Data Representation: Images and Sound

Announcements

- Lab Exam?
- Tomorrow:
  - PS 7
  - Lab 8
  - PA 7

Yesterday

- Data Representation
- Data Compression
- Information and redundancy
- Huffman Codes

ALOHA

Fixed Width:

0001 0110 1001 0011 0001

20 bits

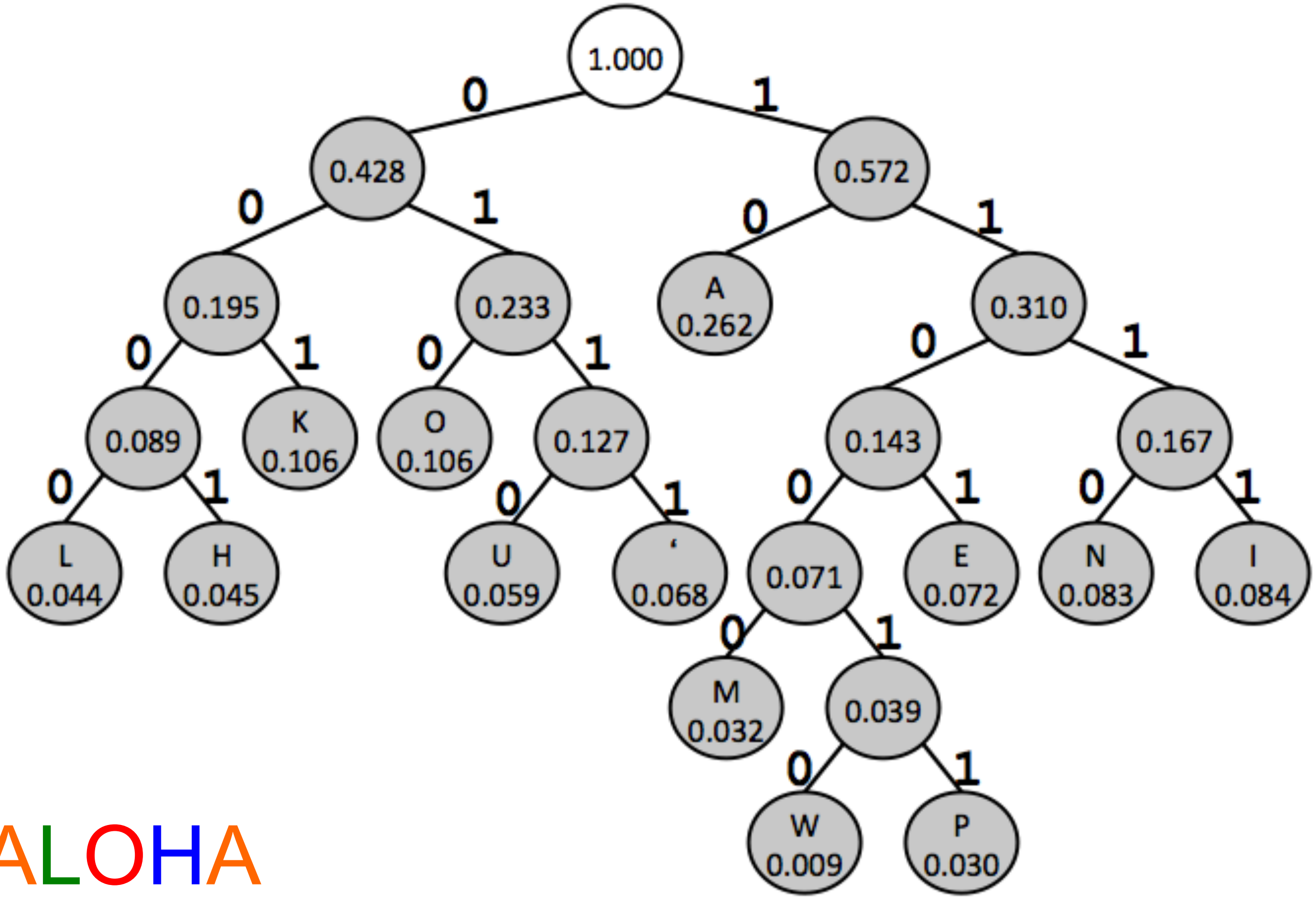
Huffman Code:

10 0000 010 0001 10

15 bits

Decoding

100000010000110



ALOHA

- To find the character use the bits to determine path from root

parity bits

error correction

Noisy Communication Channels

- Suppose we’re sending ASCII characters over the network
- Network communications may erroneously alter bits of a message
- Simple error detection method: **the parity bit**

Reminder: ASCII table

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

- 2^7 (128) characters
- 7 bits needed for binary representation
- (Not shown: control characters like tab and newline, values 0…31)

Parity

- Idea: for each character (sequence of 7 bits), count the number of bits that are 1

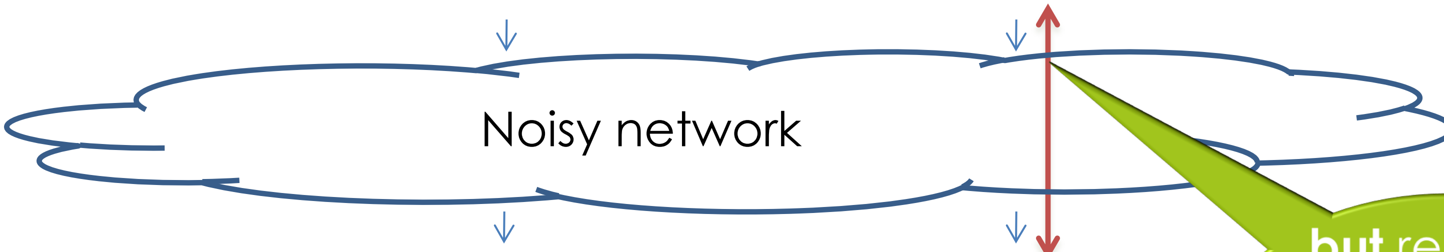
- Sender and receiver agree to use *even parity* (or *odd parity*); sender sends **extra** leftmost bit
  - Even parity: Set the leftmost bit so that the number of 1′s in the byte is even.

Parity Example

- "M" is transmitted using **even parity**.
- "M" in ASCII is 77_{10}, or 100 1101 in binary
  - four of these bits are 1
- Transmit 0 100 1101 to make the number of 1-bits **even**.
- Receiver counts the number of 1-bits in character received
  - if odd, something went wrong, request retransmission
  - if even, proceed normally
  - Two bits could have been flipped, giving the illusion of correctness. **But** the probability of 2 or more bits in error is low.

Parity Example

	H	I	
7 bit code	1 1 0 1 0 0 0	1 1 0 1 0 0 1	
Transmit	1 1 1 0 1 0 0 0	0 1 1 0 1 0 0 1	**Even parity**



Receive	1 1 1 0 1 0 0 0	0 1 1 0 0 0 0 1	

but receiver can't tell where the error is

Odd number of ones. There must be an error in transmission

Parity and redundancy

- An ASCII character with a correct parity bit contains *redundant information*
- …because the parity bit is *predictable* from the other bits
- This idea leads into the basics of information theory

Today:

- Human sensory systems and digital representations
- Digitizing images
- Digitizing sounds
- Video

human sensory systems

Why Do We Care?

- We want to represent and reproduce sensory experiences – sights and sounds
  - typically this leads to storing a huge amount of data
- Data compression for images and sounds can exploit limits on human senses
  - throw away information not needed for good-quality experience

Human Limitations

- Range
  - only certain pitches and loudnesses can be heard
  - only certain kinds of light are visible, and there must be enough / not too much light
- Discrimination
  - pitches, loudnesses, colors, intensities can't be distinguished unless they are different enough
- Coding
  - nervous systems "encode" experience, e.g. rods and cones in the eye

images

digitizing

Human Vision

- Separate receptors for gray scale/dim light and color/bright light perception
- Receptors feed nervous subsystems that respond to contrast and other factors
- Spatial and temporal limits on discrimination
  - e.g. affect frame rates in video
- Three kinds of color receptors (RGB)

Encoding Images:
Vector vs. Raster / Bit-map

- There are two major ways to store images:
  - Vector graphics:
    a series of lines or curves. Expensive to compute but smoothly rescales.
  - Raster or Bit-map graphics:
    an array of pixels. Cheap to compute, but scales poorly.

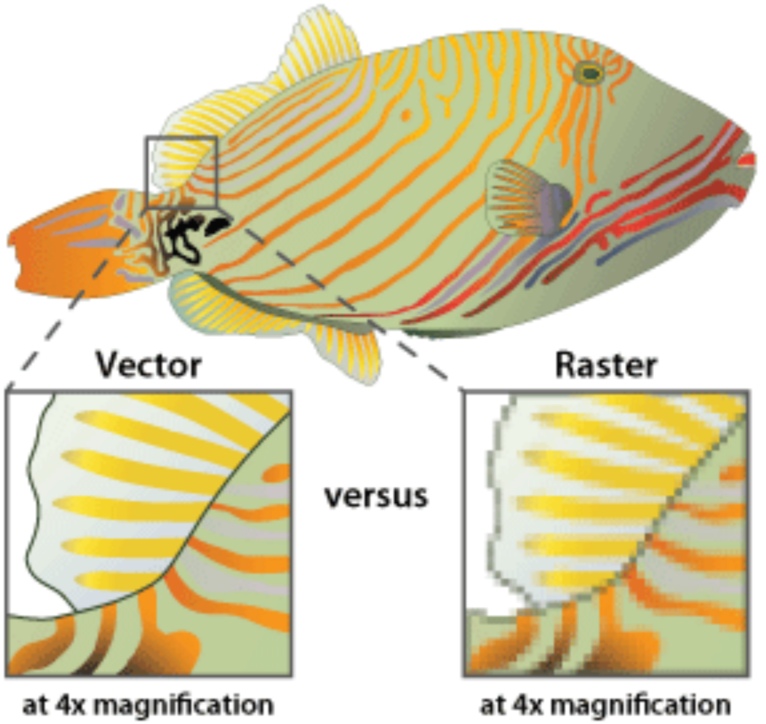


image source: ian.umces.edu

Recording Photographic Images

- How do digital cameras record images?
- Basic idea: array of receptors: bit-map
  - each receptor records a *pixel* by "counting" the number of photons that strike it during exposure
- Red, green, blue recorded separately
  - each point on image produced by group of three receptors
  - each receptor behind a color filter

“Raw” Bit-Mapped Images

- Array of pixels
  - one pixel = three numbers (RGB)
- What other information do we need to display the image?
  - look at TIFF file
  - image is just a bunch of numbers
  - we need to know how wide/high it is to make sense of it

“Raw” Bit-Mapped Image Example

255, 0, 0, 255, 0, 0, 255, 0, 0, … 255, 0, 0

Image Formats

- Exploit human perceptual system in quality/size tradeoff
- Exploit specialized types of images to get a lot of compression

Common Standards

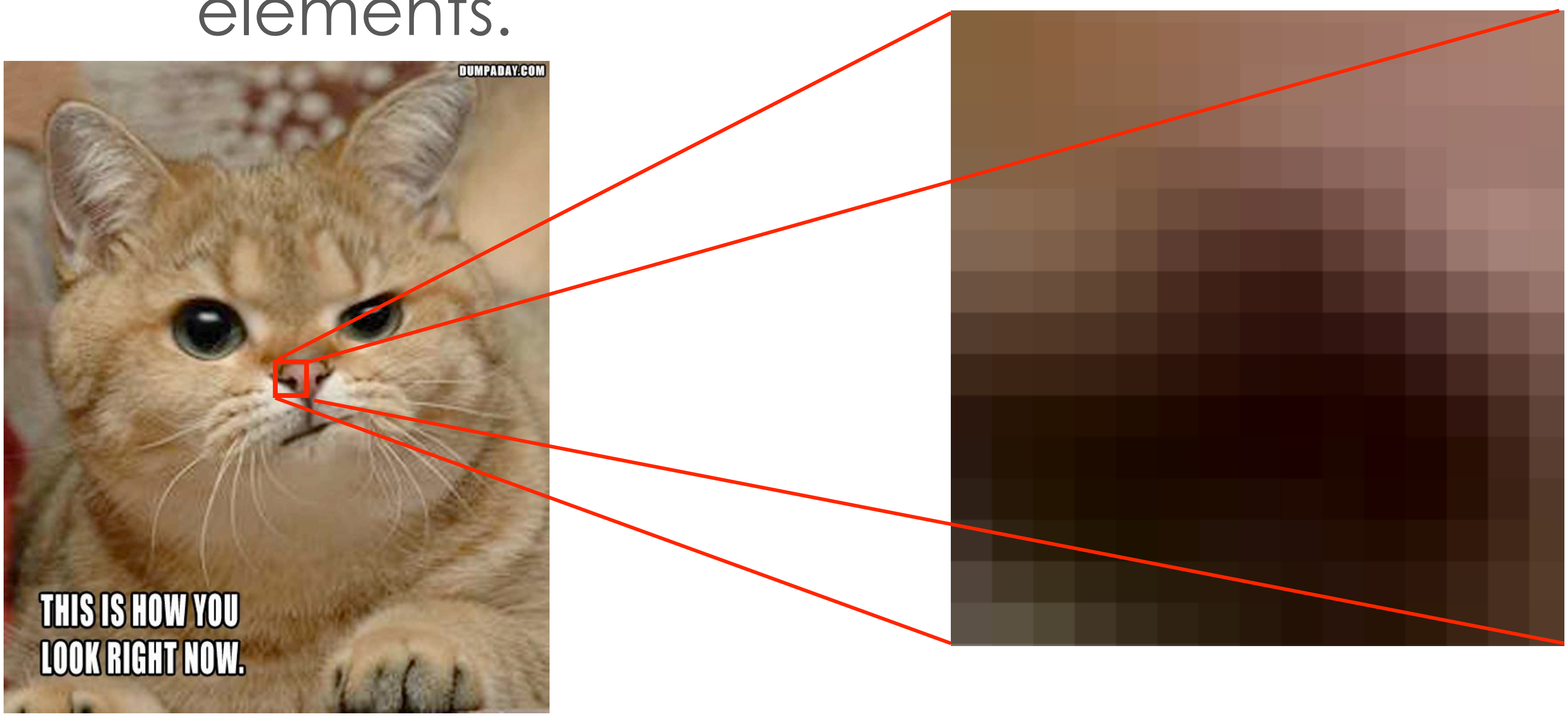
- Vector: SVG, EPS, AI, CDR.
  - Special-purpose: commonly used for high-quality illustrations, graphics, etc.
- Raster: JPEG (compression), GIF (compression, transparency), PNG (web portability), TIFF (printing, huge), BMP (huge)
  - Commonly used for photos and pretty much everything

Bit mapped images

A closer look.

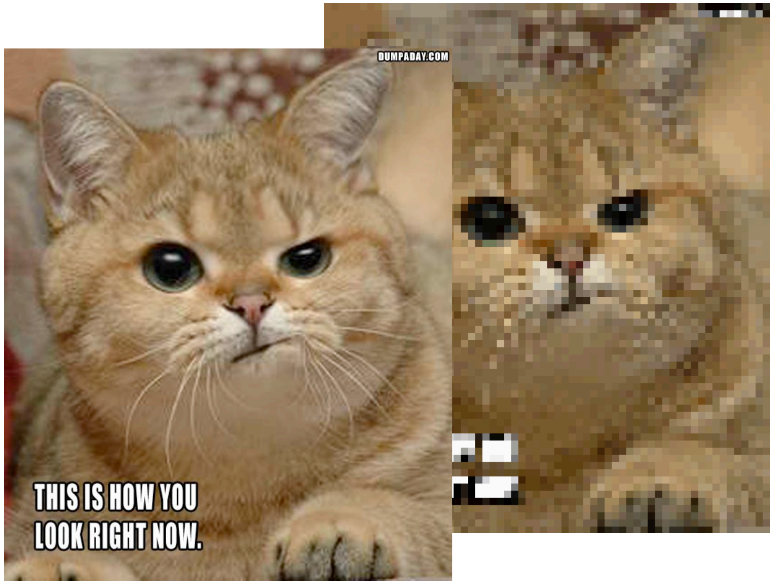
Pixels

- A bit-mapped image is stored in a computer as a sequence of *pixels*, picture elements.



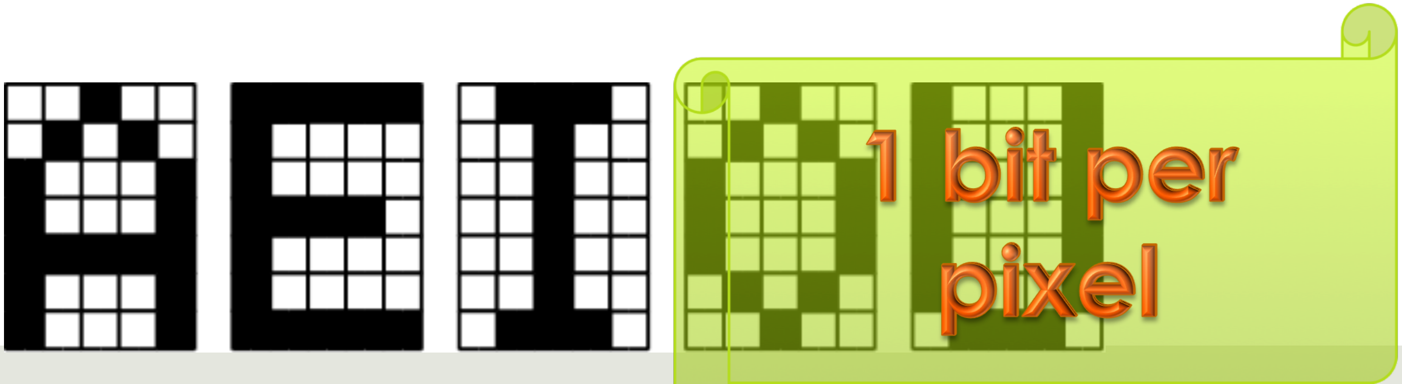
Resolution

- The resolution of an image is the number of pixels used to represent the image (e.g. 1024 X 768).
- Each pixel represents the average color in that region.
- The more pixels per area, the higher the resolution, and the more accurate the image will appear.



Storing Bitmap Images

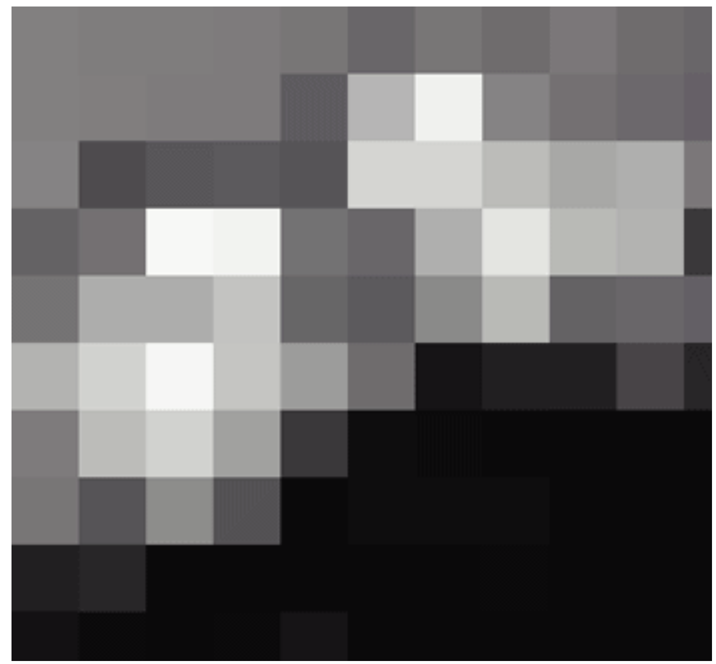
- In bitmapped images, each pixel is represented in computer memory in binary, just like other data types.
- If pixels of an image are black or white only, then we only need 1 bit per pixel to store the image, e.g. 00100 might be top row of "A".



Grayscale Images

- Grayscale images contain pixels that are various shades of gray, from black (maximum gray) to white (minimum gray).

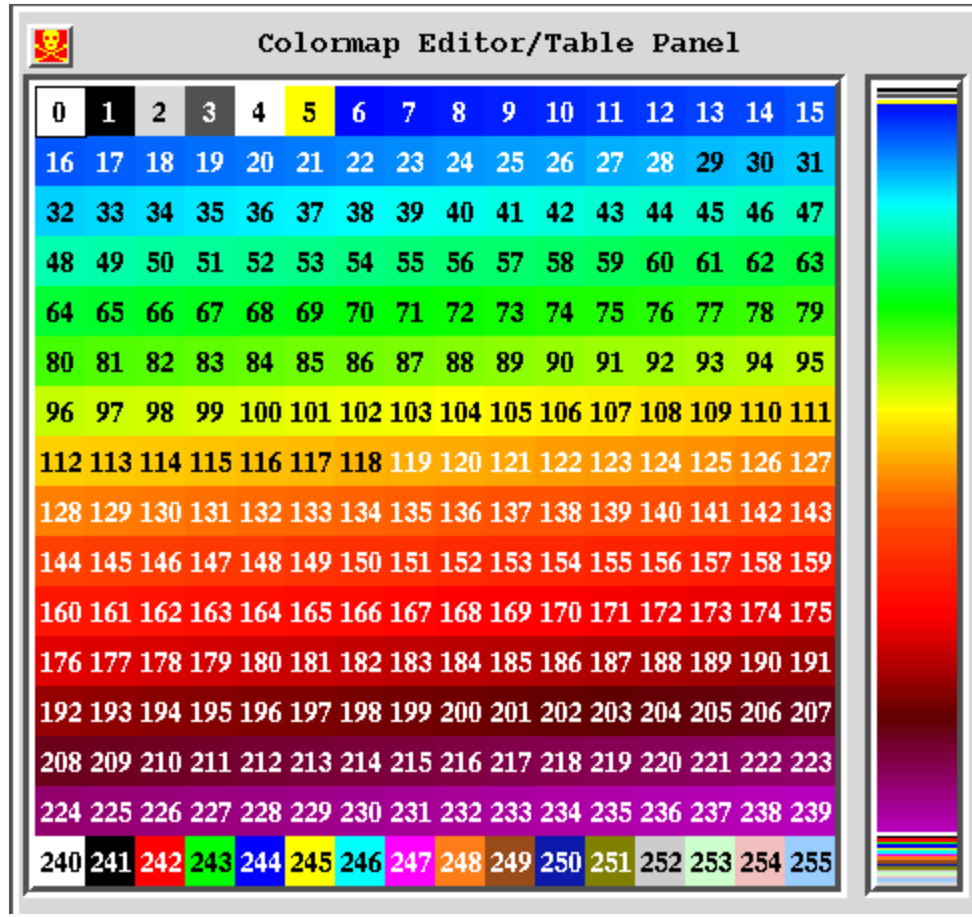
- If there are 256 levels of gray for pixels, we can represent each pixel using 8 bits.
  11111111 = white
  …   (shades of gray)
  00000000 = black



8 bits per pixel

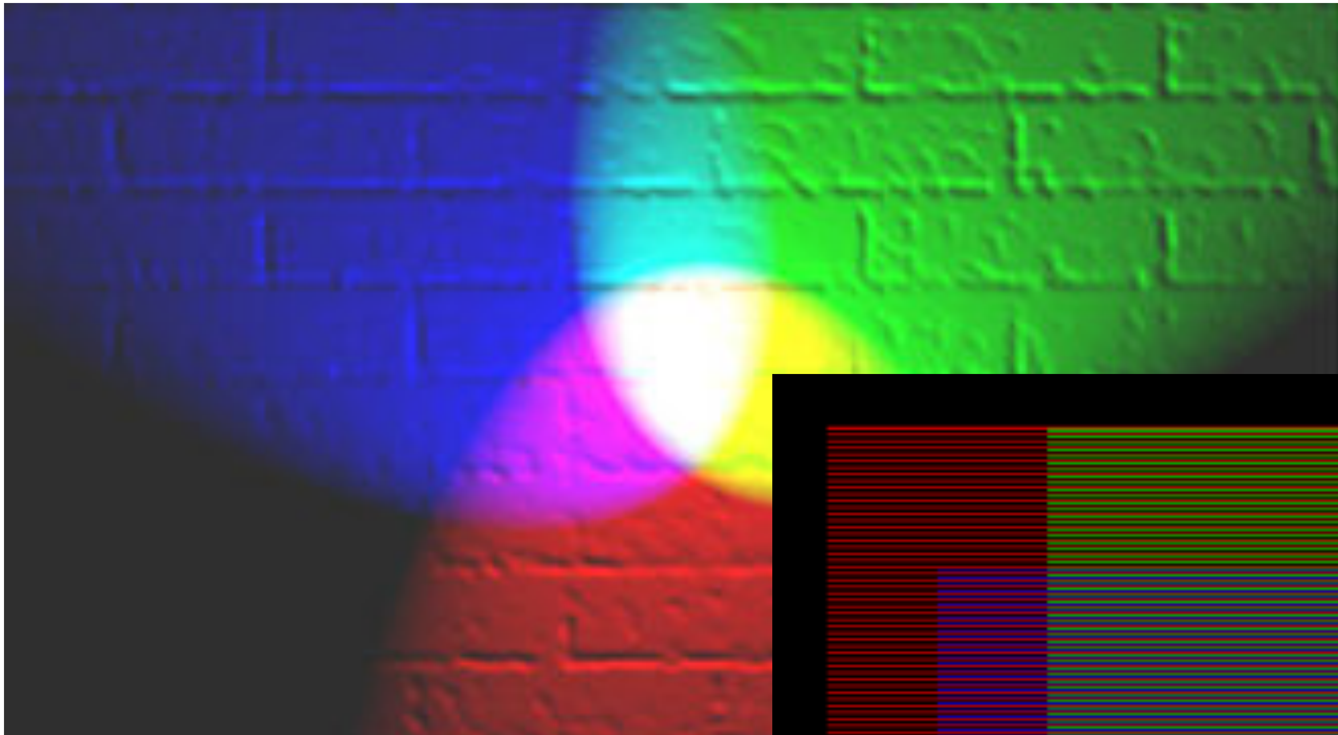
256-color images (8-bit color)

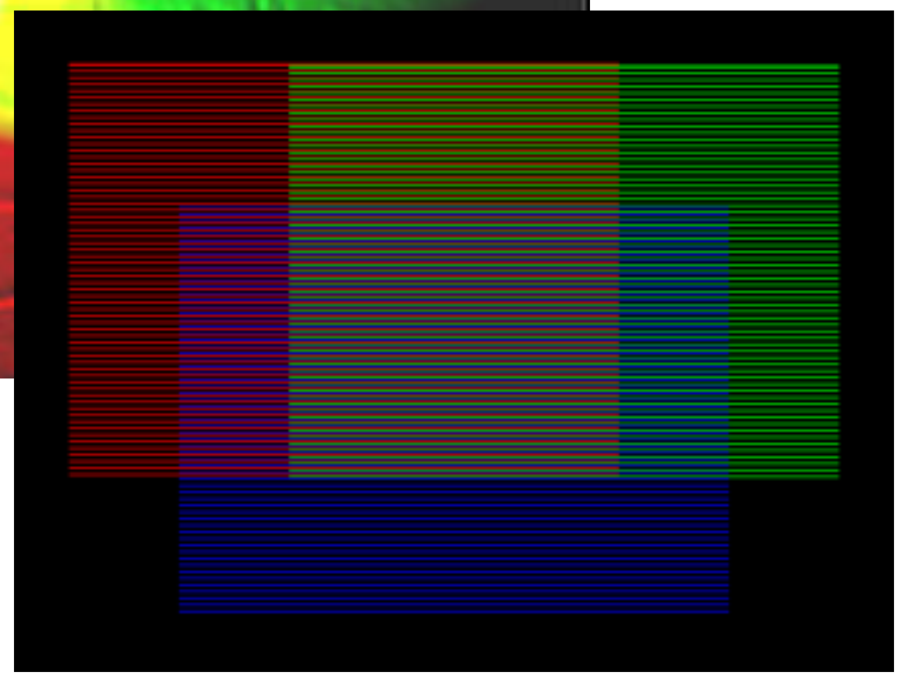
- Each pixel is represented with a 8-bit value that is an index into a *palette* of 256 colors.



8 bits per pixel

RGB color systems



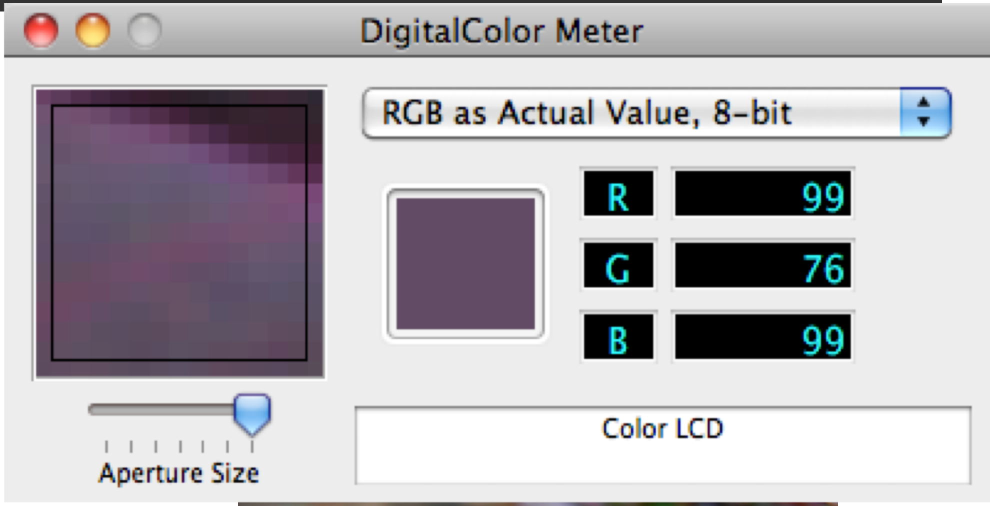


24 bits per pixel

images: Wikipedia

RGB-color images (24-bit color)

- Colors are represented as mixtures of red (R), green (G), and blue (B).
- Each pixel is represented using three 8-bit values, one for each color component.
- This representation allows for 2^{24} = 16,777,216 different colors.
- This representation is also called “true color”.
- **Explore with DigitalColor Meter**



(image from Wikipedia)

Comparing Representations

- For a 640 X 480 image (307,200 pixels), **how many bytes** needed?
  - B&W 38,400 bytes (307200/8)
  - 8-bit grayscale 307,200 bytes
  - 256-color (8-bit color) 307,200 bytes
  - 24-bit color 921,600 byte(307200*24/8)
- A single RGB image of size 1600 X 1200 requires over 5.76 million bytes!

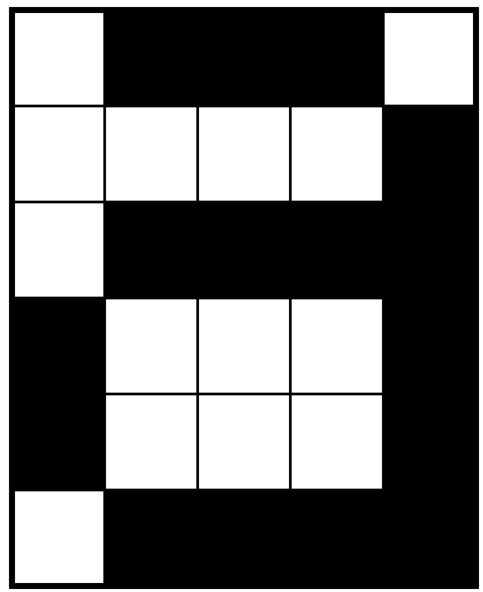
so we need compression

Compressing Raster Data

- Run-length encoding (lossless, limited)
- Color maps (GIF, good for graphics with solid areas of color)
- JPEG (lossy - a suite of techniques exploiting human visual perception)

RLE compression

- Run-Length Encoding is a <u>lossless</u> compression technique used in early image files.
- Instead of storing the 8-bit value for every pixel, we store an 8-bit value along with how many of these occur in a row (run).
- This saves a lot **when there are large runs of the same color**.



Color, Run, Color, Run, ...

255,1,0,3,255,1

255,4,0,1

255,1,0,4

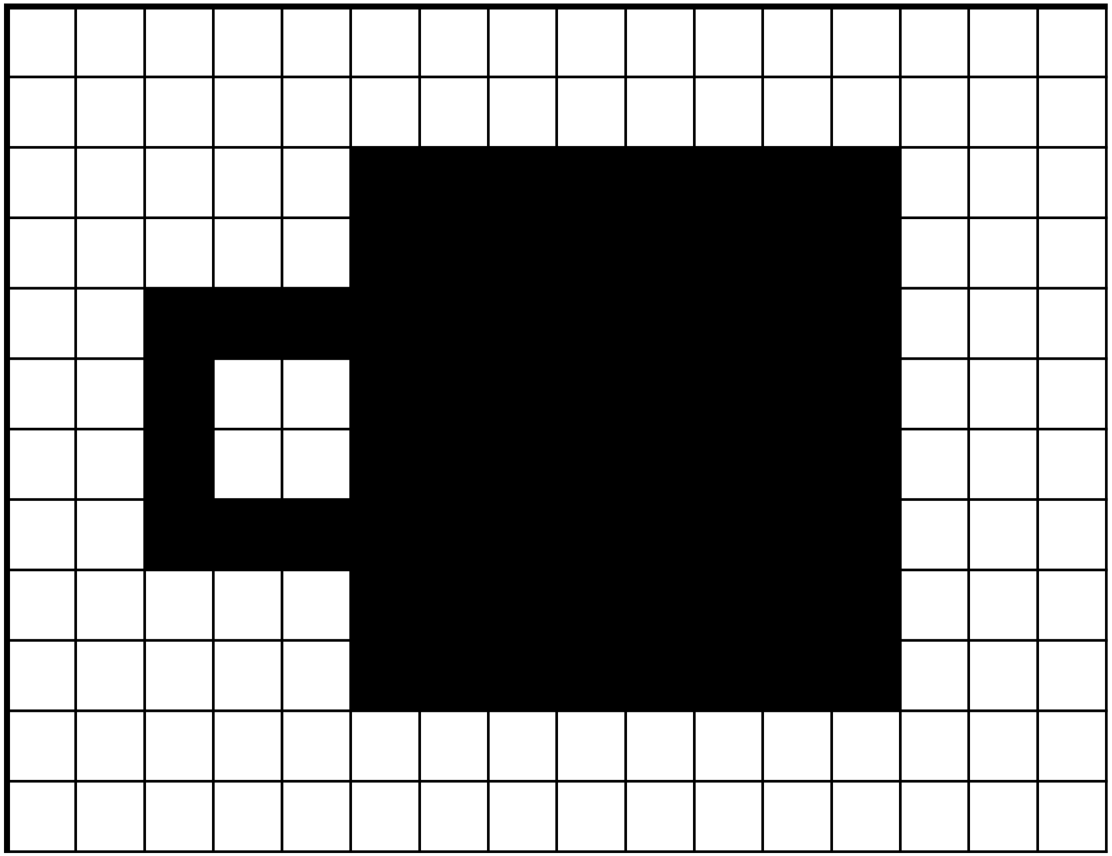
0,1,255,3,0,1

0,1,255,3,0,1

255,1,0,4

(Colors: 0=Black, 255=White)

RLE Comparison



RLE	Bitmap
2 bytes	16 bytes
2 bytes	16 bytes
6 bytes	16 bytes
6 bytes	16 bytes
6 bytes	16 bytes
10 bytes	16 bytes
10 bytes	16 bytes
6 bytes	16 bytes
6 bytes	16 bytes
6 bytes	16 bytes
2 bytes	16 bytes
2 bytes	16 bytes
64 bytes	192 bytes

GIF: Graphic Interchange Format

- 8-bit pixels, mapping to a table of 256 24-bit RGB colors.
- A *codebook* stores recurring sequences.
- Useful for representing images with fewer colors or large areas of color like company logos.

GIF and photos

Only 256 colors leads to strange effects

JPEG (JPG): Joint Photographic Experts Group

- A <u>lossy</u> compression technique for photographic images.
  - Perceptual Coding: based on what we can/cannot see.

Higher quality
Compression 2.6:1
(images from Wikipedia)

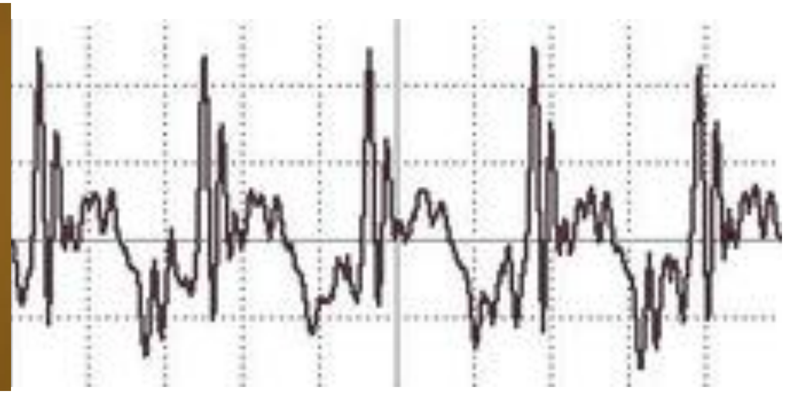
Medium quality
Compression 23:1

Lowest quality
Compression 144:1

Digitizing sound

Sound Is a Pressure Wave

- When an instrument is played or a voice speaks, periodic (many times per second) changes occur in air pressure, which we interpret as sound.



Human Sound Perception

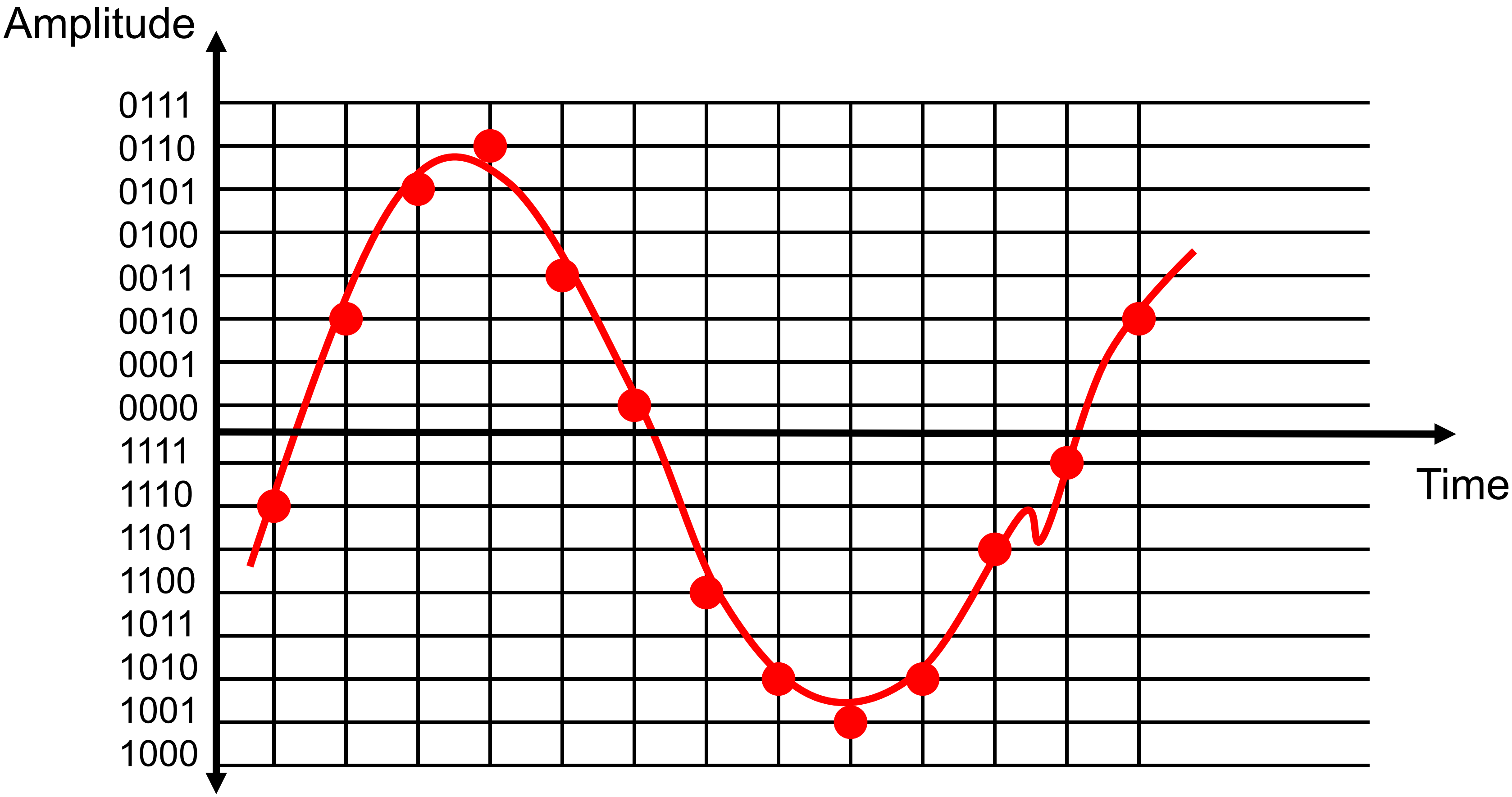
- Frequency **range**: about 20 Hz* to 20,000 Hz
- Frequency **discrimination** drops off at high part of range
- Amplitude (roughly, volume) **range**: about 10^9 (huge!)
- **Sensitivity** to volume (amplitude) drops off at ends of range

* *Hz* stands for *Hertz*, meaning *cycles per second*

Sampling

- Pressure varies **continuously–sampling** measures how much pressure at fixed intervals
- Accuracy determined by
  - Sampling rate
  - Sample size
- **Sampling rate**: how many times per second do we measure?
- **Sample size**: how many bits do we store per sample?

Sampling



1110 0010 0101 0110 0011 0000 1100 1010 1001 1010 1101 1111 0010 ....

When Sampling Is Too Slow

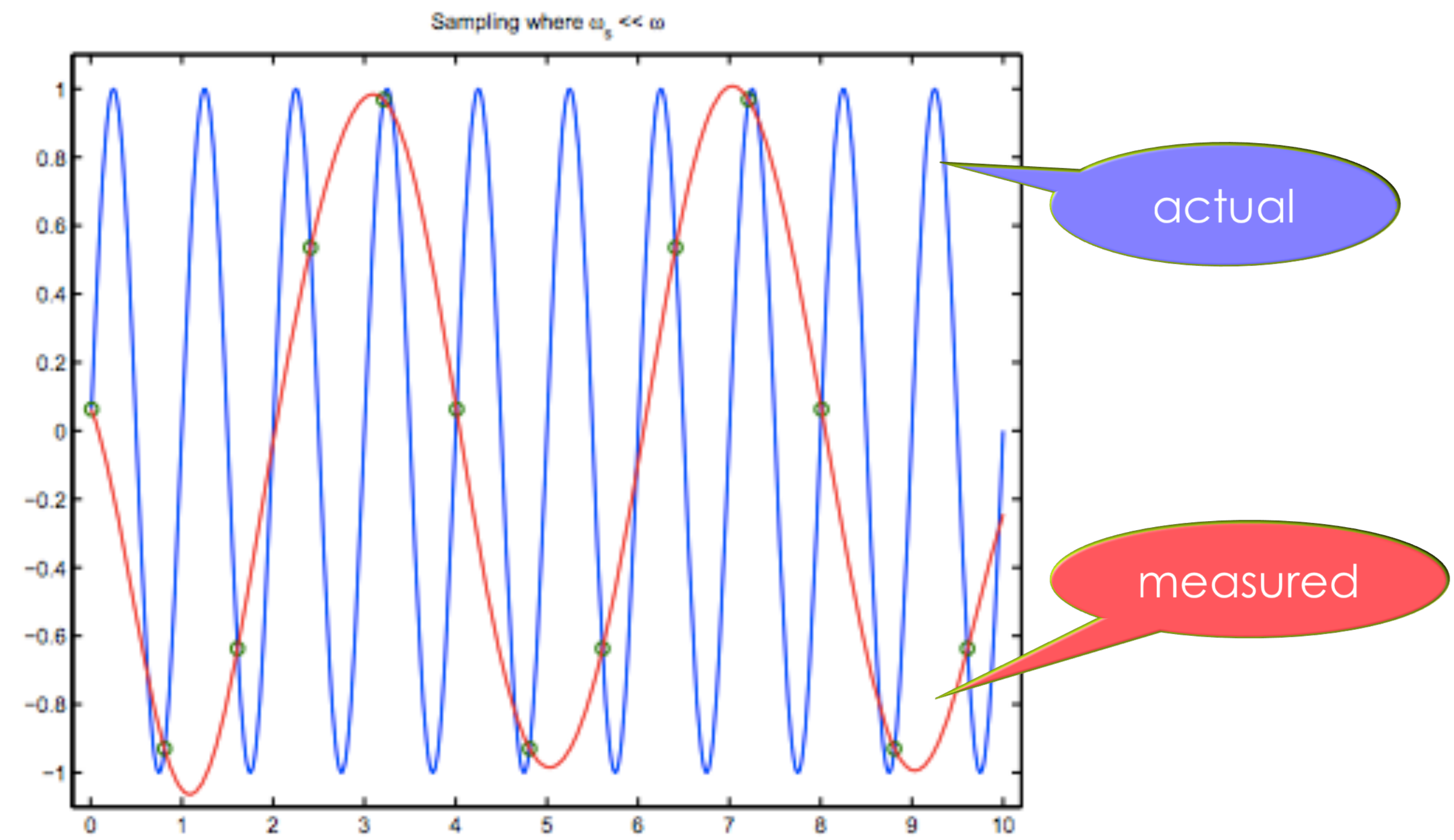


Figure 5.7: Sampling a sinusoid at too slow of a rate.

Source: http://www.princeton.edu/~cuff/ele201/kulkarni_text/digitizn.pdf

Samples Must Have Enough Bits

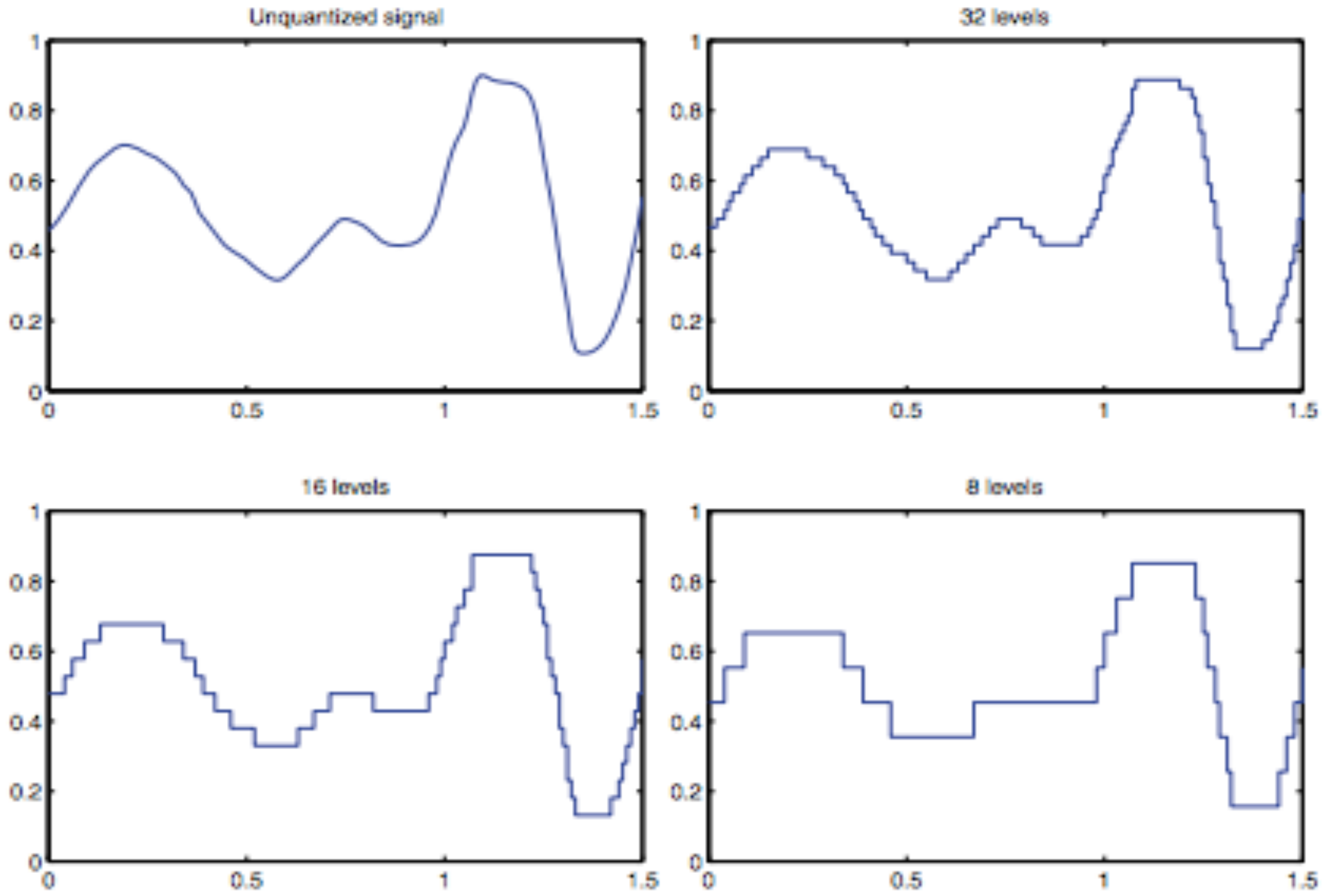


Figure 5.11: Quantized versions of an analog signal.

Source: http://www.princeton.edu/~cuff/ele201/kulkarni_text/digitizn.pdf

High-Quality Sampling

sampling rate

- Rate: 44,100 samples per second (Hertz – Hz).
  - sampling theorem: *the sampling rate must be at least twice the highest frequency in the sound* (humans can hear up to approx. 20,000 Hz.)

sample size

- Sample size: 16-bits per sample (so there are 65,536 amplitude levels that can be measured).
  - *Quantization* (rounding to integer sample values) *introduces noise*. Adding one bit cuts the noise in half.

sound file formats

Compressing Sound Files

- *codecs* (compression/decompression) implement various compression/decompression techniques
- **Lossless**: WMA Lossless, ALAC, MPEG-4 ALS, …
- **Lossy**: MPEG, like JPEG, a family of perceptually-based techniques

MP3

- MP3 (MPEG3) is a lossy compression technique.
- Takes advantage of human perception (**psychoacoustics)**
  - Our hearing is better in mid range frequencies than on the low and high ends.
  - If a loud and soft sound play at about the same time or about the same frequencies, we can't hear the soft sound: this is called *masking*
  - *Masking can hide noise introduced by compression.*

MP3 Demo

Let Me Call You Sweetheart

http://www-mtl.mit.edu/Courses/6.050/2014/notes/mp3.html

MP3 Compression

- Like JPEG, MP3 has various levels of compression:

Bit Rate	Compression Ratio	Quality
256Kbps	5:1	Supreme (near best)
192Kbps	7:1	Excellent (better)
128Kbps	11:1	(good)
96Kbps	19:1	(fair)
64Kbps	22:1	FM quality (poor)

- MP3 also has Variable Bit Rate (VBR) since compression ability can vary at different segments of the digital recording.

image + sound = video

Problem: a torrent of data

- Imagine if we used "raw" images and sound for video
  - about 5MB of image data per frame, times 30 frames/sec = about 150 MB image data per second
  - about 1400 kbps, or 175 KB sound data per second
  - 10 minutes of this: about 90.1 Gigabytes

MP4

- MP4 (MPEG4): compression technique for video
- Sophisticated engineering exploits
  - **redundancy** (next frame is likely to resemble this frame)
  - **perception** (what the eye and ear can do)
- Applications: streaming, HDTV broadcast, Digital Cinema, cameras (e.g. GoPro), phones

YouTube, Vimeo, etc.

- YouTube, Vimeo, etc. support many formats, including MP4, AVI (Microsoft), QuickTime (Apple), and Flash (Adobe).
- You can download videos from these sites in your preferred format using tools such as KeepVid
- Uploading and then downloading a video may reduce the quality due to lossy compression.

Summary

- **Samples**
  - **Pixels** are samples of the image in space; *resolution* and *number of bits* determine quality
  - **Audio samples** measure the signal in time; *sampling rate* and *number of bits* determine quality
- **Tradeoff** between quality and size
- **Compression** methods exploit
  - Coding redundancy (e.g. Huffman codes)
  - Data redundancy (e.g. run-length coding)
  - Perceptual redundancy (e.g. MP3, JPEG)