

# Image Compression

or

How to squeeze lots of pictures into an iPod

Jeffrey A. Fessler

EECS Department  
The University of Michigan

Engin 110

2008-03-27

# Cell Phones Everywhere



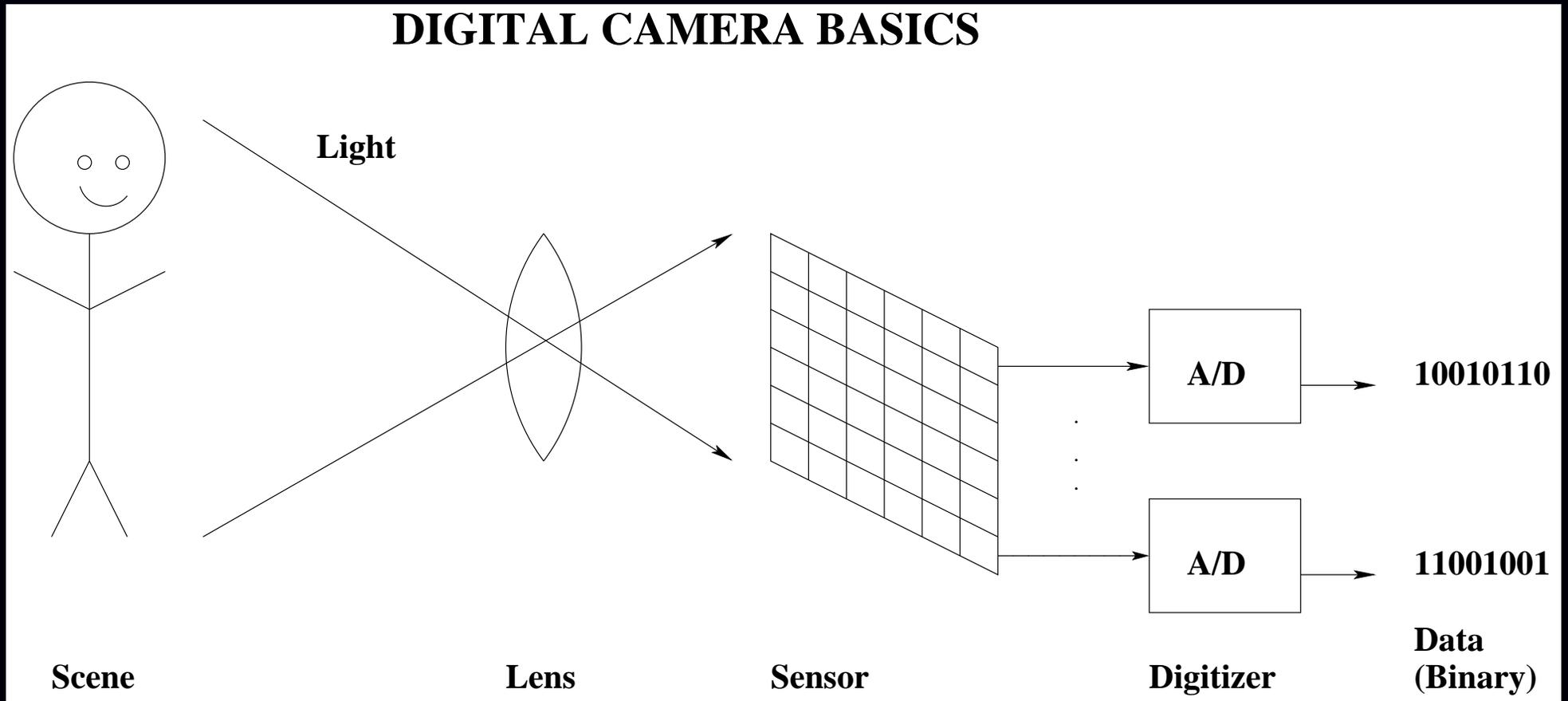
Notice anything funny about this picture?

# What is Inside an iPod?



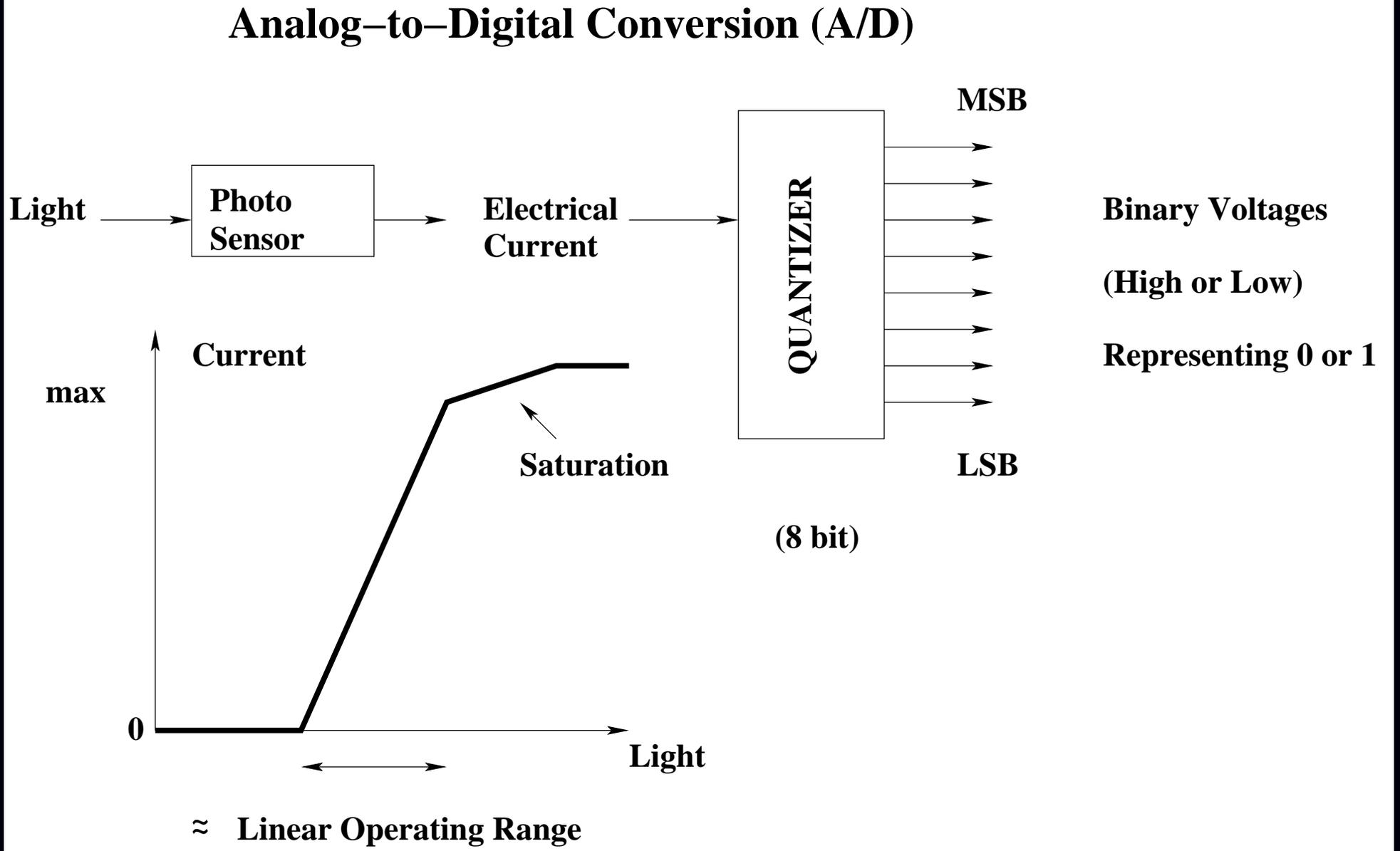
This talk concerns the invisible algorithms executing inside the chips, within iPods, digital cameras, cell phones, ...

# Digital Camera Basics



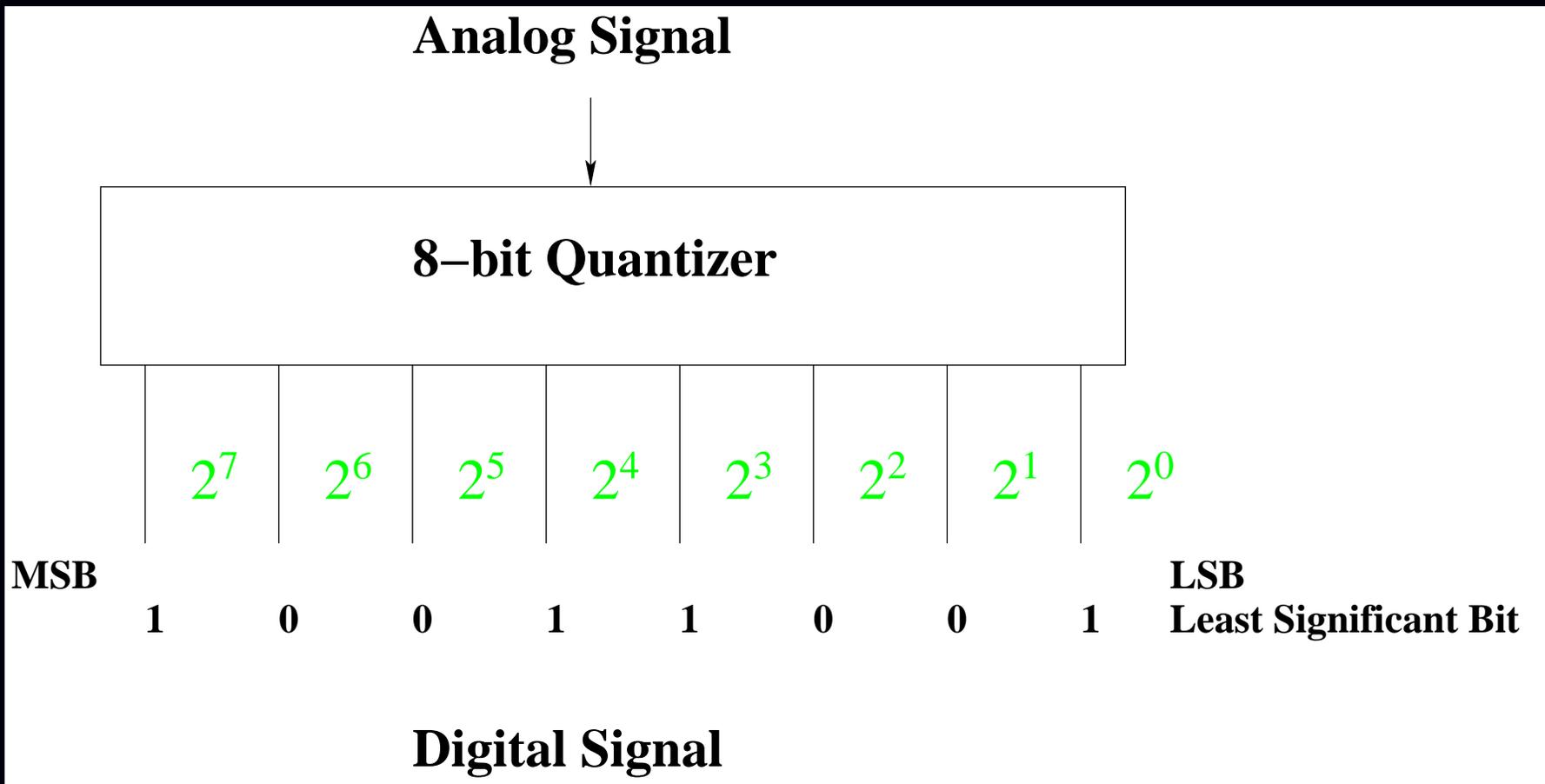
Key component: analog-to-digital converter

# Analog-to-Digital (A/D) Conversion



Key component: quantizer

# Quantizer



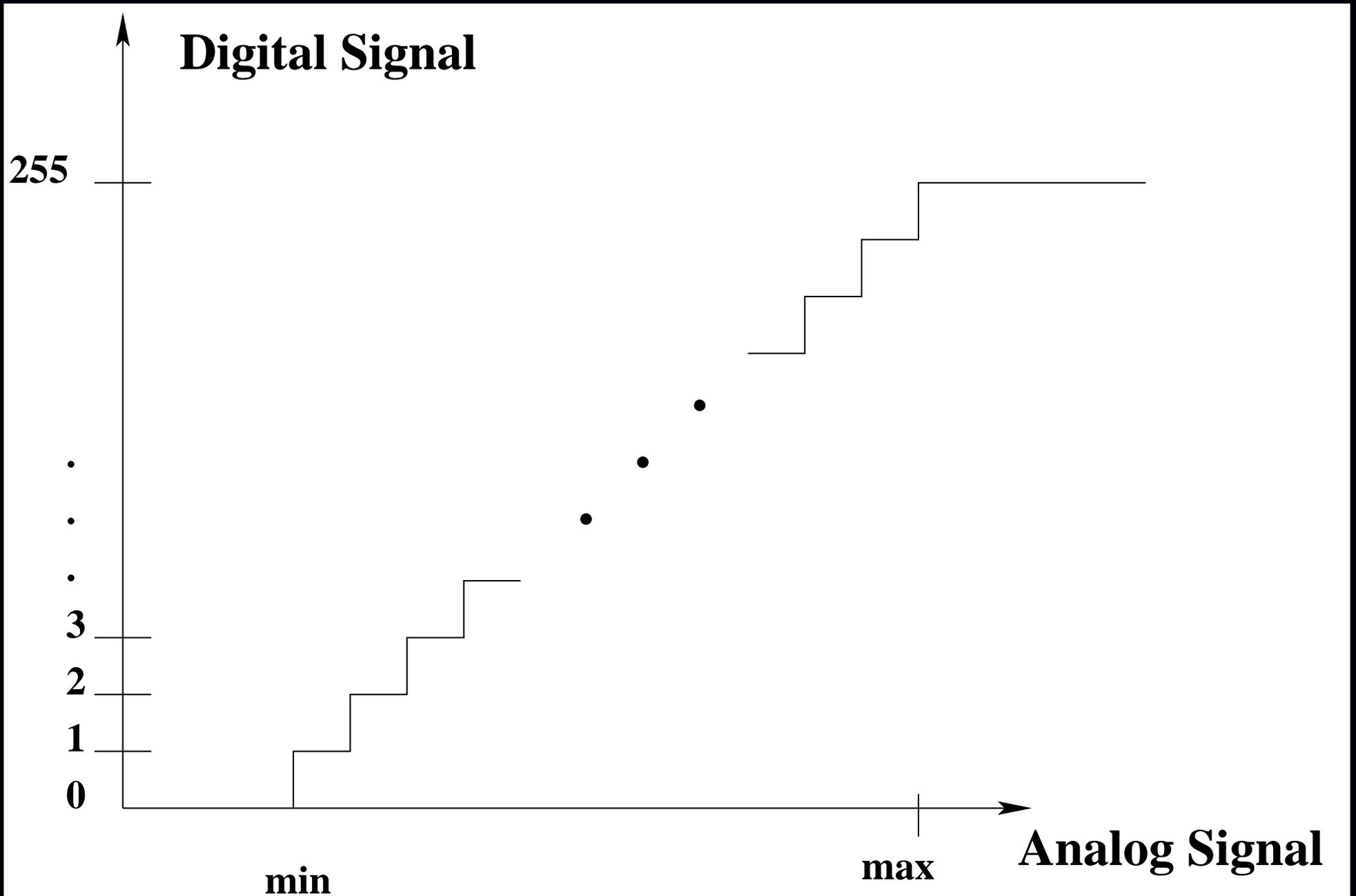
These 8-bit binary values represent:

$$2^7 + 2^4 + 2^3 + 2^0 = 128 + 16 + 8 + 1 = 153$$

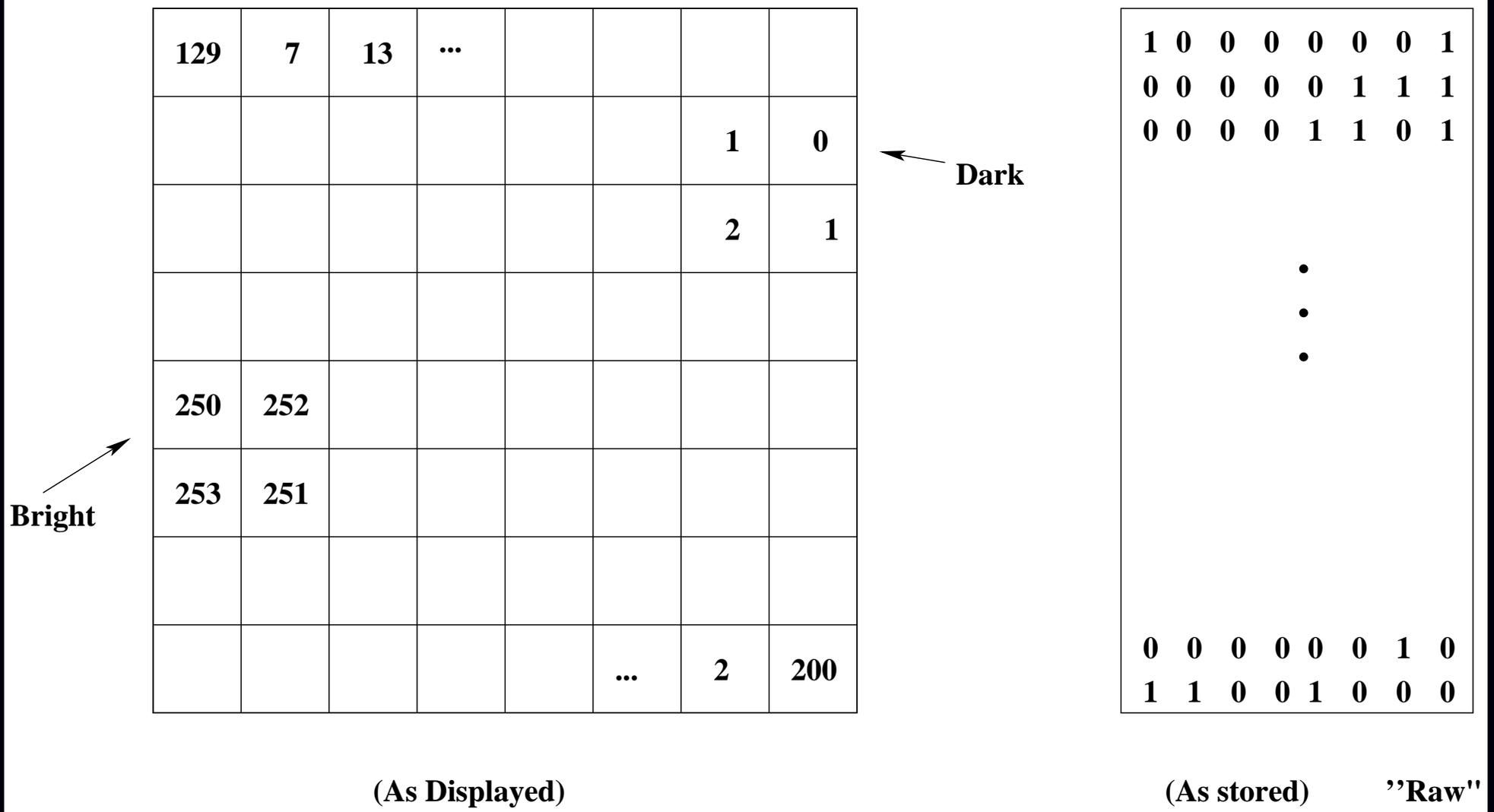
Smallest 8-bit digital value = 0.

Largest 8-bit digital value = ?

# Quantization



# Digital Gray scale (B/W) Image



$$\# \text{ bits stored} = \# \text{ pixels} \cdot \# \text{ bits per pixel (bpp)}$$

## Lots of Bits...

Example:  $3000 \times 3000$  pixel array (“9 Megapixel camera”)  
8-bit quantizer

$$\begin{aligned}\text{total \# of bits} &= 3000 \cdot 3000 \cdot 8 \\ &= 72 \cdot 10^6 \text{ bits} \\ &\approx 9 \text{ MB}\end{aligned}$$

9MB for a grayscale image is undesirably large.  
And for a RGB color image we would need  $3\times$  more.

# Image Compression aka Image Coding

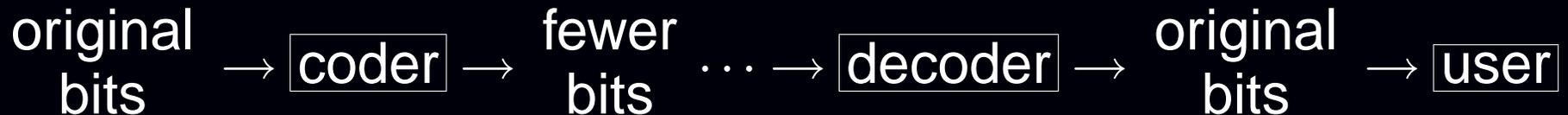
A subfield within the field of *data compression* aka *source coding*.  
Other data compression problems: audio compression (MP3), text compression (zip, ...)

Goal: reduce the # bits, while trying to preserve image quality.

- Lossless image coding: perfectly reversible
- Lossy image coding: recover an approximation

Why? For storage, transmission, ...

Concept:



The coder and decoder (“codec”) are designed together.

Examples: MP3, JPEG, MPEG4, AVI, ...

Q: How low can we go (how few bits per pixel?)

A: Claude Shannon’s information theory

# Basic Image Compression by Rounding Down

Suppose for each pixel we discard the two least significant bits (LSBs), *i.e.*, we set them to 0.

MSB 

0/1
-----

0/1
-----

0/1
-----

0/1
-----

0/1
-----

0/1
-----

0
---

0
---

 LSB

00000000 = 0

00000100 = 4

00001000 = 8

Possible gray scale values are multiples of 4:

00001100 = 12

00010000 = 16

⋮

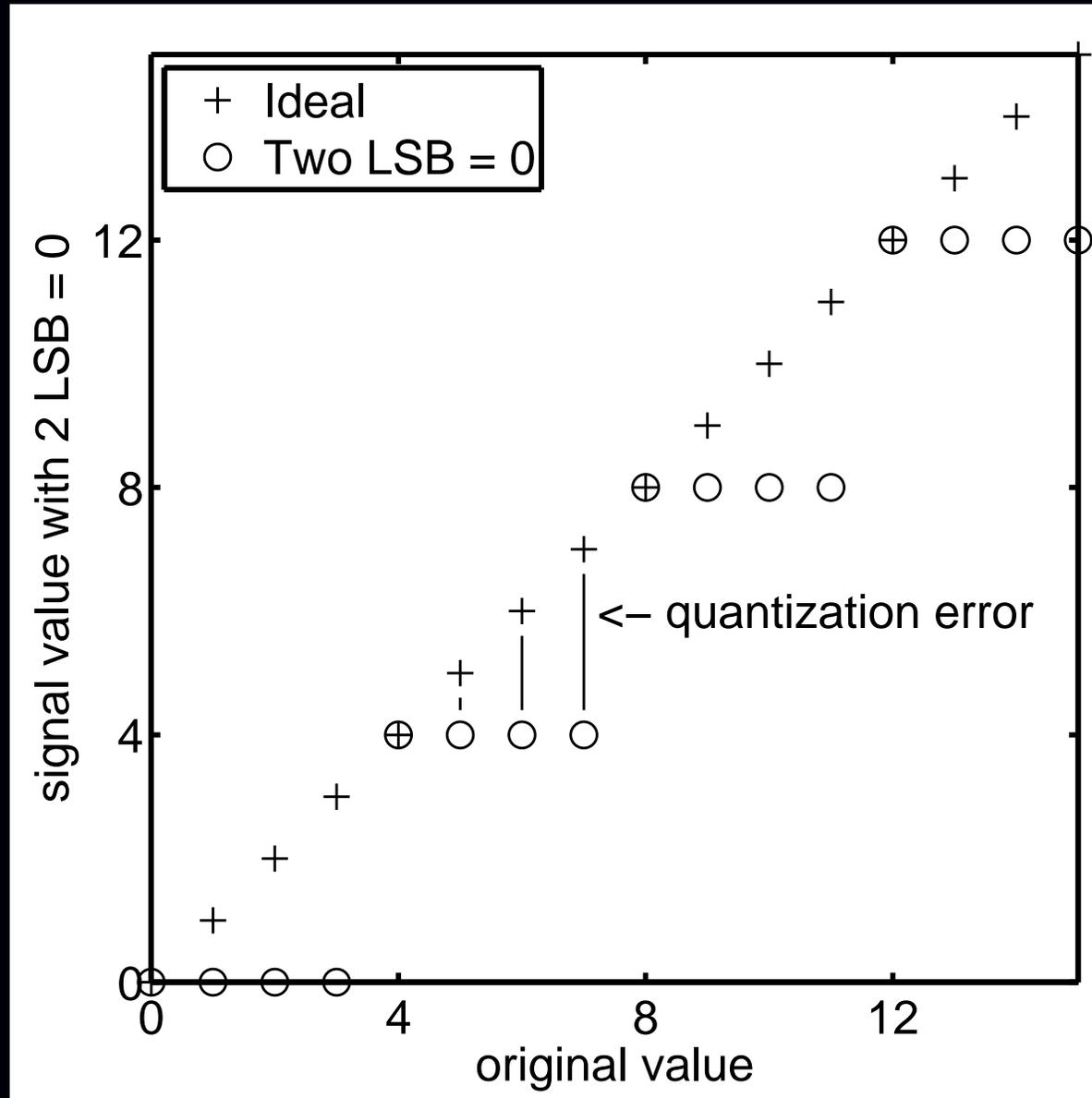
11111100 = 252

No need to store the two least-significant bits,  
so now only 6 bits per pixel are stored, instead of 8.  
25% data compression.

What happens to image quality?

# Quantization Error for Rounding Down

Setting the two LSBs to 0 reduces memory, but induces error:



# Average Quantization Error for Rounding Down

For discarding 2 bits:

$$\text{average error} = \frac{0 + 1 + 2 + 3}{4} = 1.5 \text{ gray levels}$$

For discarding  $d$  bits, where  $0 \leq d < 8$ :

$$\text{average error} = \frac{0 + 1 + 2 + \dots + 2^{d-1}}{2^d} = \frac{2^d - 1}{2} \text{ gray levels}$$

As we discard more bits, the error increases.  
Shannon called this the rate-distortion tradeoff.

Next we see what it *looks* like.

# Original Image

Original: using all 8 bits per pixel



255

0

Average error: 0.00 gray levels

# Compressed Image 1

Discarding 1 least significant bits



Average error: 0.50 gray levels

# Compressed Image 2

Discarding 2 least significant bits



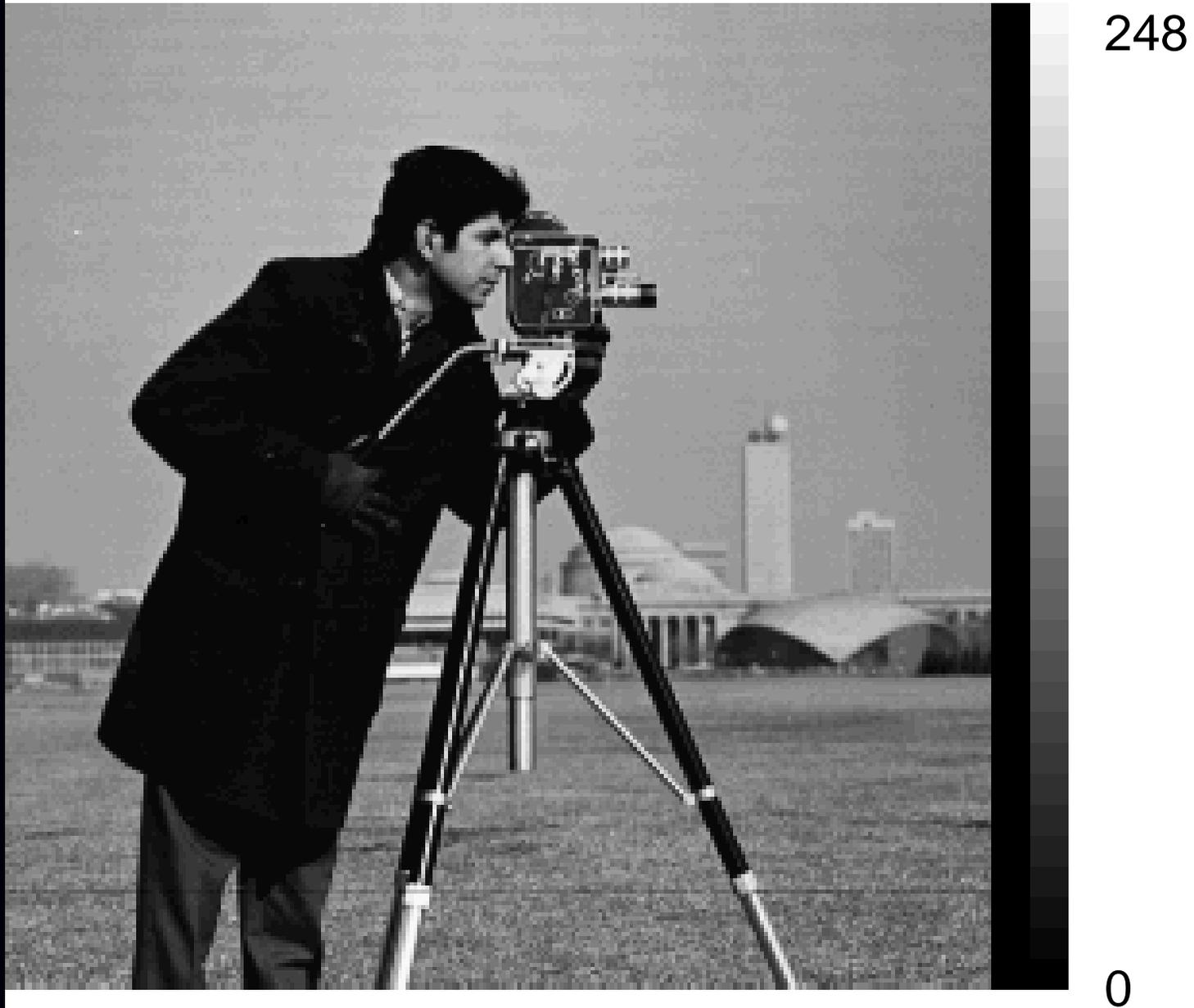
252

0

Average error: 1.49 gray levels

# Compressed Image 3

Discarding 3 least significant bits



Average error: 3.48 gray levels

# Compressed Image 4

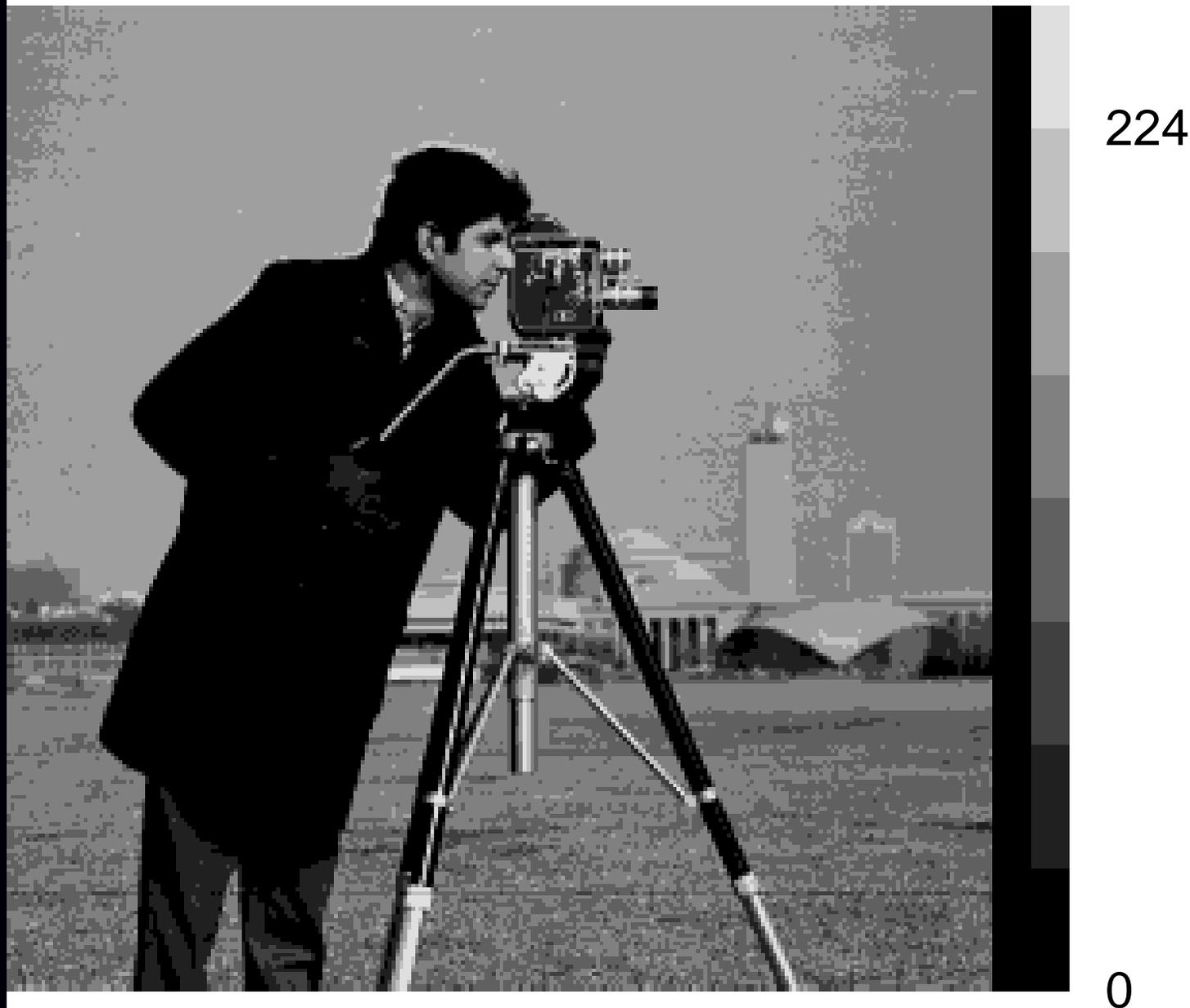
Discarding 4 least significant bits



Average error: 7.79 gray levels

# Compressed Image 5

Discarding 5 least significant bits



Average error: 14.61 gray levels

# Compressed Image 6

Discarding 6 least significant bits



Average error: 31.09 gray levels

# Compressed Image 7

Discarding 7 least significant bits

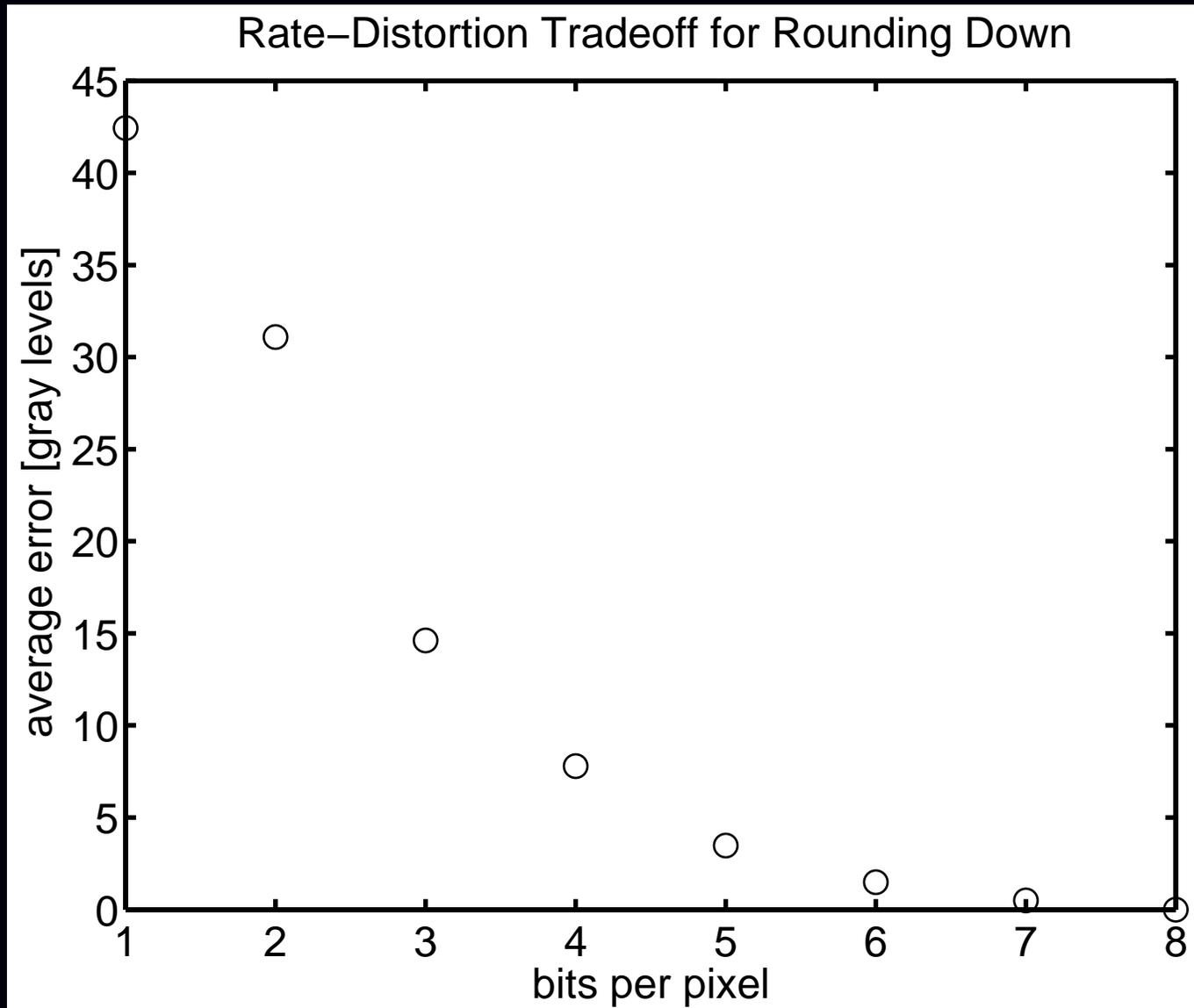


128

0

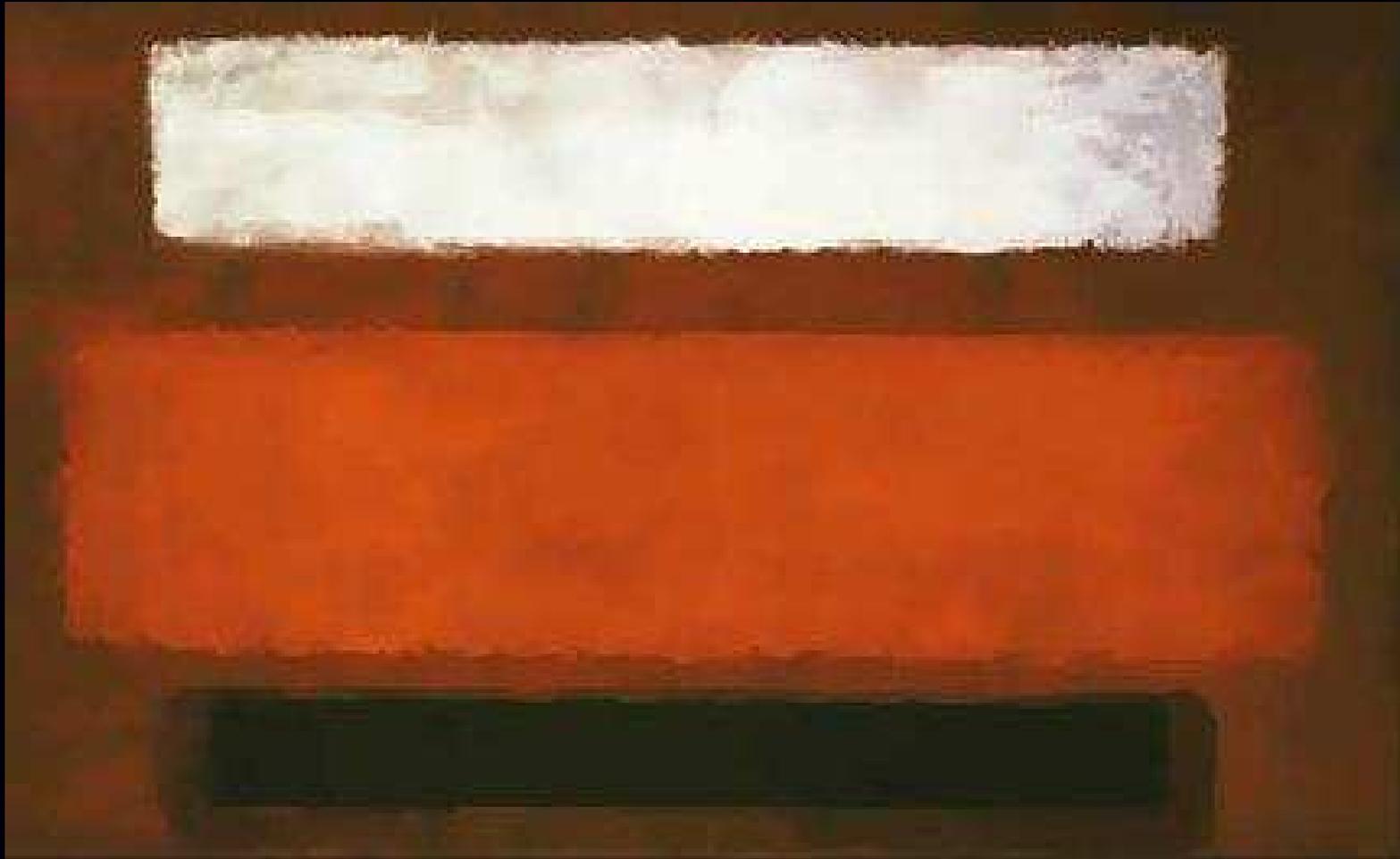
Average error: 42.44 gray levels

# Rate-Distortion Tradeoff for Rounding Down



Can we design a better image compression method?  
(What does better mean?)

# Mark Rothko's "White and Black on Wine"



# Jeff Fessler's "With Apologies to Rothko"



This image has only four distinct gray levels: 15, 120, 180, 230.  
How many bits per pixel are needed to encode it?

# Coding an image with a few distinct gray levels

Reasonable binary code:

value	code
15	00
120	01
180	10
230	11

With this code, only 2 bits per pixel are needed for this image.  
(Plus a few bits overhead to store the code table for the decoder.)

Can we do even better?

# Variable-Length Codes

So far we have been using **fixed-length codes**, where every value is represented by the same number of bits. (2 bits per value in preceding example.)

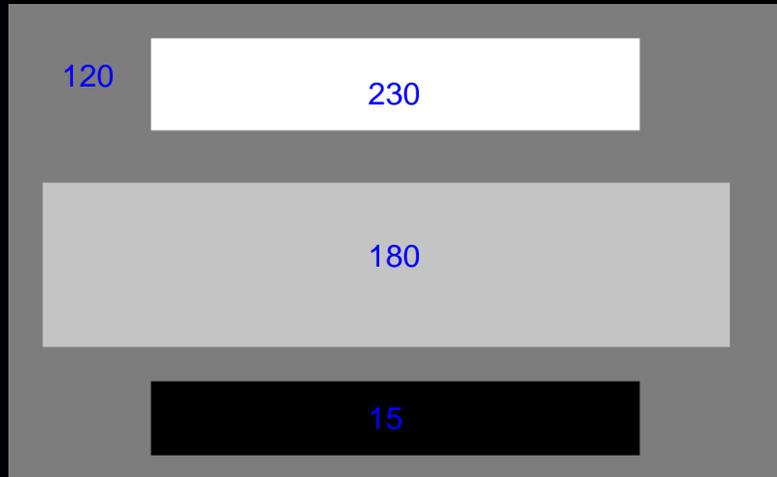
Consider (international) Morse Code (1840s):

A	. -	G	- - .	N	- .	U	. . -
B	- . . .	H	. . . .	O	- - -	V	. . . -
C	- . - .	I	. .	P	. - - .	W	. - -
D	- . .	J	. - - -	Q	- - . -	X	- . . -
E	.	K	- . -	R	. - .	Y	. - . -
F	. . - .	L	. - . .	S	. . .	Z	- - . .
		M	- -	T	-		

Why only a single “dot” for “E” ?

Idea of **variable-length codes**: use fewer bits for values that occur more frequently.

# Variable-Length Code: Example



Improved (variable-length) code:

value	proportion	code
15	9.8 %	0 1 1
120	47.5 %	1
180	30.5 %	0 0
230	12.2 %	0 1 0

How many bits per pixel *on average* are needed with this code?

$$0.098 \cdot 3 + 0.475 \cdot 1 + 0.305 \cdot 2 + 0.122 \cdot 3 = 1.745$$

Less than 2 bits per pixel!

*How is it stored?*

This is a **Huffman code** (see Matlab's `huffmandict` command).

Can we do even better?

# Shannon's Source Coding Theory

To encode numerous signal values that lie in a set with  $N$  elements with proportions (probabilities)  $p_1, p_2, \dots, p_N$ , on average we need *at least*  $H$  bits per value, where  $H$  is the **entropy**:

$$H = - \sum_{i=1}^N p_i \log_2 p_i.$$

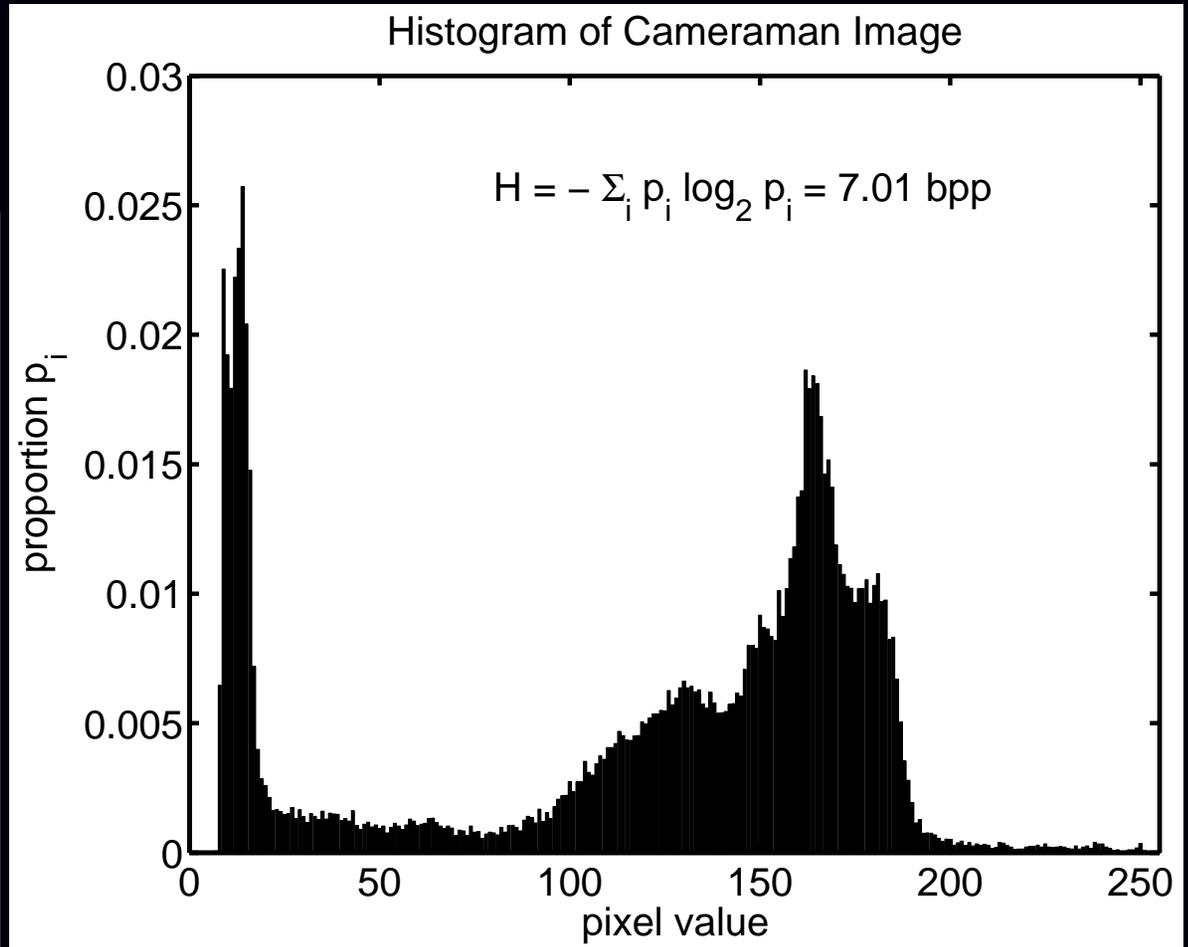
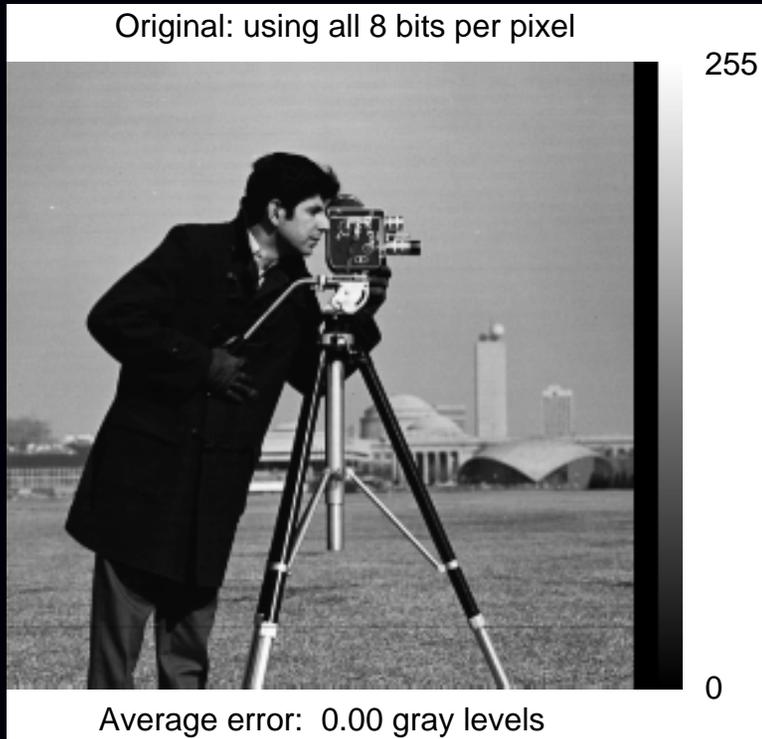
Example: for our image with  $N = 4$  gray levels, the entropy is:

$$\begin{aligned} H &= - (p_1 \log_2 p_1 + \dots + p_4 \log_2 p_4) \\ &= - (0.098 \log_2 0.098 + 0.475 \log_2 0.475 \\ &\quad + 0.305 \log_2 0.305 + 0.122 \log_2 0.122) \\ &\approx 1.7313 \text{ bits per pixel} \end{aligned}$$

Our Huffman code came remarkably close to this lower limit.

This type of “theoretical bound” is very important for *practical design*.

# More Complicated Images?



This image's pixel values also lie a finite set:  $0, 1, 2, \dots, 255$ .  
So Shannon's entropy bound applies.  
For this image, need at least 7.01 bits per pixel.  
Can we do better?

# Coding Pixel Relationships

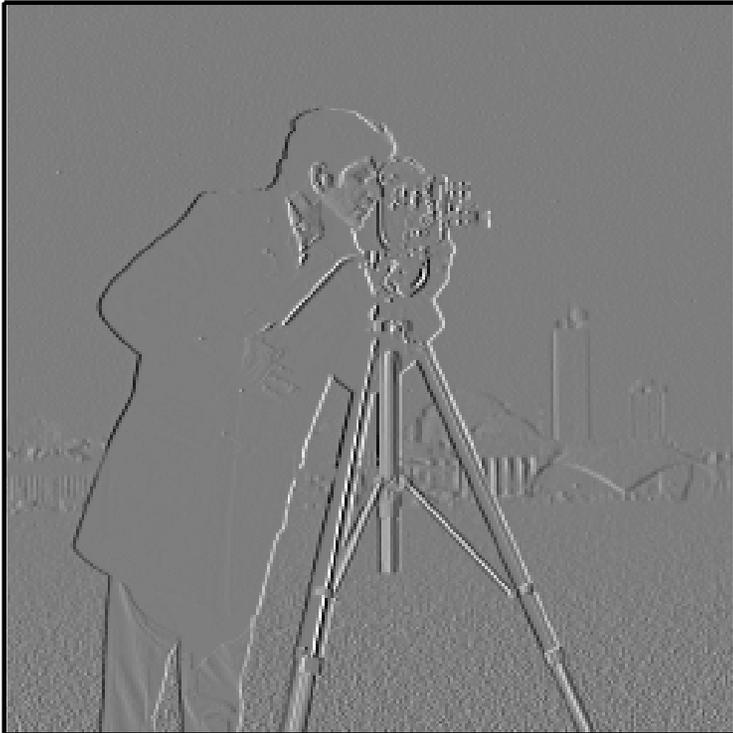
So far we have coded the *individual* pixel values directly. This does not give much data compression for most images.

For better image compression, we must consider the *relationships* between pixel values.

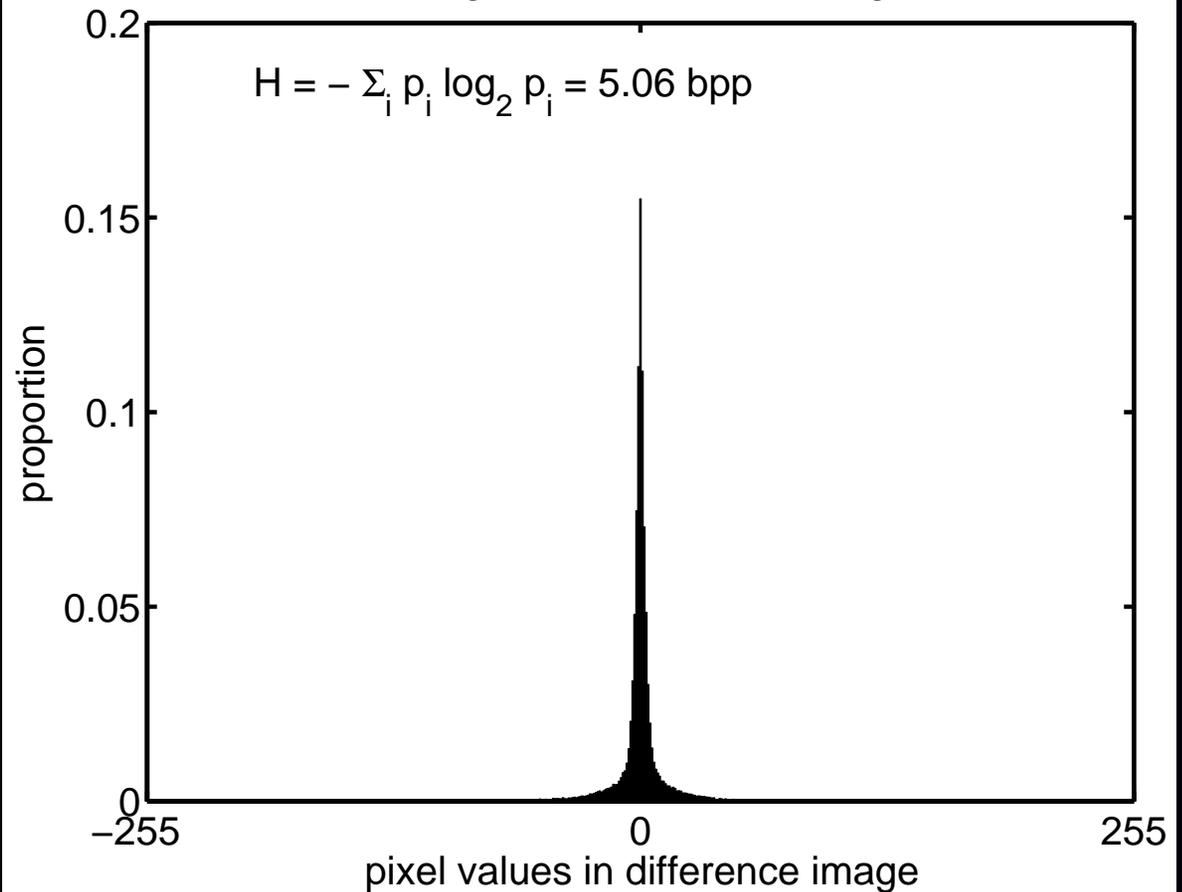
For example, neighboring pixel values tend to be similar.

# Coding Pixel Differences

Horizontal pixel differences



Histogram of Difference Image



The horizontal difference image has pixel values that lie in a finite set:  $-255, -254, \dots, 0, \dots, 255$ .

So Shannon's entropy bound applies.

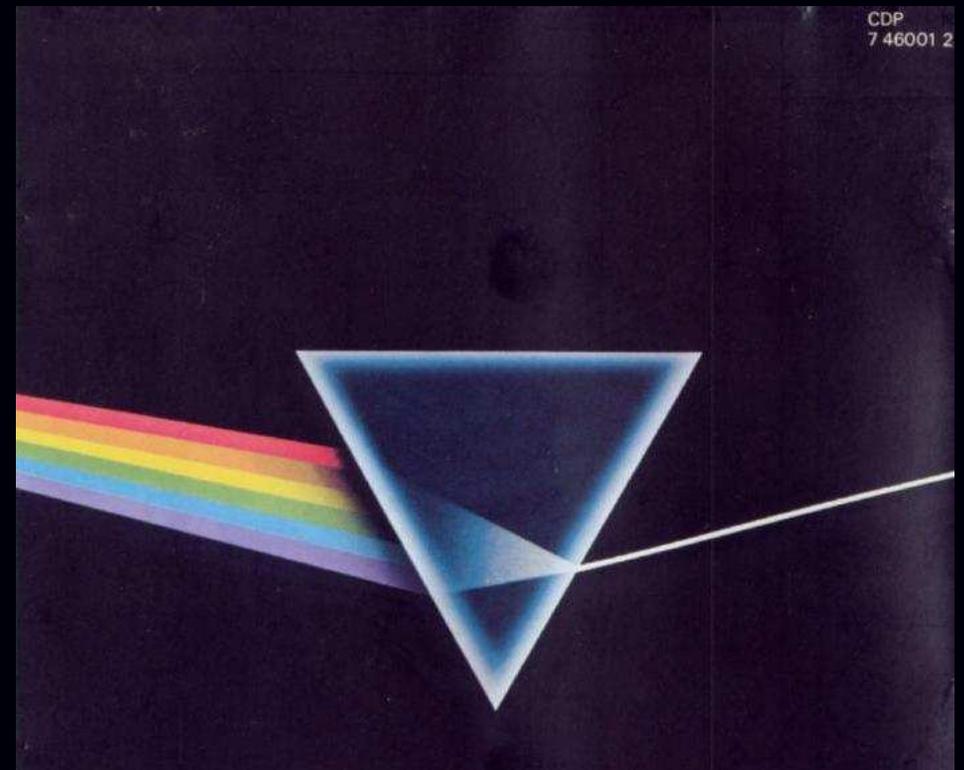
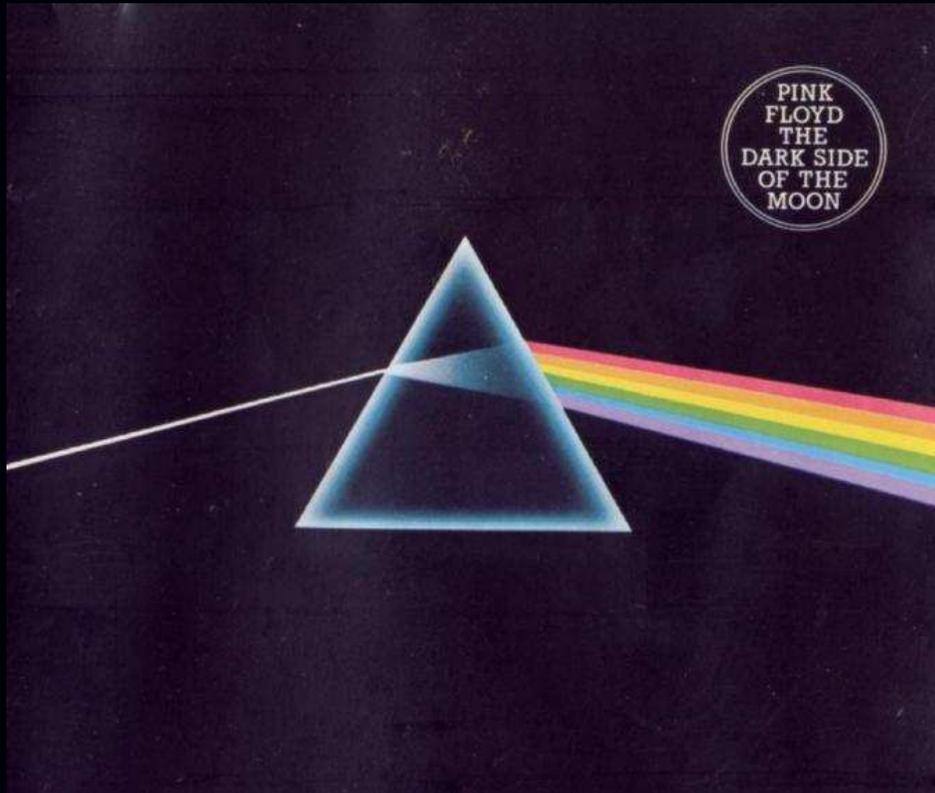
For this image, need at least 5.06 bits per pixel.

# Modern Transform-Based Image Coding

- Capture pixel relationships using:
  - **discrete Cosine transform** (JPEG)
  - **wavelet transform** (JPEG 2000)
- Substantial compression by discarding small DCT coefficients
- Lossy vs lossless compression
- For video coding,
  - use DCT within each frame
  - and differences between frames

# JPEG Image Compression and the DCT

DCT = discrete cosine transform  
relative of the discrete Fourier transform (DFT)  
practical thanks to the fast Fourier transform (FFT)  
digital equivalent of an optical prism



# Original Image

Original: using all 8 bits per pixel



# JPEG Compressed Image 1: 100%

JPEG with 0.685 bits per pixel



Average error: 0.62 gray levels

## JPEG Compressed Image 2: 95%

JPEG with 0.390 bits per pixel



Average error: 1.32 gray levels

# JPEG Compressed Image 3: 80%

JPEG with 0.185 bits per pixel



Average error: 2.85 gray levels

# JPEG Compressed Image 4: 25%

JPEG with 0.071 bits per pixel



Average error: 5.40 gray levels

# JPEG Compressed Image 5: 5%

JPEG with 0.030 bits per pixel



Average error: 10.01 gray levels

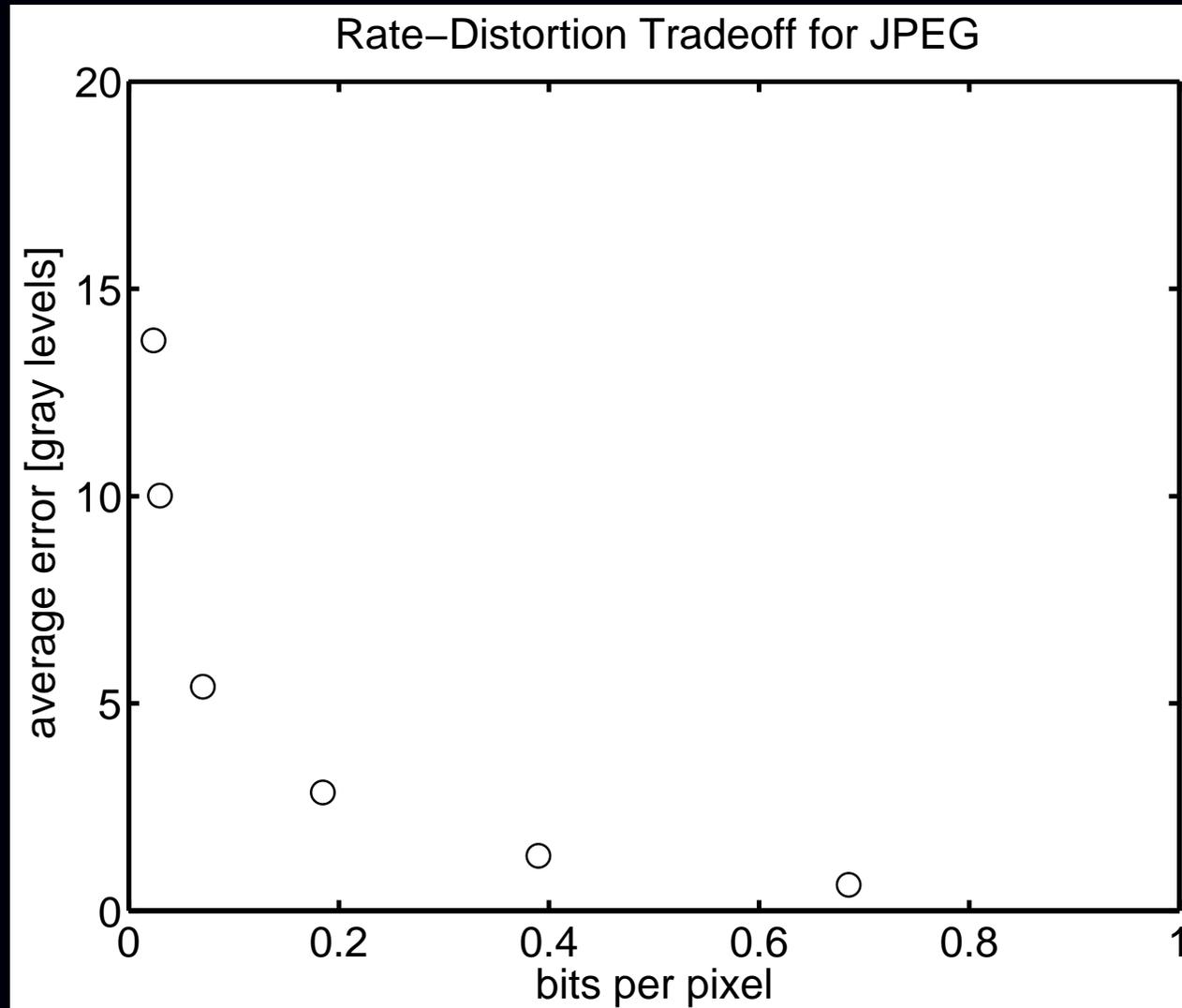
# JPEG Compressed Image 6: 1%

JPEG with 0.024 bits per pixel



Average error: 13.76 gray levels

# Rate-Distortion Tradeoff for JPEG



Recall that for compression by rounding,  
at 1 bbp the average error was 40 gray levels.

# Summary

- EEs are responsible for the principles that underly modern digital audio and video
  - both the devices / technologies (applied physics)
  - and the algorithms that they execute (applied math)
- Data compression is a very active research area
- Other considerations in data compression
  - complexity
  - hardware implementations
  - color
  - progressive encoding
  - ...
- To learn more about signals / systems / sampling / quantization:
  - EECS 216, 401, 451, 452, 455 (design / implement)
  - EECS 550, 551, 556, 650, 651, 750 (invent next generation)