Lecture 17:
A Camera's Image Processing Pipeline
Part 2

Kayvon Fatahalian
CMU 15-869: Graphics and Imaging Architectures (Fall 2011)

Today

- Finish image processing pipeline
  - Gamma
  - JPG Compression

- Auto-focus / auto-exposure

- Camera processing elements

- Smart phone processing elements

Simplified image processing pipeline

- Correct for sensor bias (using measurements of optically black pixels)
- Correct pixel defects
- Vignetting compensation
- Dark frame subtract (optional)
- White balance
- Demosaic
- Denoise / sharpen, etc.
- Color Space Conversion

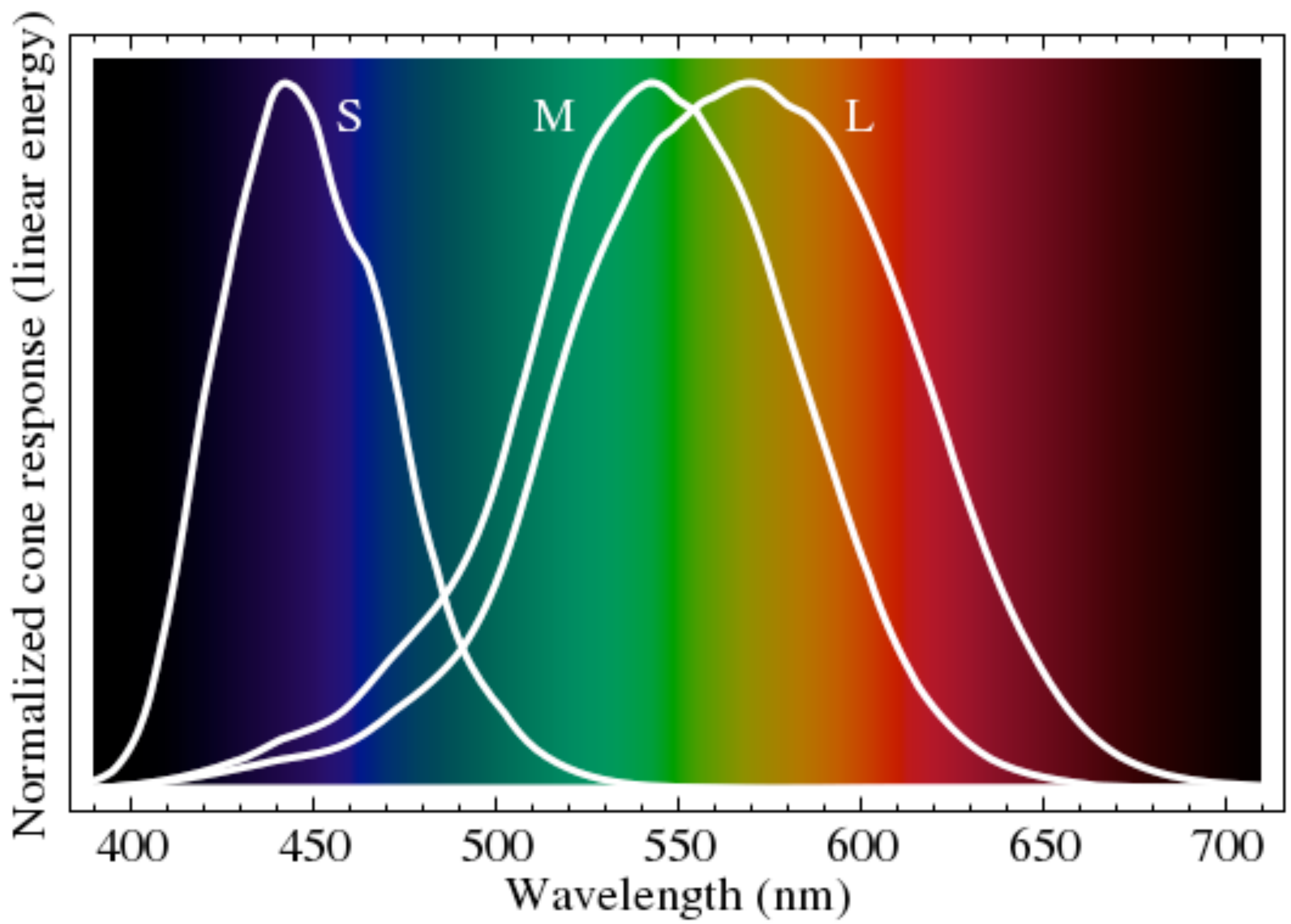
lossless compression → RAW file

Last time

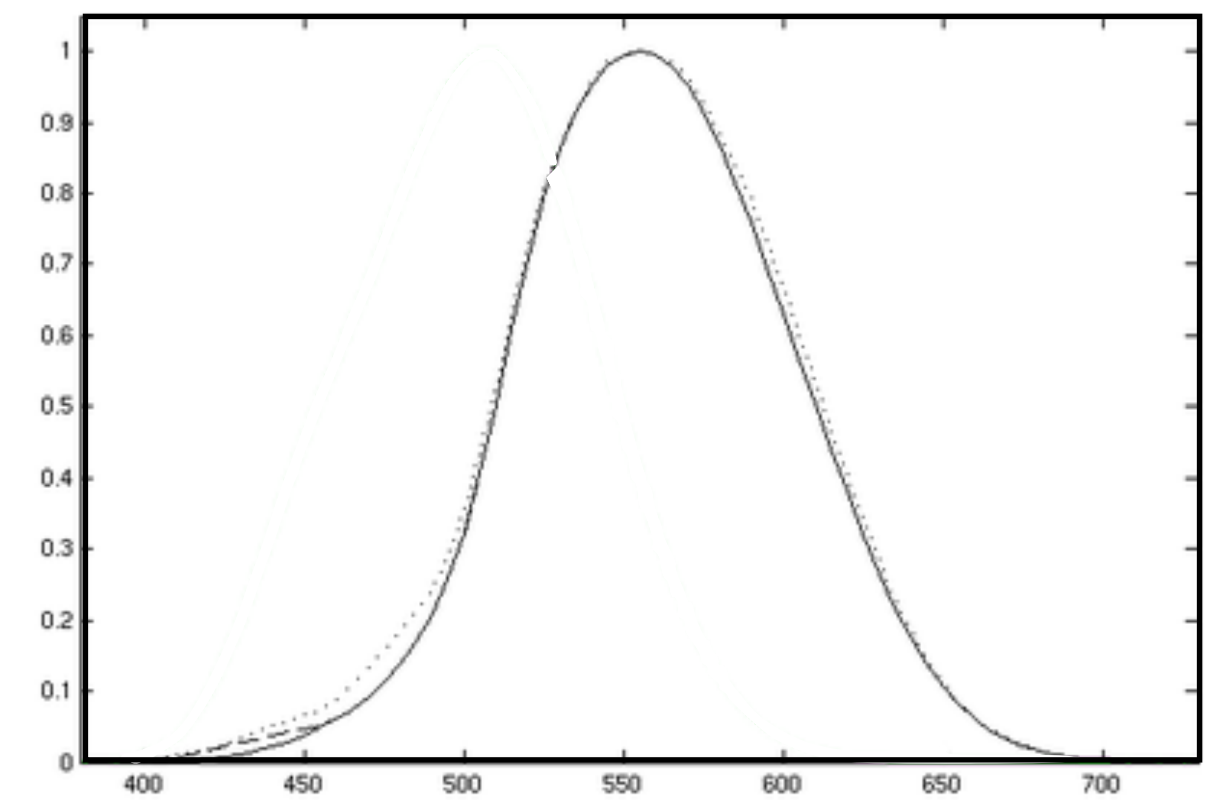
- Gamma Correction
- Color Space Conversion (Y'CbCr)
- 4:4:4 to 4:2:2 chroma subsampling
- JPEG compress (lossy)

→ JPEG file

Eye spectral response



Eye Spectral Response (S, M, L cones)



Avg. eye spectral sensitivity
(daytime-adapted)

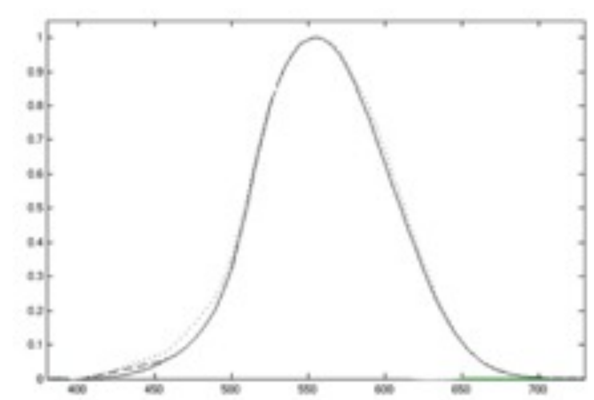
Uneven distribution of cone types
~64% of cones are L cones, ~ 32% M cones

Image credit: Wikipedia

Lightness (perceived brightness)

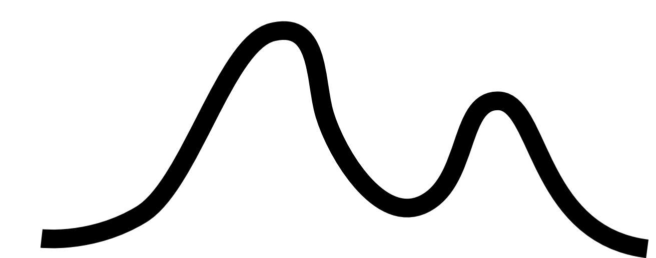
Perceived | **Physical Response**

Lightness (L*) ←? Luminance (L) = $\int_{\lambda}$



Eye spectral sensitivity

*



Radiance
(energy from scene)

Dark adapted eye: $L^* \propto L^{0.4}$

Bright adapted eye: $L^* \propto L^{0.5}$

So what does a pixel's value mean?

Gamma

Old CRT display: $L \propto \text{voltage}^{\gamma}$

$\gamma \sim 2.5$

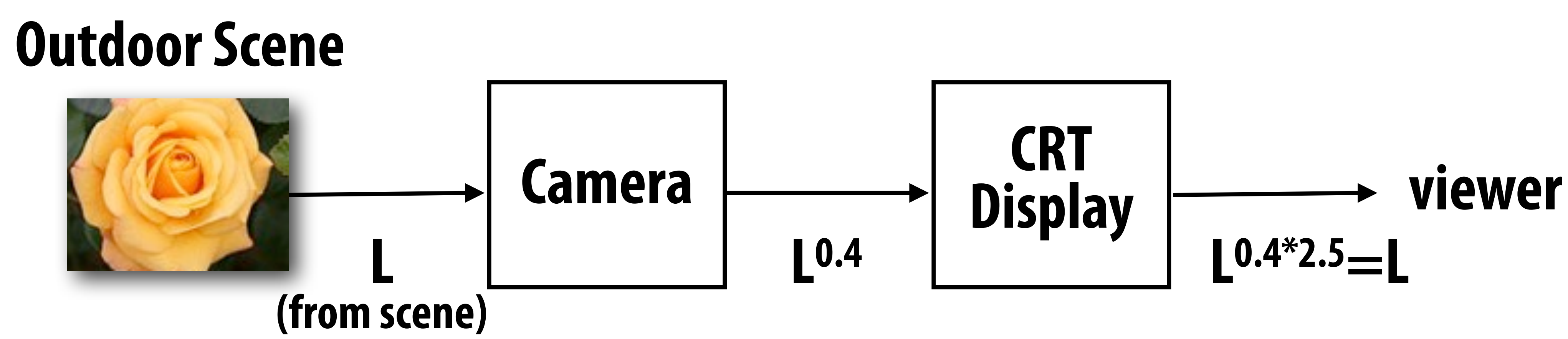
If pixels store L, what happens?

Desired Image

Gamma correction

Goal: want viewer to perceive luminance differences as if they were present in the environment where a picture is taken (note: reproducing absolute values not possible)

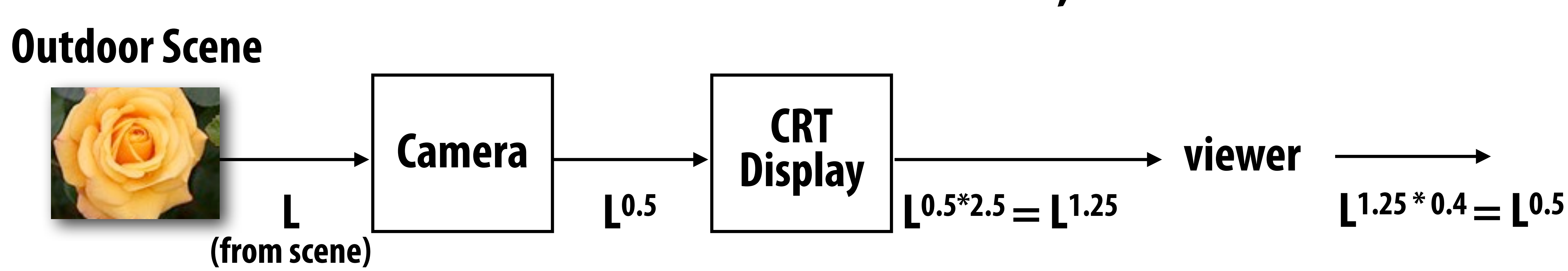
Can set TV camera to record L, store $L^{1/2.5} = L^{0.4}$



But scene is bright (viewer bright adapted) and living room is dark (TV viewer dark adapted)

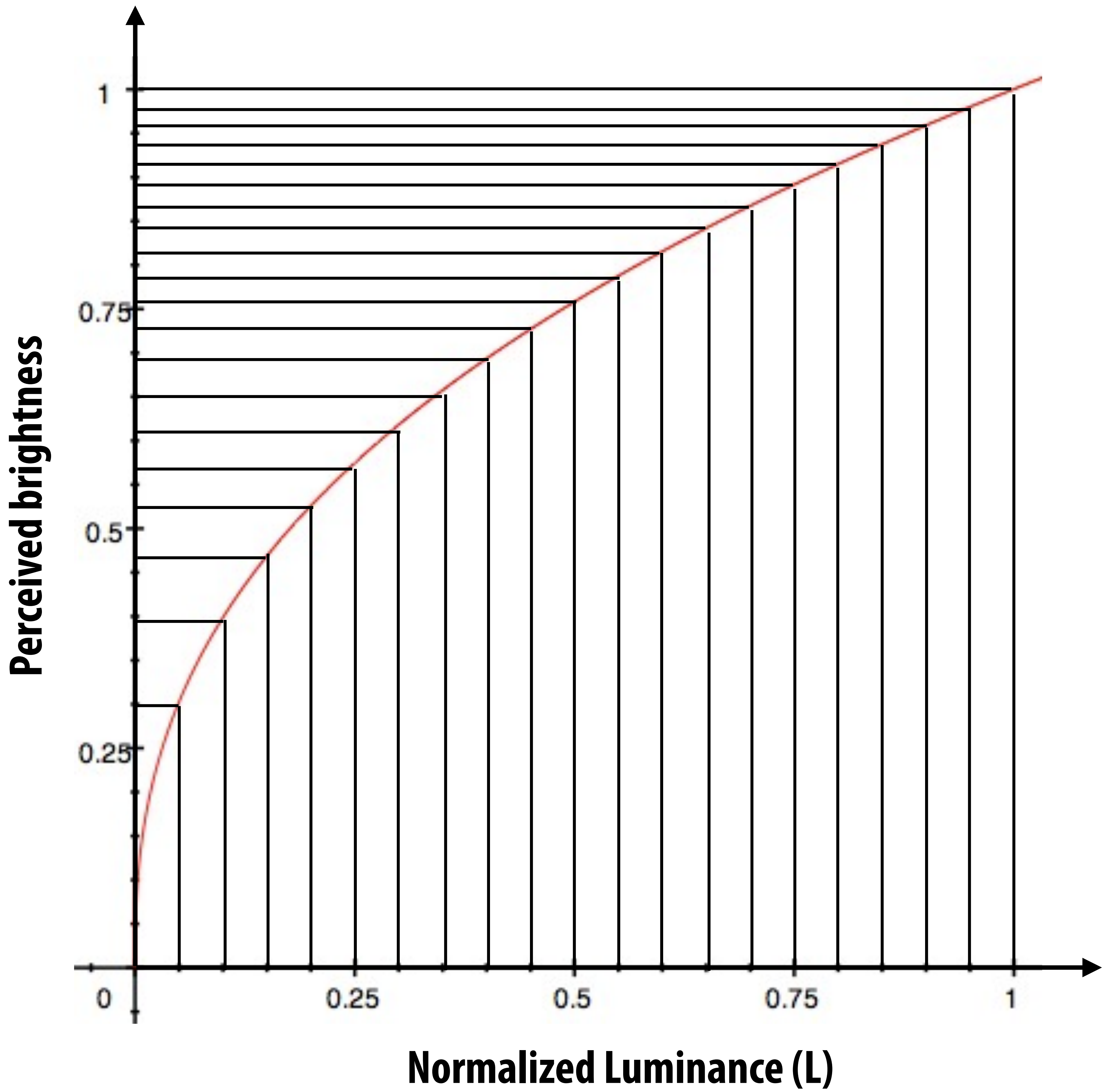
So TV viewer actually perceives $L^{0.4}$ (not the same as if viewer was "there")

Solution: TV cameras record L, store $L^{0.5}$



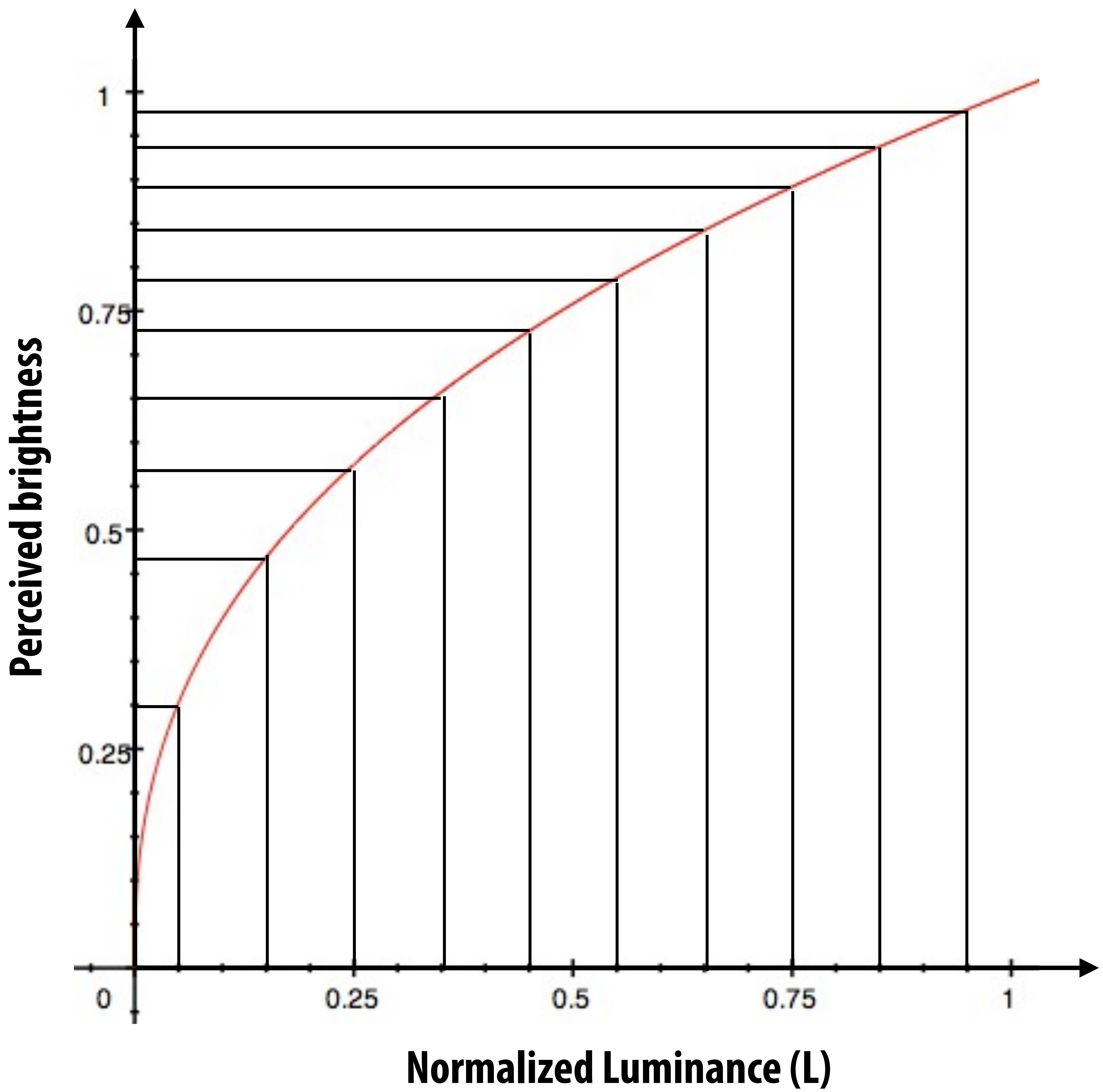
Credit: Marc Levoy, Stanford CS178

Power law



12 bit sensor pixel:
Can represent 4096 luminance values
Values are ~ linear in luminance

Quantization error

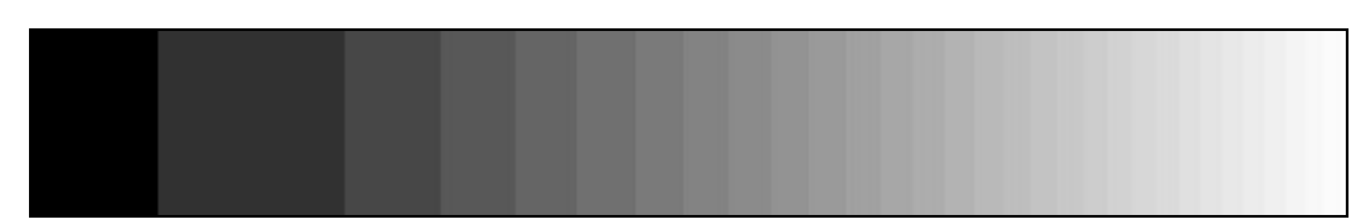


12 bit sensor pixel:
4096 representable luminance values
Values are ~ linear in luminance

Most images are not RAW files

8 bits per channel (256 unique values)
Risks quantization dark areas of image

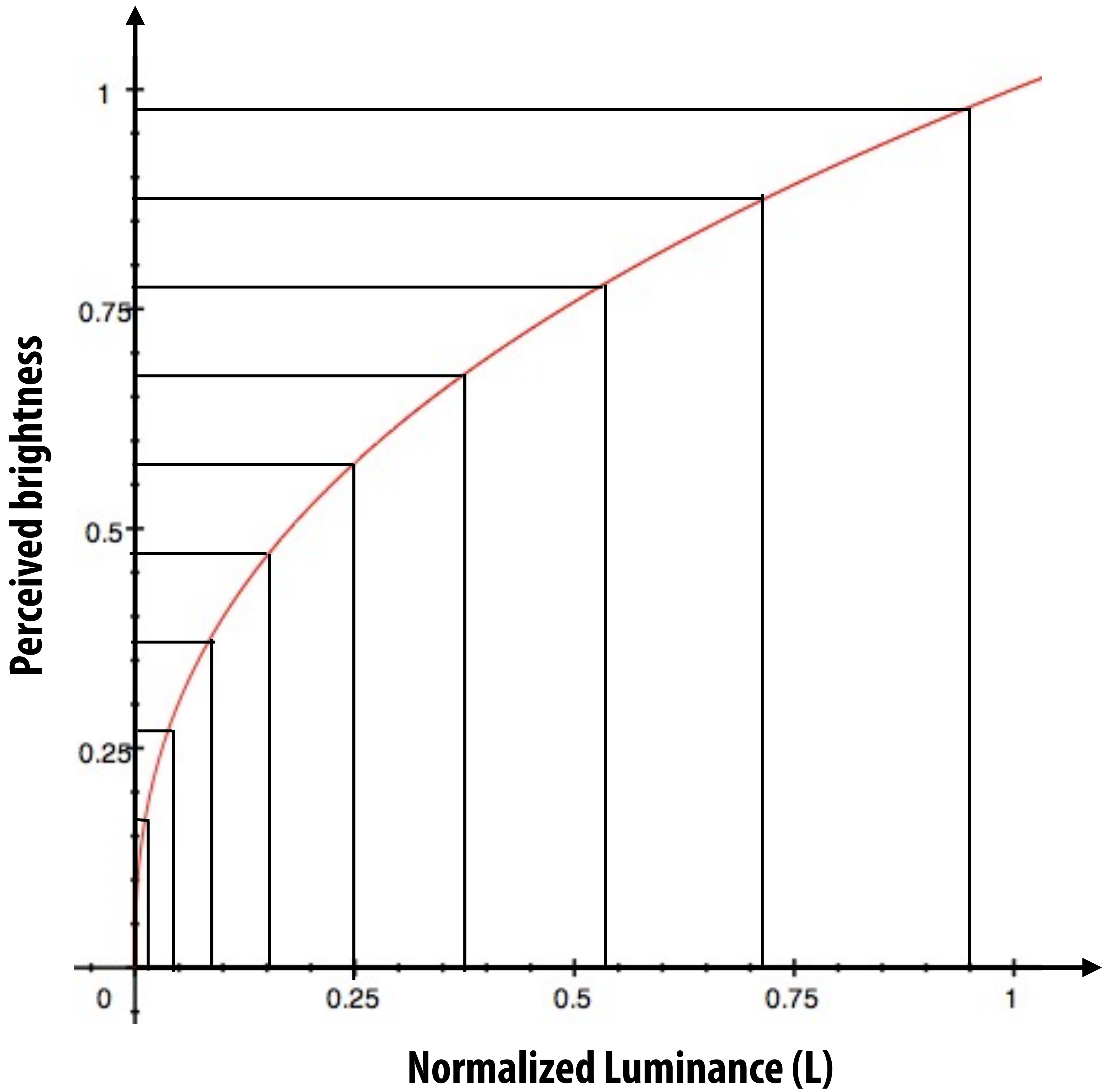
High bit depth pixels



5 bits/pixel (32 grays)
Pixel stores L

Store values linear in brightness

Evenly distribute values over perceptible range
(Make better use of available bits)



Rule of thumb: human eye cannot differentiate differences in luminance less than 1%

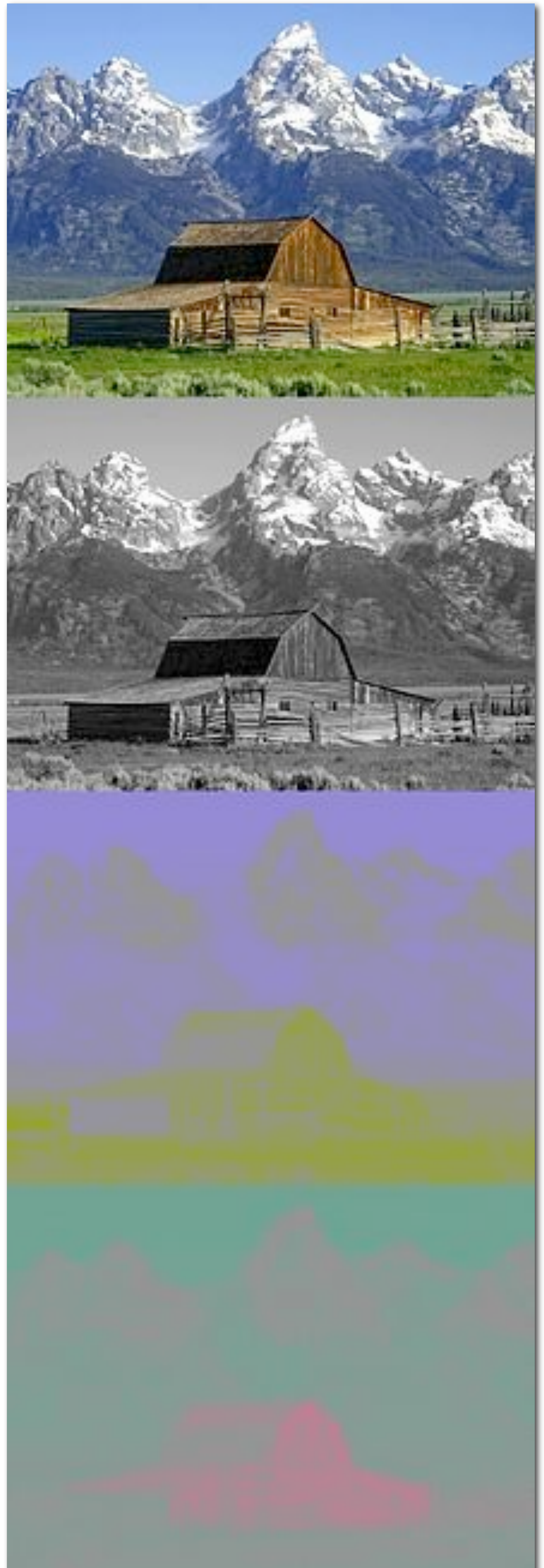
High bit depth pixels

5 bits/pixel (32 grays)
Pixel stores L

5 bits/pixel (32 grays)
Pixel stores $L^{0.45}$
Must compute $(pixel\_value)^{2.2}$ prior to display

Must take caution with subsequent pixel processing operations: should blending images average brightnesses or intensities?

Y’CbCr



Y’

Cb

Cr

Y’ = perceived luminance

Cb = blue-yellow deviation from gray

Cr = red-cyan deviation from gray

$$
\begin{aligned}
Y' &= 16 + \frac{65.738 \cdot R'_D}{256} + \frac{129.057 \cdot G'_D}{256} + \frac{25.064 \cdot B'_D}{256} \\
C_B &= 128 + \frac{-37.945 \cdot R'_D}{256} - \frac{74.494 \cdot G'_D}{256} + \frac{112.439 \cdot B'_D}{256} \\
C_R &= 128 + \frac{112.439 \cdot R'_D}{256} - \frac{94.154 \cdot G'_D}{256} - \frac{18.285 \cdot B'_D}{256}
\end{aligned}
$$

Image credit: Wikipedia

Chroma subsampling

Y'CbCr is an efficient storage (and transmission) representation because Y' can be stored at higher resolution than CbCr without much loss in perceived visual quality

4:2:2 representation:

Store Y' at full resolution

Store Cb, Cr at full vertical resolution, but half horizontal resolution

Y'_{00} Cb_{00} Cr_{00}	Y'_{10}	Y'_{20} Cb_{20} Cr_{20}	Y'_{30}
Y'_{01} Cb_{01} Cr_{01}	Y'_{11}	Y'_{21} Cb_{21} Cr_{21}	Y'_{31}

JPG Compression

JPG compression observations

- Low frequency content is predominant in images of the real-world

- Human visual system is less sensitive to high frequency sources of error

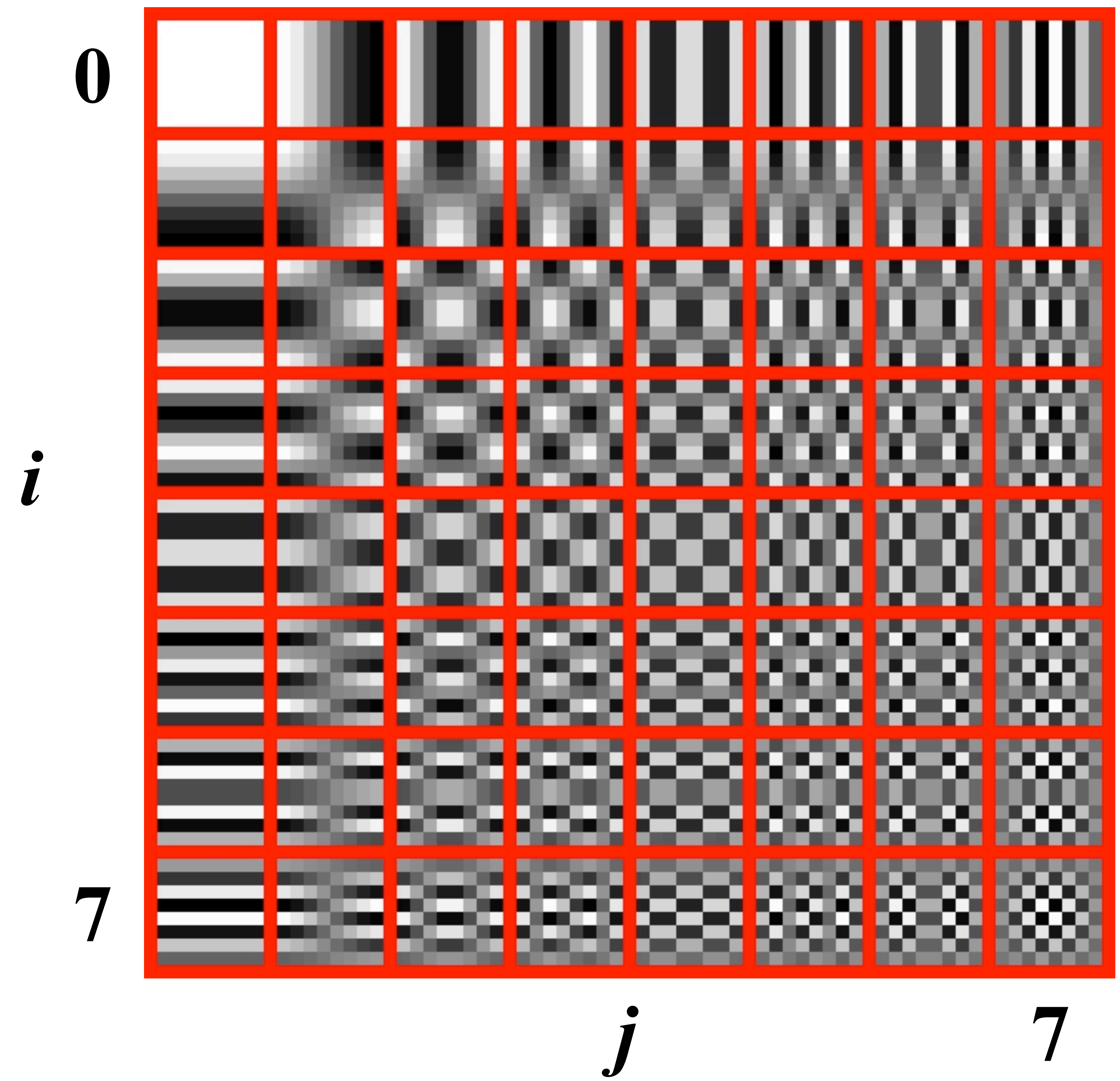
Slide credit: Pat Hanrahan

Discrete cosine transform (DCT)

Project image into its frequency components

DCT basis for 8x8 block of pixels

$$\cos\left[\pi\frac{i}{N}\left(x+\frac{1}{2}\right)\right] \times \underset{j}{\cos}\left[\pi\frac{j}{N}\left(y+\frac{1}{2}\right)\right]$$



Slide credit: Wikipedia, Pat Hanrahan

Quantization

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

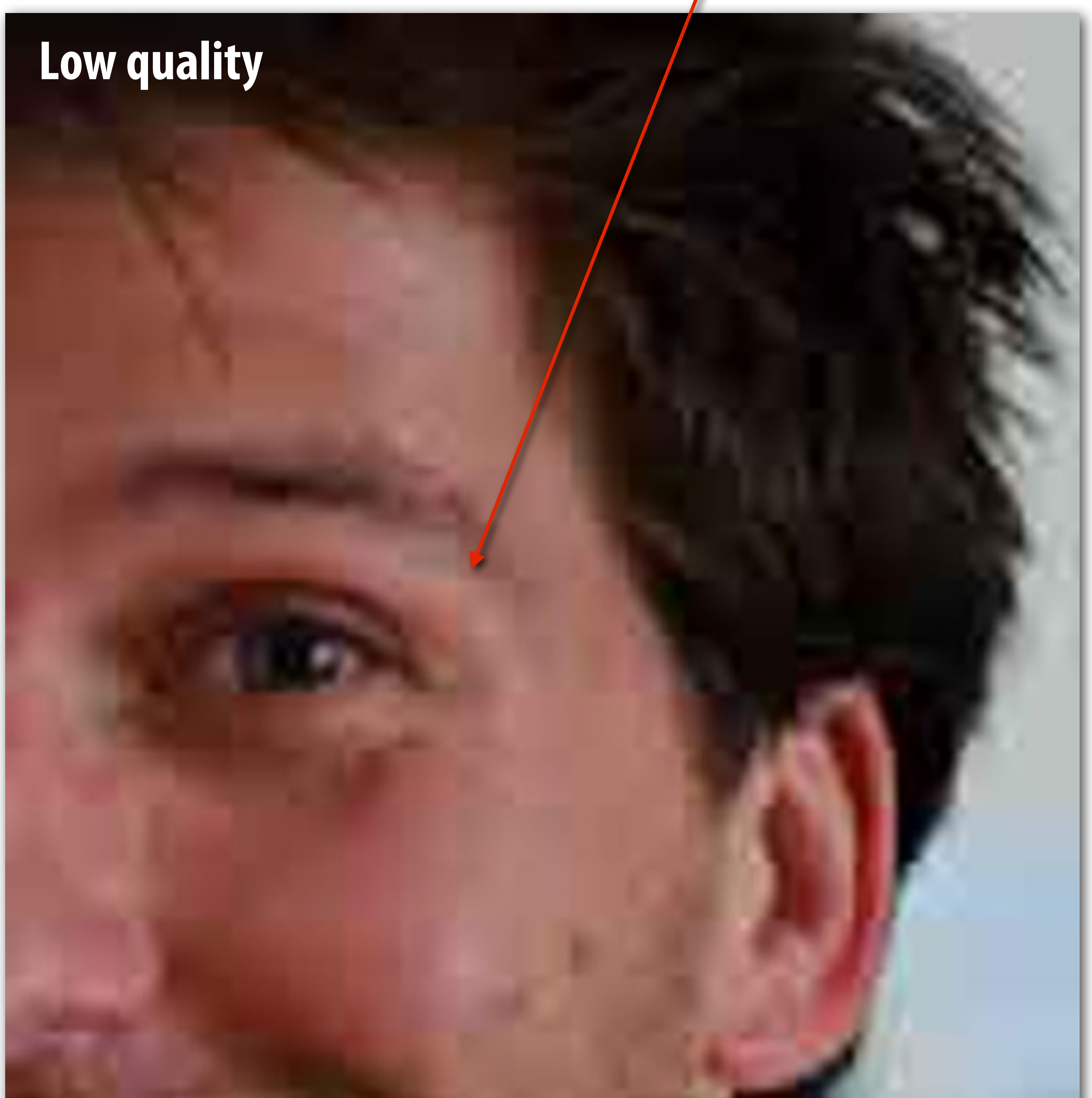
$$
= \underbrace{\begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}}_{\textbf{Quantized DCT}}
$$

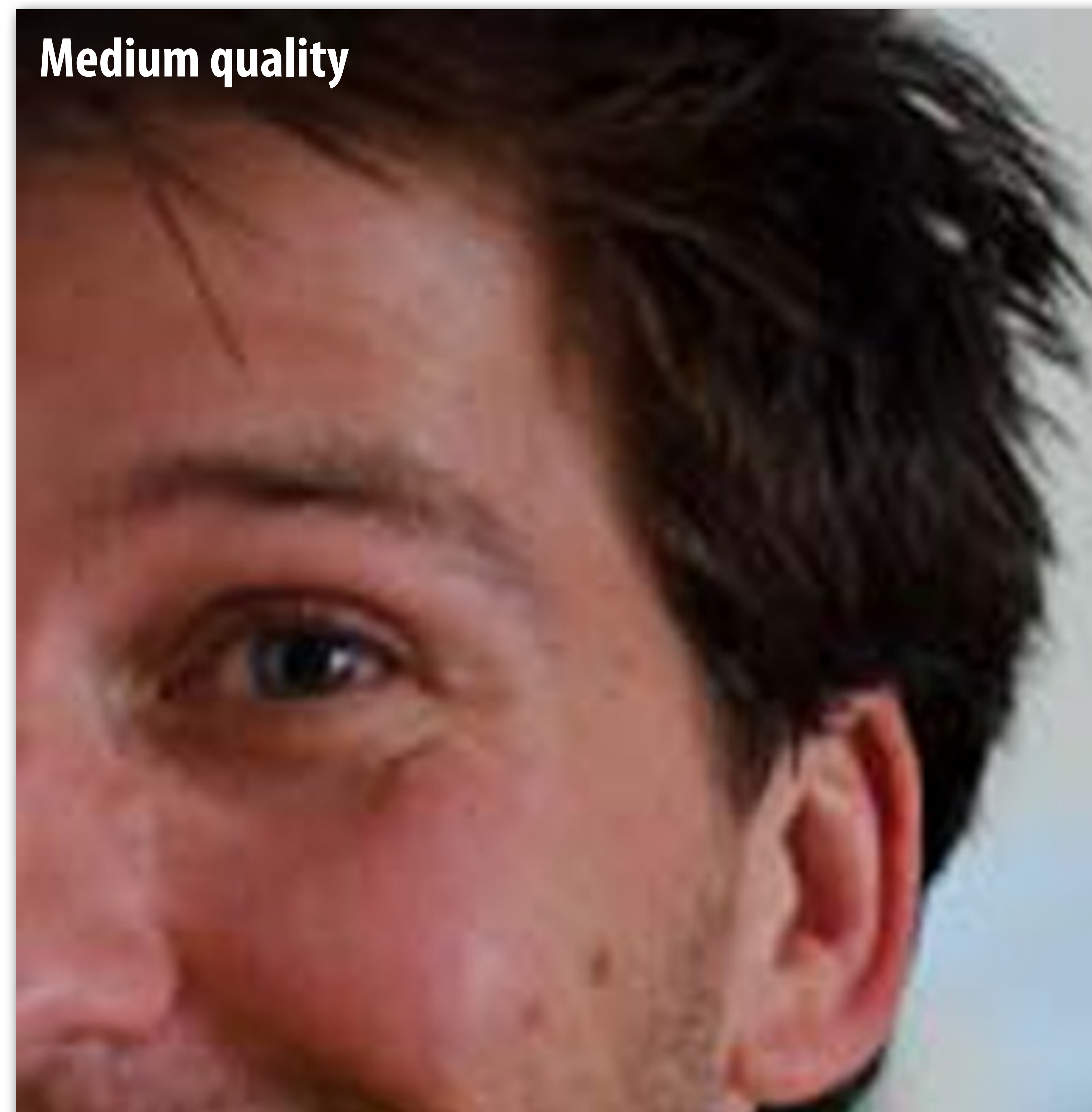
- **Quantization produces small values for coefficients**
- **Zeros out many coefficients**
- **JPEG quality setting scales coefficients**

Slide credit: Wikipedia, Pat Hanrahan

JPEG compression artifacts

8x8 pixel block boundaries

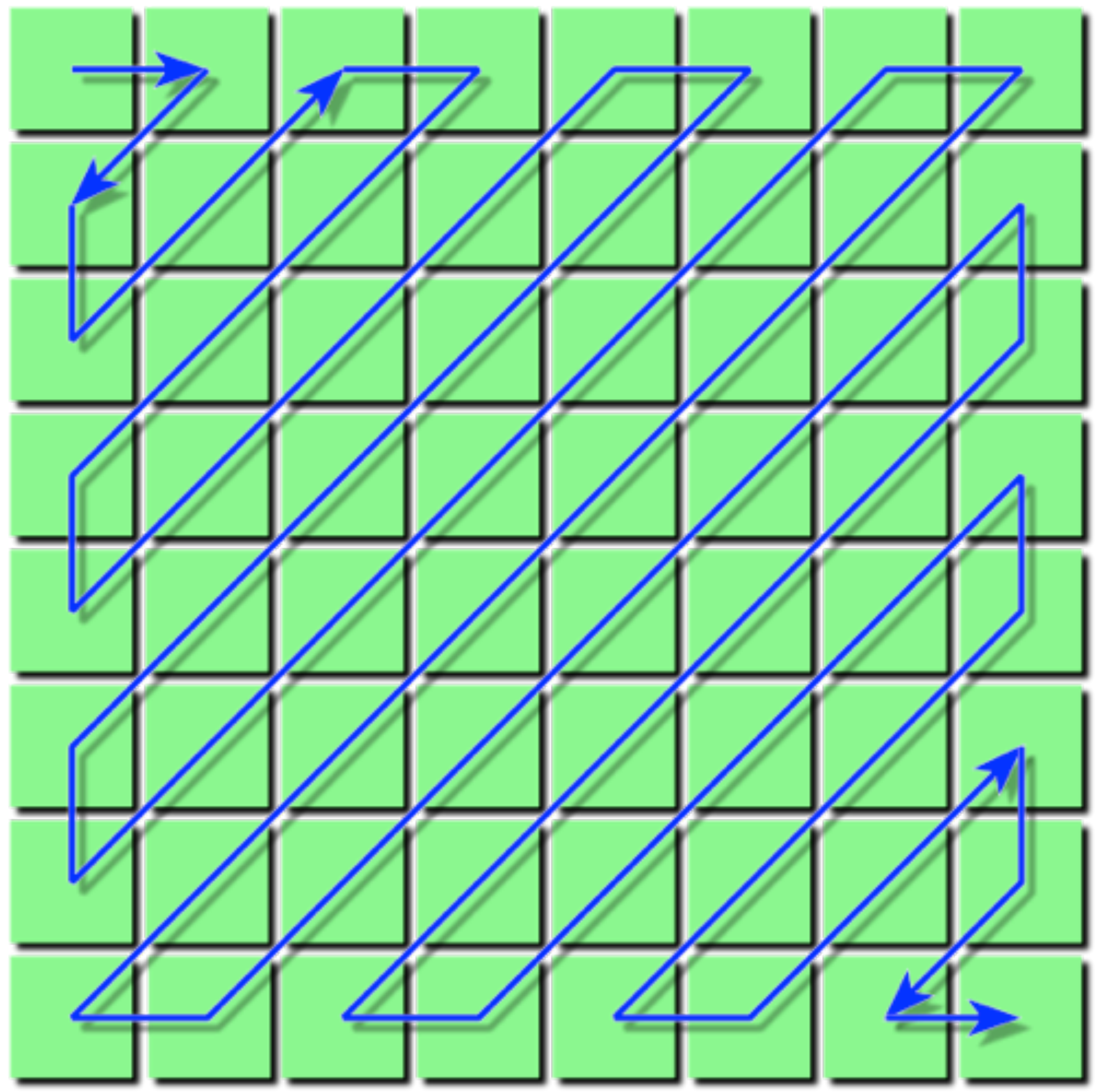




Lossless compression of quantized DCT values

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

Quantized DCT Values



Reordering

Entropy encoding: (lossless)

- **Reorder values**
- **RLE encode 0's**
- **Huffman encode non-zero values**

Image credit: Wikipedia

JPG compression summary

For each image channel

- For each 8x8 image block
  - Compute DCT
  - Quantize results (lossy)
  - Reorder values
  - RLE encode 0-spans
  - Huffman encode non-zero values

Exploiting characteristics of human perception

- Encode pixel values linearly in perceived brightness, not in luminance
- Y'CrCb representation allows reduced resolution in color channels (4:2:2)
- JPEG compression reduces file size at cost of quantization errors in high-spatial frequencies (human brain tolerates these errors at high frequencies more than at low frequencies)

Simplified image processing pipeline

- Correct for sensor bias (using measurements of optically black pixels)
- Correct pixel defects
- Vignetting compensation
- Dark frame subtract (optional)
- White balance
- Demosaic

12-bits per pixel
1 intensity per pixel
Pixel values linear in energy

- Denoise / sharpen, etc.
- Color Space Conversion
- Gamma Correction

3x12-bits per pixel
RGB intensity per pixel
Pixel values linear in energy

- Color Space Conversion (Y'CbCr)
- 4:4:4 to 4:2:2 chroma subsampling
- JPEG compress

3x8-bits per pixel
(until 4:2:2 subsampling)
Pixel values perceptually linear

Performance demo: Nikon D7000

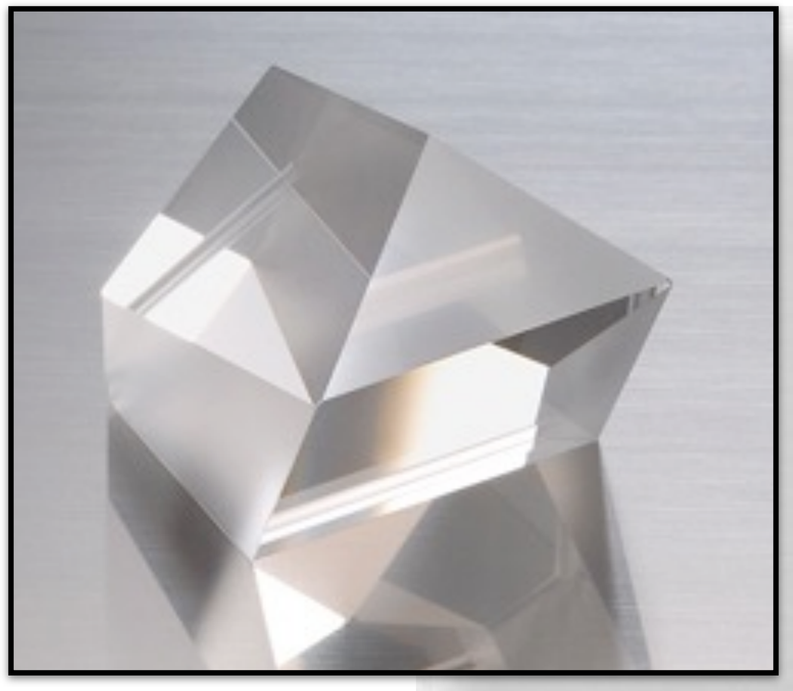
- Sensor made by Sony
  - 16 MP
  - Pixel size 4.78 x 4.78 um
  - 14 bit ADC

- 6 full-res JPG compressed shots / sec

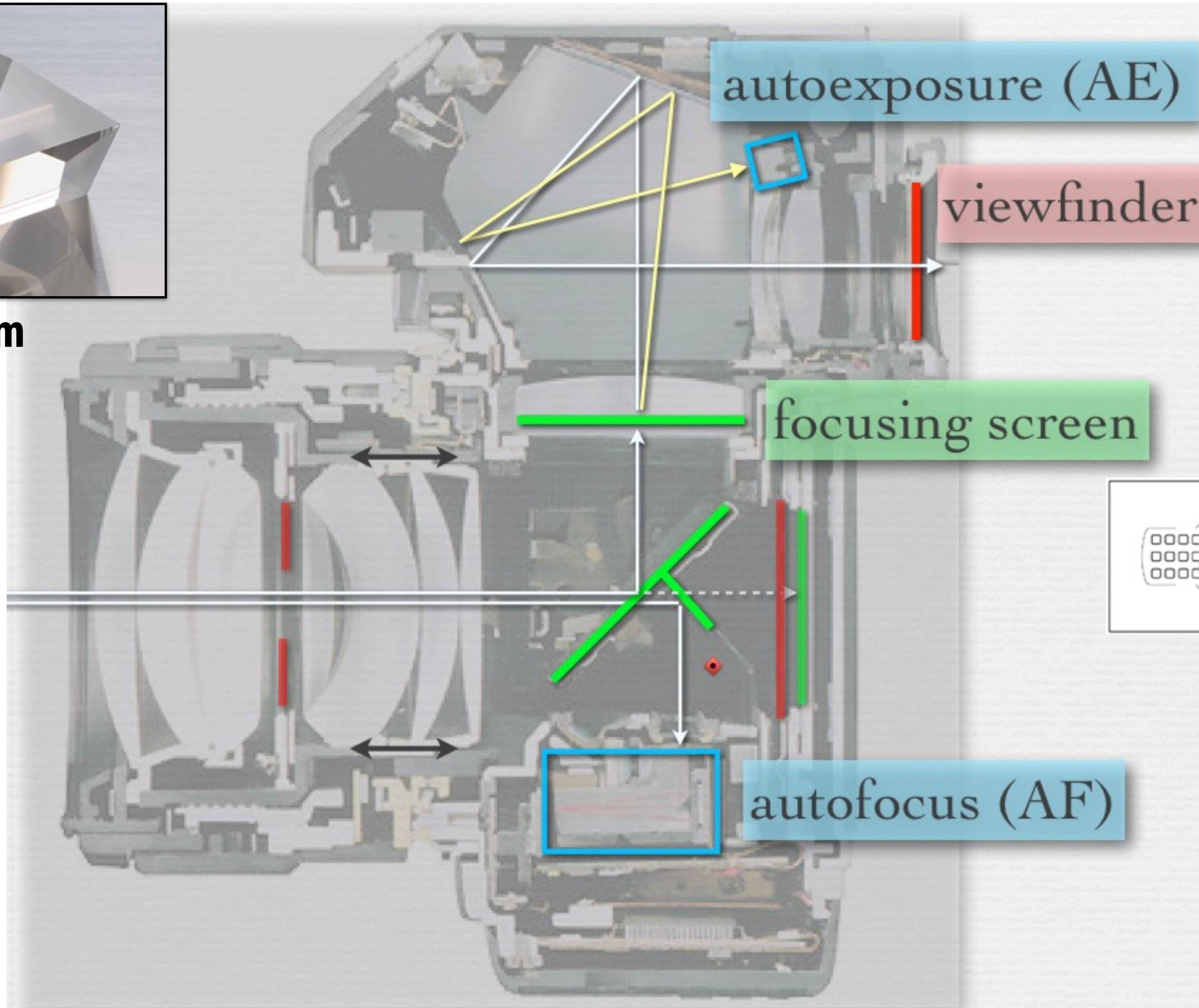
- Note: RAW to JPG conversation in Adobe Lightroom on my dual-core MacBook Pro: 6 sec / image (36 times slower)

Auto Focus / Auto Exposure

SLR Camera



Pentaprism



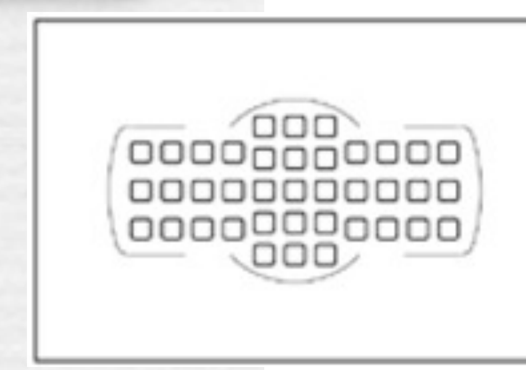


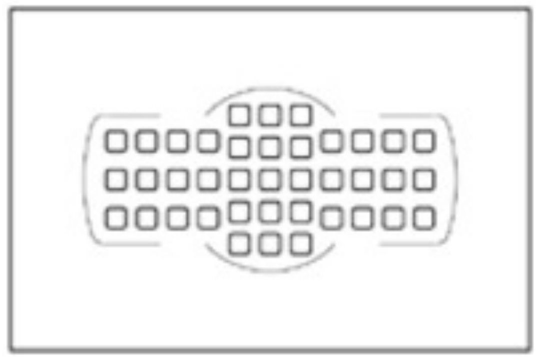
Image credits: Nikon, Marc Levoy

Demos

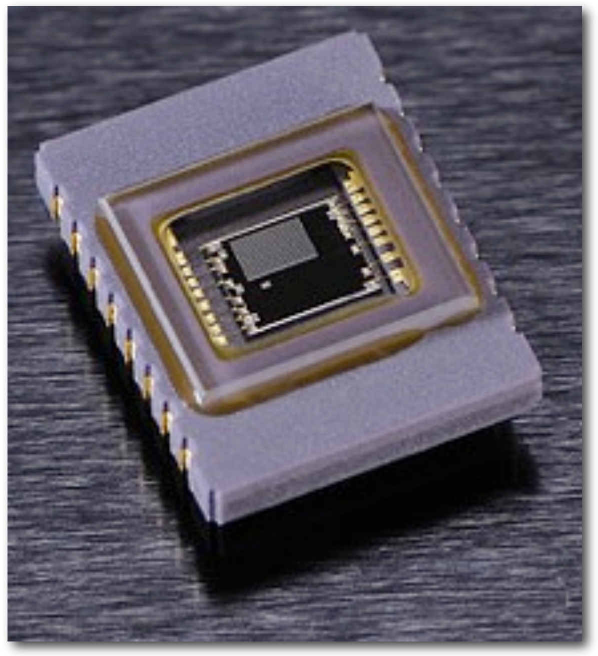
- Phase-detection
  - Common in SLRs

- Contrast-detection
  - Point-and-shoots, cell-phone cameras

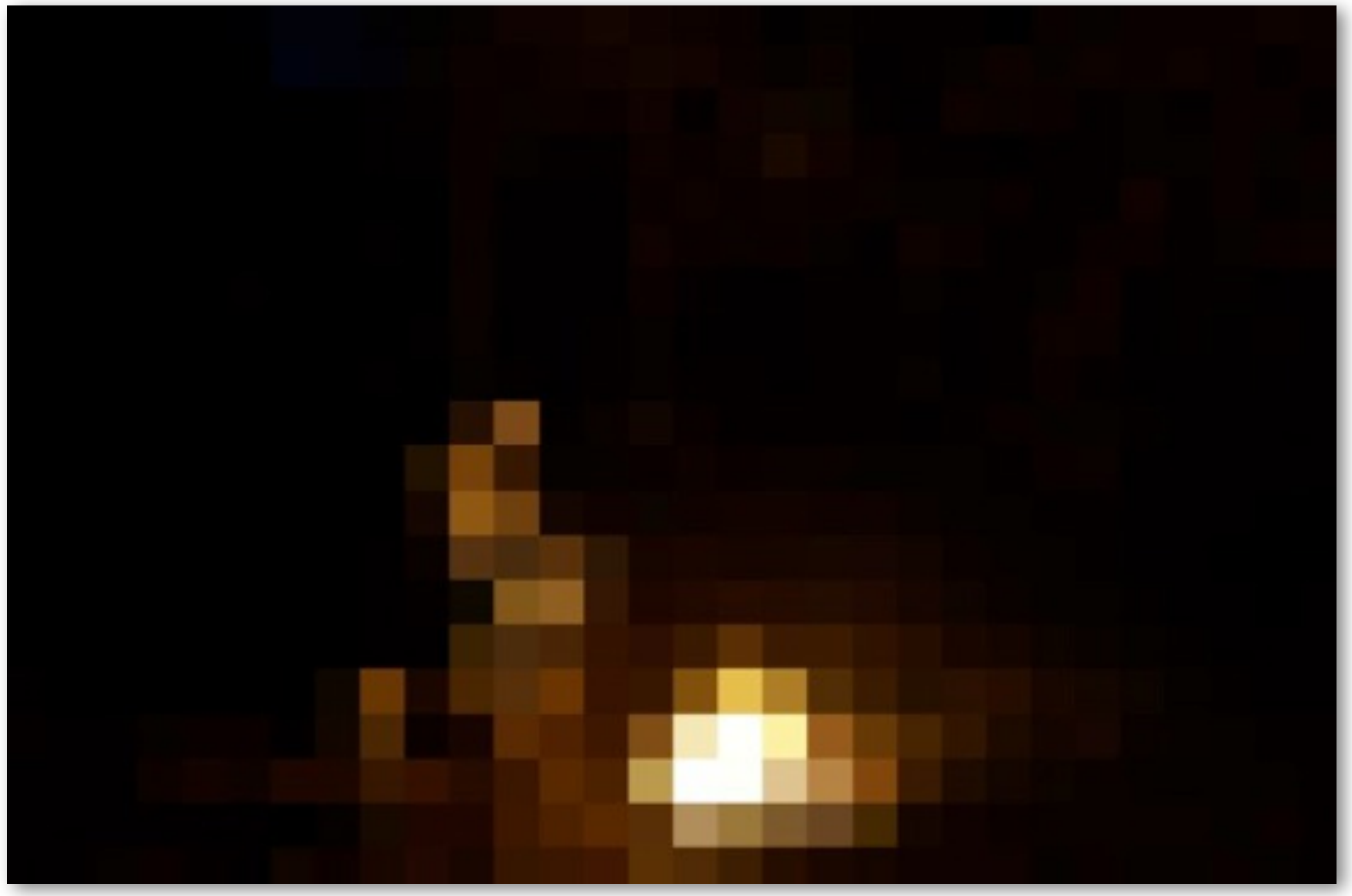
Nikon D7000



- Auto-focus sensor: 39 regions
- Metering sensor: 2K pixels
  - Auto-exposure
  - Auto-white-balance
  - Subject tracking to aid focus (predicts movement)
- Shutter lag ~ 50ms



Auto exposure



Low resolution metering sensor capture

Metering sensor pixels are large
(higher dynamic range than main sensor)

How do we set exposure?

What if a camera doesn't have a separate metering sensor?

Image credits: Marc Levoy, Andrew Adams

AF/AE summary

- DSLRs have additional sensing/processing hardware to assist with the "3A's" (auto-focus, auto-exposure, auto-white-balance)
  - Phase detection AF: optical system directs light to AF sensor
  - Nikon metering sensor: large pixels to avoid over-saturation

- Point-and-shoots, cell-phone cameras make these measurements by processing data from the main sensor
  - Contrast detection AF: search for lens position that produces large image gradients
  - Exposure metering: if pixels are saturating, meter again with lower exposure

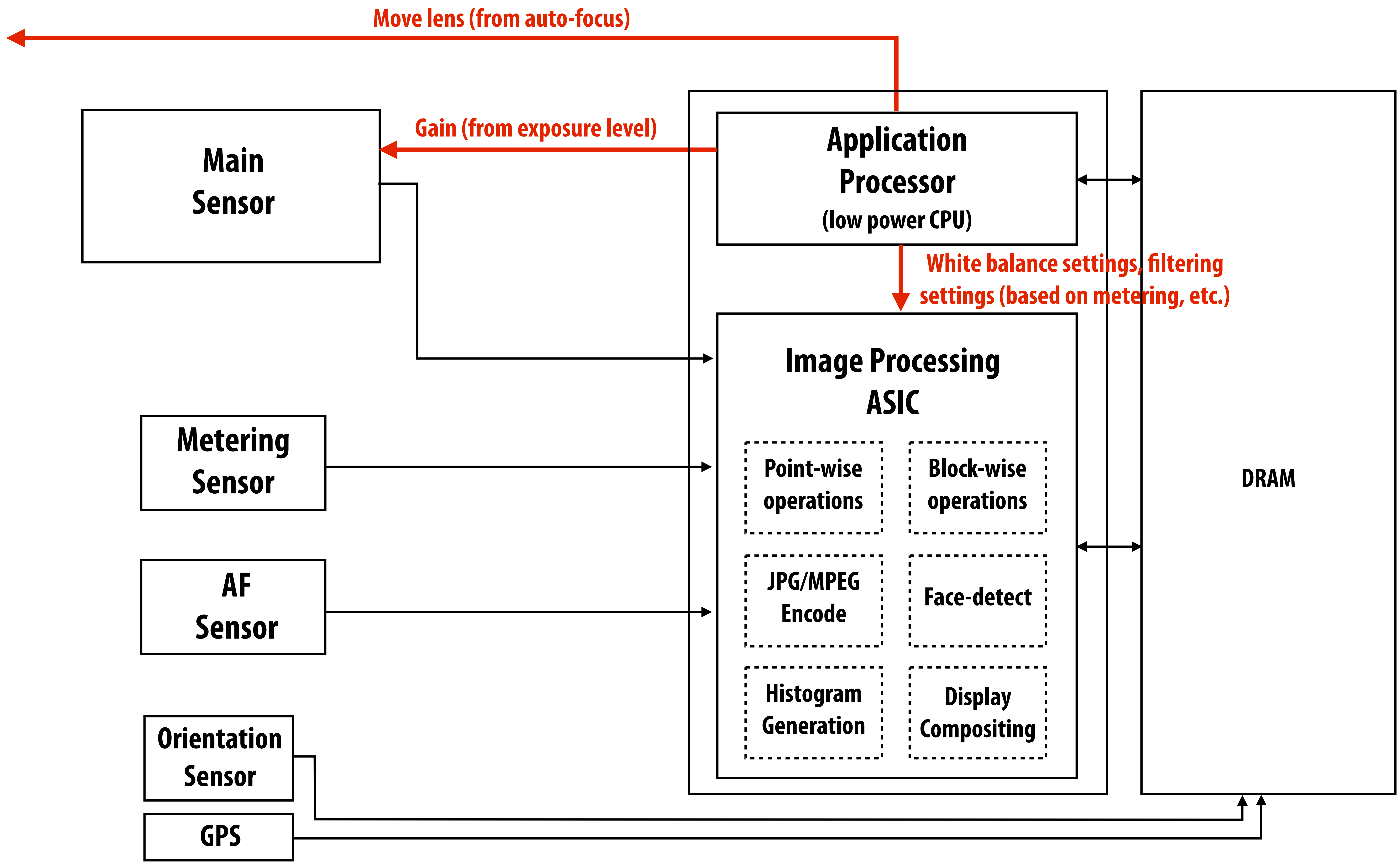
- In general AF/AE/AWB is a computer vision problem
  - Understand the scene well enough to choose image capture/processing parameters that approximate the image a human would perceive
  - As processing/sensing power increases, these algorithms are getting a lot smarter

Camera processing resources

Generic SLR camera

Consider everything that happens from shutter press to image!

Do designers care about latency or throughput?



Smart phone processing resources

Texas Instruments OMAP 4

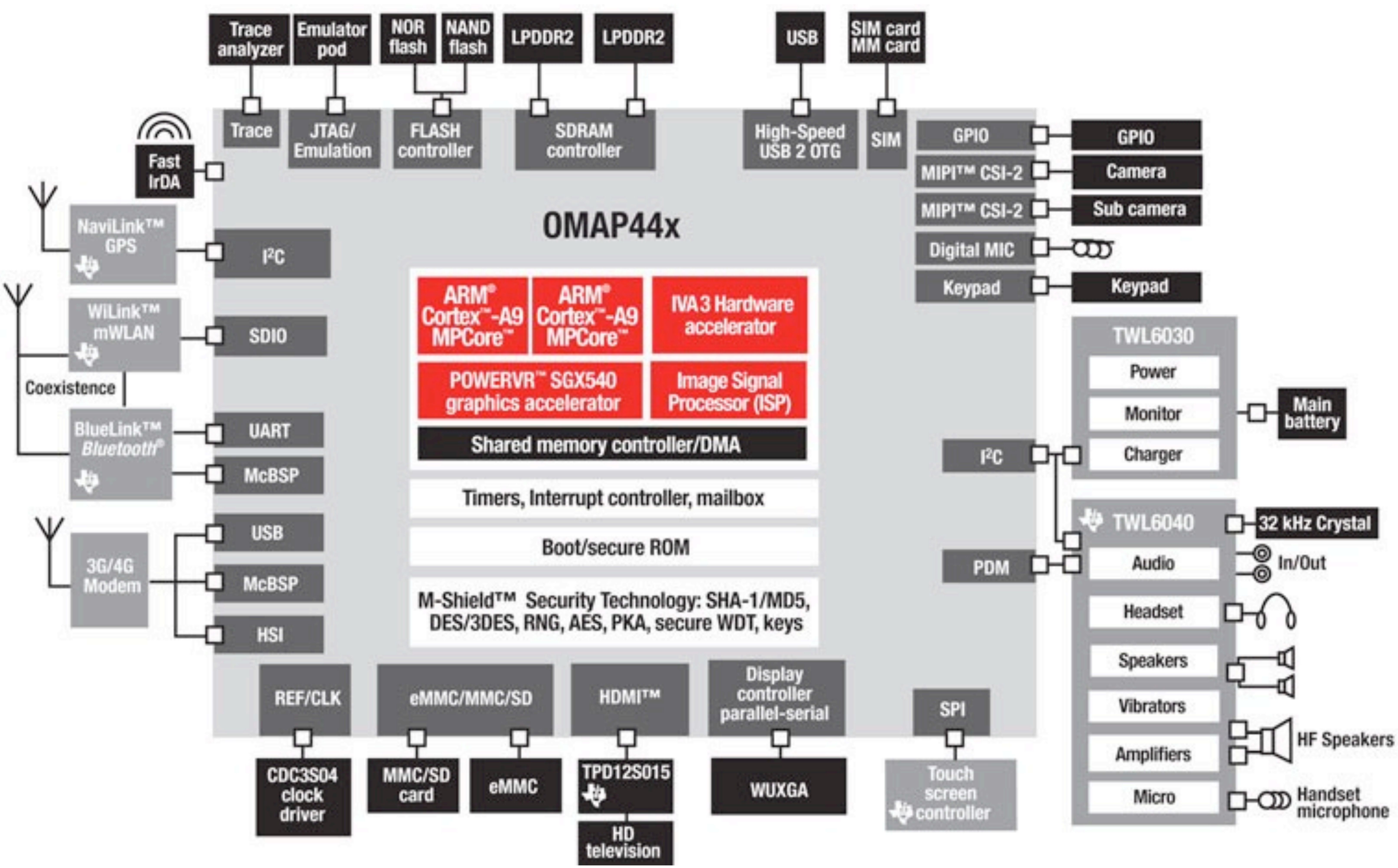


Image credit: TI

NVIDIA Tegra 3

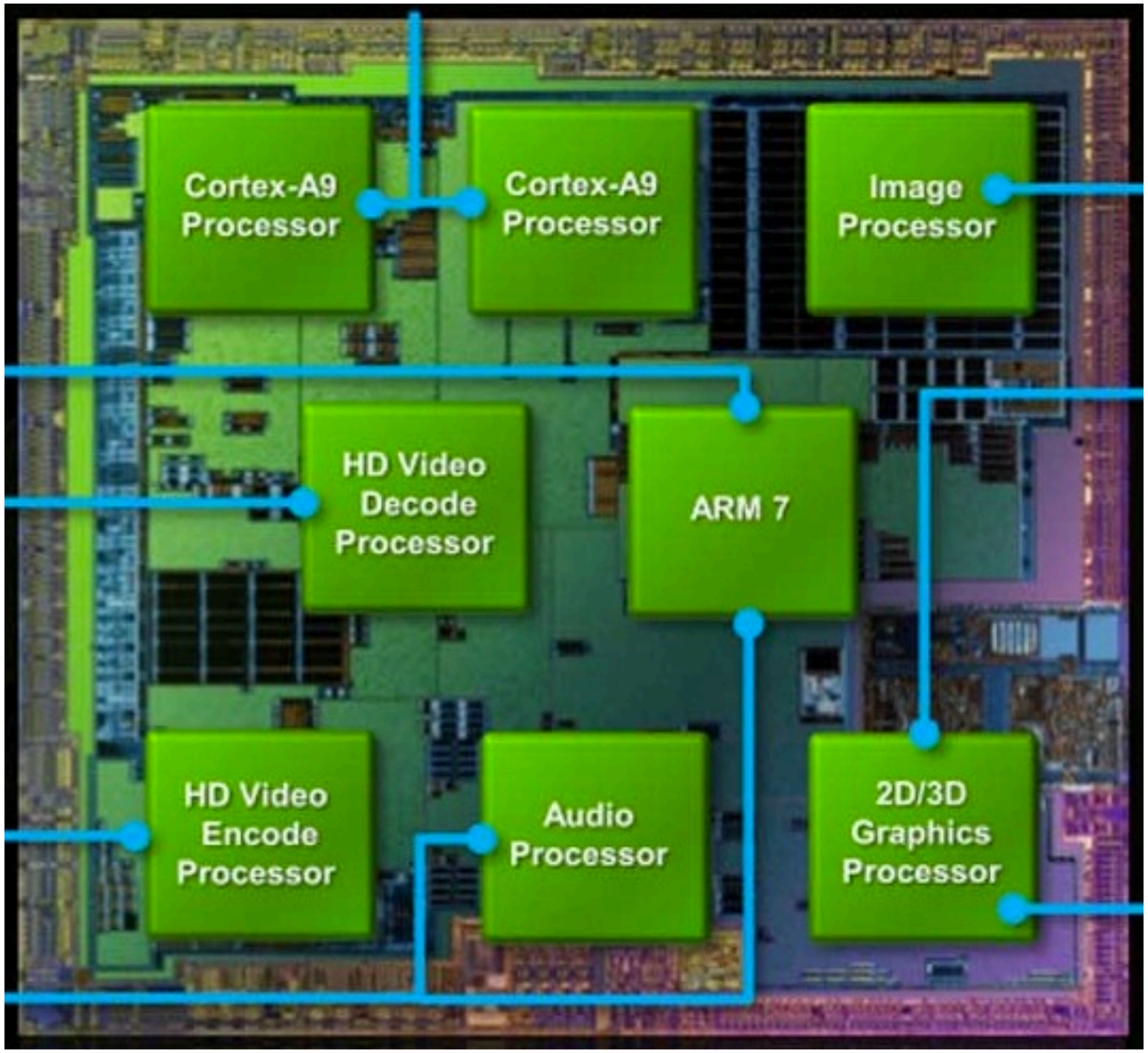


Image credit: NVIDA

Texas Instruments OMAP 5

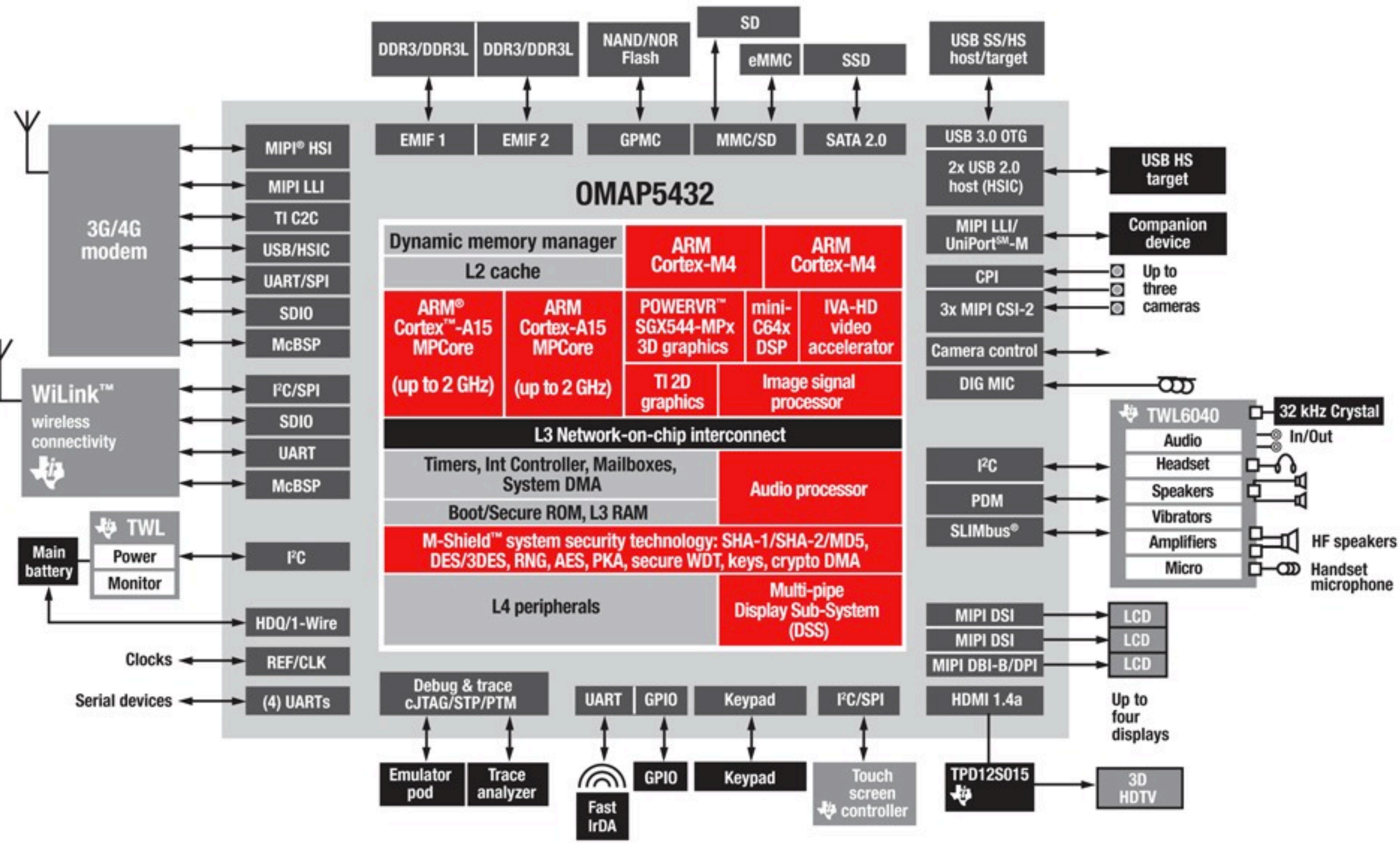


Image credit: TI

Discussion

- Traditional rule of thumb in system design is to design simple, general-purpose components. This is not the case with mobile processing systems (perf/watt)

- Needs of high bandwidth sensing and media processing are a big part of these designs [image/video/audio processing, 2D/3D graphics]
  - User interfaces are visually rich
  - Games
  - Speech recognition
  - Photography/video:
    - Acquire signal, compute images (not directly measuring an image)
    - More processing --> smarter sensing
    - More processing --> more flexibility?

- Questions:
  - Re-homogenize, or become increasingly heterogeneous?
  - How does an application developer think about these systems?

Next time

Light field cameras

Next, next time

Depth cameras

Next, next, next time

Frankencamera
(programming systems for cameras)