Image Processing

Burcu KIR SAVAŞ, PhD.

Point Operations
Histogram Processing

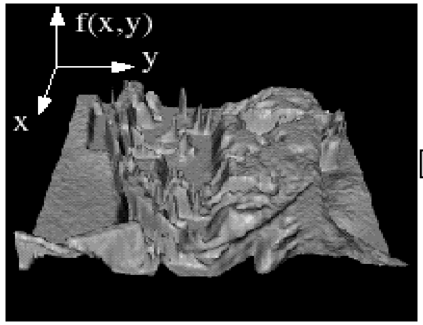
Today's topics

- Point operations
- Histogram processing

Digital images

- <u>Sample</u> the 2D space on a regular grid
- <u>Quantize</u> each sample (round to nearest integer)

- Image represented as a matrix of integer values.

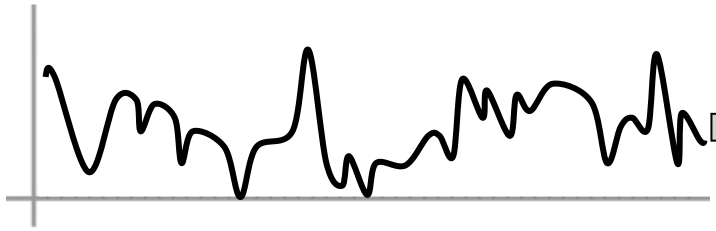


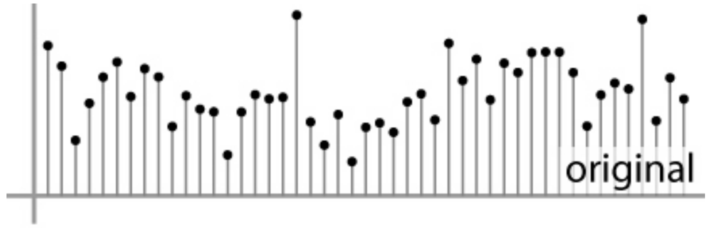
$j \longrightarrow$

$i \downarrow$

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

2D





1D

Slide credit: K. Grauman, S. Seitz

Image Transformations

- $g(x,y)=T[f(x,y)]$

$g(x,y)$: output image

$f(x,y)$: input image

T: transformation function

1. Point operations: operations on single pixels
2. Spatial filtering: operations considering pixel neighborhoods
3. Global methods: operations considering whole image

Point Operations

- Smallest possible neighborhood is of size 1x1
- Process each point independently of the others
- Output image *g* depends only on the value of *f* at a single point (x,y)
- Map each pixel's value to a new value
- Transformation function *T* remaps the sample's value:

$$s = T(r)$$

where

- *r* is the value at the point in question
- *s* is the new value in the processed result
- *T* is a *intensity transformation* function

Point operations

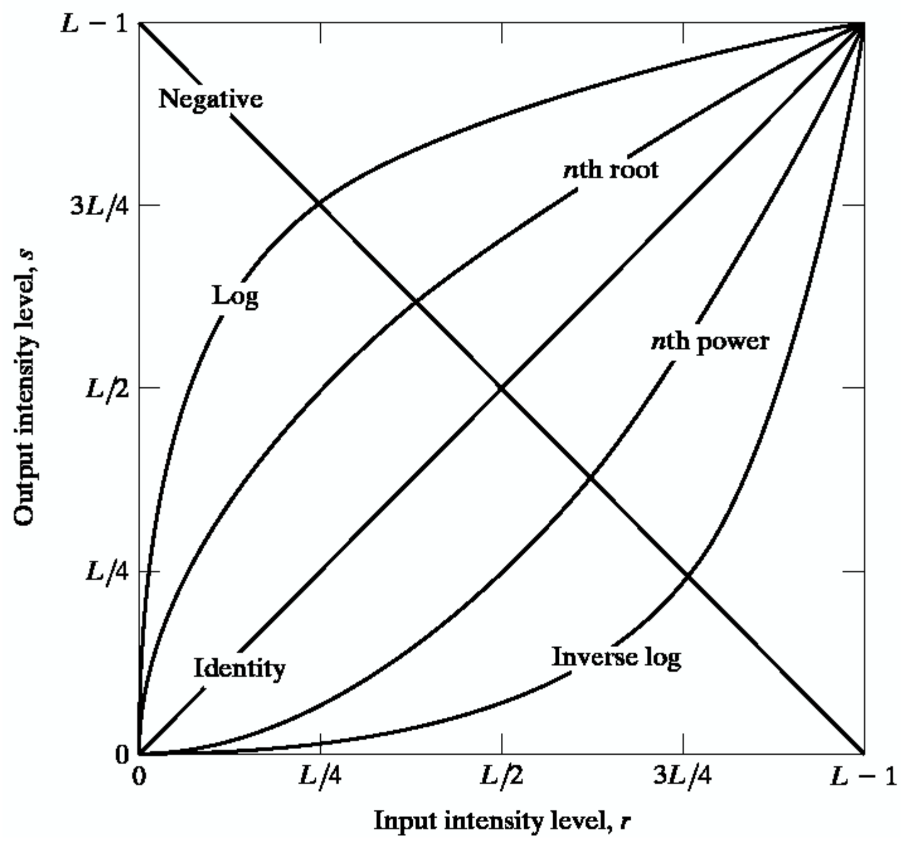
- Is mapping one color space to another (e.g. RGB2HSV) a point operation?
- Is image arithmetic a point operation?
- Is performing geometric transformations a point operation?
  - Rotation
  - Translation
  - Scale change
  - etc.

Sample intensity transformation functions

- Linear function

  Image negatives and identity transformations

- Logarithm function
  - Log and inverse-log transformation
  - Compresses the dynamic range of images
- Power-law function
  - Gamma correction
  - nth power and nth root transformations



Sample intensity transformation functions

- **Linear function**
  Linear transformations include identity transformation and negative transformation.
- In identity transformation, the input image is the same as the output image.
  s = r
- The negative transformation is:
- s = L - 1 - r = 256 - 1 - r = 255 - r
- L = Largest gray level in an image. The negative transformation is suited for enhancing white or gray detail embedded in dark areas of an image, for example, analyzing the breast tissue in a digital image.

Sample intensity transformation functions

- **Logarithmic**

  a) General Log Transform

- The equation of general log transformation is: s = c * log(1 + r)
- s,r: denote the gray level of the input pixel and the output pixel.
- ‘c’ is a constant; to map from [0,255] to [0,255], c = 255/LOG(256)
- the base of a common logarithm is 10

In the log transformation, the low-intensity values are mapped into higher intensity values. It maps a narrow range of low gray levels to a much wider range. Generally speaking, the log transformation works the best for dark images.

Sample intensity transformation functions

- **Inverse Log Transform**
  The inverse log transform is opposite to log transform. It maps a narrow range of high gray levels to a much wider range. The inverse log transform expands the values of light-level pixels while compressing the darker-level values.
- s = power(10, r * c)-1
- s,r: denote the gray level of the input pixel and the output pixel.
- 'c' is a constant; to map from [0,255] to [0,255], c =LOG(256)/255

Sample intensity transformation functions

Power-Law (Gamma) Transformation

The gamma transform, also known as exponential transformation or power transformation, is a commonly used grayscale non-linear transformation. The mathematical expression of the gamma transformation is as follows:

s = c * power(r, γ), where

s,r: denote the gray level of the input pixel and the output pixel;

'c' is a constant

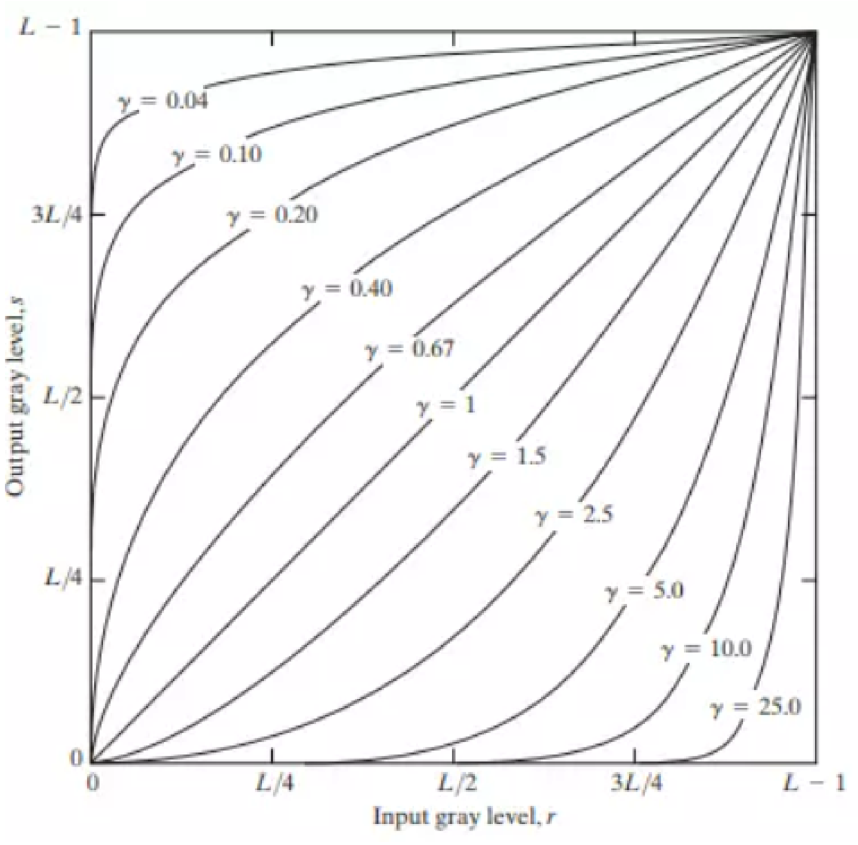
'γ' is also a constant and is the gamma coefficient.

Plots of the equation [formula] for various values of γ (c = 1 in all cases)

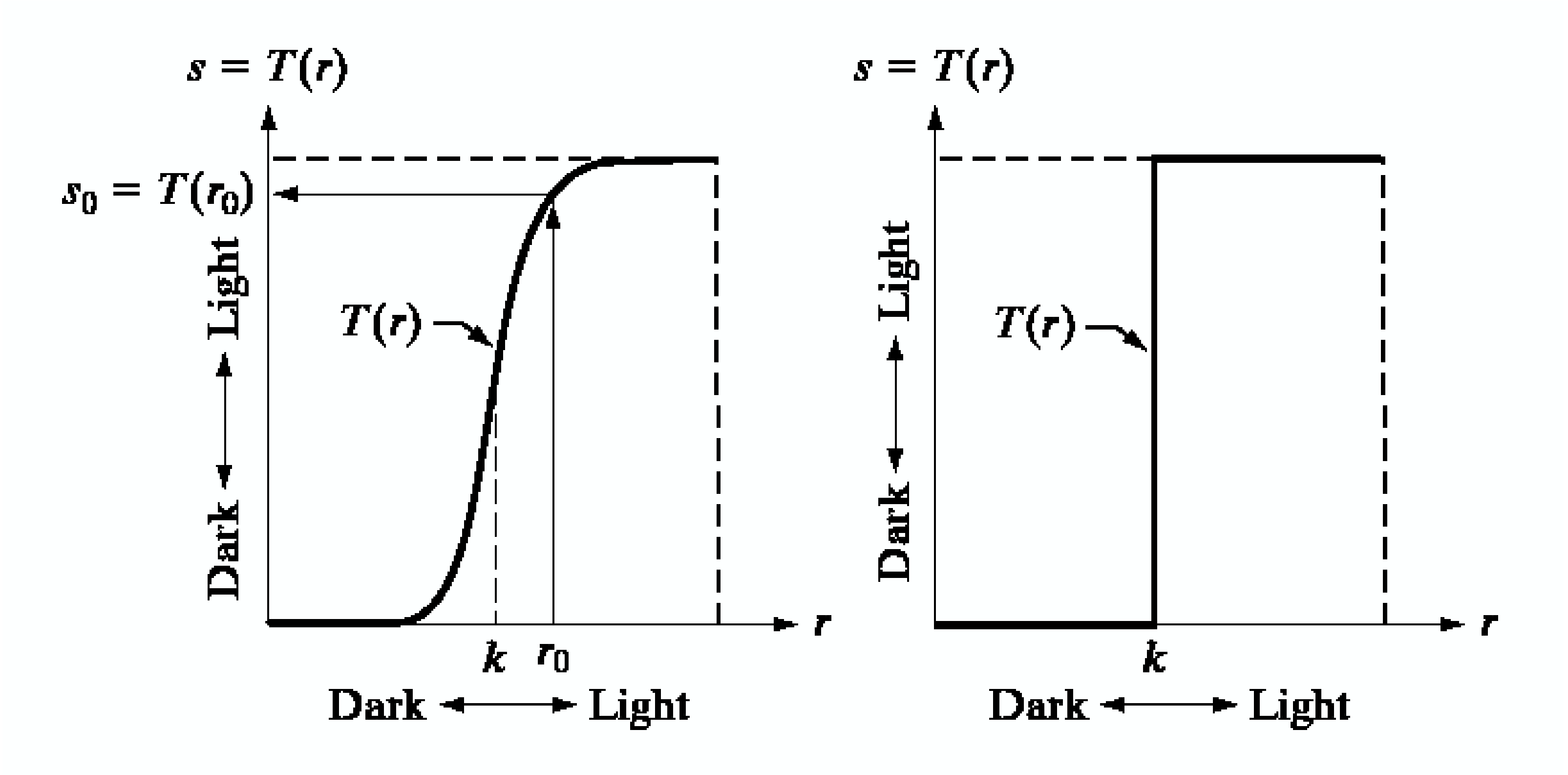
Sample intensity transformation functions

Power-Law (Gamma) Transformation

All curves were scaled to fit in the range shown. X-axis represents input intensity level ‘r’, and y-axis represents the output intensity level ‘s’.



Point Processing Examples



- produces an image of higher contrast than the original

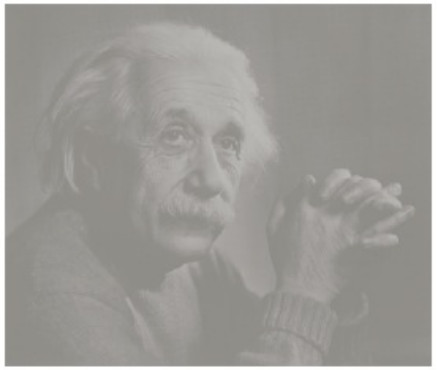
darkening the intensity levels below k n the original image

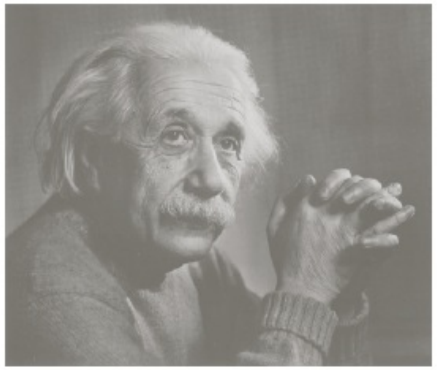
brightening intensities above k in the original image

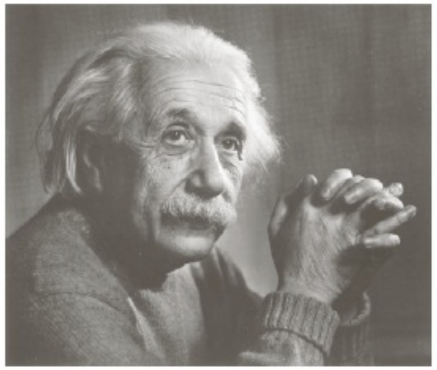
produces a binary (two-intensity level) image

Dynamic range

- Dynamic range $R_d = I_{\mathrm{max}} \; / \; I_{\mathrm{min}}$, or $(I_{\mathrm{max}} + k) \; / \; (I_{\mathrm{min}} + k)$
  - determines the degree of image contrast that can be achieved
  - a major factor in image quality
- Ballpark values
  - Desktop display in typical conditions: 20:1
  - Photographic print: 30:1
  - High dynamic range display: 10,000:1


low contrast

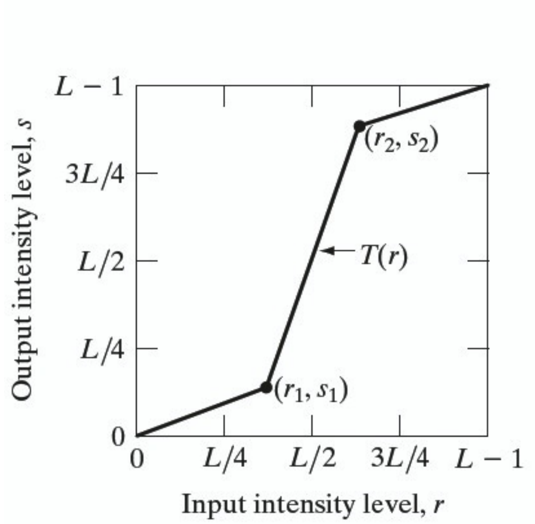

medium contrast

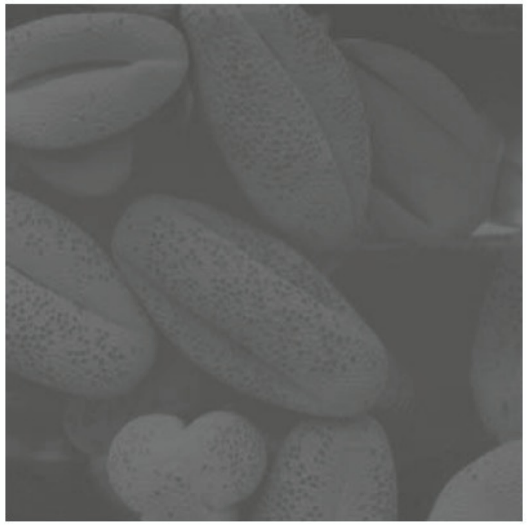
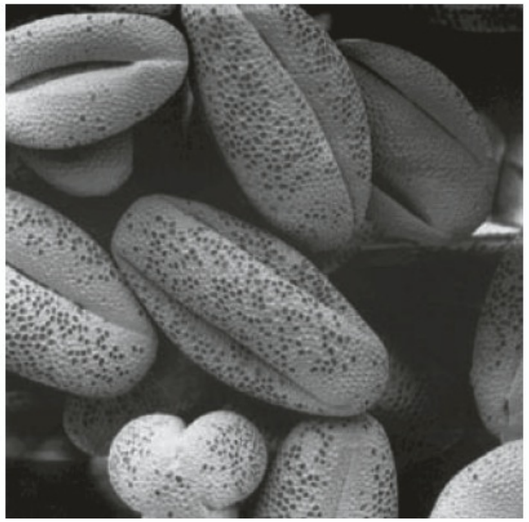
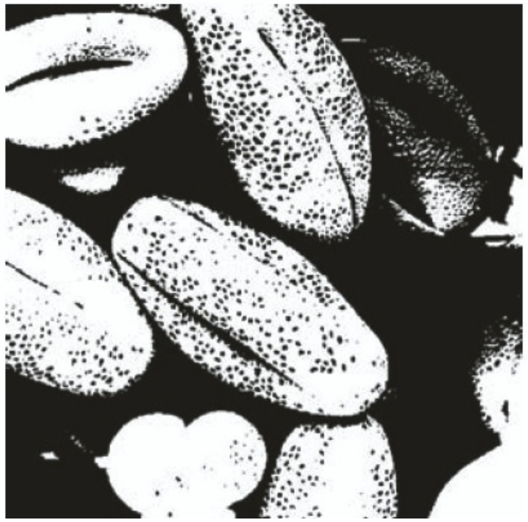

high contrast

Slide credit: S. Marschner

Point Operations: Contrast stretching and Thresholding

- Contrast stretching: produces an image of higher contrast than the original
- Thresholding: produces a binary (two-intensity level) image



Point Operations: Intensity-level Slicing

- highlights a certain range of intensities

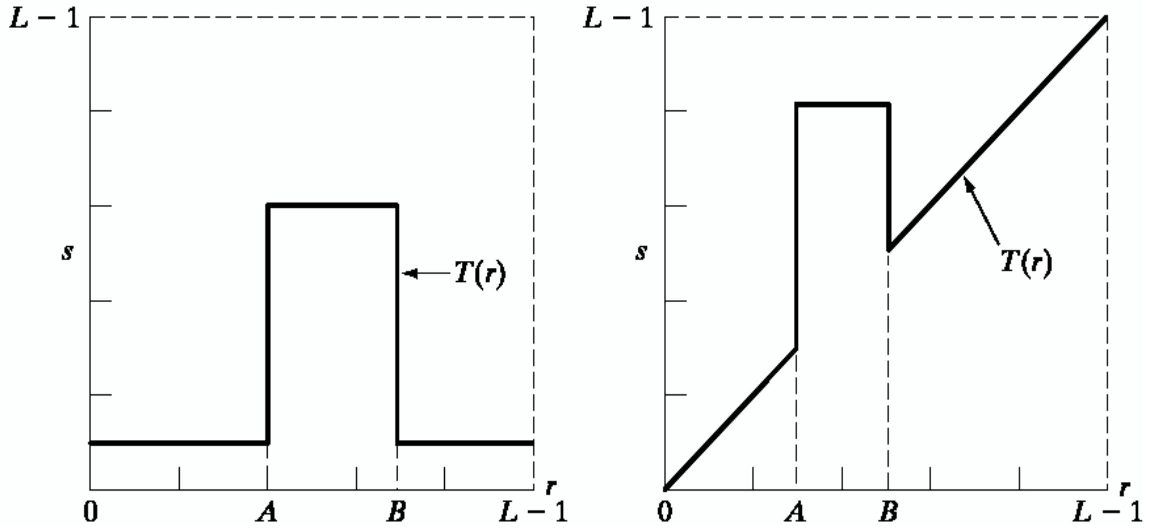
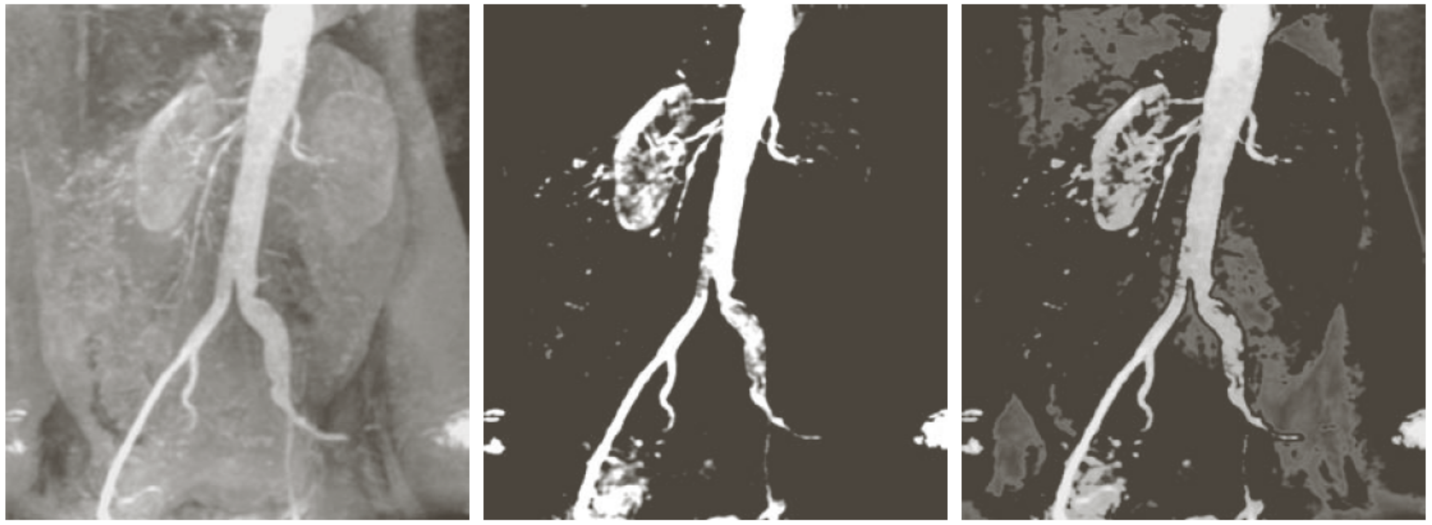


FIGURE 3.11 (a) This transformation highlights intensity range $[A, B]$ and reduces all other intensities to a lower level. (b) This transformation highlights range $[A, B]$ and preserves all other intensity levels.



Intensity encoding in images

- Recall that the pixel values determine how bright that pixel is.
- Bigger numbers are (usually) brighter
- *Transfer function*: function that maps input pixel value to luminance of displayed image

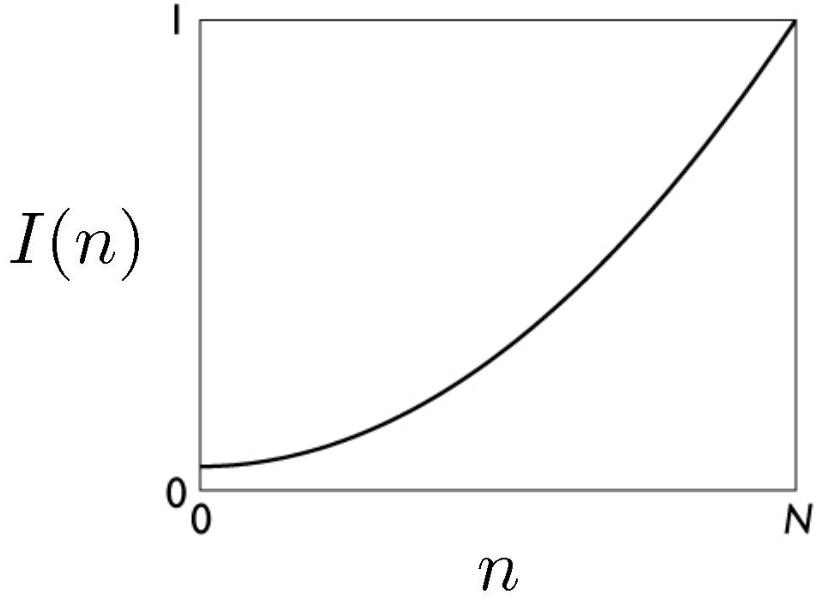
$$I = f(n) \quad f : [0, N] \to [I_{\min}, I_{\max}]$$

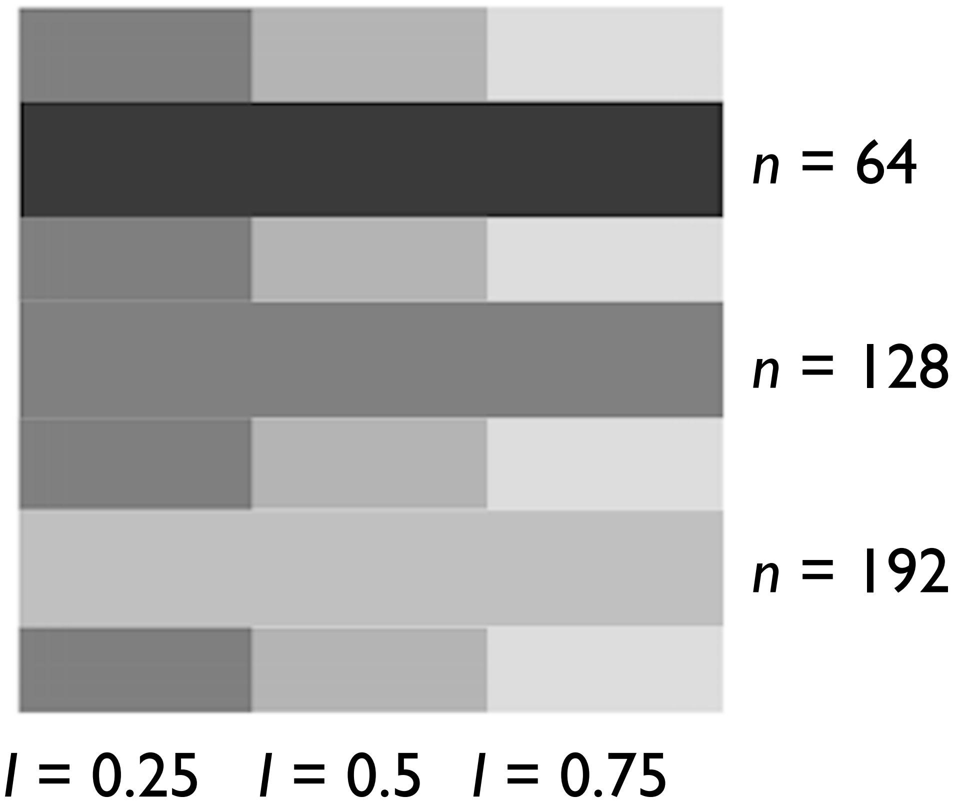
- What determines this function?
  - physical constraints of device or medium
  - desired visual characteristics

adapted from: S. Marschner

What this projector does?

- Something like this:





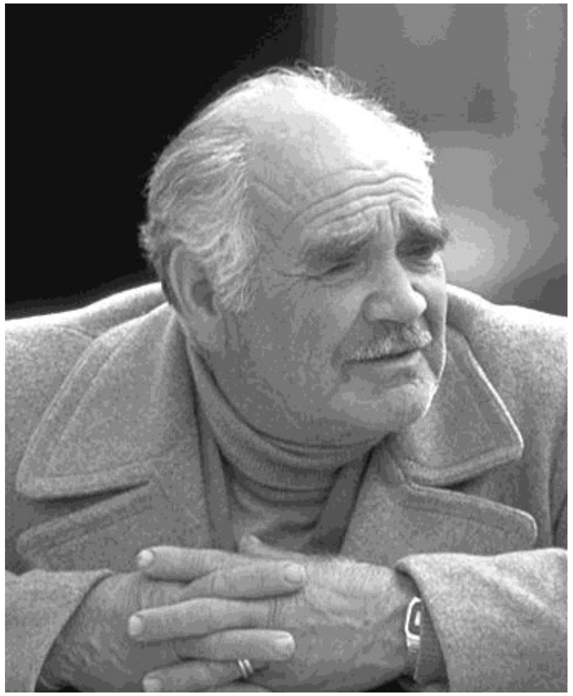
adapted from: S. Marschner

Constraints on transfer function

- Maximum displayable intensity, I_{max}
  - how much power can be channeled into a pixel?
    - LCD: backlight intensity, transmission efficiency (<10%)
    - projector: lamp power, efficiency of imager and optics
- Minimum displayable intensity, I_{min}
  - light emitted by the display in its "off" state
    - e.g. stray electron flux in CRT(A cathode-ray tube (CRT)), polarizer quality in LCD
- Viewing flare, *k*: light reflected by the display
  - very important factor determining image contrast in practice
    - 5% of I_{max} is typical in a normal office environment [sRGB spec]
    - much effort to make very black CRT and LCD screens
    - all-black decor in movie theaters

Transfer function shape

- Desirable property: the change from one pixel value to the next highest pixel value should not produce a visible contrast
  - otherwise smooth areas of images will show visible bands
- What contrasts are visible?
  - rule of thumb: under good conditions we can notice a 2% change in intensity
  - therefore we generally need smaller quantization steps in the darker tones than in the lighter tones
  - most efficient quantization is logarithmic


an image with severe *banding*

[Philip Greenspun]

Slide credit: S. Marschner

How many levels are needed?

- Depends on dynamic range
  - 2% steps are most efficient:

$$0 \mapsto I_{\min}; 1 \mapsto 1.02 I_{\min}; 2 \mapsto (1.02)^2 I_{\min}; \ldots$$

  - log 1.02 is about 1/120, so 120 steps per decade of dynamic range
    - 240 for desktop display
    - 360 to print to film
    - 480 to drive HDR display
- If we want to use linear quantization (equal steps)
  - one step must be < 2% (1/50) of $I_{\min}$
  - need to get from ~0 to $I_{\min} \cdot R_d$ so need about 50 R_d levels
    - 1500 for a print; 5000 for desktop display; 500,000 for HDR display
- Moral: 8 bits is just barely enough for low-end applications
  - but only if we are careful about quantization

Slide credit: S. Marschner

Intensity quantization in practice

- Option 1: linear quantization $I(n) = (n/N)\, I_{\max}$
  - pro: simple, convenient, amenable to arithmetic
  - con: requires more steps (wastes memory)
  - need 12 bits for any useful purpose; more than 16 for HDR
- Option 2: power-law quantization $I(n) = (n/N)^{\gamma}\, I_{\max}$
  - pro: fairly simple, approximates ideal exponential quantization
  - con: need to linearize before doing pixel arithmetic
  - con: need to agree on exponent
  - 8 bits are OK for many applications; 12 for more critical ones
- Option 2: floating-point quantization $I(x) = (x/w)\, I_{\max}$
  - pro: close to exponential; no parameters; amenable to arithmetic
  - con: definitely takes more than 8 bits
  - 16–bit “half precision” format is becoming popular

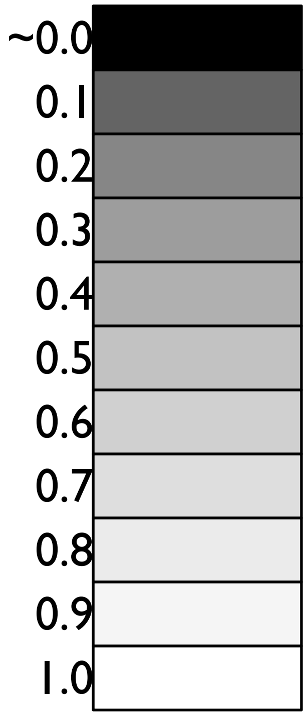
Slide credit: S. Marschner

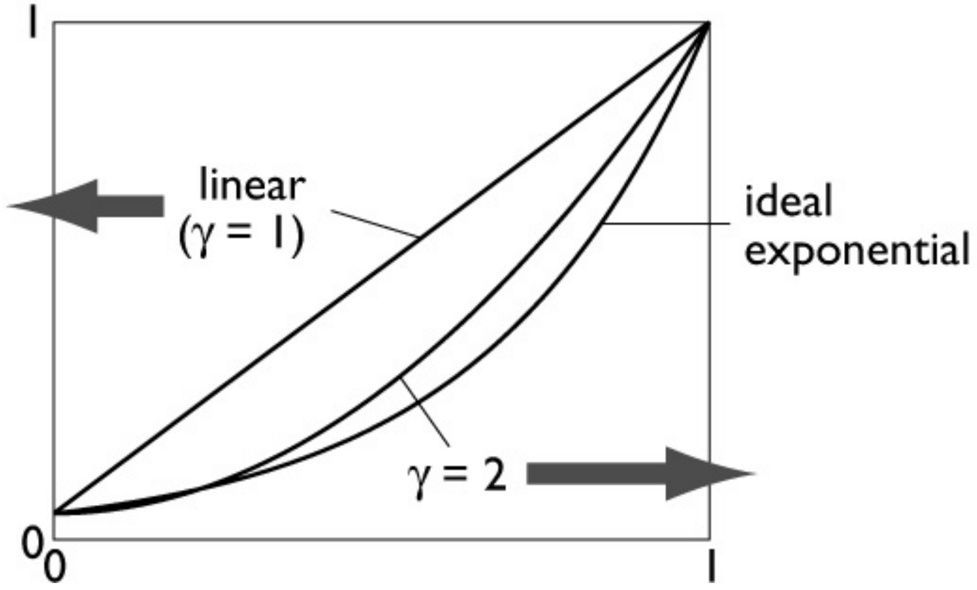
Why gamma?

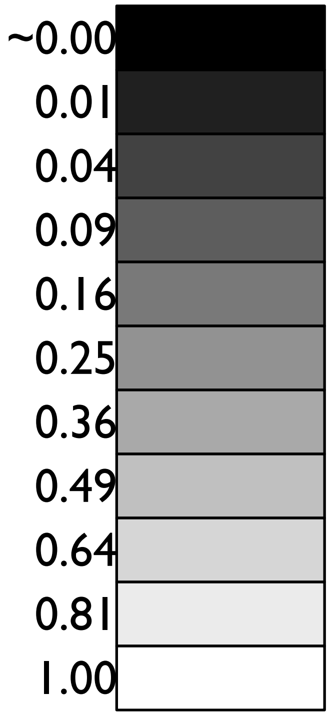
- Power-law quantization, or *gamma correction* is most popular
- Original reason: CRT
  - intensity on screen is proportional to (roughly) voltage^2
- Continuing reason: inertia + memory savings
  - inertia: gamma correction is close enough to logarithmic that there's no sense in changing
  - memory: gamma correction makes 8 bits per pixel an acceptable option

Slide credit: S. Marschner

Gamma quantization







- Close enough to ideal perceptually uniform exponential

Slide credit: S. Marschner

Gamma correction

- Sometimes (often, in graphics) we have computed intensities *a* that we want to display linearly
- In the case of an ideal monitor with zero black level,

$$I(n) = (n/N)^\gamma$$

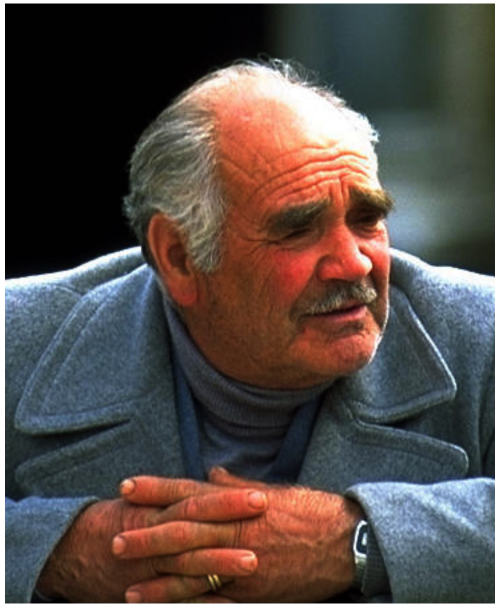
(where N = 2^n – 1 in *n* bits). Solving for *n*:

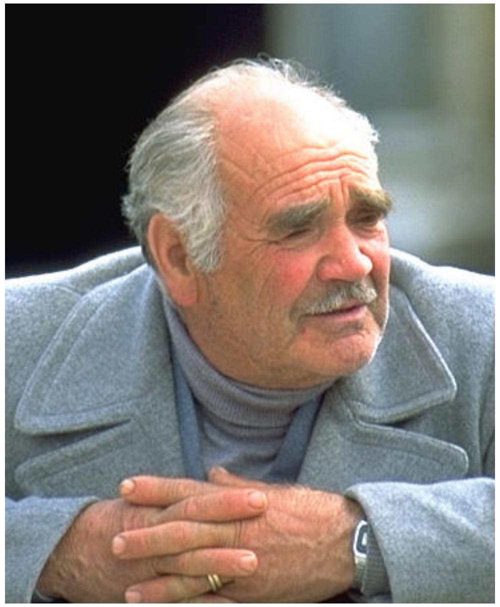
$$n = Na^{\frac{1}{\gamma}}$$

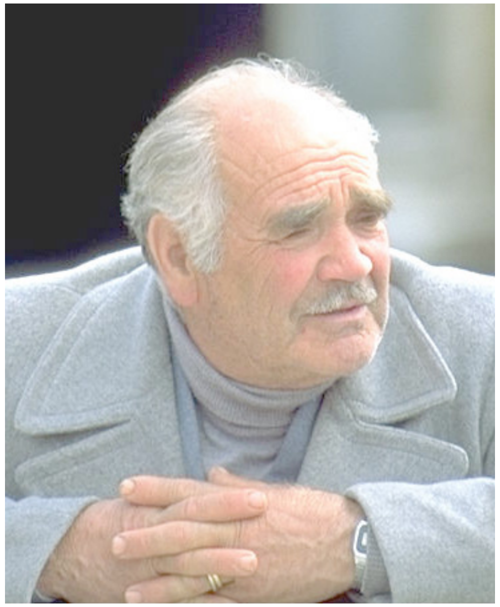
- This is the "gamma correction" recipe that has to be applied when computed values are converted to 8 bits for output
  - failing to do this (implicitly assuming gamma = 1) results in dark, oversaturated images

Slide credit: S. Marschner

Gamma correction


corrected for γ lower than display


OK

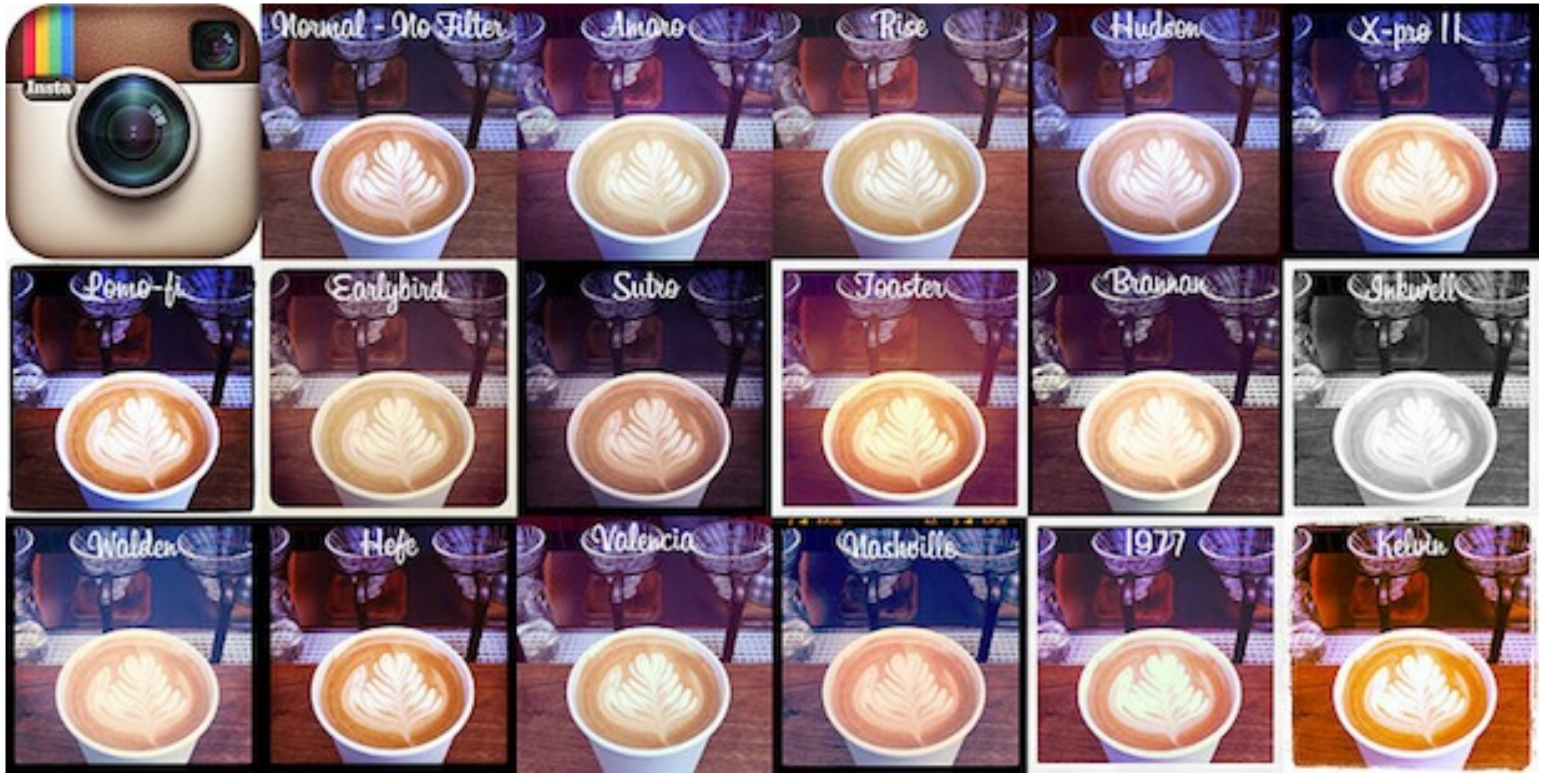

corrected for γ higher than display

[Philip Greenspun]

Slide credit: S. Marschner

Instagram Filters

- How do they make those Instagram filters?

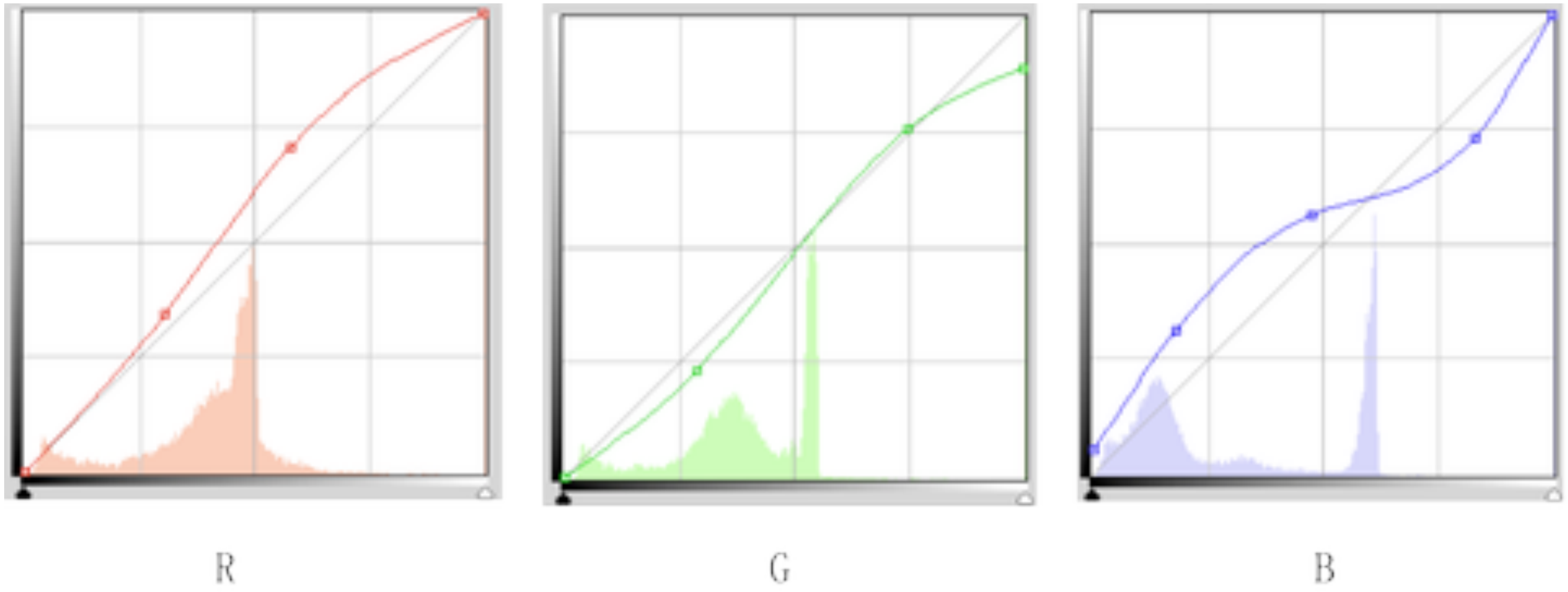


"It's really a combination of a bunch of different methods. In some cases we draw on top of images, in others we do pixel math. It really depends on the effect we're going for." --- Kevin Systrom, co-founder of Instagram

Source: C. Dyer

Example Instagram Steps

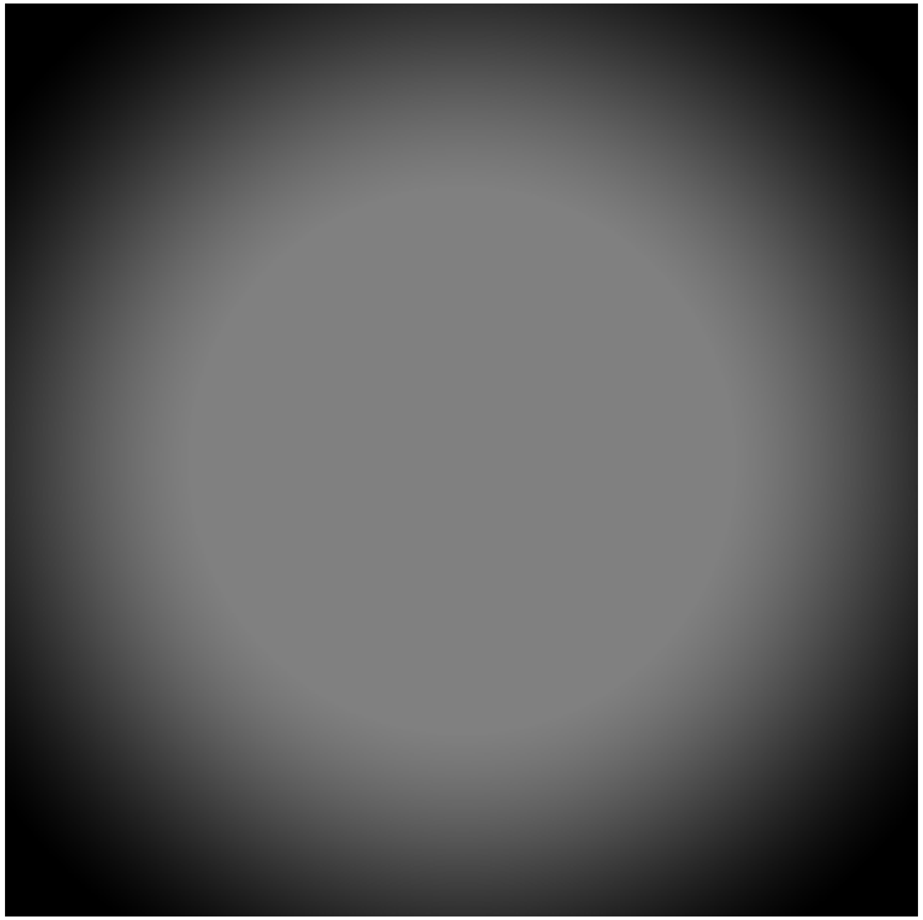
1. Perform an independent RGB color point transformation on the original image to increase contrast or make a color cast



Source: C. Dyer

Example Instagram Steps

2. Overlay a circle background image to create a vignette effect



Source: C. Dyer

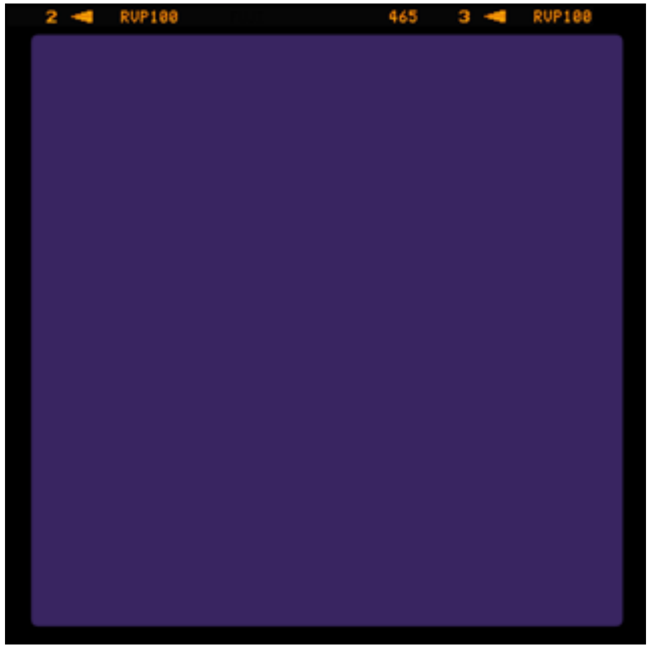
Example Instagram Steps

3. Overlay a background image as decorative grain

Source: C. Dyer

Example Instagram Steps

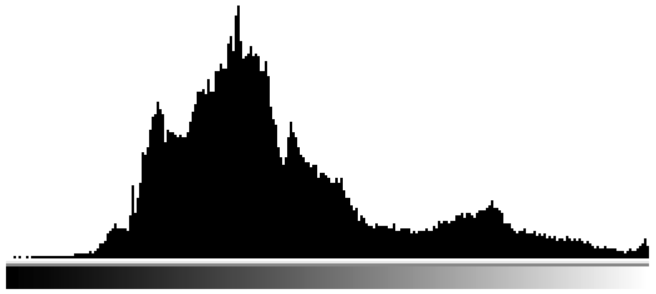
4. Add a border or frame



Source: C. Dyer

Histogram

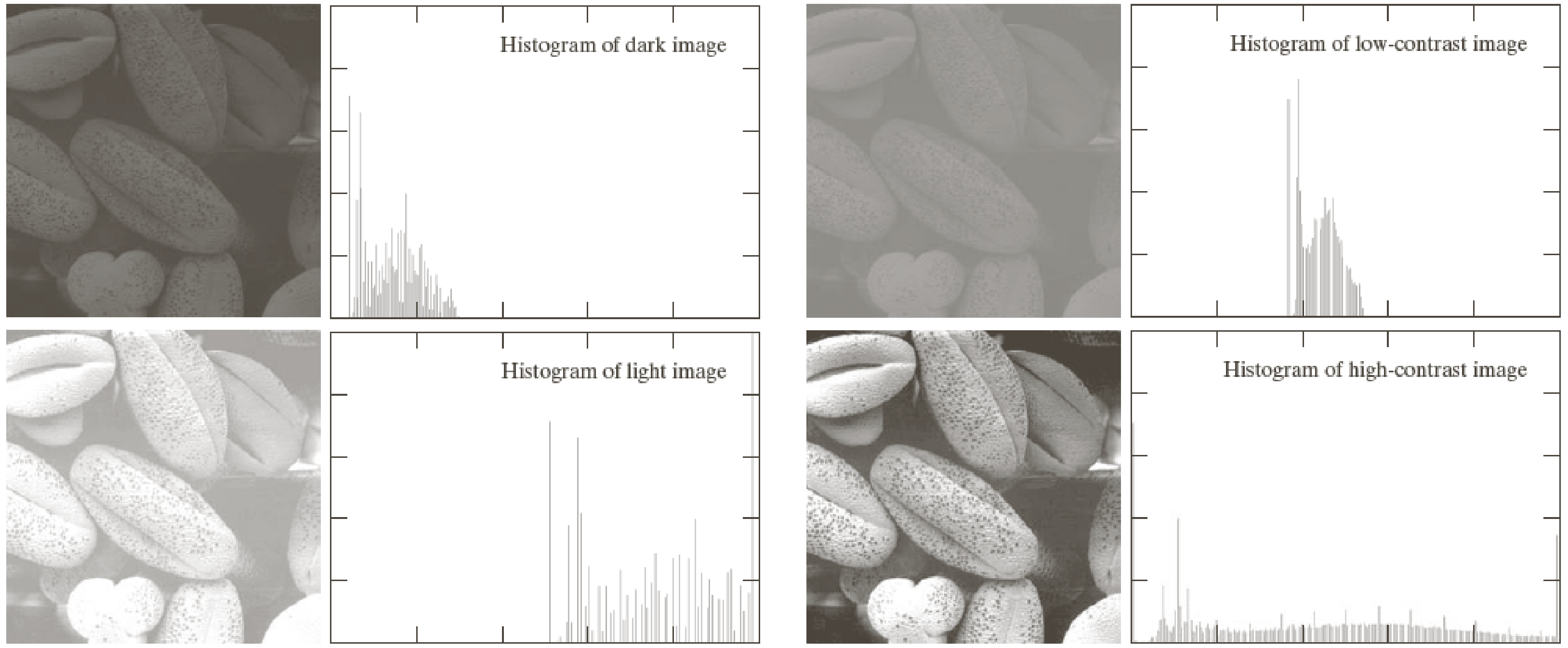
- Histogram: a discrete function *h(r)* which counts the number of pixels in the image having intensity *r*
- If *h(r)* is normalized, it measures the probability of occurrence of intensity level *r* in an image



- What histograms say about images?
- What they don't?
  - No spatial information

A descriptor for visual information

Images and histograms



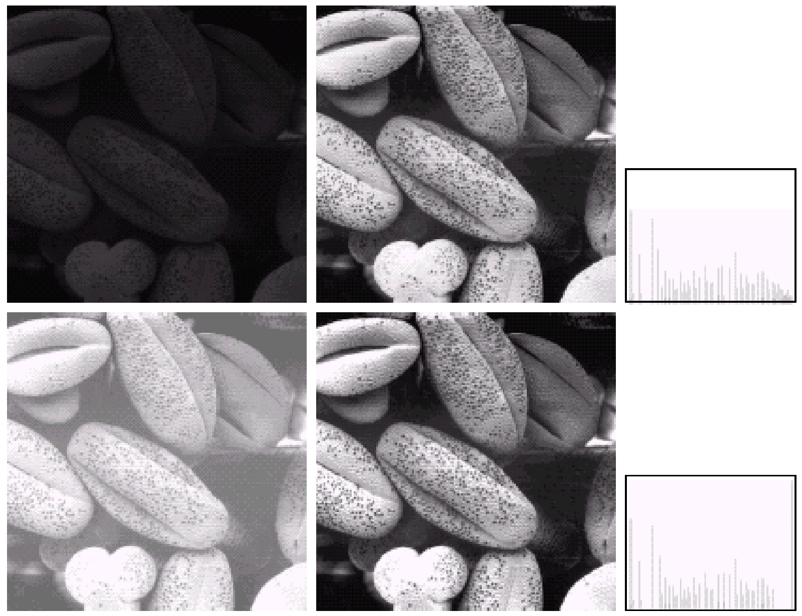
- How do histograms change when
  - we adjust brightnesss? shifts the histogram horizontally
  - we adjust constrast? stretches or shrinks the histogram horizontally

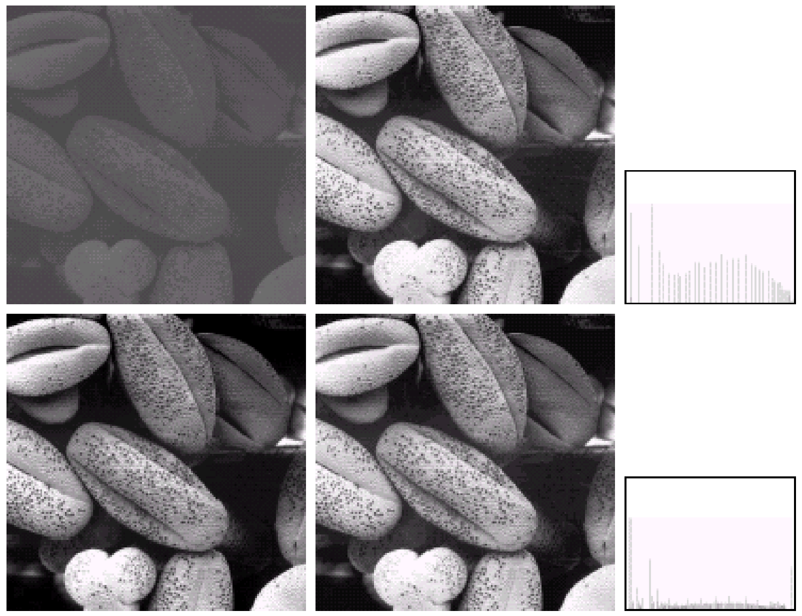
Histogram equalization

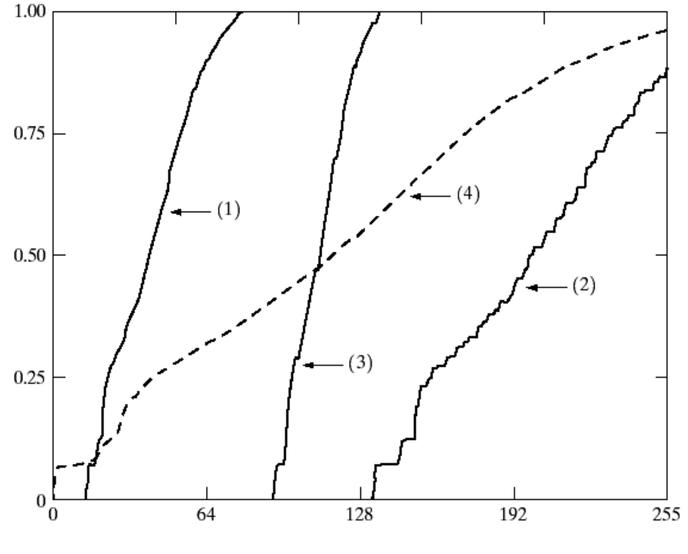
- A good quality image has a nearly uniform distribution of intensity levels. Why?
- Every intensity level is equally likely to occur in an image

- *Histogram equalization:* Transform an image so that it has a uniform distribution
  - create a lookup table defining the transformation

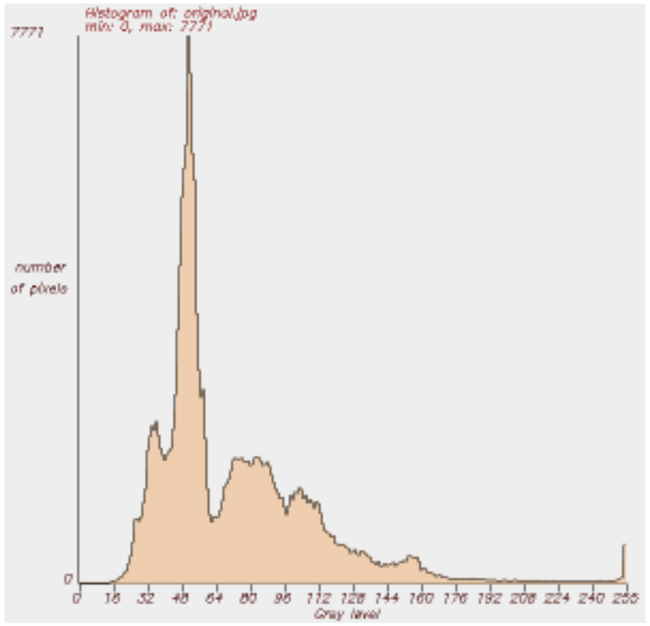
Histogram equalization examples

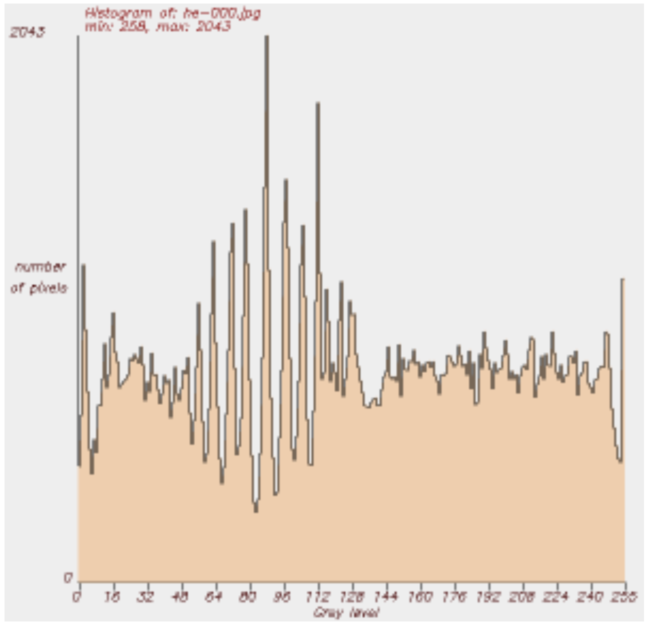


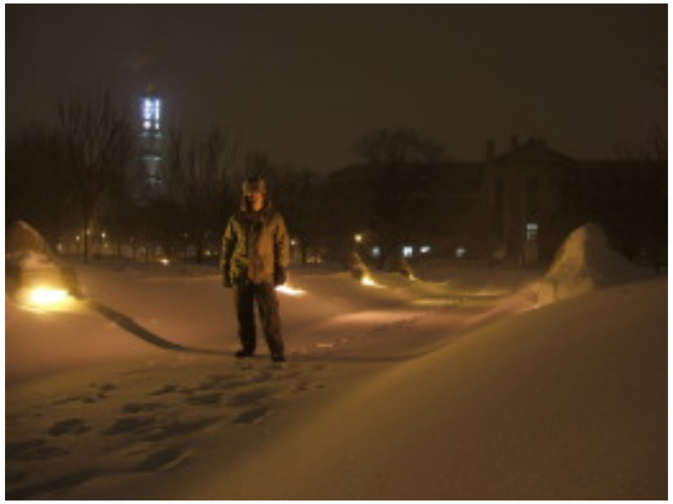


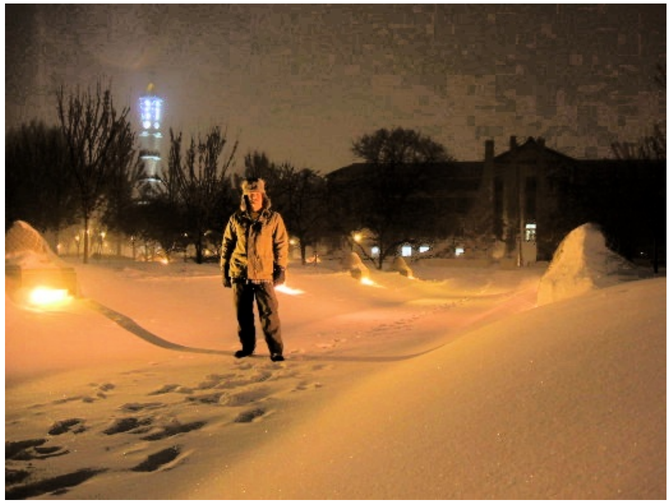


Histogram Equalization









Source: C. Dyer

Histogram as a probability density function

- Recall that a normalized histogram measures the probability of occurrence of an intensity level *r* in an image
- We can normalize a histogram by dividing the intensity counts by the area

$$p(r) = \frac{h(r)}{Area}$$

Histogram equalization: Continuous domain

- Define a transformation function of the form

$$s = T(r) = (L-1) \underbrace{\int_0^r p(w)\,dw}_{\text{cumulative distribution function}}$$

where
- r is the input intensity level
- s is the output intensity level
- p is the normalized histogram of the input signal
- L is the desired number of intensity levels

(Continuous) output signal has a uniform distribution!

Histogram equalization: Discrete domain

- Define the following transformation function for an MxN image

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} \frac{n_j}{MN} = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$$

$$\text{for} \quad k = 0,\dots,L-1$$

where

- r_k is the input intensity level
- s_k is the output intensity level
- n_j is the number of pixels having intensity value j in the input image
- L is the number of intensity levels

(Discrete) output signal has a nearly uniform distribution!

Histogram Specification

- Given an input image *f* and a specific histogram $p_2(r)$, transform the image so that it has the specified histogram

- How to perform histogram specification?

- Histogram equalization produces a (nearly) uniform output histogram
- Use histogram equalization as an intermediate step

Histogram Specification

1. Equalize the histogram of the input image

$$T_1(r) = (L-1)\int_0^r p_1(w)\,dw$$

2. Histogram equalize the desired output histogram

$$T_2(r) = (L-1)\int_0^r p_2(w)\,dw$$

3. Histogram specification can be carried out by the following point operation:

$$s = T(r) = T_2^{-1}(T_1(r))$$

Programming Assignment

1. Image Processing
2. Read, save, and display images with plot
3. NumPy for Images
4. Image Channels
5. Arithmetic Operations
6. Image Histogram with NumPy and plot