[Videos](#)
[Pages](#)
[Groups](#)
[Events](#)
[Payments](#)
[Marketplace](#)

How can I make sure that my photos display in the highest possible quality?

[Computer Help](#) [Mobile Help](#) [Share Article](#)

We automatically resize and format your photos when you upload them to Facebook. To help make sure your photos appear in the highest possible quality, try these tips:

- Resize your photo to one of the following supported sizes:
 - Regular photos: 720px, 960px or 2048px wide
 - Cover photos: 851px by 315px
- To avoid compression when you upload your cover photo, make sure the file size is less than 100KB
- Save your image as a JPEG with an sRGB color profile

You can also change your settings so that your photos are uploaded in HD by default.

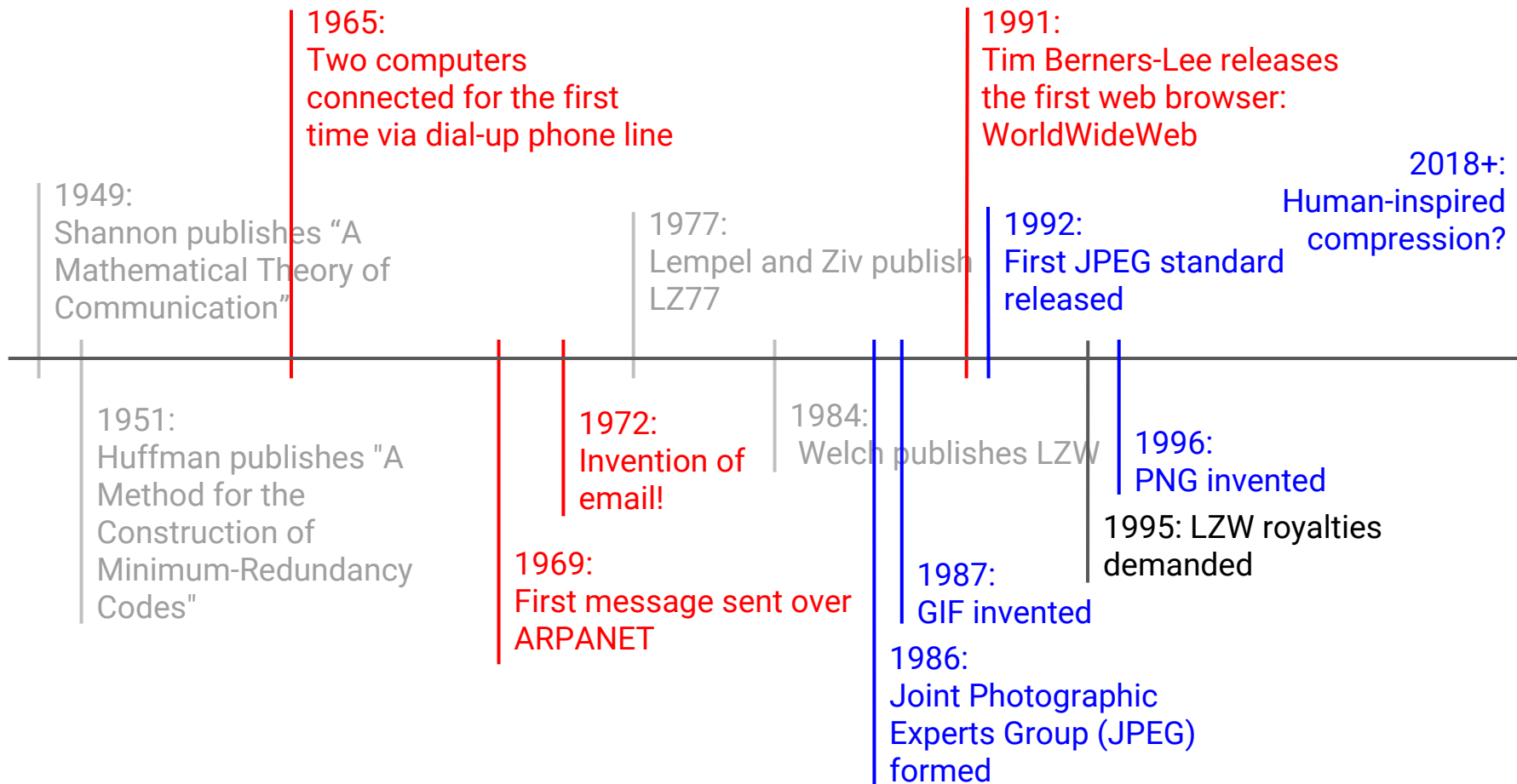
Was this information helpful?

Yes No

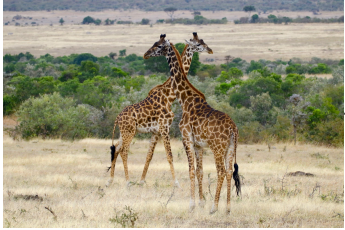
WHEN TO BACK UP

Roaming

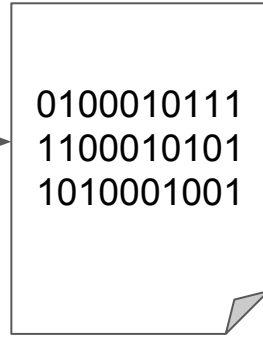




Original image
(bitmap)



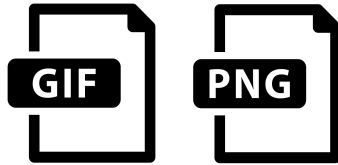
Compressed
bitstream/bytes



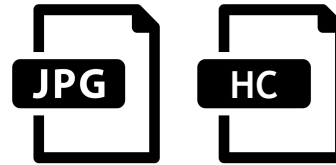
Reconstructed
Image
(bitmap)



lossless compression



lossy compression



Lossless compression, part 1: GIF



GIF: the basics

Graphics Interchange Format

One of the earliest developed image compression algorithms (1987)

Limited to 8-bit color space--each GIF image can contain only up to 256 different colors selected from a 24-bit RGB color space

Uniquely supports animations

Based on LZW compression scheme

Lempel-Ziv-Welch (LZW) compression

“A Technique for High-Performance Data Compression”

by Terry Welch, IEEE Computer 1984

Initialize table to contain single-character strings.

Read first input character \rightarrow prefix string ω

Step: Read next input character K

If no such K (input exhausted): code (ω) \rightarrow output; EXIT

If ωK exists in string table: $\omega K \rightarrow \omega$; repeat Step.

else ωK not in string table: code (ω) \rightarrow output;

$\omega K \rightarrow$ string table;

$K \rightarrow \omega$; repeat Step.

Lempel-Ziv-Welch (LZW) compression

Input

a b a b c b a b a b a a a a a

Initial string table/dictionary	
Index	Entry
1	a
2	b
3	c

Note

Initial dictionary is assumed to be known to both encoder and decoder. For example: for text compression the initial dictionary is the table of ASCII characters.

Lempel-Ziv-Welch (LZW) compression

Input character, K	ω_k	In string table?	Output, $\text{code}(\omega)$	ω_k / string table ind	ω
a		yes	nothing	none	a
ab	ab	no	1	ab / 4	b
aba	ba	no	1, 2	ba / 5	a
abab	ab	yes	no change	none	ab
ababc	abc	no	1, 2, 4	abc / 6	c
ababcb	cb	no	1, 2, 4, 3	cb / 7	b
ababcba	ba	yes	no change	none	ba
ababcbab	bab	no	1, 2, 4, 3, 5	bab / 8	b
ababcbaba	ba	yes	no change	none	ba
ababcbabab	bab	yes	no change	none	bab
ababcbababa	baba	no	1, 2, 4, 3, 5, 8	baba / 9	a
ababcbababaa	aa	no	1, 2, 4, 3, 5, 8, 1	aa / 10	a
ababcbababaaa	aa	yes	no change	none	aa
ababcbababaaaa	aaa	no	1, 2, 4, 3, 5, 8, 1, 10	aaa / 11	a
ababcbababaaaaa	aa	yes	no change	none	aa
ababcbababaaaaaa	aaa	yes	1, 2, 4, 3, 5, 8, 1, 10, 11	none	aaa

Lempel-Ziv-Welch (LZW) compression

Dictionary	
Index	Entry
1	a
2	b
3	c
4	ab
5	ba
6	abc
7	cb
8	bab
9	baba
10	aa
11	aaa

Encoded output

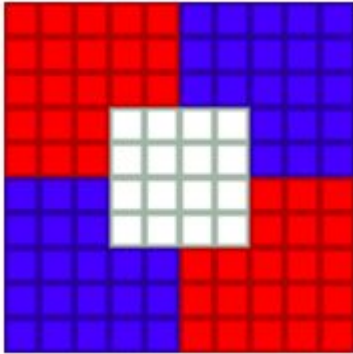
1, 2, 4, 3, 5, 8, 1, 10, 11

Input length: 17

Encoded output length: 9

LZW in GIFs part 1: pixels to index stream

*Note: All subsequent examples for the GIF section are from <http://giflib.sourceforge.net/whatsinagif/>

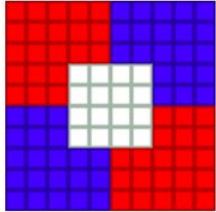


(10x10)

Index	Color
0	White
1	Red
2	Blue
3	Black

Index stream for first five lines: 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 0, 0, 0, 0, 2, 2, 2, 1, 1, 1, 0, 0, 0, 0, 2, 2, 2, ...

LZW in GIFs part 2: index stream to code stream



Code	Color index
#0	0 (White)
#1	1 (Red)
#2	2 (Blue)
#3	3 (Black)
#4	Clear Code
#5	End of Information Code

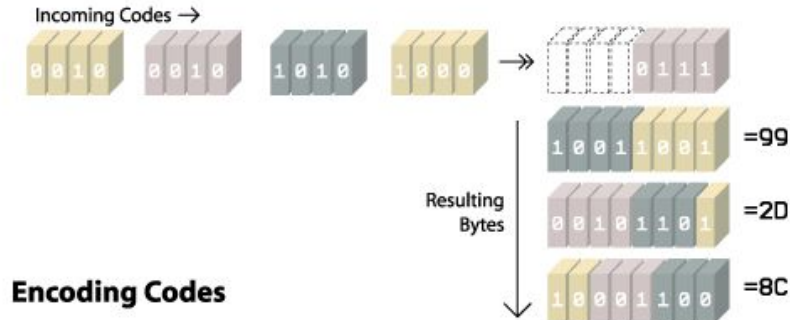
Step	Action	Index Stream	New Code Table Row	Code Stream
0	Init	1 1 1 1 2 2 2 2 2 1 1 1 1...		#4
1	Read	1 1 1 1 1 2 2 2 2 2 1 1 1 1...		#4
2	Not Found	1 1 1 1 1 2 2 2 2 2 1 1 1 1...	#6 - 1, 1	#4 #1
3	Read	1 1 1 1 1 1 2 2 2 2 2 1 1 1 1...		#4 #1
4	Found	1 1 1 1 1 1 2 2 2 2 2 1 1 1 1...		#4 #1
5	Read	1 1 1 1 1 1 2 2 2 2 2 1 1 1 1...		#4 #1
6	Not Found	1 1 1 1 1 1 2 2 2 2 2 1 1 1 1...	#7 - 1, 1, 1	#4 #1 #6
7	Read	1 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1...		#4 #1 #6
8	Found	1 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1...		#4 #1 #6
9	Read	1 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1...		#4 #1 #6

Final code stream 10x10 sample image: #4 #1 #6 #6 #2 #9 #9 #7 #8 #10 #2 #12 #1 #14 #15 #6 #0 #21 #0 #10 #7 #22 #23 #18 #26 #7 #10 #29 #13 #24 #12 #18 #16 #36 #12 #5

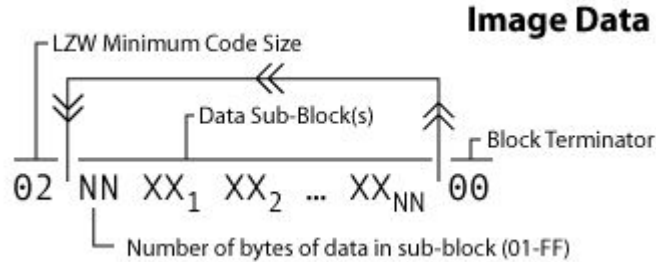
LZW in GIFs part 3: code stream to bitstream

1. Convert code stream into binary using a flexible code size
 - a. Avoids limiting max code to just 255 (8 bits)
 - b. Avoid wasting bits for small codes
2. Flexible code size is increased as soon as you write out code equal to $2^{(\text{current code size} - 1)}$
3. Start with minimum code size

Final code stream 10x10 sample image: #4 #1 #6 #6 #2 #9 #9 #7 #8 #10 #2 #12 #1 #14 #15 #6 #0 #21 #0 #10 #7 #22 #23 #18 #26 #7 #10 #29 #13 #24 #12 #18 #16 #36 #12 #5

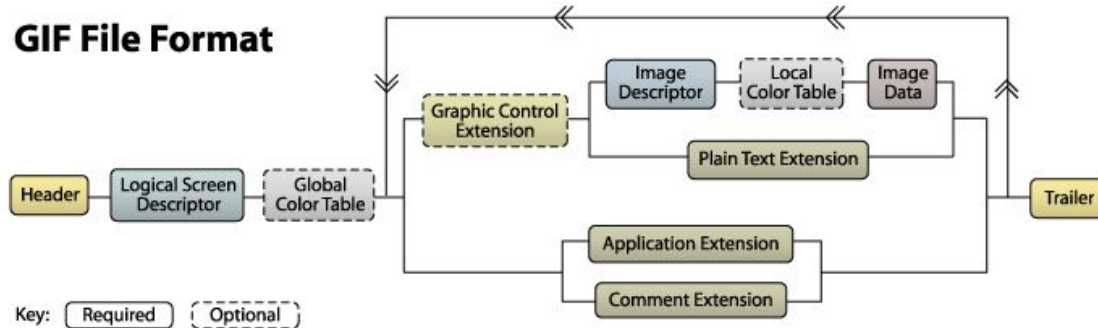


GIF files: all together now!

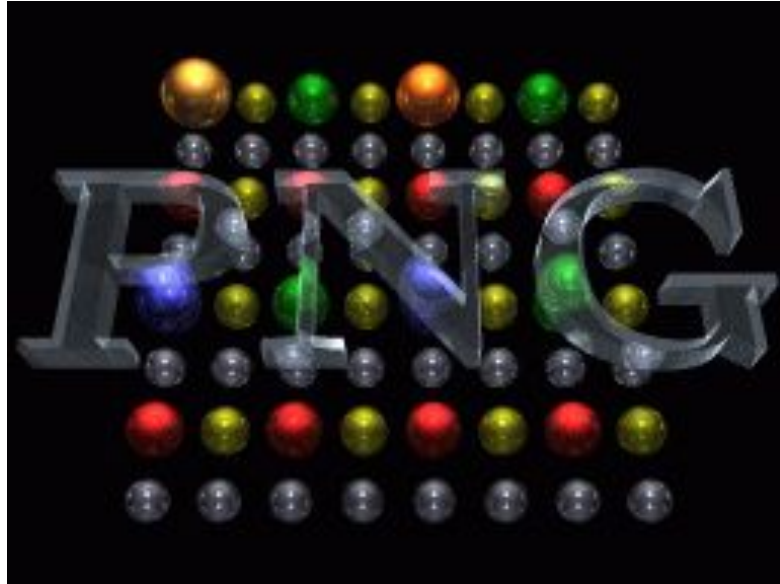


```
47 49 46 38 39 61 0A 00 0A 0091 00 00 FF
FF FF FF 00 00 0000 FF 00 00 00 21 F9 04 00
0000 00 00 2C 00 00 00 00 0A 000A 00 00 02
16 8C 2D 99 87 2A1C DC 33 A0 02 75 EC 95
FA A8DE 60 8C 04 91 4C 01 00 3B
```

GIF File Format



Lossless compression, part 2: PNG



PNG: the basics

PNG's Not GIF

Developed in 1996 to be lossless and patent-free

Adopted as international standard ISO/IEC 15948 in 2003

Supports both 8-bit and up to 48-bit color

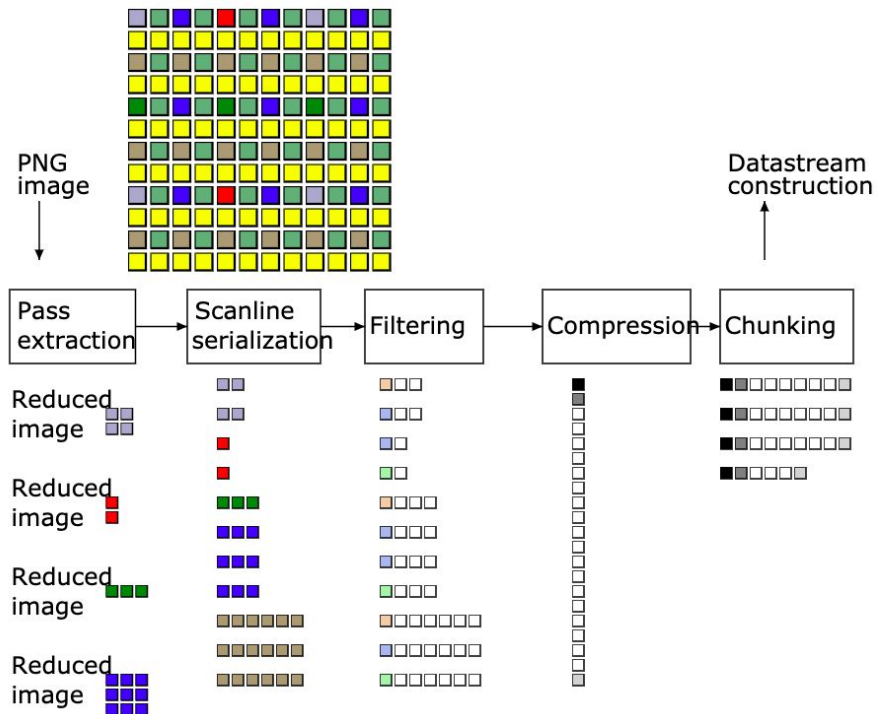
Uniquely supports full transparency

Based on DEFLATE (LZ77 + Huffman coding)

PNG workflow

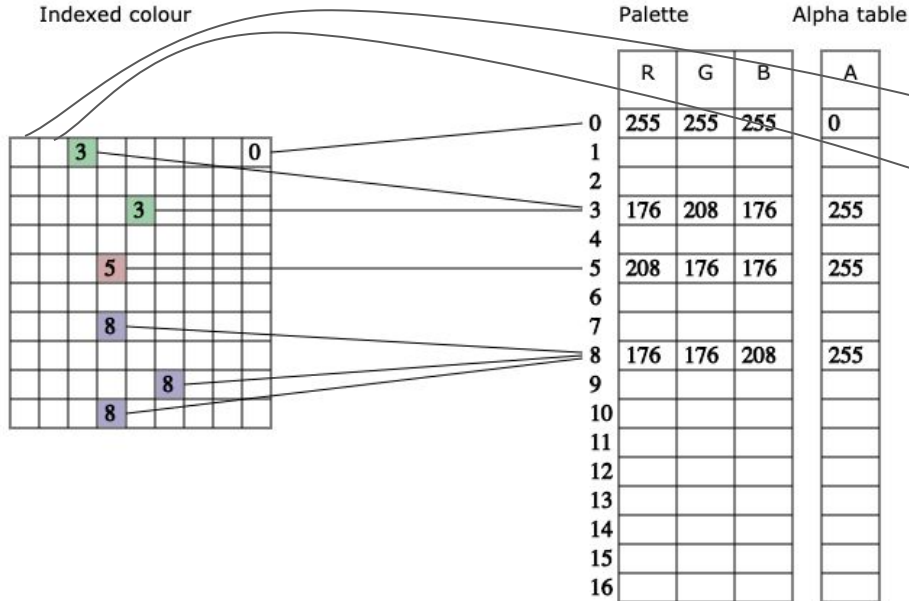
*Note: All subsequent information for the PNG section are from

<https://www.w3.org/TR/2003/REC-PNG-20031110/#4Concepts.PNGImageTransformation>

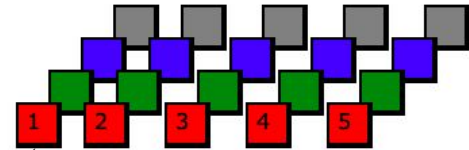


PNG part 1: pixels to index stream

As in GIF, pixels are first mapped to an index stream



Each row of pixels is called a "scanline" containing up to 4 channels per pixel



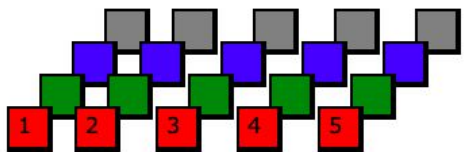
PNG part 2: filtering

Filtering allows patterns (e.g. regions of slowly varying colors) to be taken advantage of even when the data is incompressible (i.e. no repeated patterns). 5 Possible filter types:

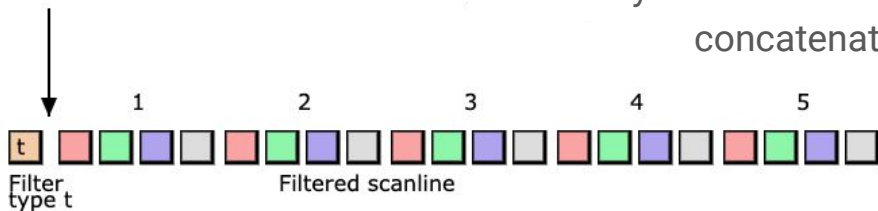
	c	b	
	a	x	

$\text{Raw}(x)$ = unfiltered byte value for pixel at position x

1. None
2. $\text{Sub}(x) = \text{Raw}(x) - \text{Raw}(a)$
3. $\text{Up}(x) = \text{Raw}(x) - \text{Raw}(b)$
4. $\text{Avg}(x) = \text{Raw}(x) - (\text{Raw}(a) + \text{Raw}(b))/2$
5. Paeth(x) = linear function of (a,b,c)



Filtering encodes each pixel x with the difference between the filtered (predicted) value and actual byte value at x . Rows of filtered pixels are concatenated for compression.



PNG part 3: DEFLATE compression

Defined by Phil Katz in the early 1990s

A “descendant” of LZ77 which uses:

1. A sliding window of up to 32 kilobytes and match length between 3 and 258 bytes. In other words: it looks for matches in pixel values between 3 and 258 pixels in length within the last 32,768 pixels.
2. A Huffman encoder to further encode the LZ77 codewords

PNG vs GIF

A comparison between PNG and GIF for 256 x 256, 8-bit greyscale images which all require 65,536 bytes (from <http://www.libpng.org/pub/png/book/LCH-png-chapter.pdf>)

image name	GIF (bytes)	standard PNG	
		(bytes)	vs. GIF
bird	47516	32474	-31.7%
bridge	76511	48511	-36.6%
camera	55441	38248	-31.0%
circles	1576	1829	+16.1%
crosses	1665	2000	+20.1%
goldhill	68450	44941	-34.3%
horiz	683	693	+1.5%
lena	71115	41204	-42.1%
montage	42449	24096	-43.2%
slope	29465	11683	-60.3%
squares	931	640	-31.3%
text	4205	2339	-44.4%
total size	400007	248658	-37.8%

GIF typically 4:1 - 10:1, PNG typically 10-30% smaller than GIFs

Can we do better?

YES.

But only if we throw out some data.

Lossy compression, part 1: JPEG



■ JPEG ■

JPEG: the basics

Developed by the Joint Photographic Experts Group in 1992

Adopted as international standard ISO/IEC 10918 in 1994

Unlike GIF and PNG, JPEG standard specifies the *codec* (coder/decoder), not the file format--those are specified by Exif and JFIF standards

Like all lossy compression algorithms, JPEG throws information out based on assumptions about how human perceive images

JPEG performs lossy compression through two steps (color space sampling, DCT coefficient quantization) and lossless Huffman coding

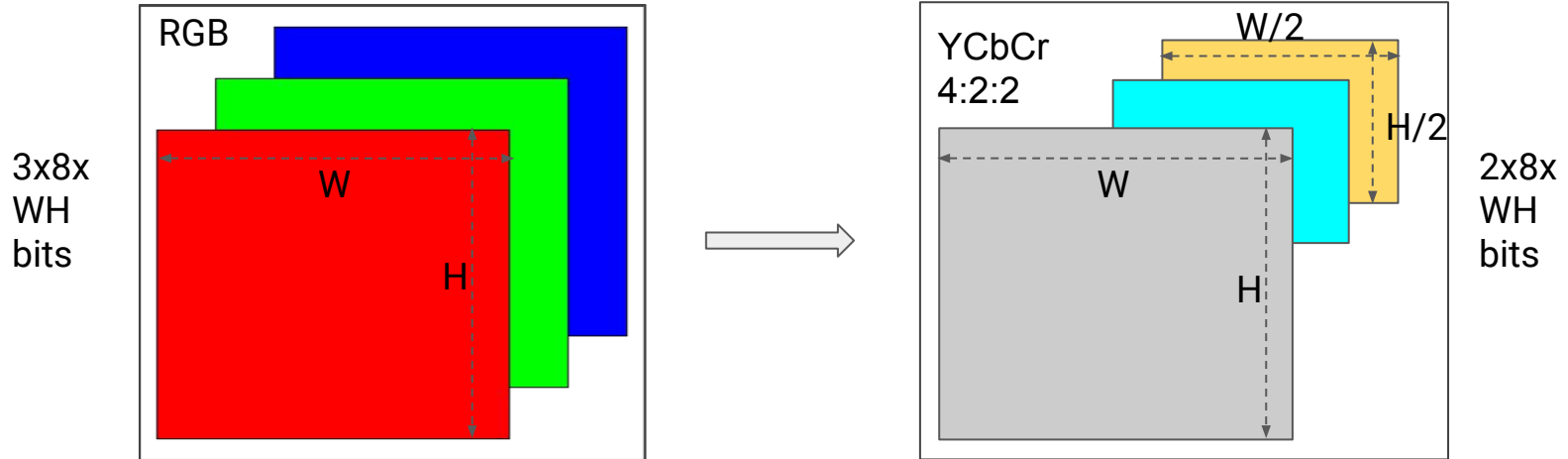
JPEG part 1: color space transformation

First, RGB pixels are converted to YCbCr colorspace:

Y = luma, or how bright the pixel is

Cb and Cr are color difference components: Cb = blue - luma and Cr = red - luma

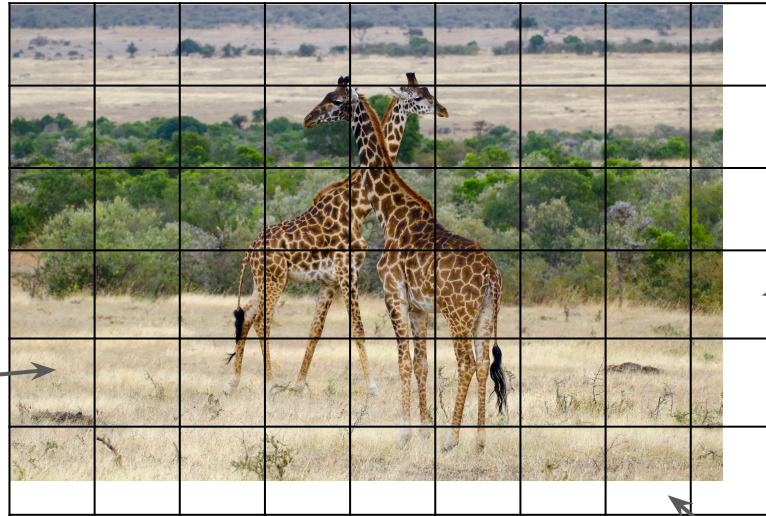
Psychovisual experiments have shown that humans discriminate brightness much more finely than color, so Cb and Cr downsampled to half the Y resolution



JPEG part 2: block splitting

Each channel is split into 8x8 pixel blocks, with padding as necessary (which might later lead to artifacts)

8x8 pixel
blocks (a
historical
artifact!)



Padding of
right boundary
blocks

Padding of
lower boundary
blocks

JPEG part 3: discrete cosine transform

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} \xrightarrow{\text{Center to } [-128, 127]} \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

$$G_{u,v} = \frac{1}{4} \alpha(u) \alpha(v) \sum_{x=0}^7 \sum_{y=0}^7 g_{x,y} \cos \left[\frac{(2x+1)u\pi}{16} \right] \cos \left[\frac{(2y+1)v\pi}{16} \right]$$

$$G = \begin{matrix} & \begin{matrix} u \\ \rightarrow \end{matrix} \\ \begin{matrix} \downarrow v \end{matrix} & \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \end{matrix}$$

JPEG part 4: quantization

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

$$B_{j,k} = \text{round} \left(\frac{G_{j,k}}{Q_{j,k}} \right) \text{ for } j = 0, 1, 2, \dots, 7; k = 0, 1, 2, \dots, 7$$

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

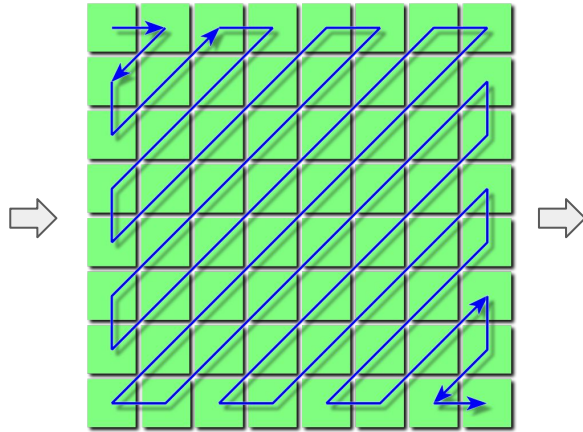
$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As with chroma sampling, quantization is based on psychovisual experiments: the human eye is generally more sensitive to low frequencies than it is to high frequencies

JPEG part 5: Entropy coding

Step 1: perform run-length coding on AC values (any after DC component) in tuples of (run length of zero, next non-zero value). Done in a zigzag, since the DCT coefficients “radiate” out from the top left corner :

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



(0,-3), (1,-3), (0,-2), (0,-6), (0, 2)...

Step 2: use Huffman coding on run-length encoded tuples

JPEG: past and present

1992: JPEG (Joint Photographic Experts Group)

Typically achieves compression ratios between 5:1 to 120:1, typically ~10:1

2000: JPEG 2000

Supports lossless compression

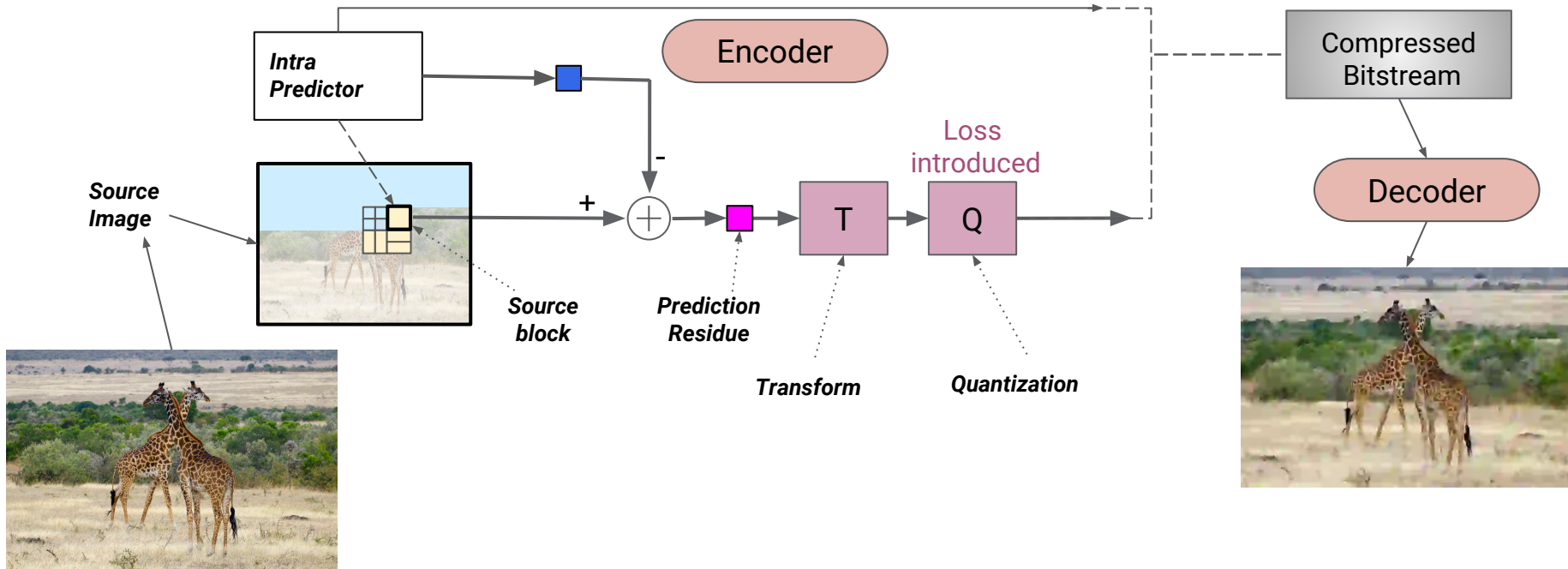
Less visible artifacts and blocking compared to JPEG

Similar compression ratios to JPEG

Never made it mainstream due to compatibility issues :(

Modern lossy compression: WebP

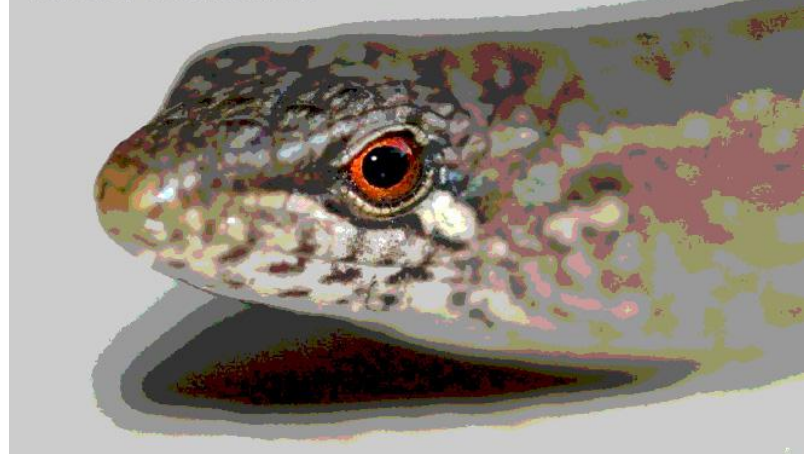
Developed in 2010, achieves (25%-30% improvement over JPEG) using block prediction and entropy coding



Flaws of lossy compression

At very low bitrates, image reconstructions aren't very good

Type of artifacts include: staircase noise (aliasing) along curving edges, blockiness, and posterization



Evaluating lossy compressors: fidelity metrics

Fidelity metrics operate pixel-by-pixel, e.g.:

Mean square error

Peak signal-to-noise ratio

Mean absolute error

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2$$

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

$$= 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right)$$

$$= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE)$$

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n}$$

Evaluating lossy compressors: SSIM

Fidelity metrics are simple and have clear physical meanings, but do not reflect **perceived visual quality**

The structural similarity (SSIM) metric was developed in 2004 to incorporate high-level properties of the human visual system:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad \text{MSSIM}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^M \text{SSIM}(\mathbf{x}_j, \mathbf{y}_j)$$

$$c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$$

SSIM is calculated on various windows x and y of common size $N \times N$ from two images, using averages, variances, covariances, and dynamic ranges obtained from the images (plus some constants)

Intuitively, SSIM tries to

SSIM vs. MSE

A number of distorted versions all with the same MSE = 210:



(a)



(b)



(c)



(d)



(e)



(f)

Index	Distortion type	MSSIM
(a)	None (original)	1
(b)	Contrast-stretched	0.9168
(c)	Mean-shifted	0.99
(d)	JPEG compressed	0.6949
(e)	Blurred image	0.7052
(f)	Salt-pepper impulsive noise contaminated	0.7748

Evaluating lossy compressors: perceptual metrics

Perceptual PSNR (P-PSNR)

$$\text{P-PSNR} = 10 \log_{10} \frac{255^2}{D_p^2}$$

based on image distortion D_p (and a bunch of other parameters measured using psychovisual experiments):

$$D_p = \frac{1}{N} \left(\sum_{i,k} \max \left\{ \frac{|\hat{b}(i,k) - b(i,k)|}{t(i,k)}, 1 \right\}^{Q_s} \right)^{\frac{1}{Q_s}}$$

2018: Guetzli (Google)

Based on butteraugli, a metric that “estimates the psychovisual similarity of two images.” Also generally achieves 30% more compression than JPEG, fairly memory-intensive

Lossy compression, part 2: Human compression

Human compression: the basics

We wondered: can we design a better compression paradigm by directly asking humans what they care about?

Key components: 1) humans as “describers” and “reconstructors,” 2) side information in the form of publicly available images on internet, 3) evaluations also based on humans

Lossy human compression

Components: two humans!

Plus photoshop. Plus Skype. Plus Google Images.

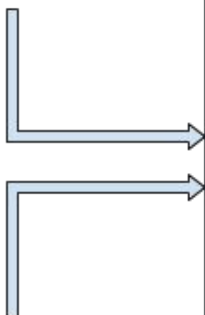
“Describer”: uses text-based commands to describe a selected image

“Reconstructor”: attempts to recreate the image using photo-manipulation software



Original image

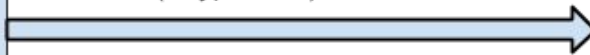
Only Describer sees the original image



Links of Public Images from the Internet



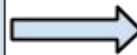
Text chat (Skype chat)



Audio feedback (voice chat)
Visual feedback (screen-share on Skype)



Both have access to the internet



Photoscape X



Final Reconstruction

Reconstructor Stages

Skype Chat Excerpts



<https://www.worldwildlife.org/habitats/grasslands>



Try transformations
elongate the fence bit
only focus on the vertical...



there's a line of shrubbery that goes across the middle third of the image...
that's the largest bush in the pic
so keep the others sizes equal to or smaller that that and make it look
continuous
and make sure to make the bushes smaller as you work your way up so
that there's a sense of depth...



there will be a line of tiny shrubs along that line...
the line itself starts about a quarter from the left...



try and make the grass look less tall on the bottom...



Final reconstruction



Original image

when you're done with that take a look at these
https://public-media.smithsonianmag.com/file/32/f2/32f24473-b380-43f5-94df-da0e58644439/16301090250_acf80be87f_o.jpg
<https://img.purch.com/w/192/aHR0cDovL3d3dy5saXZlc2NpZW5jZS5jb20vaW1hZ2VzL2kvMDAwLzA2OC8wOTQvaTMwMC9naXJhZmZlLmpwZz8xNDA1MDA4NDQy>
sure
while you're editing that giraffe
its spots are too dark
make it look like the other giraffe...

• make the right one bigger than the left
make the heads level
wait back
put the left one where it was before
good
now move the right giraffe to the left so that their necks cross
good
move them both to the center
make them both taller as well
their heads should be above the middle line of shrubs...

• there's a ridgeline in the back
of very dense shrubs
but let's try something
I want you to place a shrub on the very top of the image
and stretch it from left to right...
it should be less green make it look hazier if that makes sense...

Creating compressed files

The compression experiment proceeds until the describer is satisfied with the reconstructed image. Then:

- Timestamps are removed from the describer's text transcript
- The transcript is compressed using the bzip2 compressor
- We also use WebP to generate a compressed image of a similar size as the compressed text transcript

```
k nice
ok gimme a sec
just a heads up its a
photo with a sunset and a
bunch of balloon
im trying to find similar
sunsets and ballons rn
*hot air ballons
https://
www.stockcutouts.com/Hot-
Air-Balloon-
Silhouette#.Wx7BZl0UvGI
cut this out some how
like maybe screenshot it?
ok ok that works
```

balloons_data.txt

Plain Text - 5 KB

Evaluation

We compare the quality of compressed images using human scorers (workers) on Amazon Mechanical Turk, a platform for conducting large scale surveys

For each image, we display the original image and the human reconstruction and ask the workers to rate the reconstruction on a discrete scale of 1 to 10

To capture the effects of human perception, the scale represents a general “level of satisfaction” with the reconstruction rather than a specific metric like accuracy

We perform identical experiments for the WebP reconstructions. For every experiment, we collect 100 survey responses and obtain summary statistics

What a worker would see:

Instructions

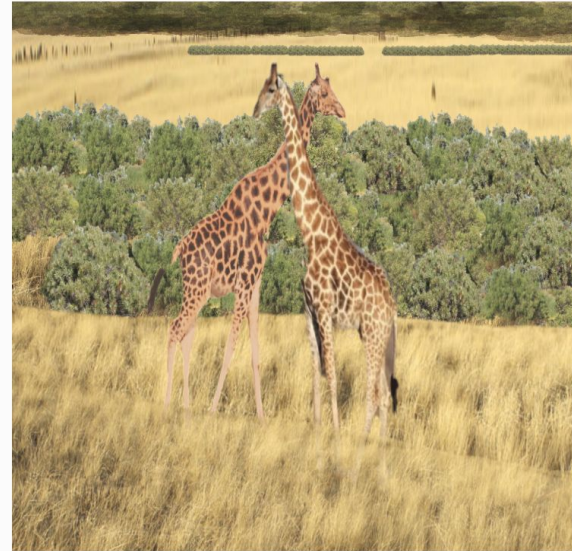
The second image is a reconstruction of the first image.

- Compare the two images and rate your level of satisfaction from the reconstruction using the scale below (1=completely unsatisfied, 10=completely satisfied).

Original Image:



Image Reconstruction:



Level of Satisfaction:

- 1 (completely unsatisfied) 2 3 4
 5 6 7 8 9
 10 (completely satisfied)

Preliminary results

→ Mturk scores for Human and WebP reconstruction

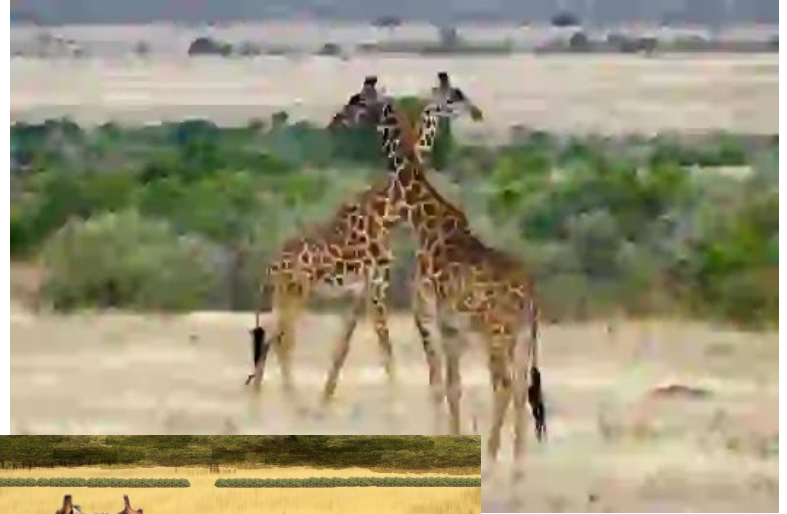
Image	Original size (KB)	Compressed chat size (KB)	WebP size (KB)	Mean score		Median score	
				Human	WebP	Human	WebP
arch	1119	3.805	3.840	4.04	5.1	3	5
balloon	92	1.951	2.036	6.22	5.45	7	6
beachbridge	3263	4.604	4.676	4.34	3.92	4	4
eiffeltower	2245	4.363	4.394	5.98	5.77	6	6
face	1885	2.649	2.762	2.95	5.47	3	6
fire	4270	2.407	2.454	6.74	5.09	7	5
giraffe	5256	3.107	3.144	6.28	4.48	7	4
guitarman	1648	2.713	2.730	4.88	4.07	5	4
intersection	3751	3.157	3.238	6.8	4.15	7	4
rockwall	4205	6.613	6.674	4.41	4.85	4	5
sunsetlake	1505	4.077	4.088	5.08	4.82	5	5
train	3445	1.948	2.024	6.85	3.62	7	3
wolfsketch	1914	0.869	0.922	8.25	3.46	9	3

Selected Visual Results

Original (5.3 MB)



WebP score: 4/10



Human
Compressed
(3.1 KB)
score: 7/10

WebP score: 3/10

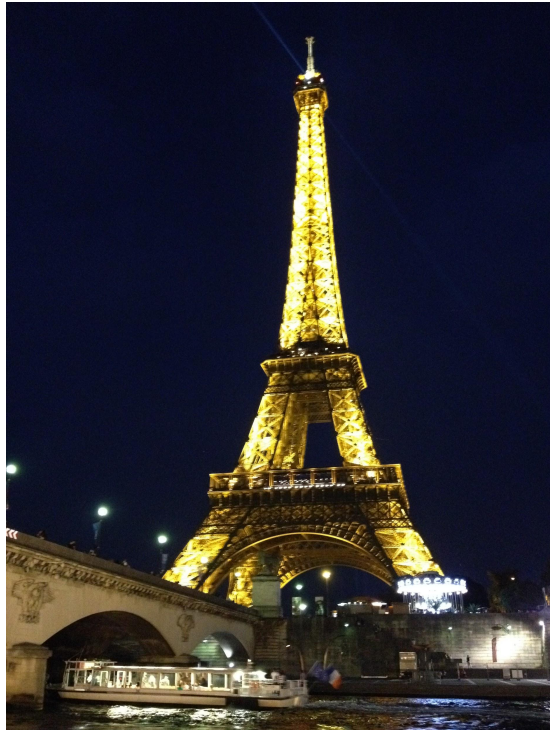


Original (3 MB)

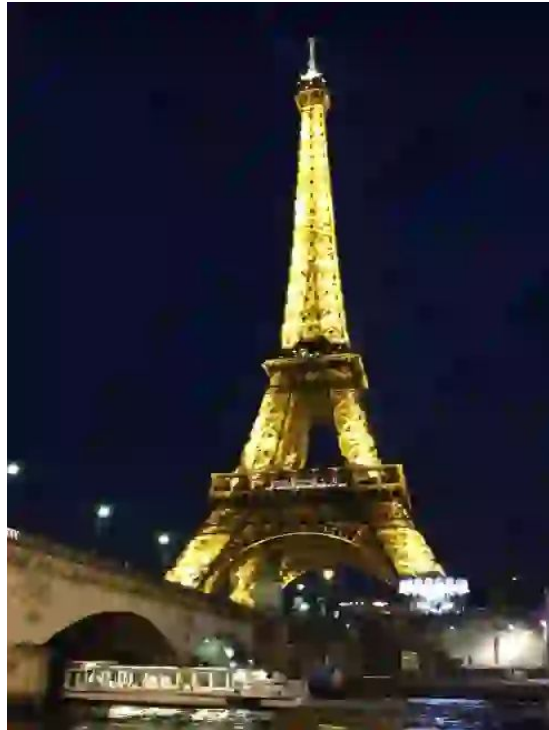


Human
Compressed (2 KB)
score: 7/10

Original (2.2 MB)



WebP score: 6/10



**Human Compressed (4.4 KB)
score: 6/10**



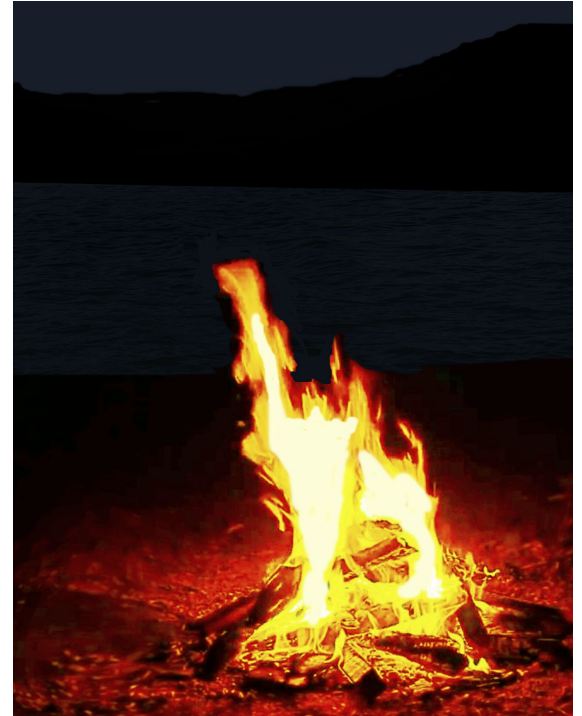
Original (4.2 MB)



WebP score: 5/10



**Human Compressed (2.4 KB)
score: 7/10**



Original (92 KB)



WebP score: 6/10



**Human
Compressed (2 KB)
score: 7/10**

Original (1.5 MB)



WebP score: 5/10



**Human Compressed
(4 KB) score: 5/10**

Original (1.9 MB)



WebP score: 6/10



**Human Compressed (2.6 KB)
score: 3/10**



Human compression takeaways

Not a practical compression scheme at the moment, but

Our experiment shows that human-centric compression based on human language can be more powerful than traditional compression at very low bit rates

We learned that utilization of semantically and structurally similar images can dramatically improve compression ratio

Showed that leveraging the growing library of publicly-available images could be the backbone of a new lossy compression framework

Future of human compression

Components of a practical version of the scheme might be implemented using artificial intelligence, e.g. <http://www.wave.one/face-compression>

Not necessarily tied to natural language

Use GANs to both “describe” and “reconstruct” images

Use neural networks to predict human scores of image reconstructions

Leveraging “side information” in the form of publicly-available images