Cameras, Displays and Compression

CS475 / 675, Fall 2016

Siddhartha Chaudhuri

Henri Cartier-Bresson, Hyères, 1932

Digital Camera

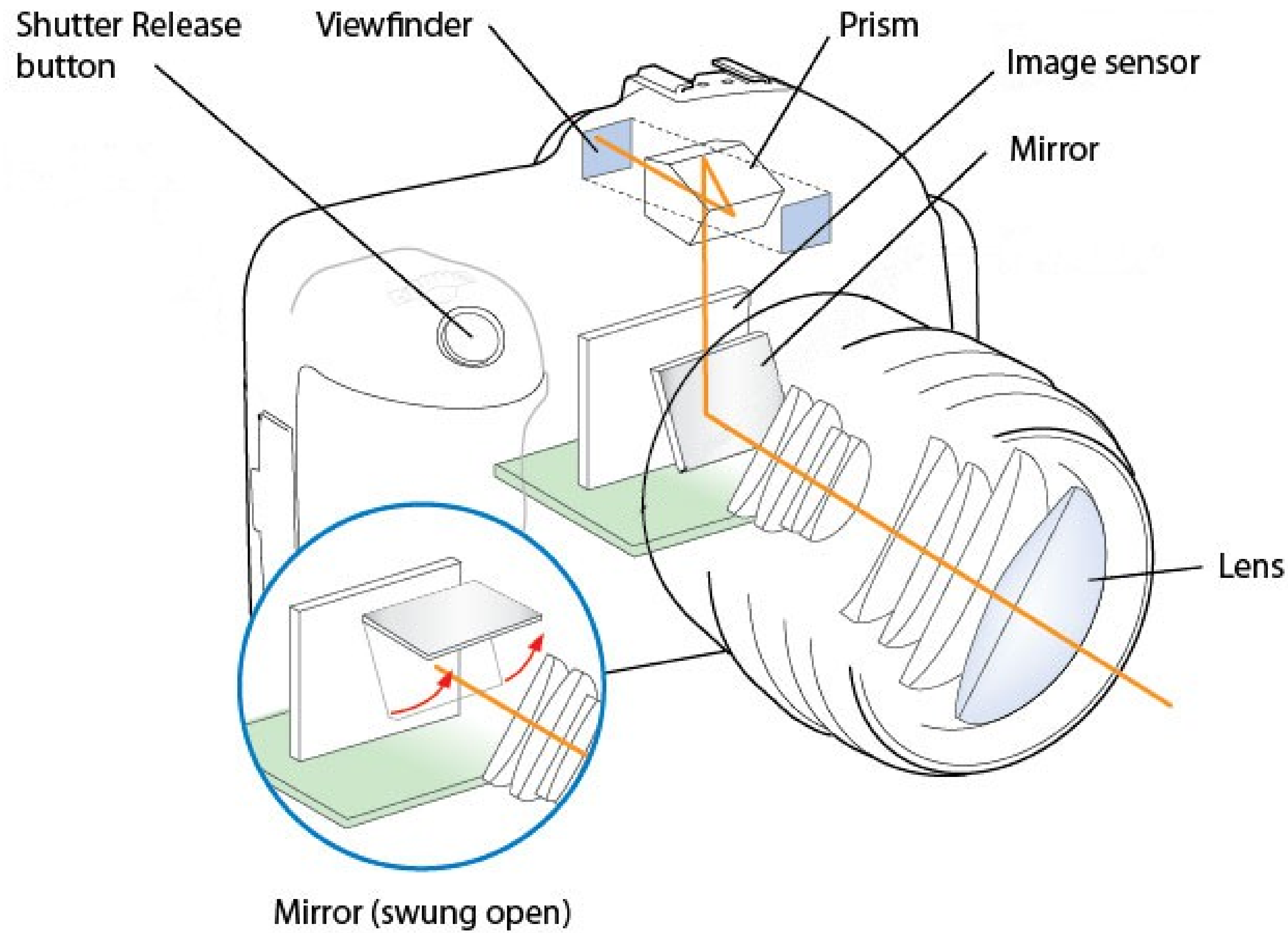
- Captures images on a digital sensor (CCD/CMOS/...)
- Structurally similar to film cameras
- Digital tech allows
  - Instant review
  - WYSIWYG viewing without bulky SLR mechanism
  - In-camera adjustments
  - Various advanced shooting aids
- We'll mostly study *still* cameras, not video
  - Our comments will apply in large part to video cameras

Common Types of Digital Still Cameras

Category	Illustration	Examples
Compact (Small Sensor)	 	Canon Powershot ELPH 180 Panasonic Lumix LX7 Apple iPhone
Compact/Mirrorless (Large Sensor)		Olympus EM1 Panasonic GX8
SLR		Nikon D3300, D5 Canon EOS 1DX
Rangefinder		Leica M
Medium/Large Format		Hasselblad H6D Phase One 645 system

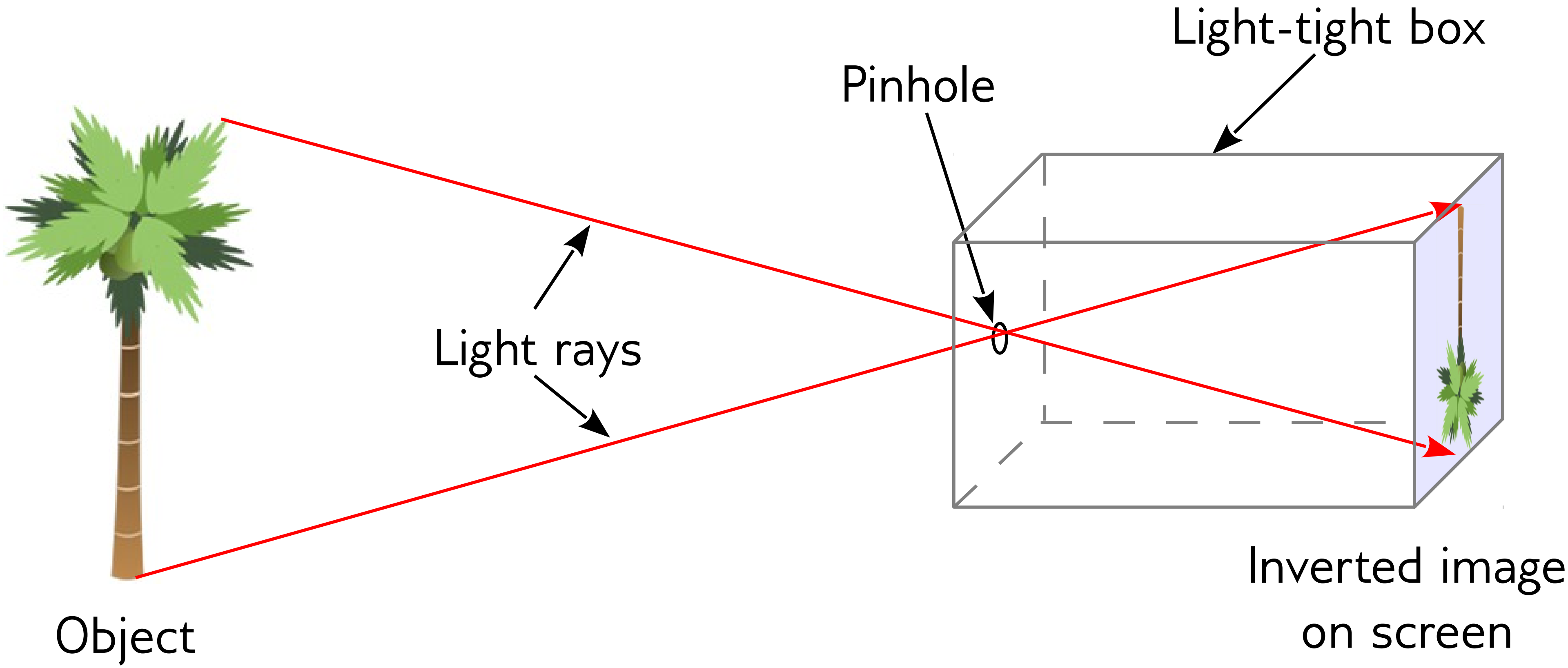
(Images: dpreview.com)

Image Formation in an SLR



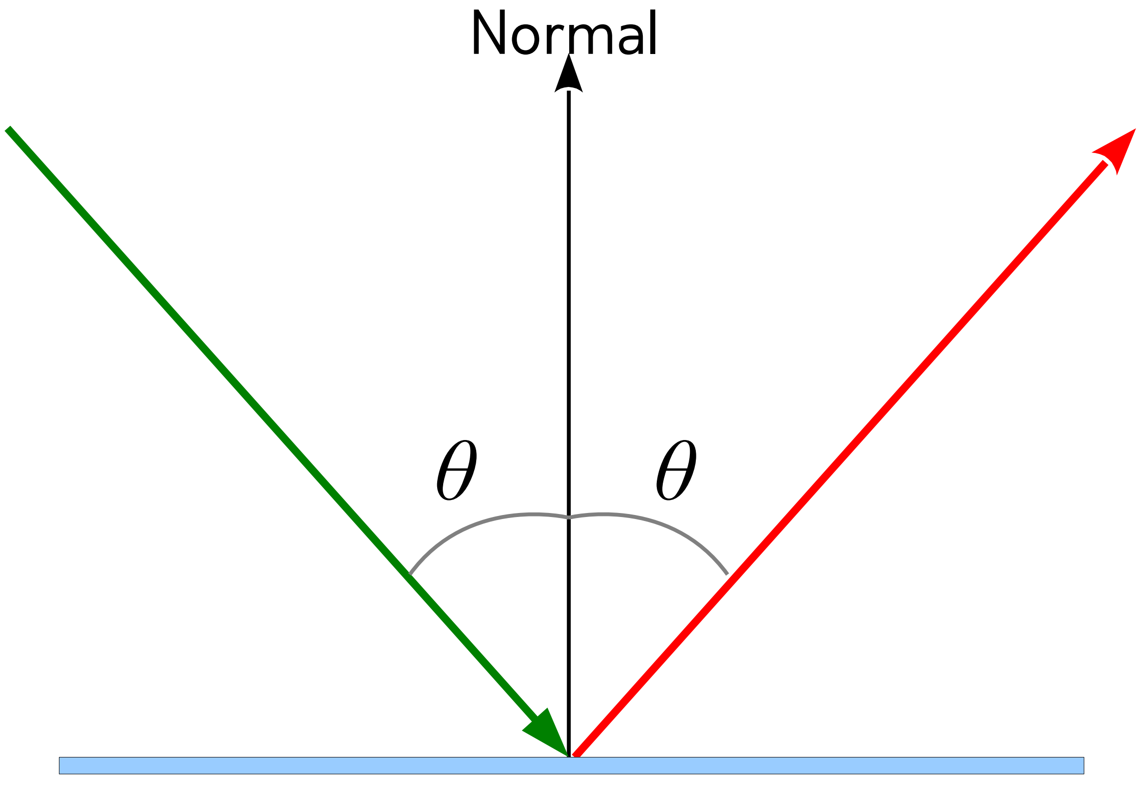
(Image: Blue Moon Productions, 2009)

Pinhole Camera



- Problem: Very little light gets through (for an ideal pinhole, just one ray per object point)
- Solution: Use a lens to gather more light

Optics Review: Reflection



Angle of incidence = Angle of reflection

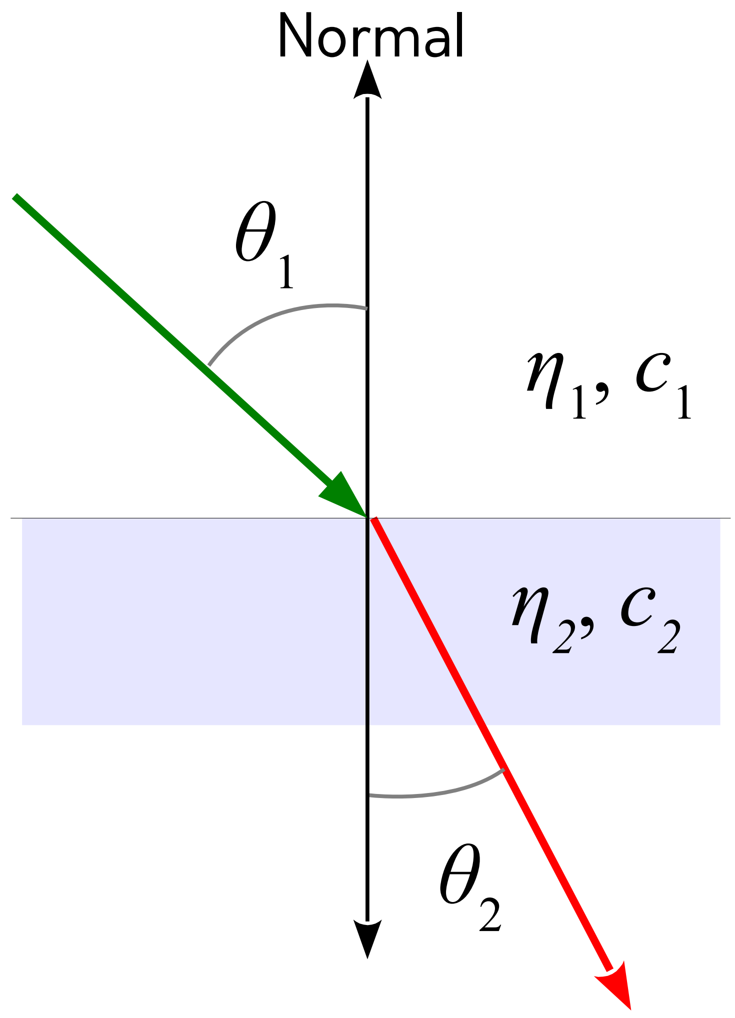
Optics Review: Refraction

- Light bends at interface between media
- *Snell's Law*:

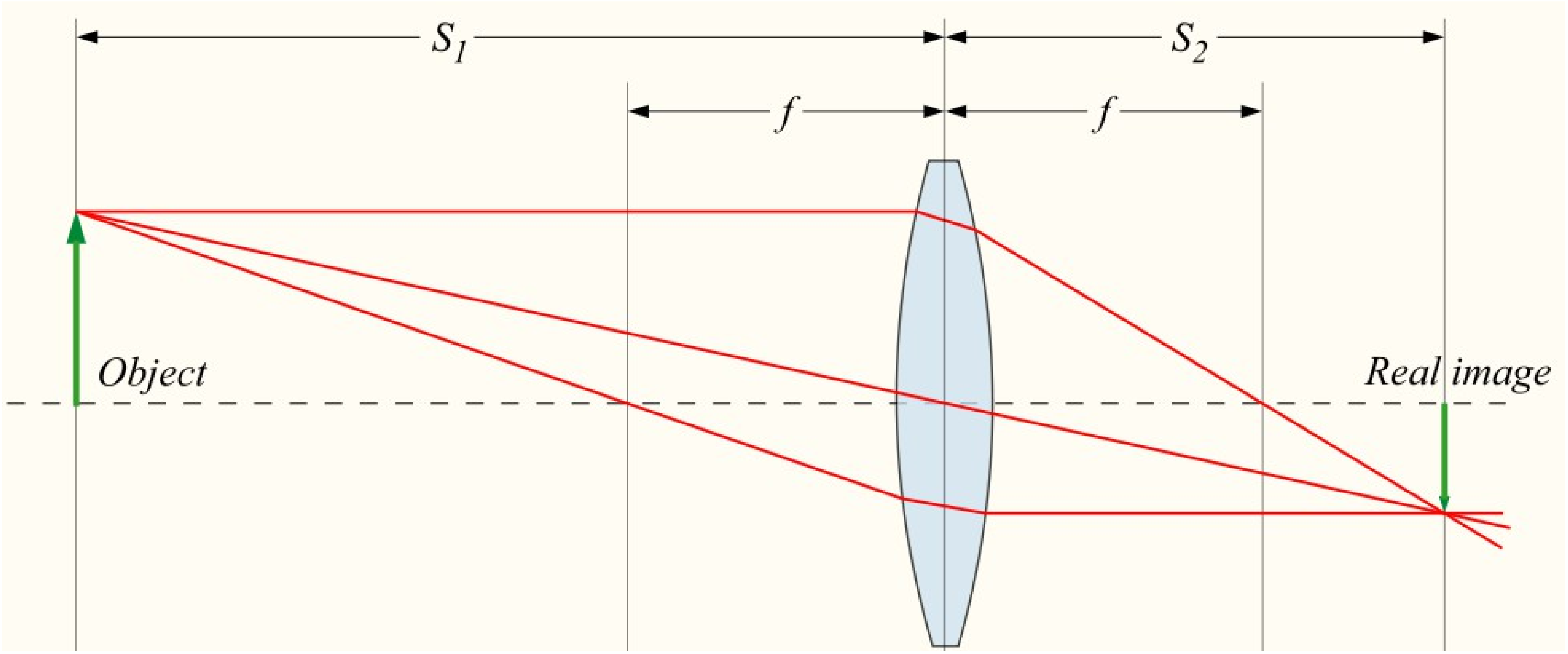
$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\eta_2}{\eta_1} = \frac{c_1}{c_2}$$

η_1, η_2 – refractive indices
c_1, c_2 – speeds of light

- *Total internal reflection*: If $(\eta_1/\eta_2)\sin\theta_1 > 1$, light reflects back into source medium



Optics Review: Thin Lens

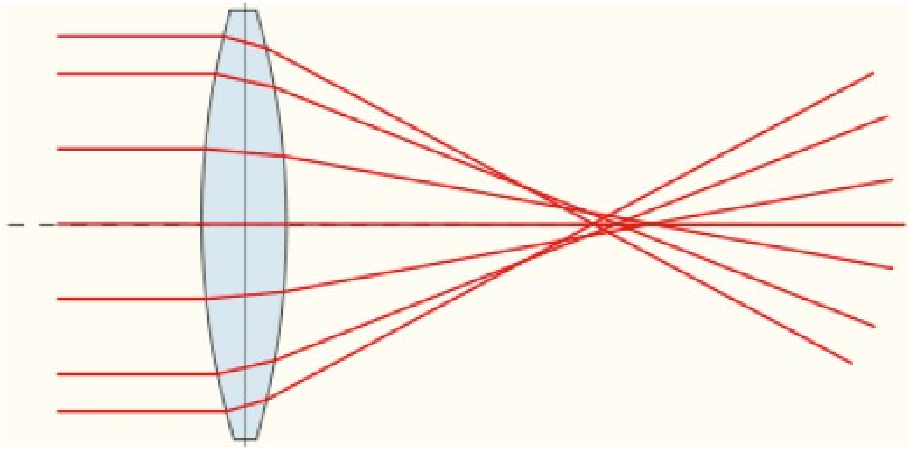


(Wikipedia)

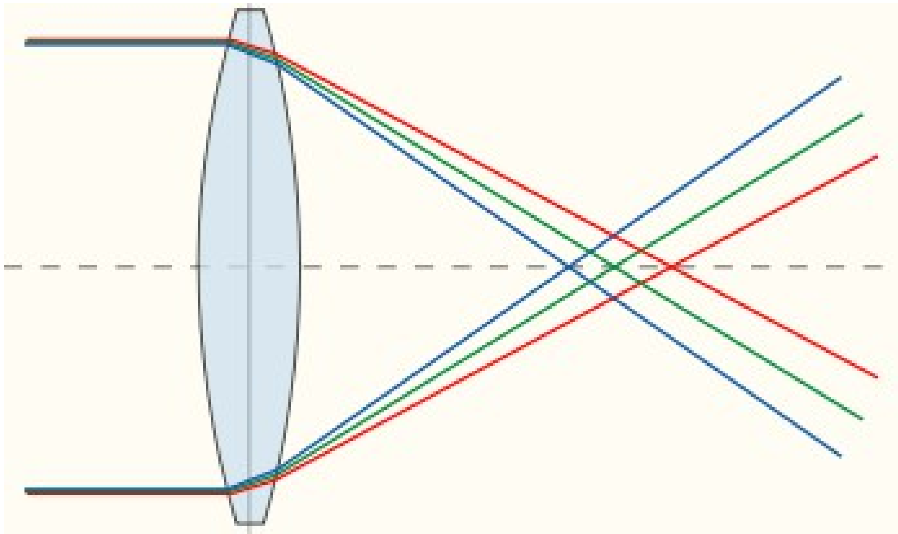
- Thin lens equation: $$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}$$
  - f – *focal length*
  - S_1 – object distance, S_2 – image distance
- Represents ideal imaging system: all rays from an object point converge on a single image point

When Conditions Aren't Ideal

- *Spherical aberration*
  - Rays from single object point don't converge at same image point
  - Reduces image sharpness



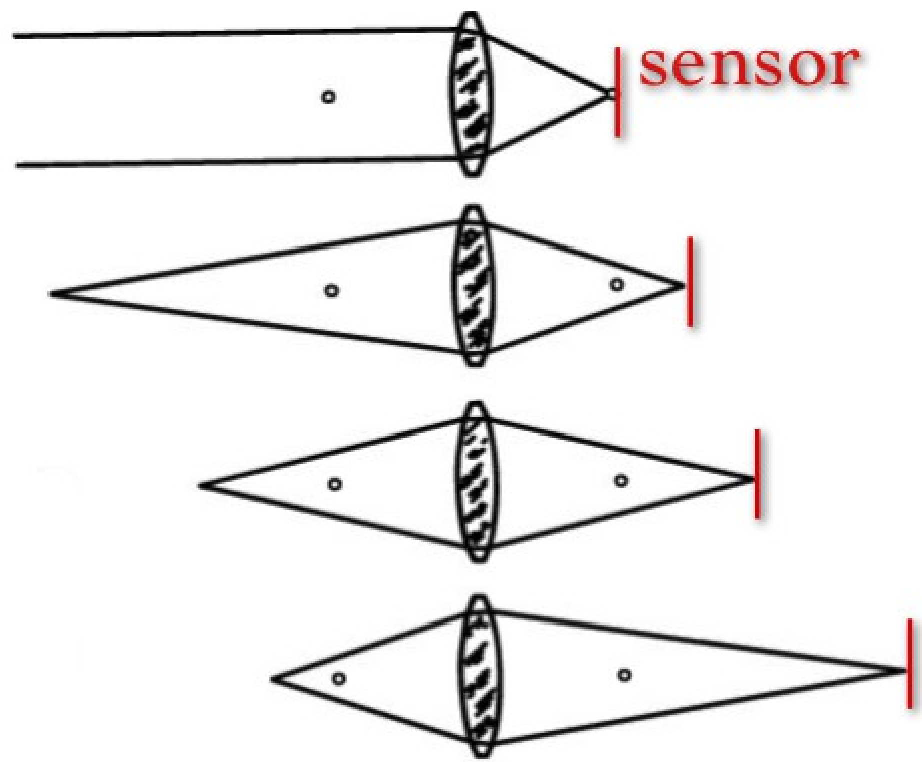
- *Chromatic aberration*
  - Different wavelengths refract differently
    - different refractive index for each wavelength
  - Color fringes at object edges



(Images: Wikipedia)

Focusing a Camera

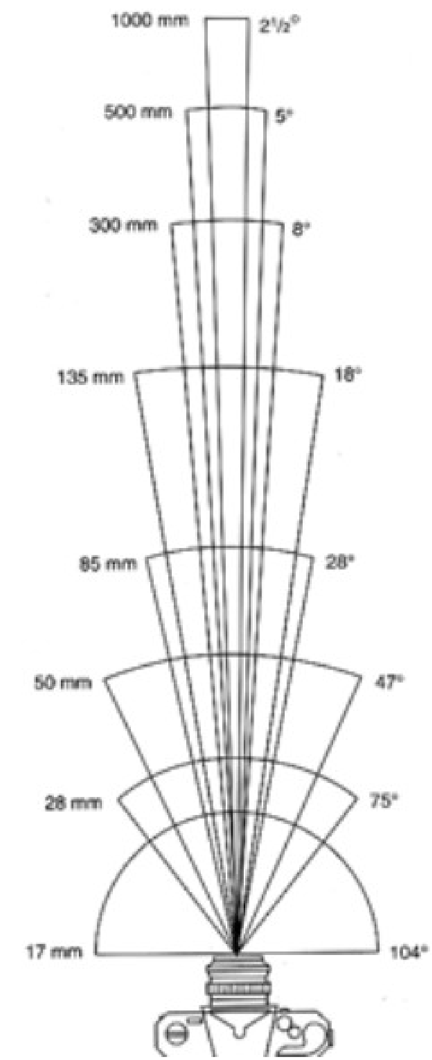
- Unlike pinhole cameras, lens cameras have only one object plane in perfect focus
- Distance between lens and sensor governs plane of focus
- Most cameras move (elements of) the lens relative to the rest of the camera body



Changing the Focal Length of the Lens

- *Wide-angle* lenses (small f) capture more of the scene
- *Telephoto* lenses (long f) capture less of the scene
- The angular range captured by a lens on a given sensor is called its *field of view* (FOV)

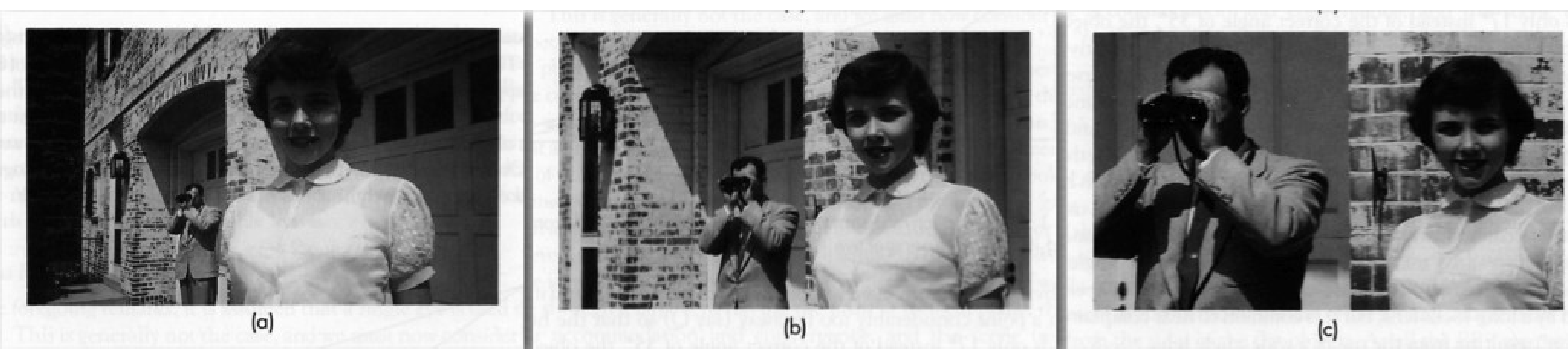
Top: Wide-angle (f = 24mm)
Bottom: Telephoto (f = 392mm)
(sensor: 35mm film)



(Photos: dpreview.com)

Changing the Focal Length of the Lens

- Wide-angle lenses accentuate depth differences, telephotos compress them
  - That's why a cricket bowler and batsman look the same size on TV (shot with a telephoto), although one's much further away than the other



Wide-angle Standard Telephoto

Changing the Focal Length of the Lens

- Wide-angle portraits can look wonky
  - Nose is nearest camera and looks bigger
  - Ears and hair are further away and look smaller

Wide-angle

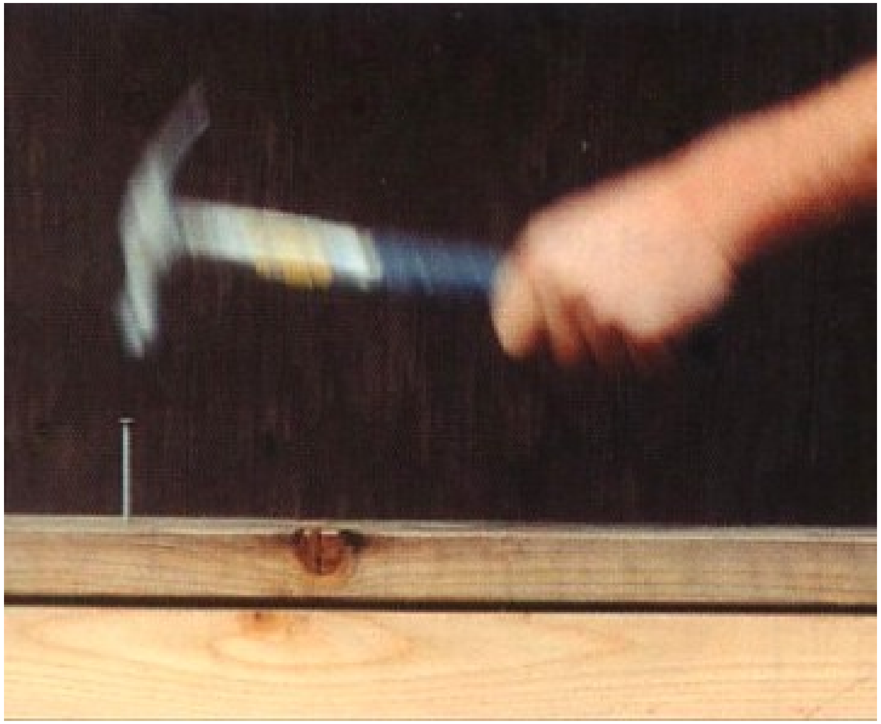
Standard

Telephoto

Exposure

- To take a picture, shutter is opened and light hits sensor for a specified time
- *Exposure*: Amount of light hitting sensor while shutter is open
- Exposure is affected by:
  - *Shutter Speed*: How long shutter is open
  - *Aperture*: Size of lens opening

Shutter Speed Affects Motion Blur

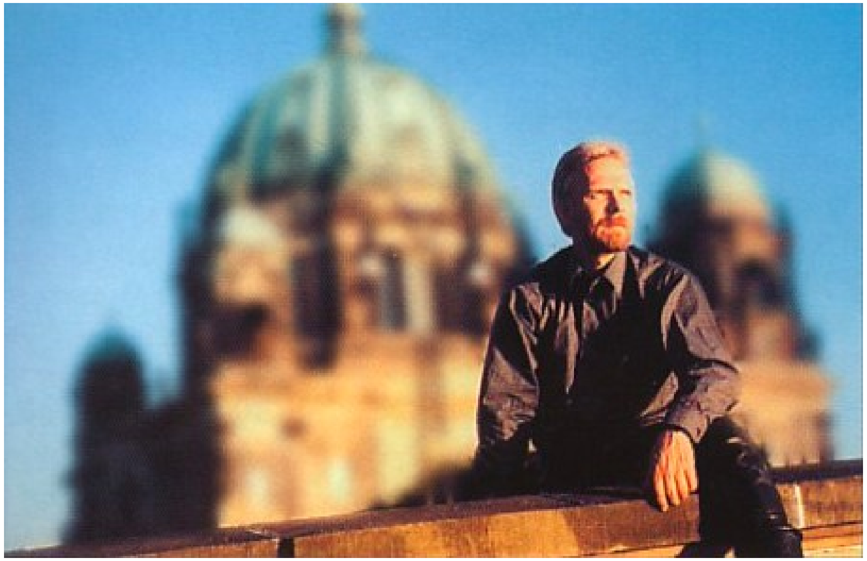


Slow shutter speed

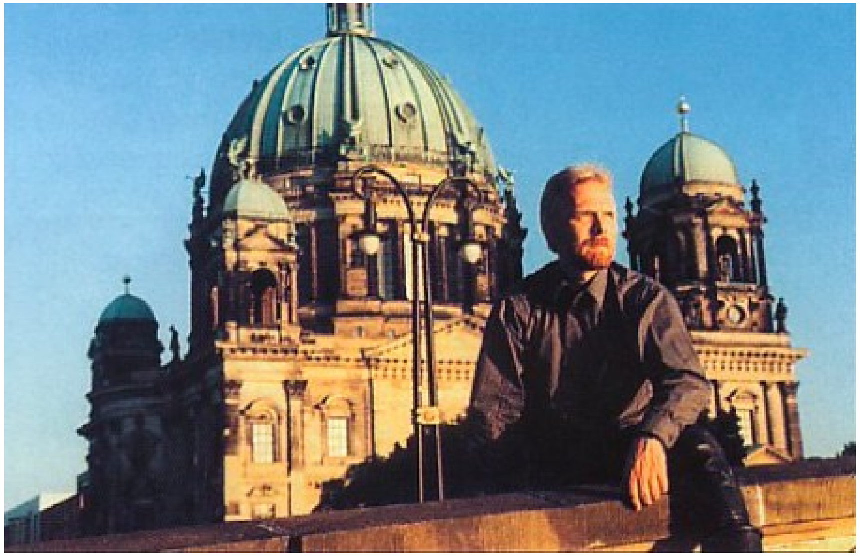
Fast shutter speed

Aperture Affects Background Blur

Formally called *depth of field* (DOF): the depth range that is approximately in focus



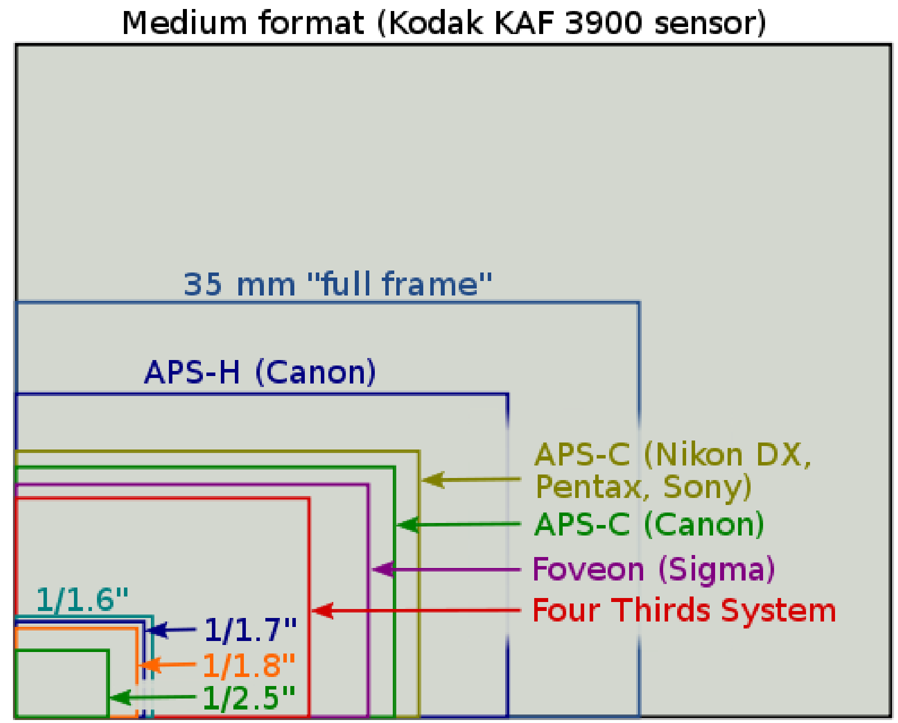
Large aperture: small DOF



Small aperture: large DOF

Sensor Characteristics

- *Size*
  - Larger sensors have larger FOV for a given lens



- *Sensitivity* (aka ISO)
  - High sensitivity $\Rightarrow$ more noise

ISO 100

ISO 800

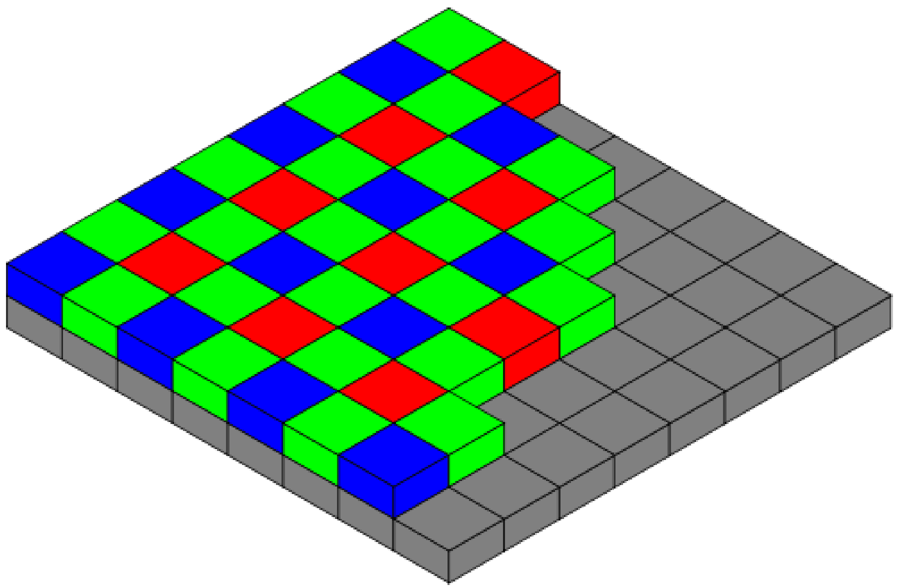
(Images: Wikipedia and dpreview.com)

Digital Sensor Characteristics

- *Resolution*: Number of pixels, e.g. 10 megapixels
    - "Megapixels don't matter!"
- *Pixel Pitch*: Size of a single pixel
    - This dœs matter
    - Larger pixels capture more light, hence have less noise at the same sensitivity rating
    - A 10 megapixel DSLR typically has much less noise than a 10 megapixel compact camera

Digital Sensor Layout

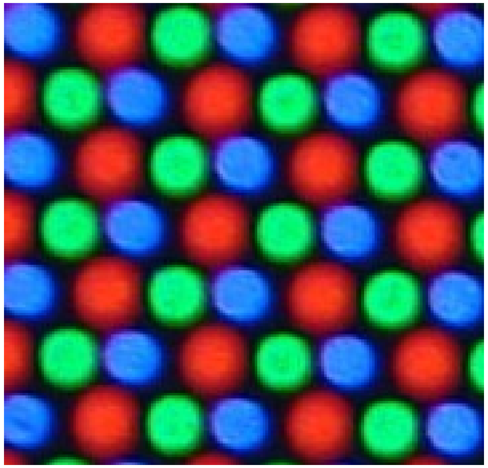
- A typical basic sensor is monochromatic (only captures intensity variations)
- A *Bayer filter* placed over the sensor passes red, green and blue light to different pixels
  - Twice as many green pixels as red or blue
  - Full RGB data at each pixel is computed by interpolation (*demosaicing*)
  - 2/3 data is reconstructed!
- There are other less common layouts as well



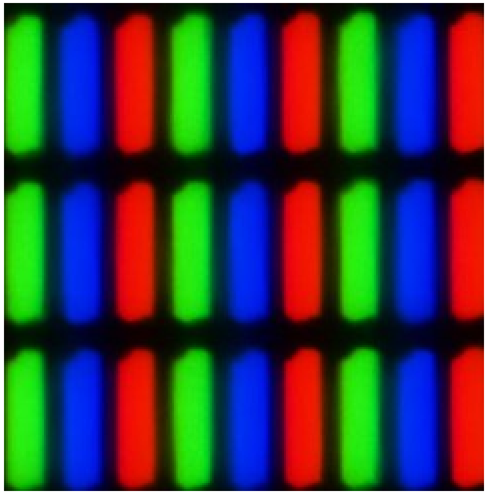
(Wikipedia)

RGB Displays

- Monitors and TVs: triads of red, green and blue subpixels
  - *Cathode Ray Tube* (CRT): Electron gun hits red, green or blue phosphor
  - *Liquid Crystal Display* (LCD): R/G/B filter placed over each subpixel
- Projectors: RGB components projected onto same pixel location, either simultaneously or in rapid succession



Shadow Mask Layout



Aperture Grille Layout

(Images: Wikipedia)

Compression

(thanks to Pat Hanrahan for much of this section)

Typical Image And Video Data Rates

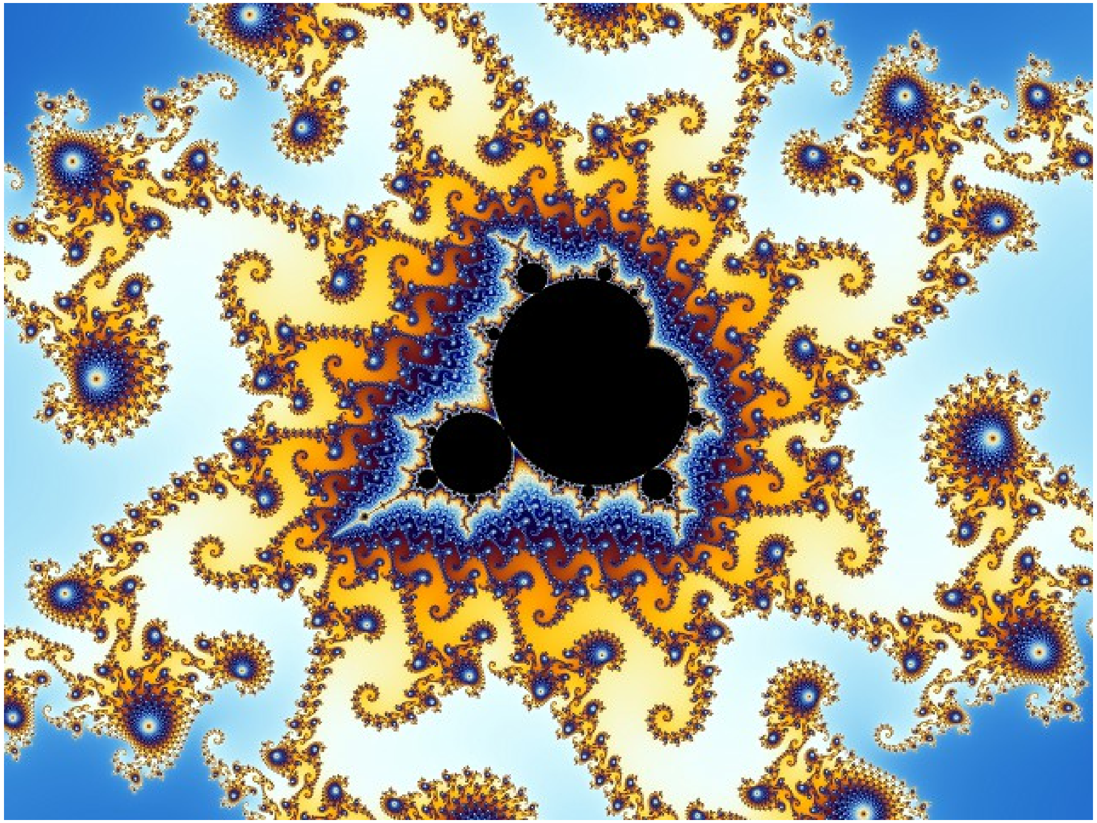
- Image
  - 640 x 480 x 24b = ~0.75 MB
  - 1024 x 768 x 24b = ~2.5MB
- DVD
  - 720 x 480 x 24b x 30f/s = ~30 MB/s
- High Definition DVD
  - 1920 x 1080 x 24b x 30f/s = ~178MB/s
- Digitized film and high-end digital video
  - 4000 x 3000 x 36b x 30f/s = ~1.5GB/s
  - 8 TB for one 90 minute movie!

Lossless vs Lossy Compression

- *Lossless*
  - All information stored
  - Exact original can be reconstructed
  - Typically used for illustrations
  - e.g. BMP, PNG
- *Lossy*
  - Some information discarded
  - Goal: discard information humans won't notice
  - Much higher compression ratios possible
  - Typically used for photographs
  - e.g. JPEG

Kolmogorov Complexity

- Length of shortest program that can *exactly* generate the data



$$z \leftarrow z^2 + c$$

program < 1KB

vs

2560×1920 PNG, 8.7 MB

Lossless: Run-Length Encoding (RLE)

BWBBBBBBBBBBBBWWWWWWBBBW

BW{12}B{6}W{3}BW

Lossless: Huffman Coding

- Given: set of m symbols with occurrence frequencies $p_1, p_2, \ldots, p_m \in [0, 1]$
  - E.g. sequence of bytes contains 256 possible symbols
- Problem: Assign binary string (*codeword*) of length b_i to each symbol s.t.
  - No codeword is a prefix of another (for unique decoding)
  - $\sum b_i p_i$ (normalized length of encoding) is minimized
- Can be approximately solved in $O(m \log m)$ time using a binary tree and a priority queue
  - Requires symbol frequencies to be independent

Huffman Coding: Basic Algorithm

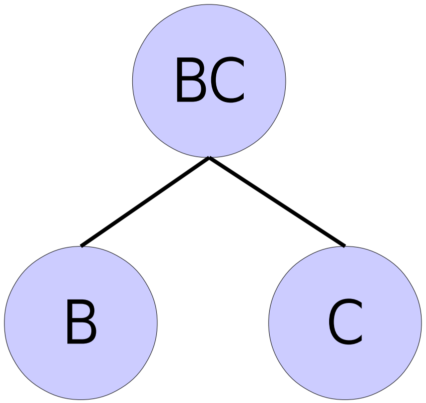
- Build tree bottom-up
  - Add a node for each symbol to the priority queue, sorted by increasing frequency (rarest first)
  - Repeat until queue has a single node
    - Pop first two nodes
    - Make them children of a new node
    - New node frequency = sum of child frequencies
    - Enqueue new node
  - Surviving node is root of final tree
- The two descendant edges of each node in the final tree are labeled 0 and 1
- Codeword of symbol (leaf) is label sequence of path from root

Huffman Coding: Demo

Front

B	0.1
C	0.1
A	0.2
D	0.2
E	0.4

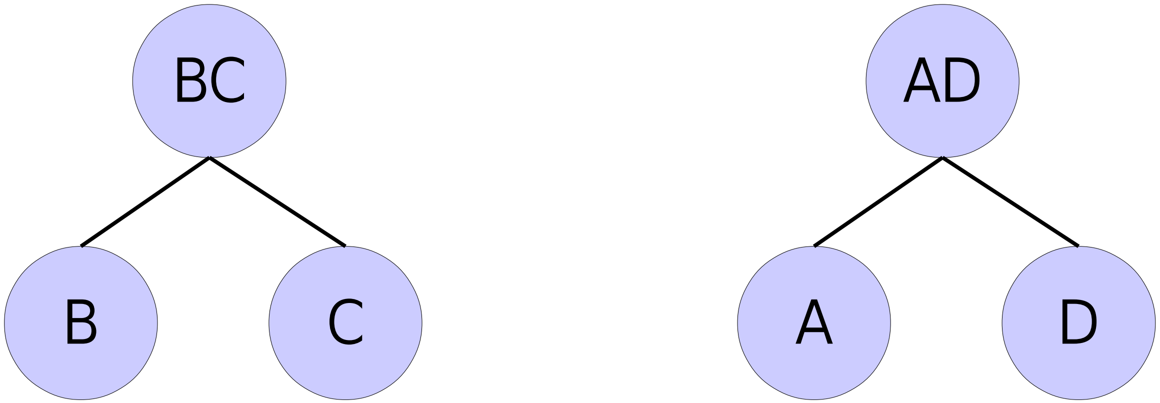
Huffman Coding: Demo



Front

A	0.2
D	0.2
BC	0.2
E	0.4

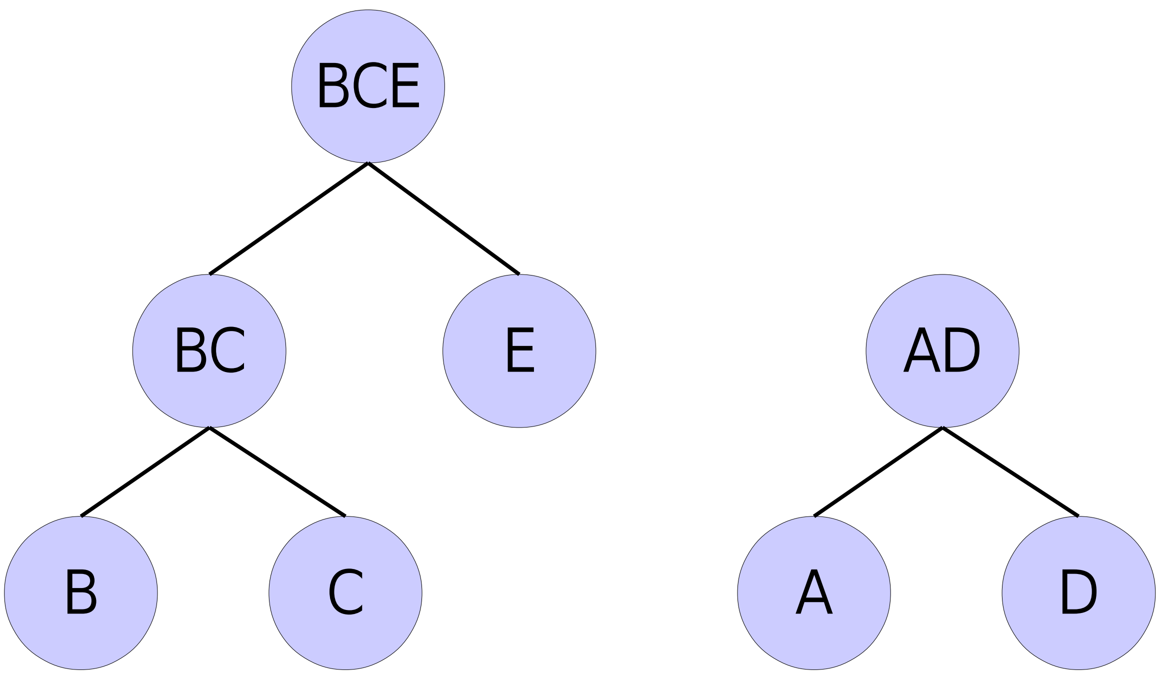
Huffman Coding: Demo



Front

BC	0.2
E	0.4
AD	0.4

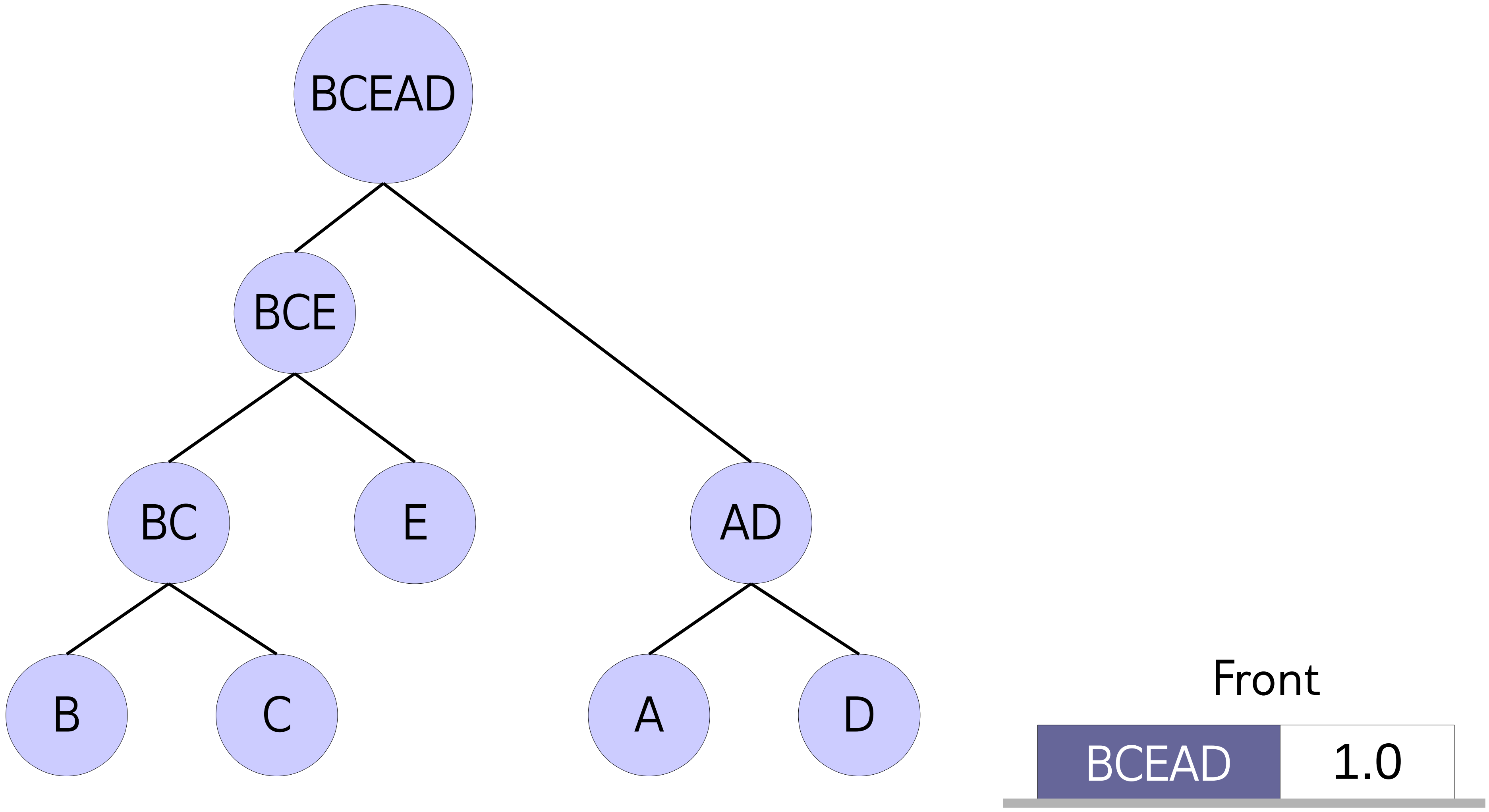
Huffman Coding: Demo



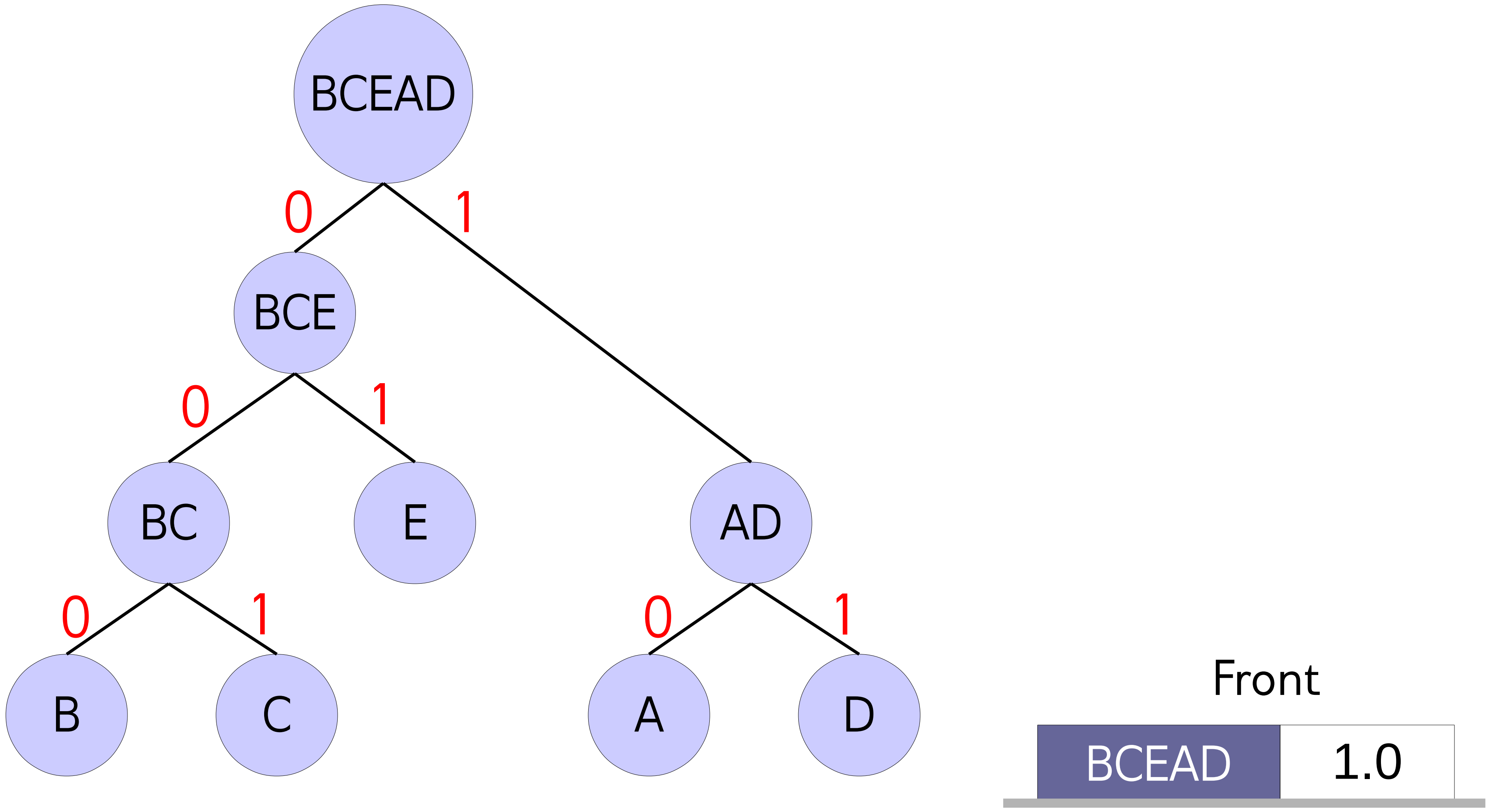
Front

AD	0.4
BCE	0.6

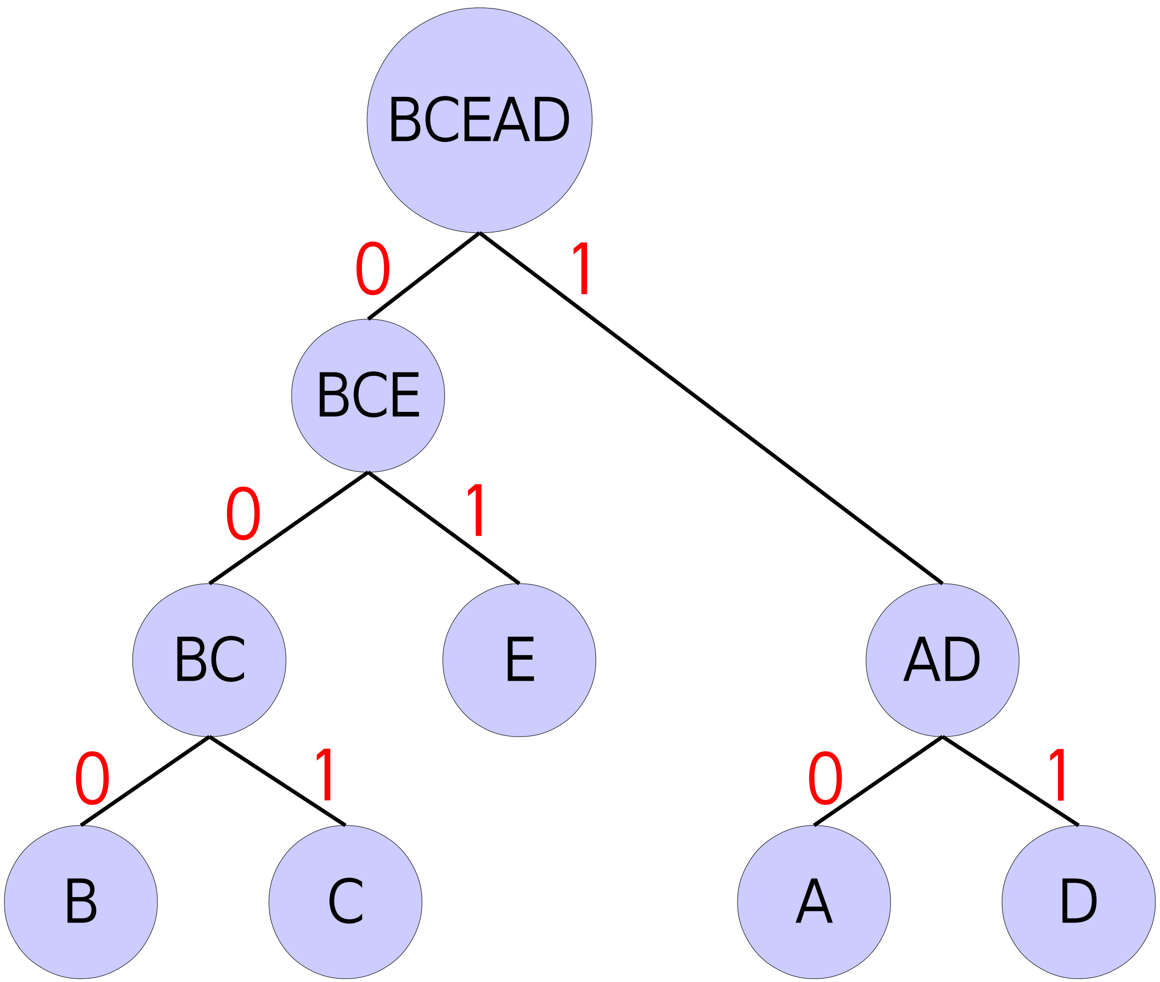
Huffman Coding: Demo



Huffman Coding: Demo



Huffman Coding: Demo



Codewords

A: 10

B: 000

C: 001

D: 11

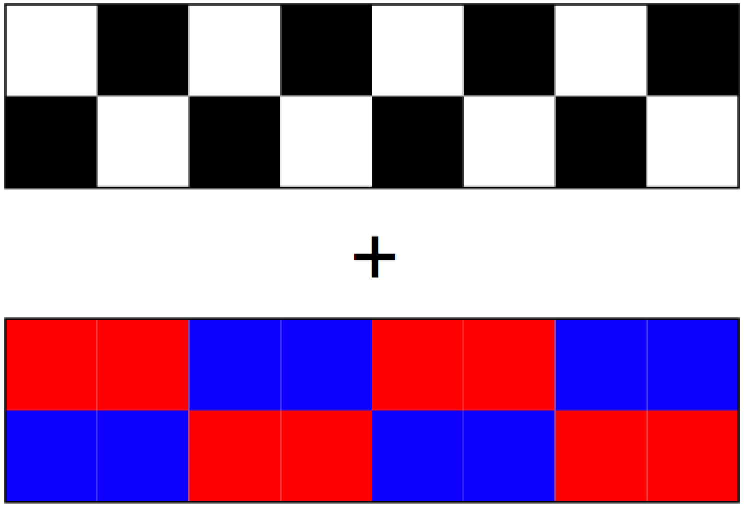
E: 01

Huffman Coding: Demo

- Encoding text with letters ABCDE
- Naïve 3-bit coding
  - e.g. A: 000, B: 001, C: 010, D: 011, E: 100
  - **3 bits/letter**
- Huffman coding
  - A: 10, B: 000, C: 001, D: 11, E: 01
  - (0.1 + 0.1) * 3 + (0.2 + 0.2 + 0.4) * 2 = **2.2 bits/letter**

Lossy: Chroma Subsampling

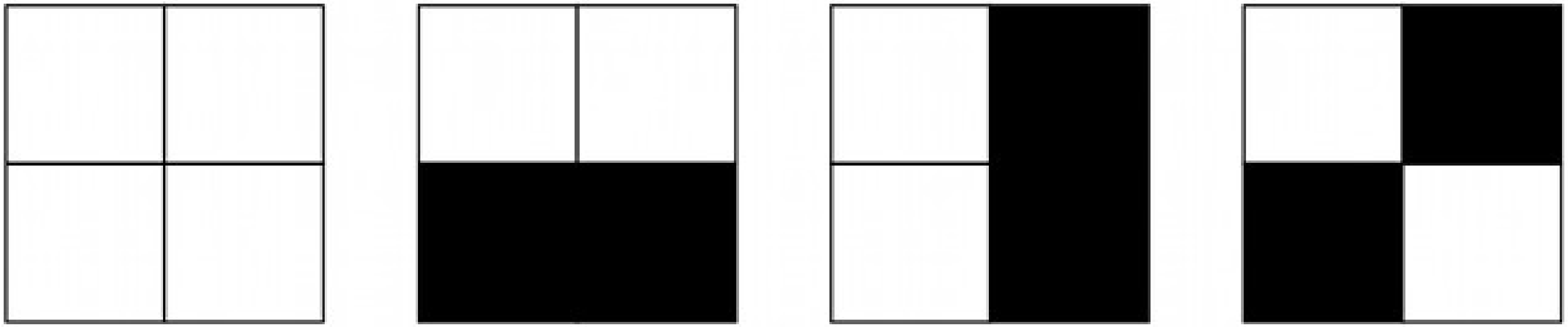
- General idea: more important to preserve contrast than color
- Separate image into luminance and chroma channels
- Reduce resolution of chroma channel



Lossy: Transform Coding

- General idea: project vector of n values (e.g. pixels of image) to another n-space where only a few dimensions hold the majority of the data
- Change of basis, like Principal Component Analysis
  - Map to $a_1\mathbf{b}_1 + a_2\mathbf{b}_2 + \ldots + a_n\mathbf{b}_n$, where $\{\mathbf{b}_i\}$ is new basis
  - $(a_1, a_2, ..., a_n)$ encodes data
  - Less significant coefficients a_i can be approximated or discarded (this is the lossy step)
- *Discrete Cosine Transform*: JPEG
- *Wavelet Transform*: JPEG2000

Example of 2x2 (4D) Pixel Basis

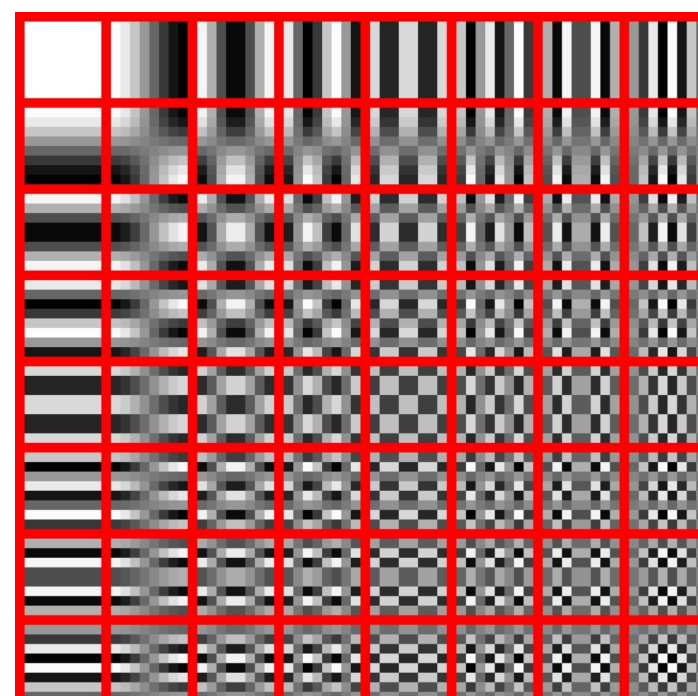


Discrete Cosine Transform (DCT)

- Similar to Fourier: decompose into low-frequency (base) and high-frequency (detail) components
  - Basis is sequence of (discrete) cosine waves of increasing frequency
- One-dimensional, for $k = 0, \ldots, N - 1$:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right]$$

  - Higher $k \Rightarrow$ higher frequency
- Multi-dimensional: product of 1D functions



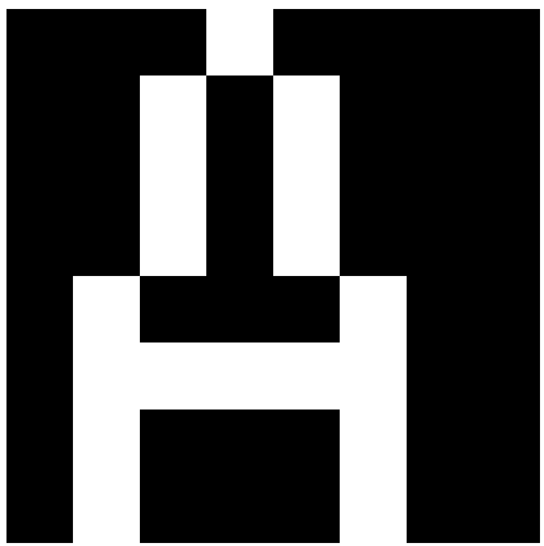
(Wikipedia)

Lossy: JPEG

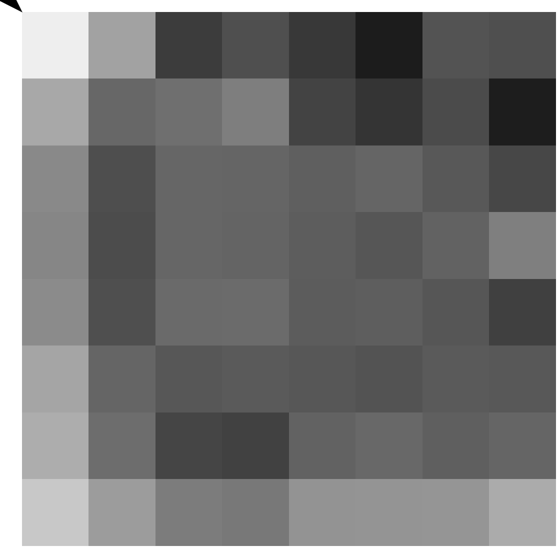
- For every 8x8 block of pixels
  - **Compute DCT** coefficients
  - **Quantize** coefficients (round to discrete steps)
    - Humans are bad at judging exact high-frequency brightness variation, so higher coefficients are quantized more coarsely
    - This is the lossy step
- **Entropy-encode** coefficients
  - Huffman code based on entire image
  - Incorporates block-based run-length data

JPEG DCT

This corner is stored in highest fidelity



8x8 block of pixels



DCT coefficient matrix
(lighter color ⇒ higher value)

This corner is stored in lowest fidelity

(Watson, 1994)

JPEG Quantization

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \div \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

DCT coefficients (rounded) Quantization matrix

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

Quantized coefficients

JPEG Pipeline

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

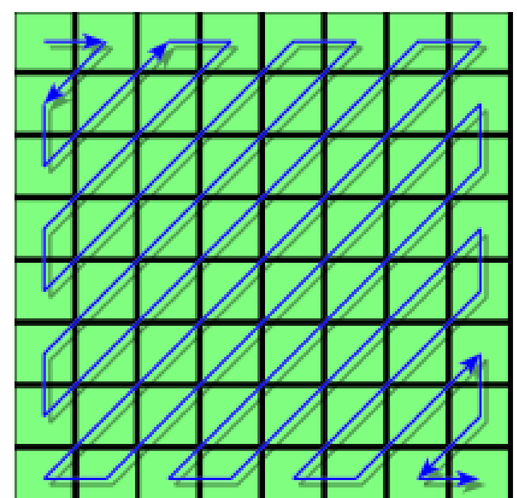
Original (greyscale) image

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

DCT coefficients (rounded)

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

Quantized coefficients



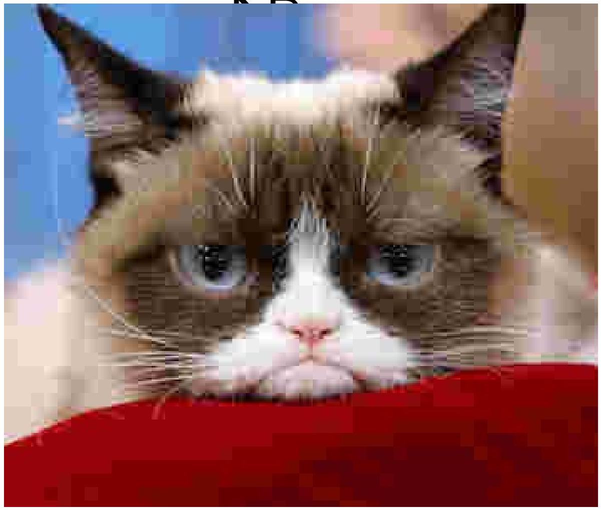
Entropy encoding order

(Wikipedia)

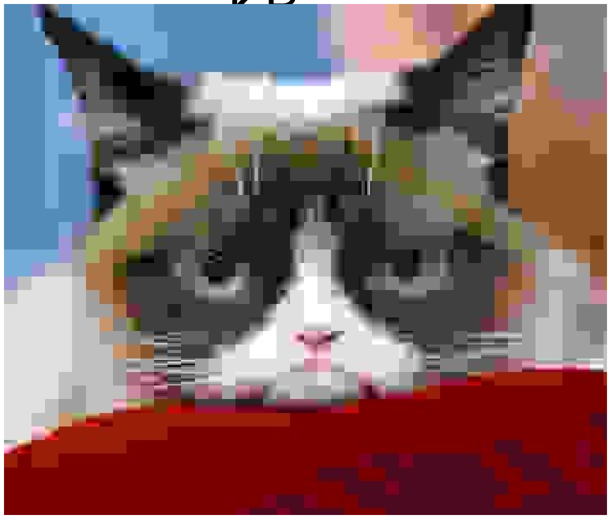
JPEG Compression Levels

Lossless 354 x 300 PNG - 214 KB

JPEG quality 75 – 20.2 KB



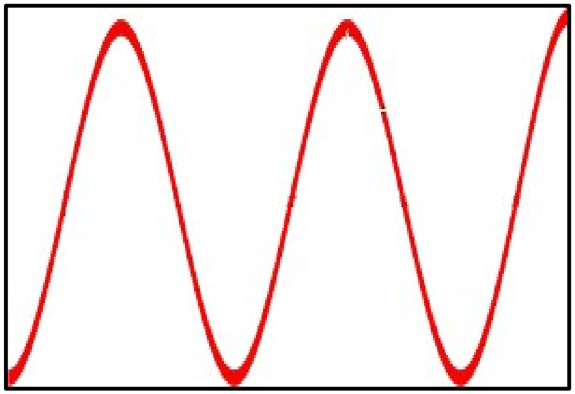
JPEG quality 10 – 4.6 KB



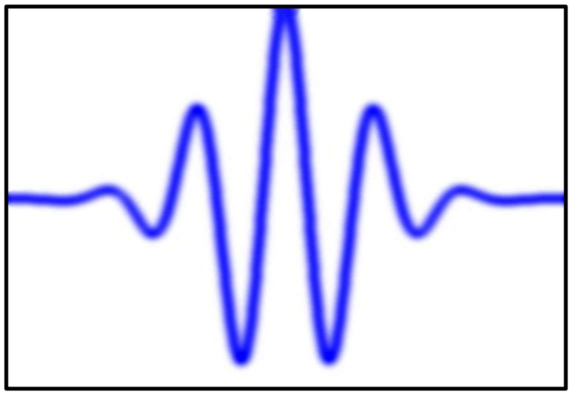
JPEG quality 1 – 2.4 KB

Wavelets

- *Support*: Region where function is non-zero



Cosine waves have global support



Wavelets have local support

- Wavelets don't require subdividing the image into blocks, since they are themselves local functions
  - Reduces blocky artifacts

Simplest Wavelet: Haar

- *"Store the difference and pass the sum"*
- Represent every two successive values A, B by
  - $(A + B) / 2$ (average)
  - $(A - B) / 2$ (detail)
- Allows perfect reconstruction
- A sequence of n values becomes two sequences of $n/2$ values each

Haar Wavelet Example

- Let's recursively compute a pyramid of averages and detail coefficients for the sequence [9 7 3 5]

Resolution	Averages	Detail coefficients
4	[9 7 3 5]	
2	[8 4]	[1 −1]
1	[6]	[2]

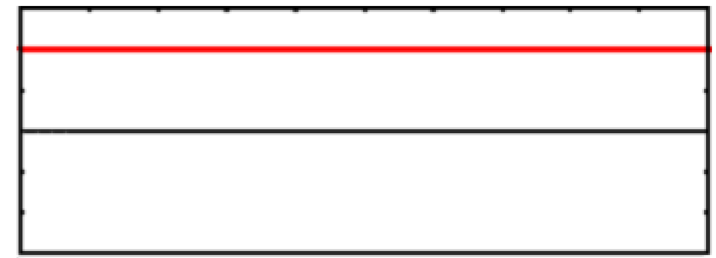
- Wavelet transform of sequence = [6 2 1 -1]

Average value (6)
Low-res detail (2)
High-res detail (1 -1)

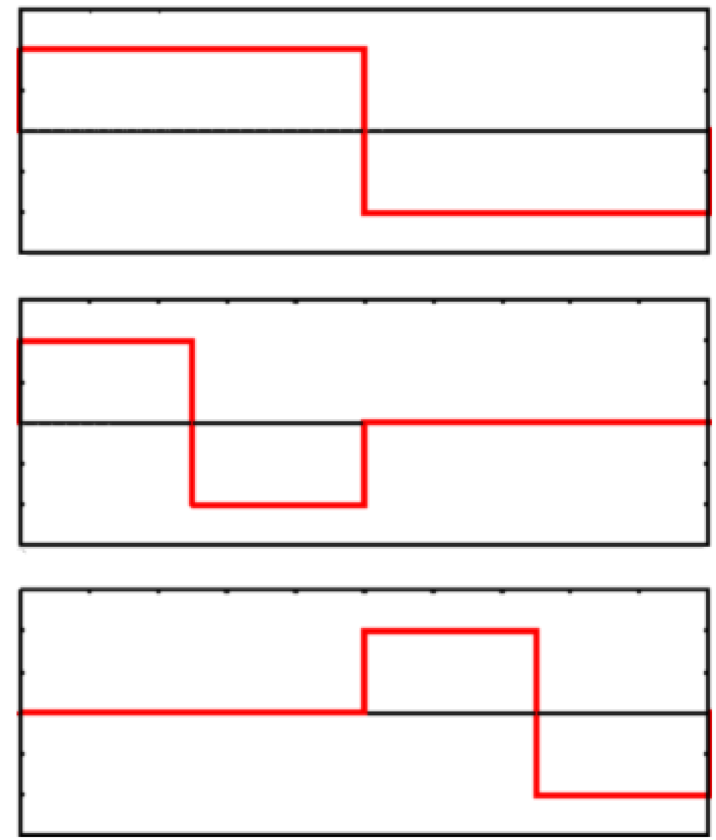
(Stollnitz, DeRose & Salesin, 1995)

Scaling and Wavelet Functions

- The average is computed via a *scaling function*

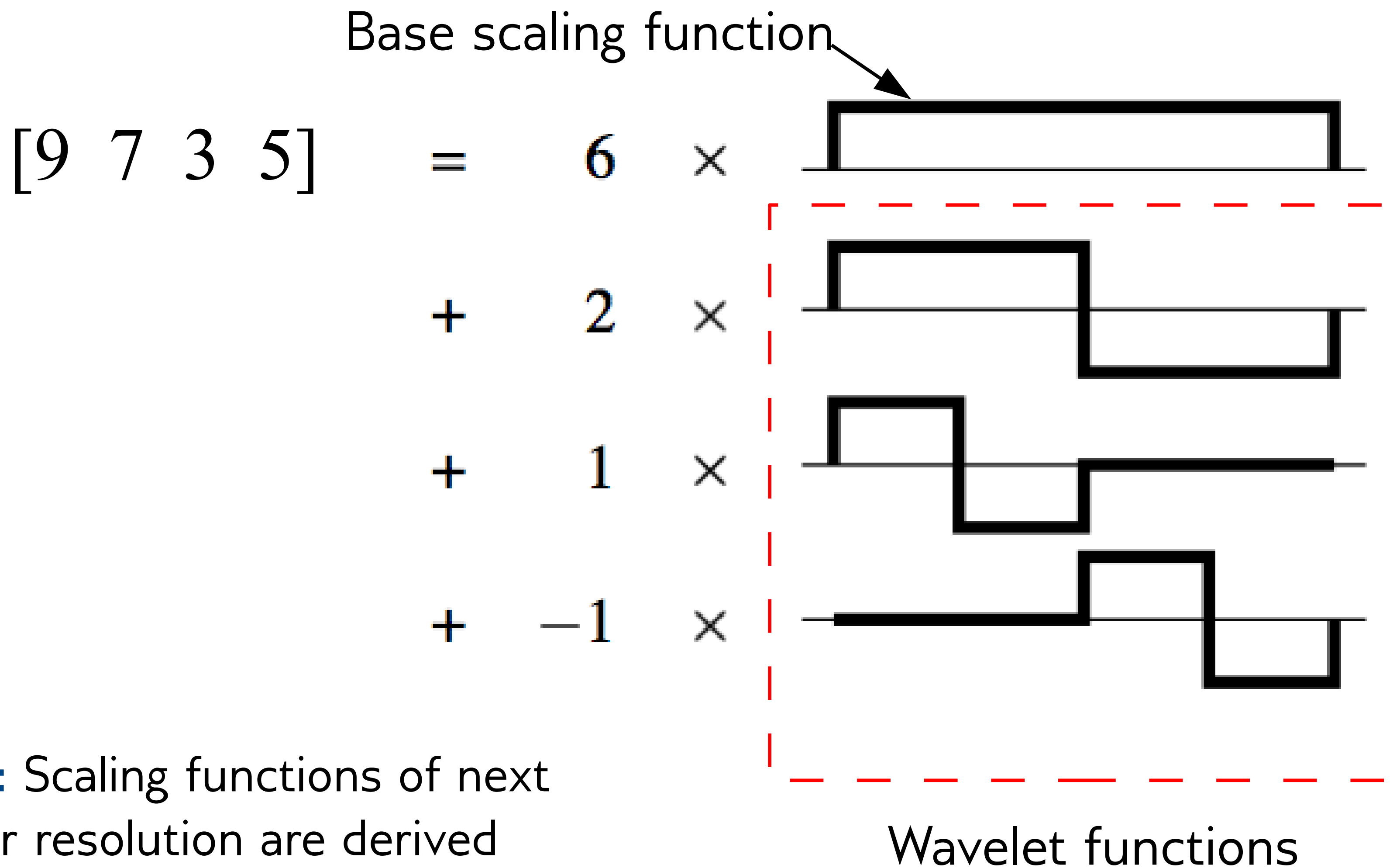


  - Low-pass filter
  - Gives lower resolution, smoothed version of image
- The detail coefficients are computed via *wavelet functions*



  - High-pass filter
  - Capture local deviations
  - Can be discarded/quantized for lossy compression

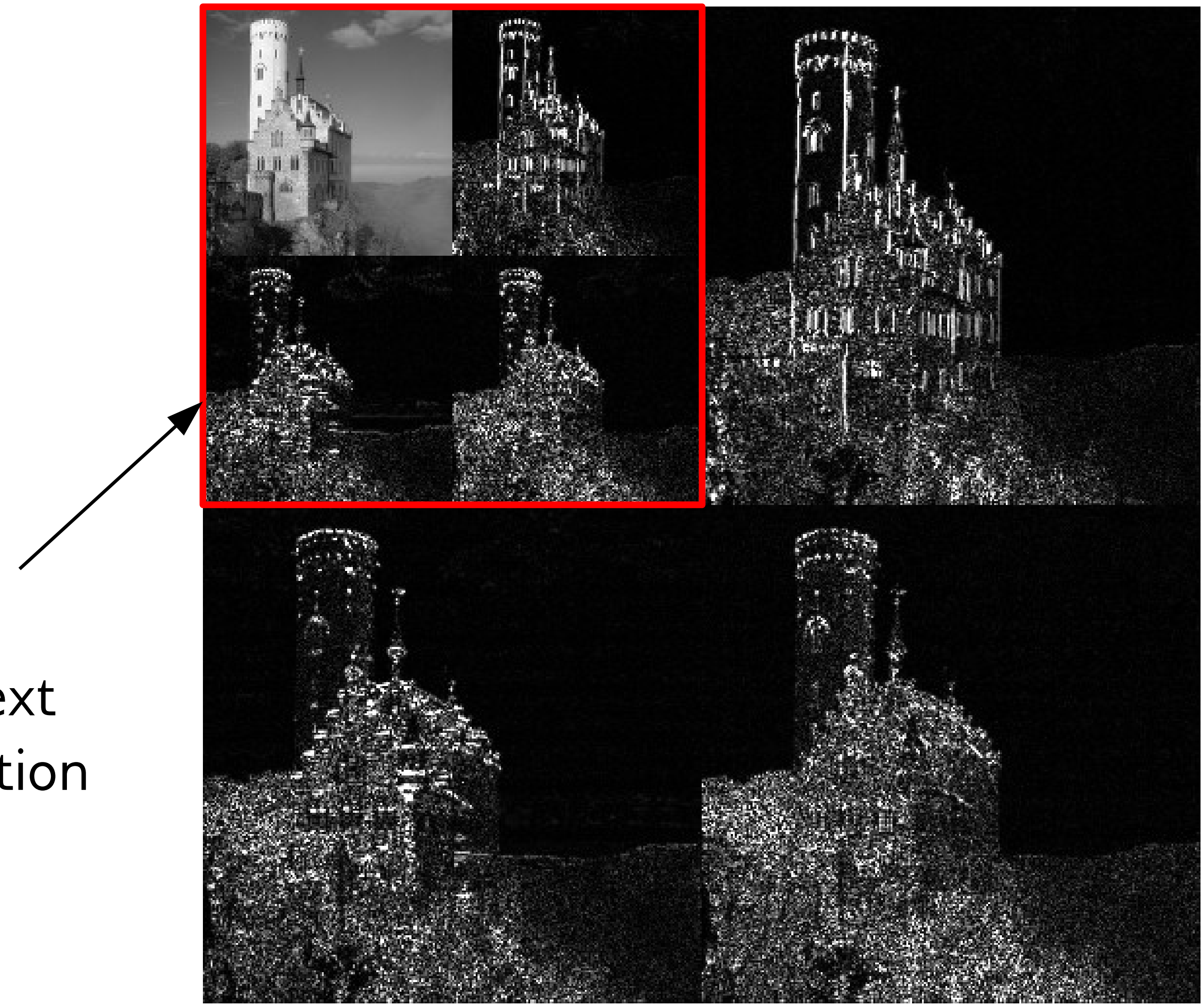
The Example Again



Note: Scaling functions of next higher resolution are derived from scaling + wavelet functions of current resolution

(Stollnitz, DeRose & Salesin, 1995)

JPEG2000: Incrementally add detail



Combined to give "average image" for next higher resolution