# Computational Photography

Week 3, Winter 2009

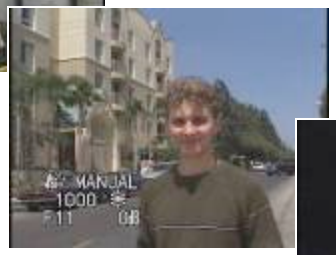
Instructor: Prof. Ko Nishino

# Problem: Dynamic Range


1


1500


25,000



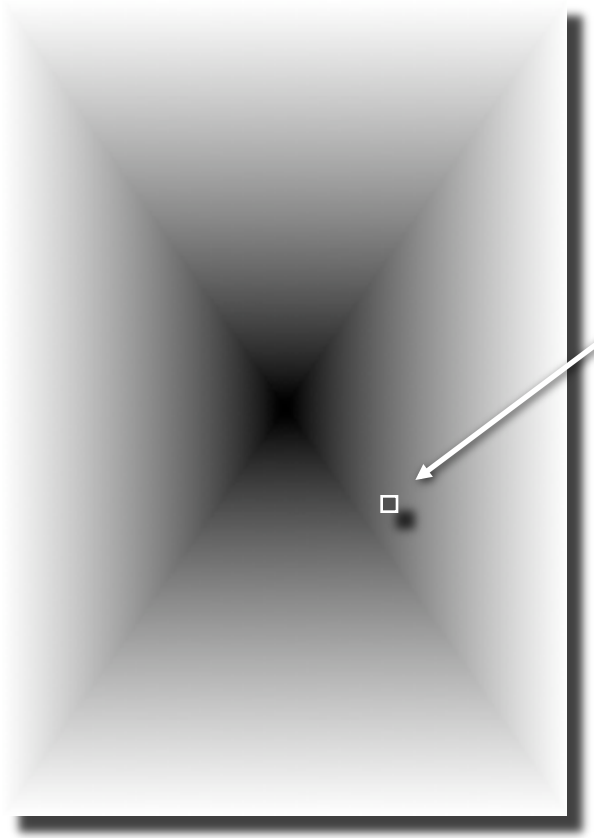

400,000


2,000,000,000

The real world is high dynamic range

# Is camera a photometer?

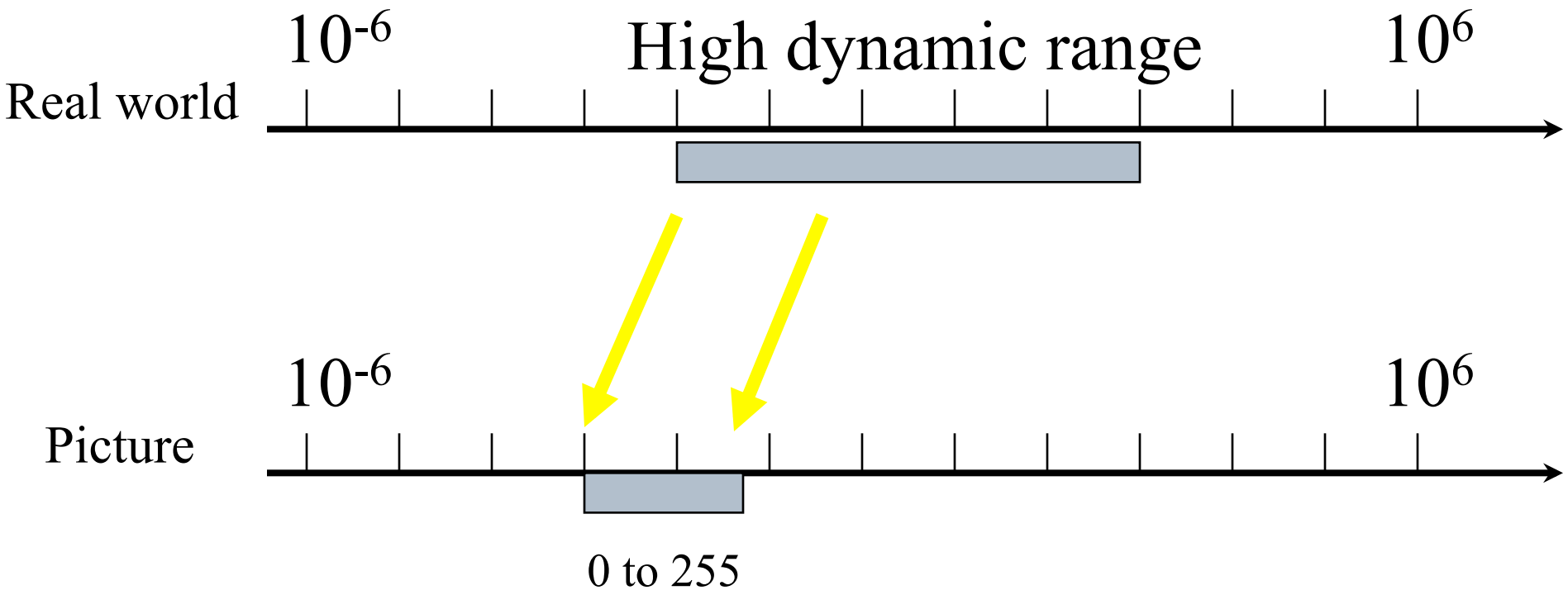
Image



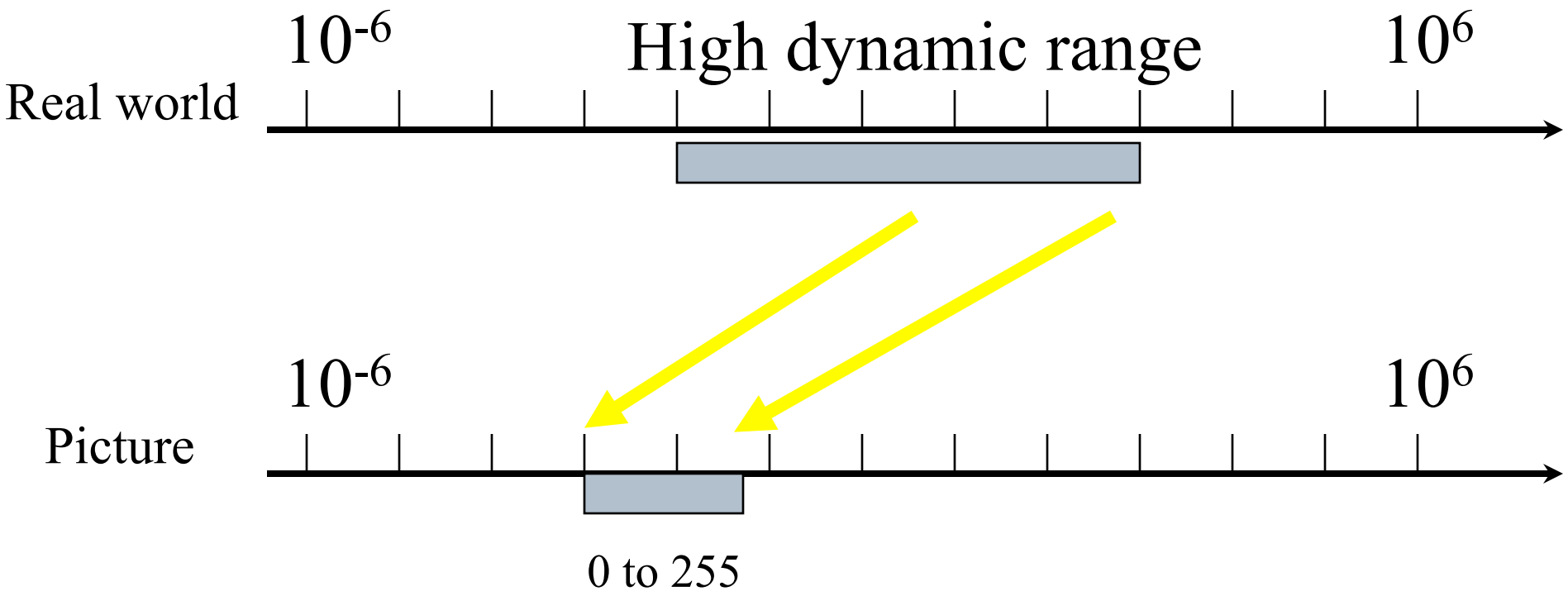
pixel (312, 284) = 42

42 photos?

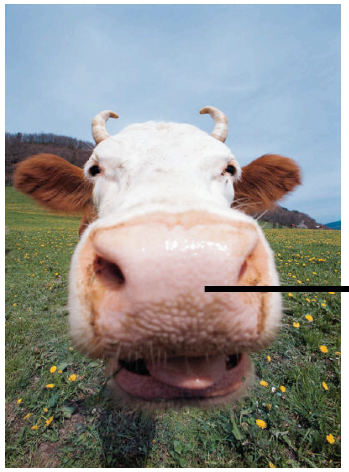
# Long Exposure

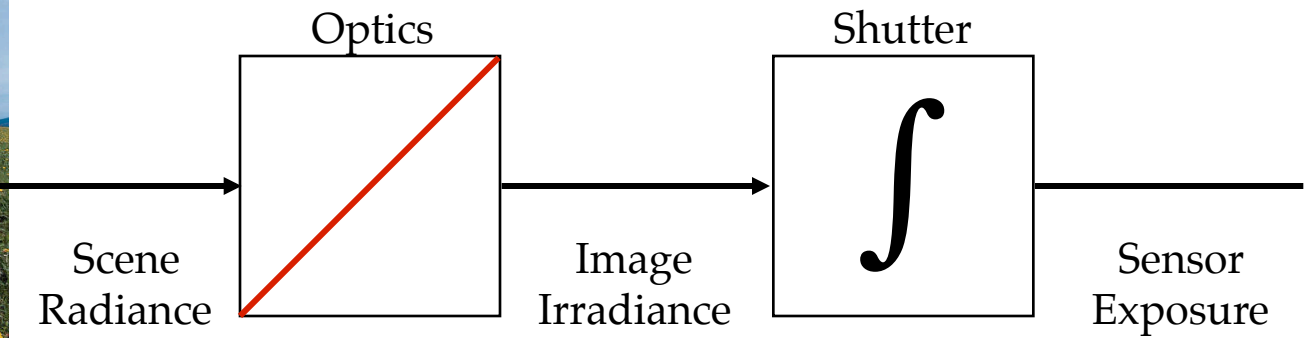


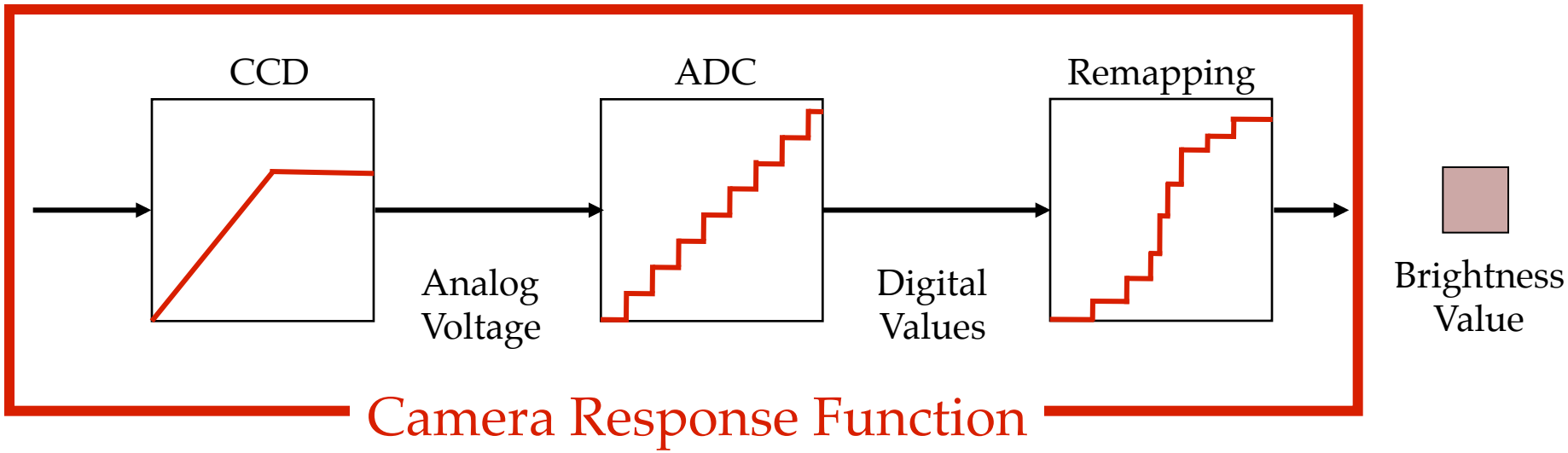
# Short Exposure



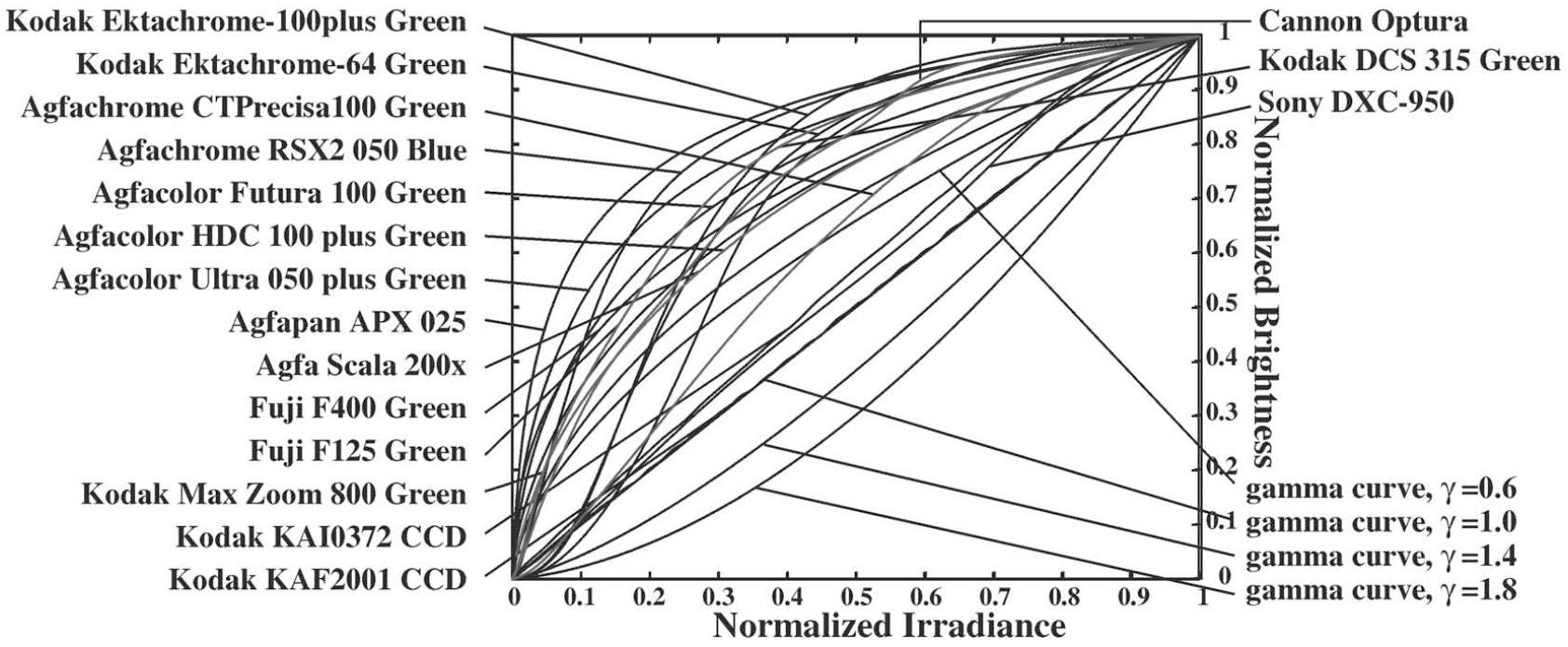
# Radiometric Pipeline





# Camera Response Function




by M. Grossberg and S.K. Nayar

# Dynamic Range

P. Debevec and J. Malik, SIGGRAPH 1997

# Extending the Dynamic Range

Figure 6: *Sixteen photographs of a church taken at 1-stop increments from 30 sec to* $\frac{1}{1000}$ *sec. The sun is directly behind the rightmost stained glass window, making it especially bright. The blue borders seen in some of the image margins are induced by the image registration process.*

How do we combine all these brightness values at each pixel?

Have to estimate camera response function!

# Extending the Dynamic Range

- http://www1.cs.columbia.edu/CAVE//software/rascal/rrgallery.php

# What does the eye see?

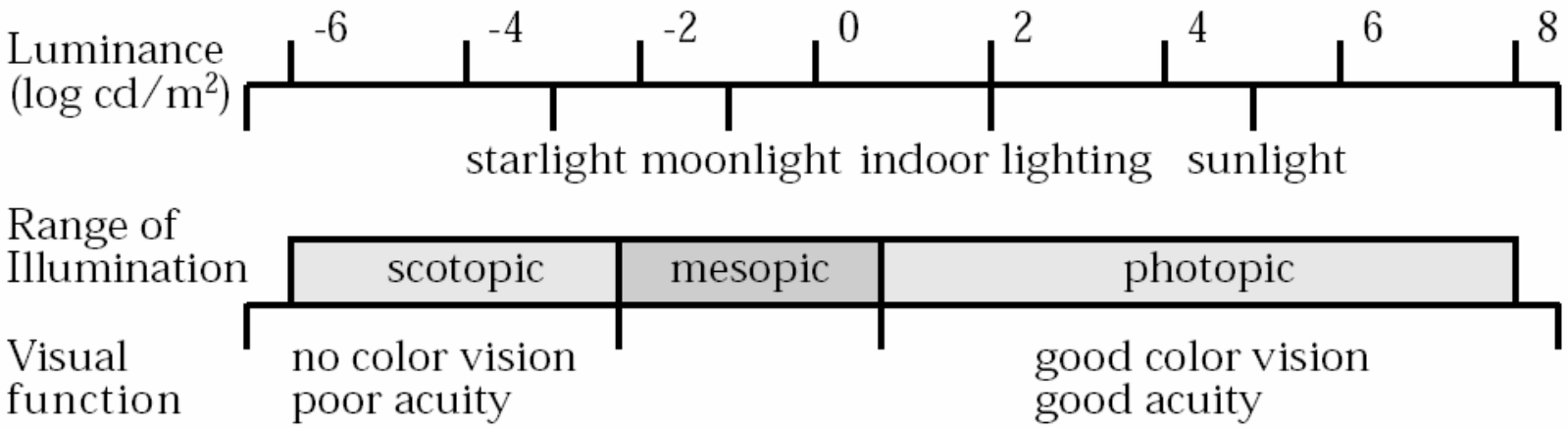



Figure 1: The range of luminances in the natural environment and associated visual parameters. After Hood (1986).

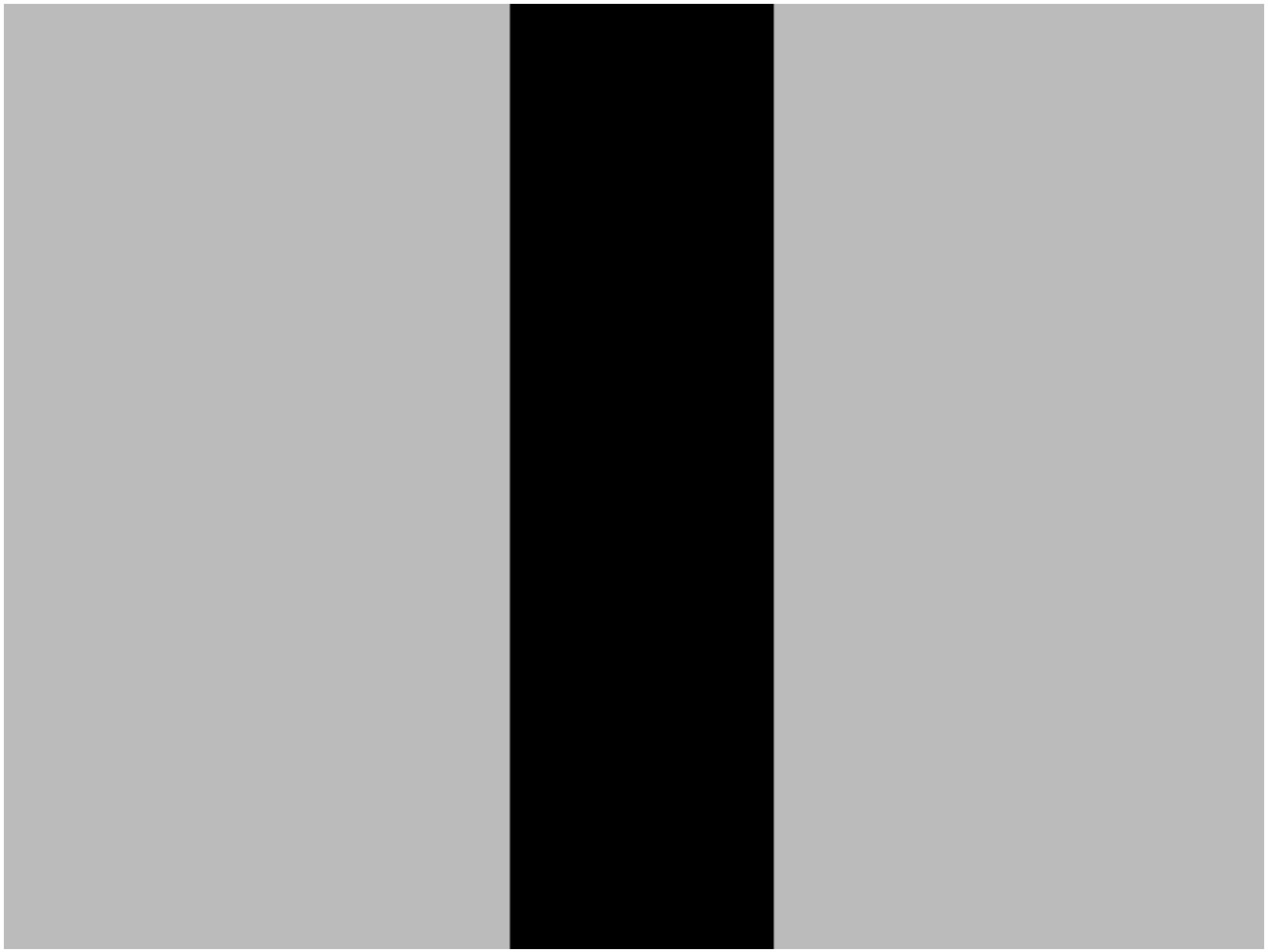
The eye has a huge dynamic range
Do we see a true radiance map?

# Eye is not a photometer!

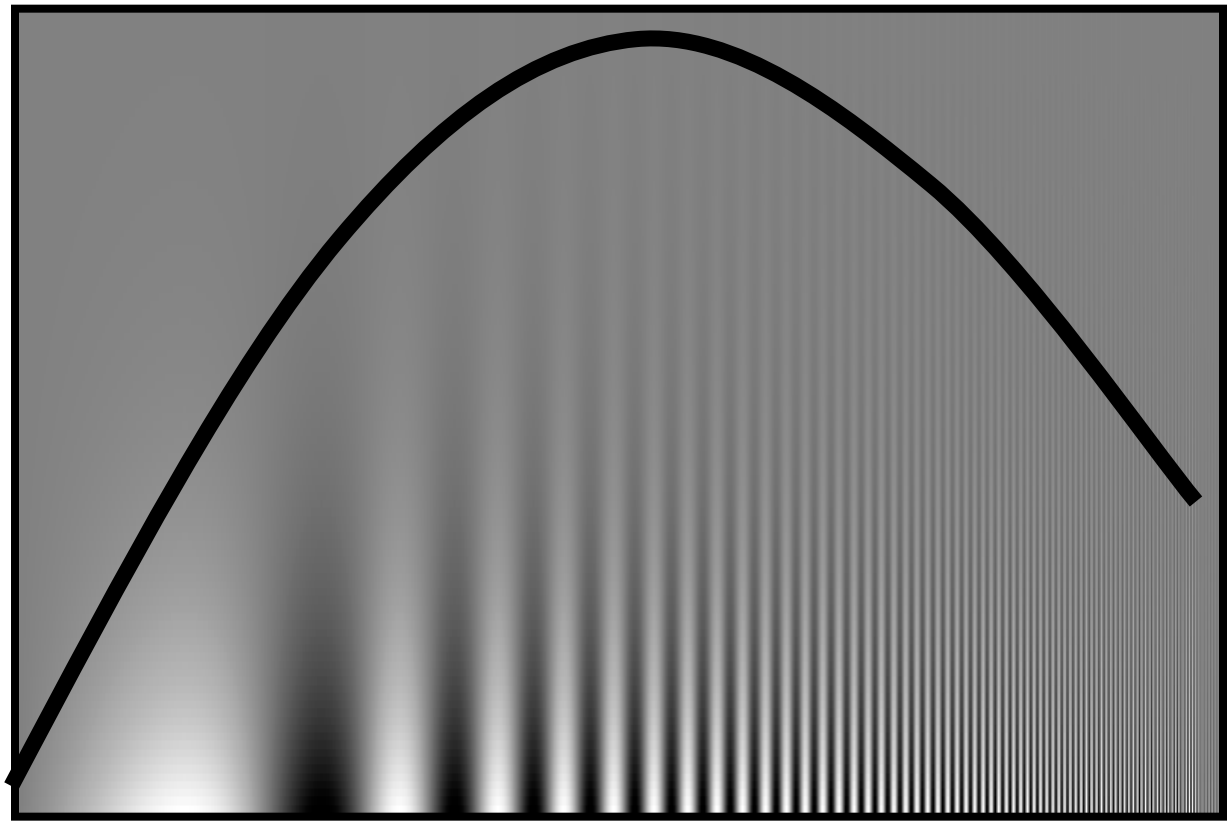
*"Every light is a shade, compared to the higher lights, till you come to the sun; and every shade is a light, compared to the deeper shades, till you come to the night."*

— John Ruskin, 1879

# Cornsweet Illusion

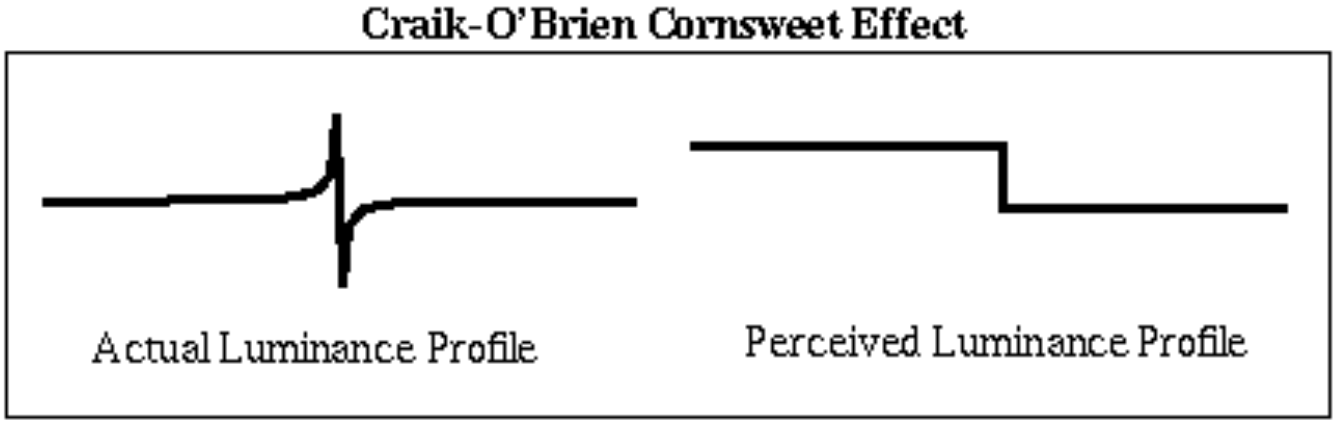


# Sine Wave



Campbell-Robson contrast sensitivity curve

# Metameric Failure




Eye is sensitive to changes

Hybrid Images, Oliva and Torralba

# Image Filtering

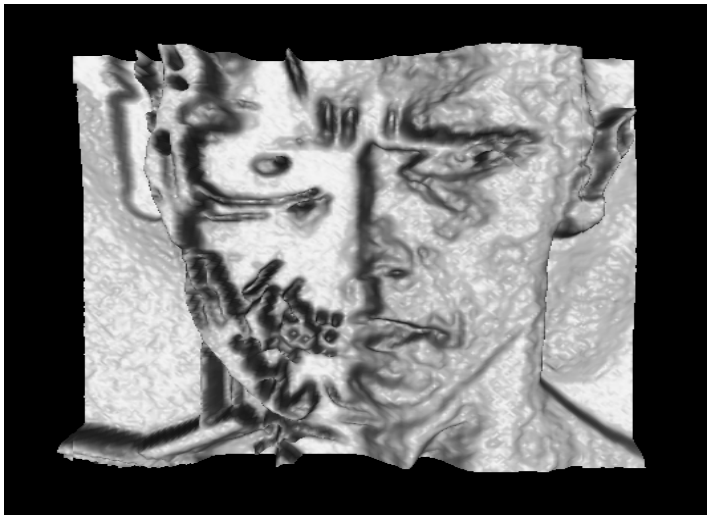
# What is an image?

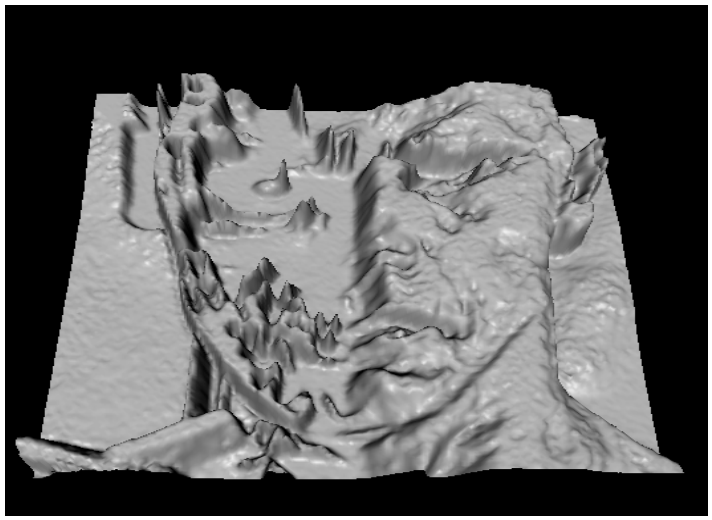
- We can think of an **image** as a function, $f$, from $R^2$ to R:
  - $f(x, y)$ gives the **intensity** at position $(x, y)$
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - $f$: $[a,b]$x$[c,d]$ → [0,1]

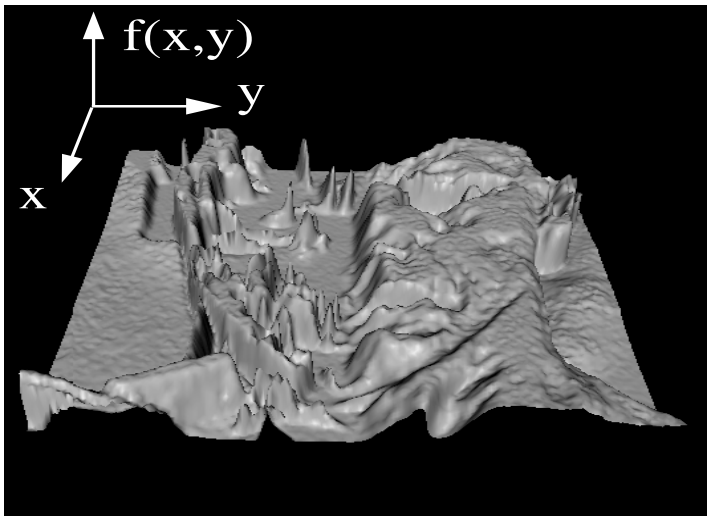
- A color image is just three functions pasted together. We can write this as a "vector-valued" function:

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

# Images as Functions





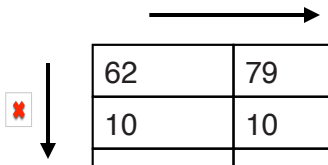


# What is a digital image?

- We usually operate on **digital** (**discrete**) images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- If our samples are $\Delta$ apart, we can write this as:

$$f[i,j] = \text{Quantize}\{ f(i\,\Delta, j\,\Delta) \}$$

- The image can now be represented as a matrix of integer values



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

# Image Processing

- An **image processing** operation typically defines a new image *g* in terms of an existing image *f*.
- We can transform either the range of *f*.

$$g(x, y) = t(f(x, y))$$

- Or the domain of *f*:

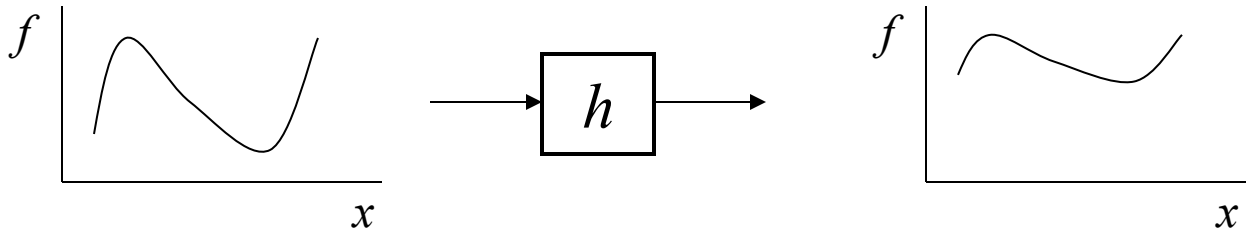
$$g(x, y) = f(t_x(x, y), t_y(x, y))$$

- What kinds of operations can each perform?

# Image Processing

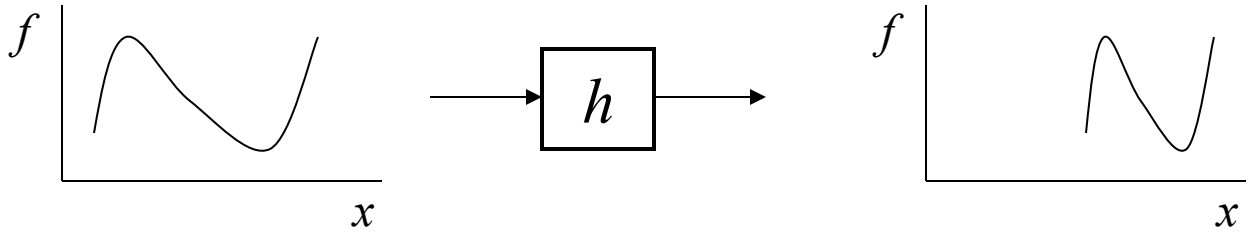
- image filtering: change ***range*** of image

$$g(x) = h(f(x))$$




- image warping: change ***domain*** of image

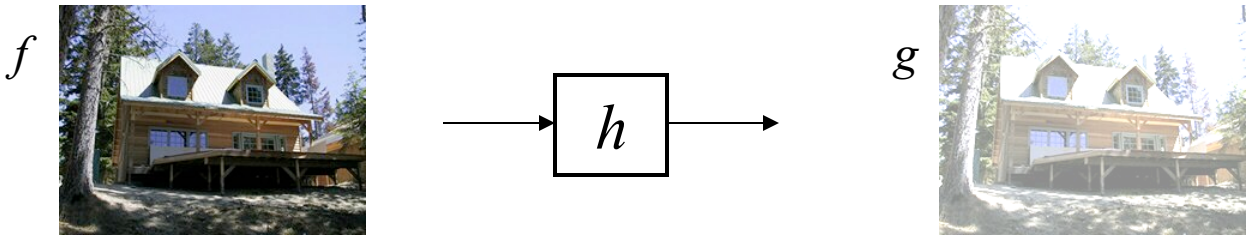
$$g(x) = f(h(x))$$



# Image Processing

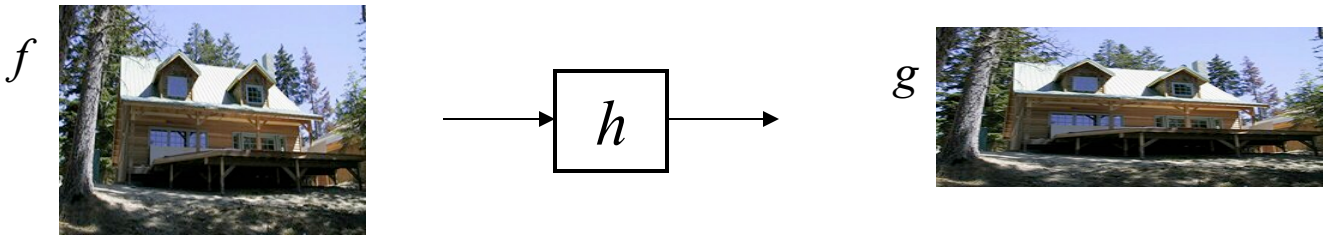
- image filtering: change ***range*** of image

$$g(x) = h(f(x))$$




- image warping: change ***domain*** of image

$$g(x) = f(h(x))$$



# Point Processing

- The simplest kind of range transformations are these independent of position x,y:

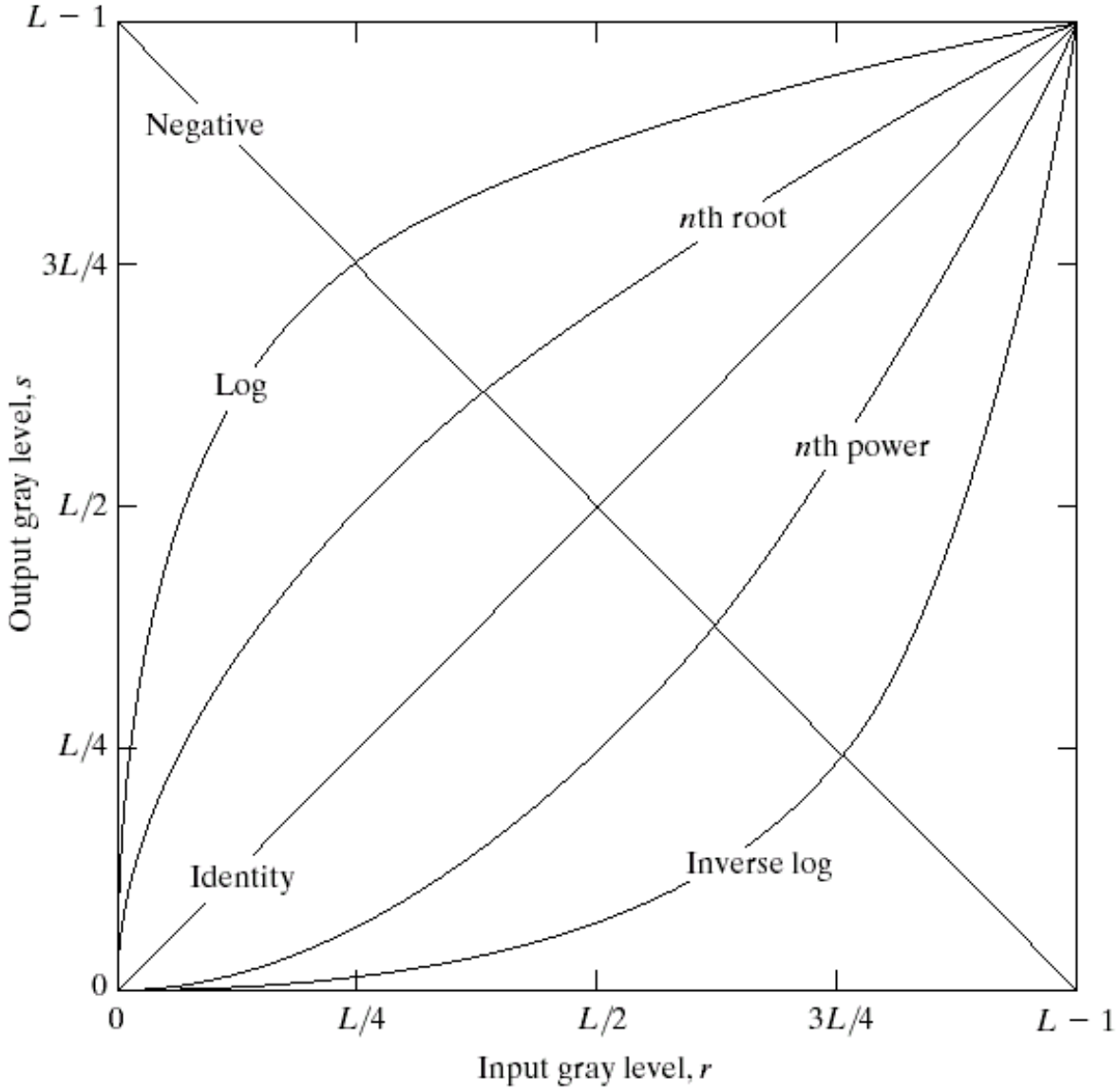
$$g = t(f)$$

- This is called point processing.

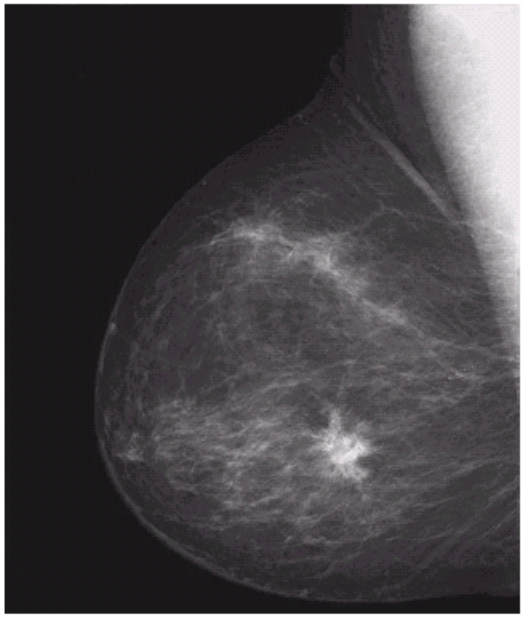
- What can they do?
- What's the form of *t*?

- **Important:** every pixel for himself – spatial information completely lost!

# Basic Point Processing



# Negative



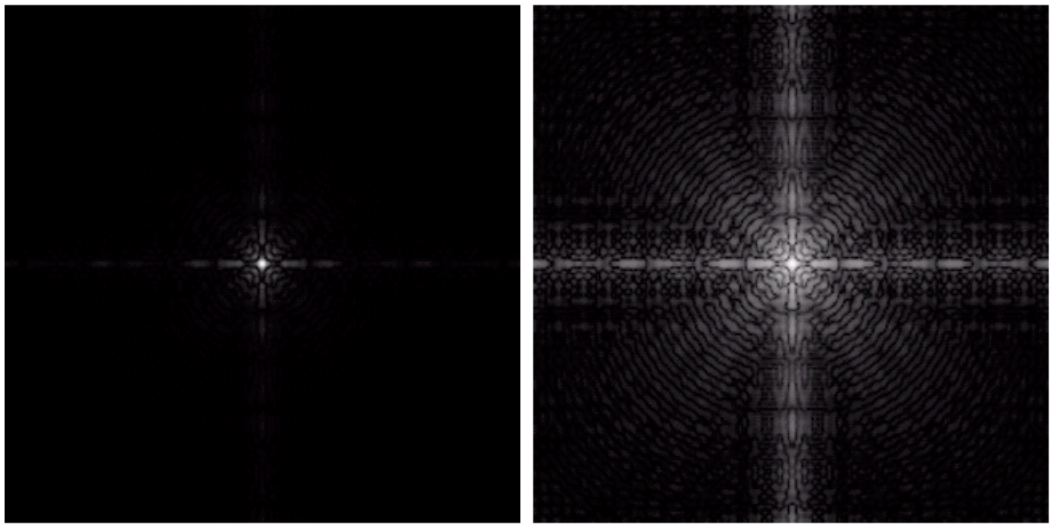
a b

**FIGURE 3.4**
(a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

# Log

a b

**FIGURE 3.5**
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.



# Power-law Transformations

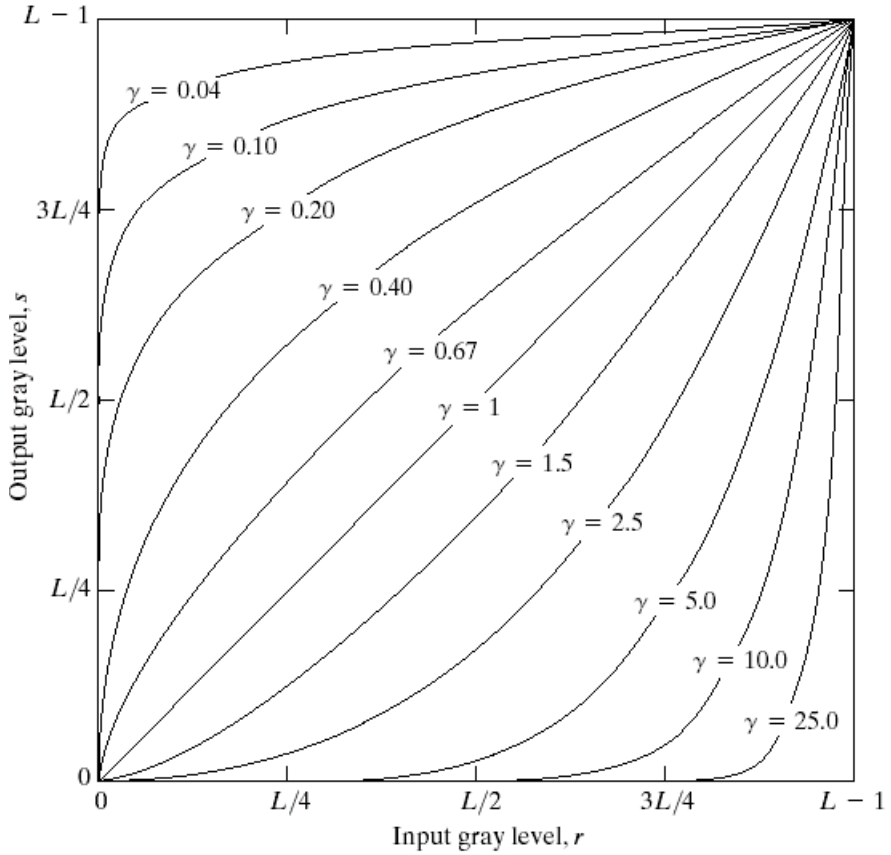



**FIGURE 3.6** Plots of the equation $s = cr^{\gamma}$ for various values of $\gamma$ ($c = 1$ in all cases).

$$s = cr^{\gamma}$$

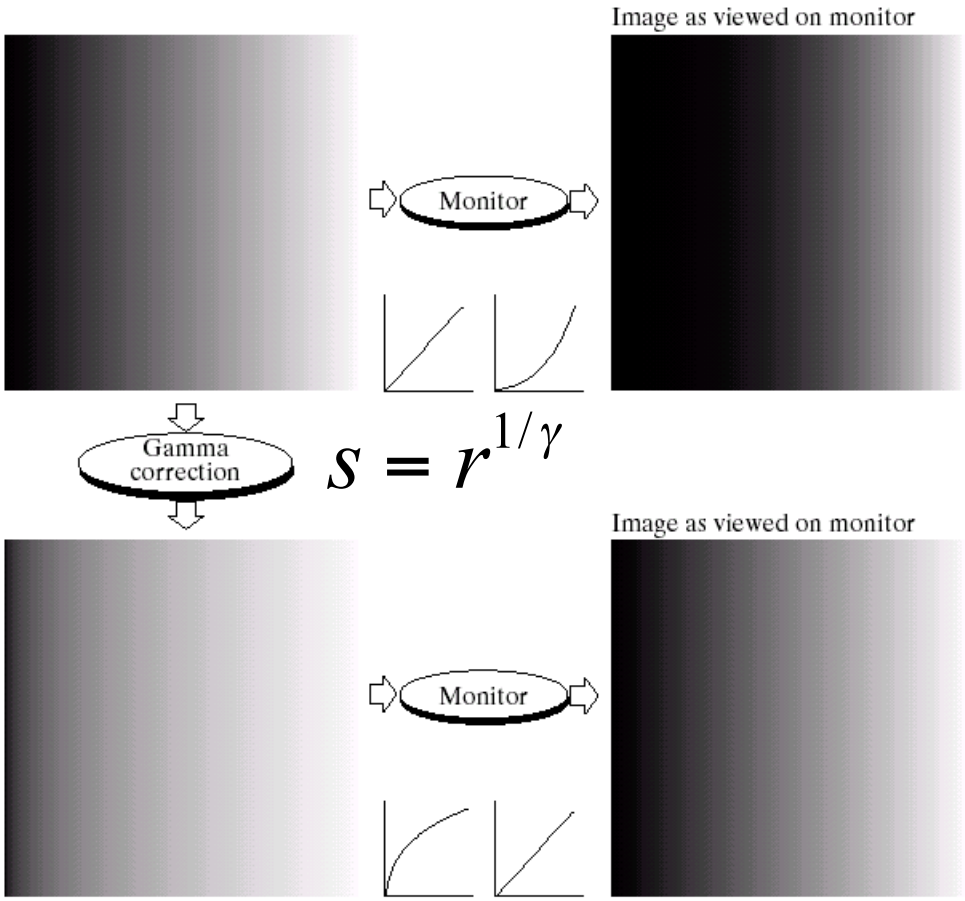
# Image Enhancement

$\gamma = 3.0$

$\gamma = 4.0$

$\gamma = 5.0$

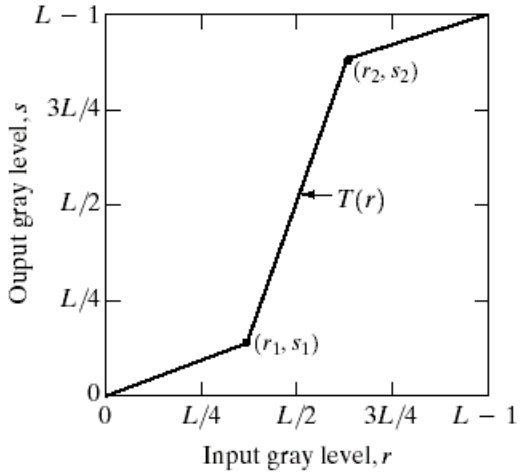
# Example: Gamma Correction

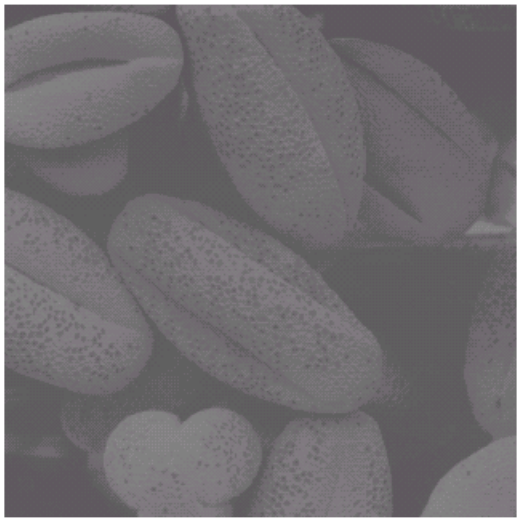


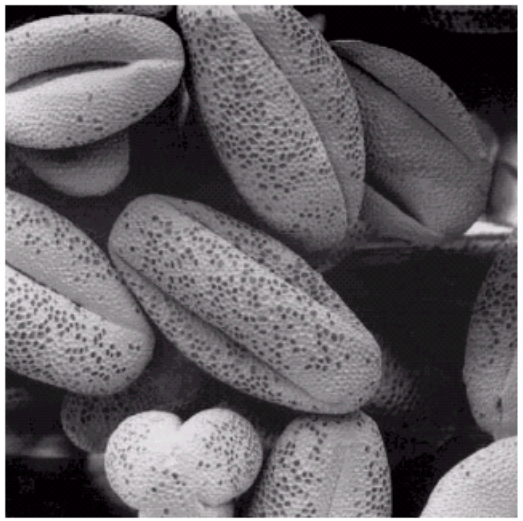

$$s = r^{\gamma}$$

$$e.g.\ \ 0.25 = 0.5^{2.0}$$

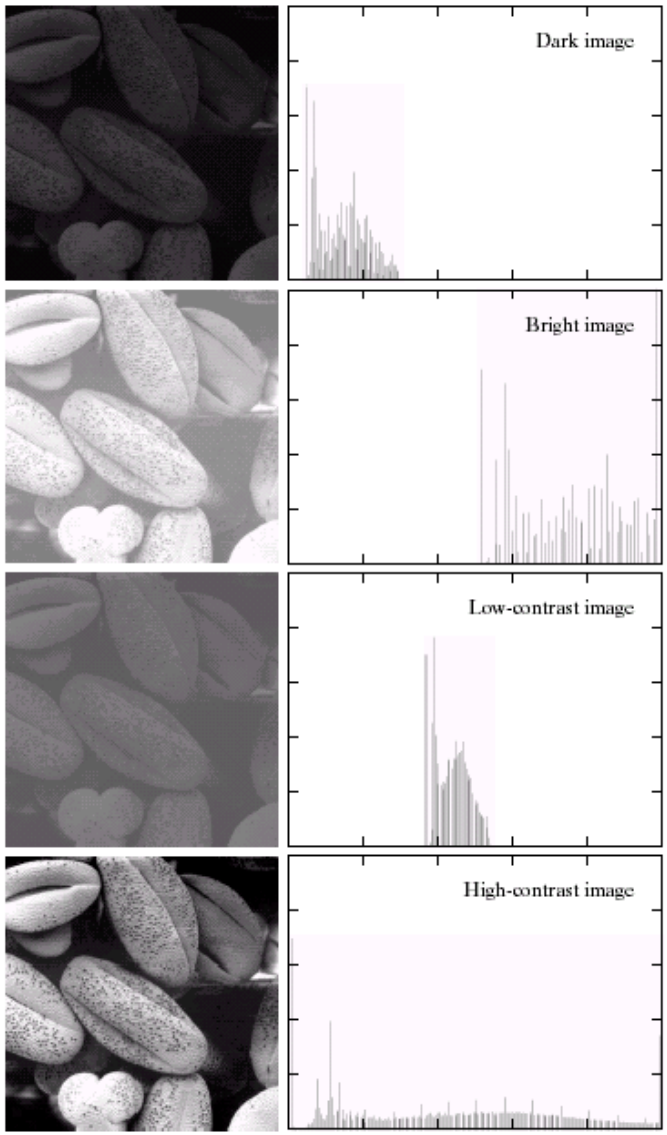
# Contrast Stretching

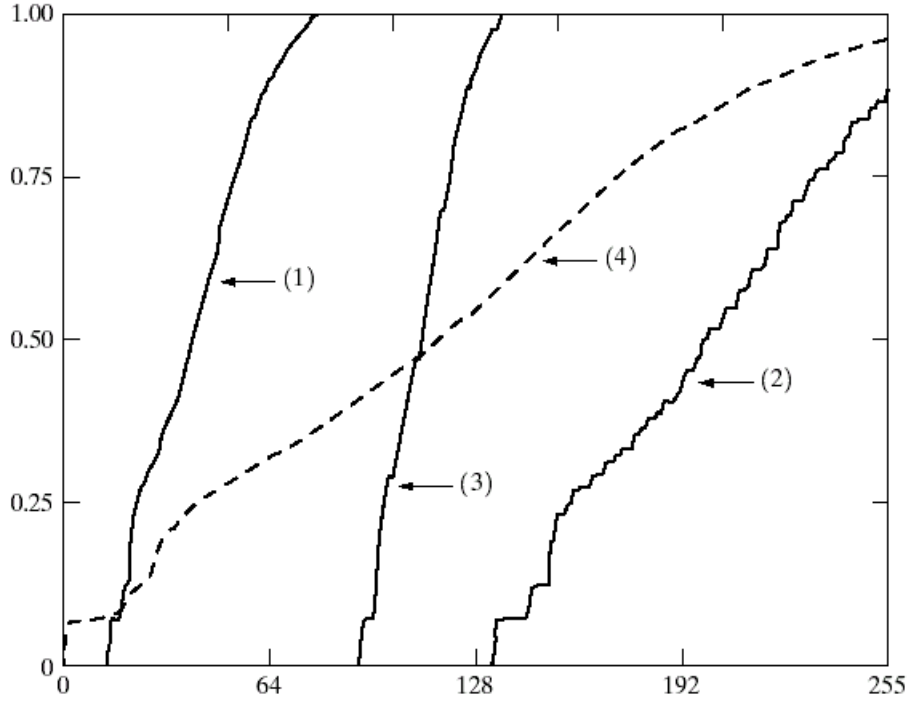




# Image Histograms

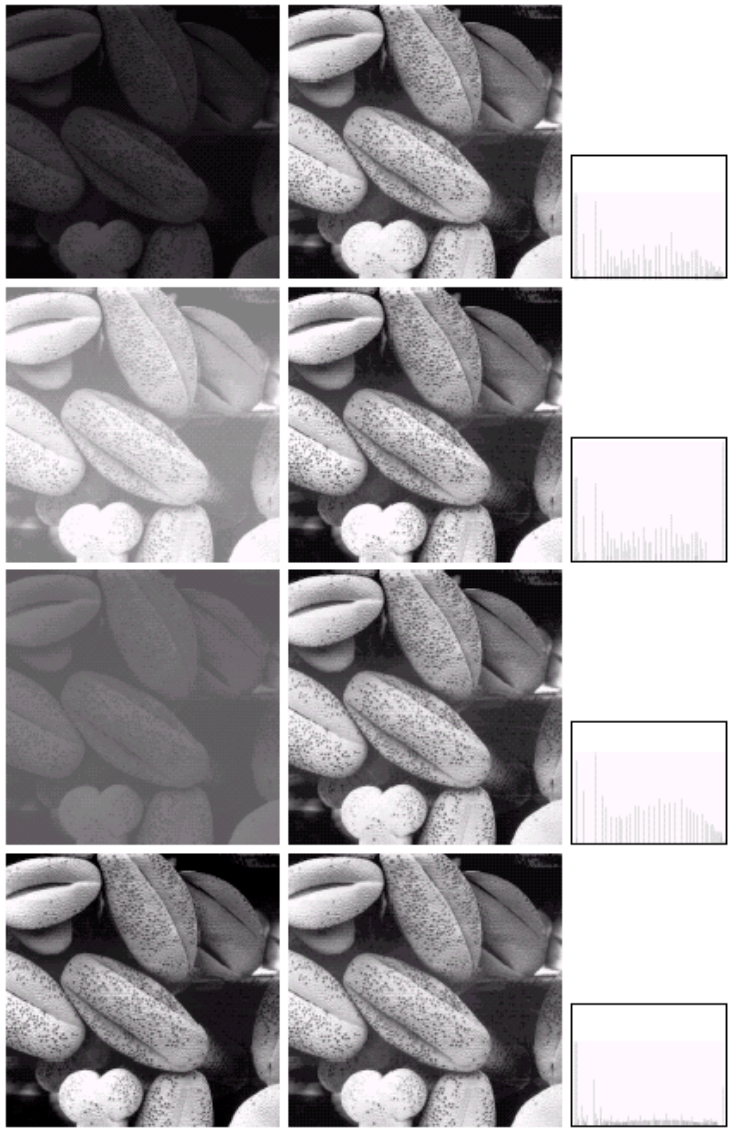


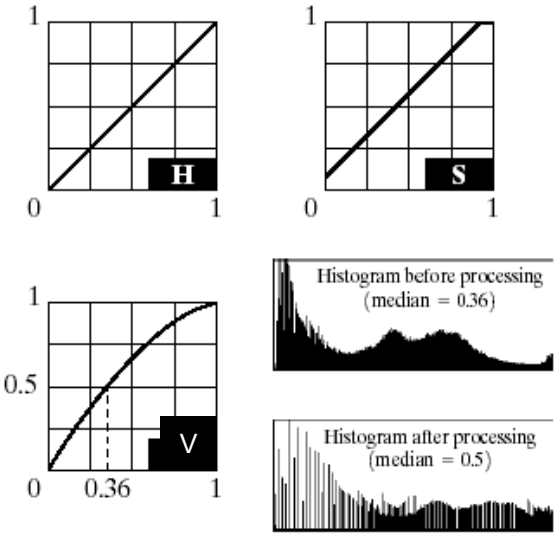

Cumulative Histograms

$s = T(r)$

# Histogram Equalization



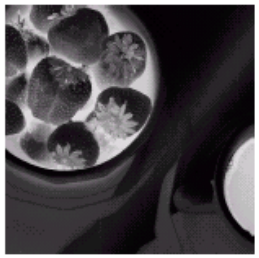
# Histogram Equalization



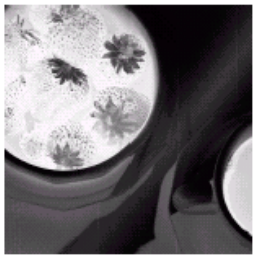

Histogram equalize the brightness (V)

# Color Channels

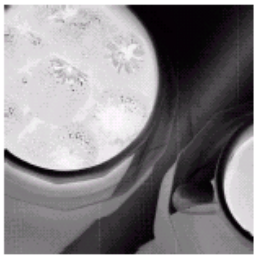
Full color



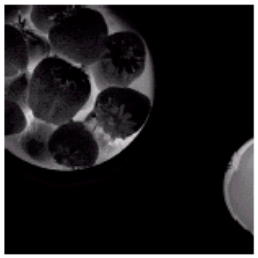
Cyan



Magenta



Yellow

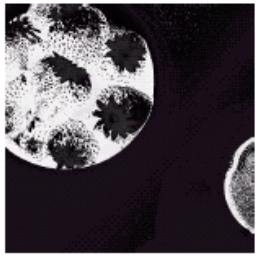


Black

Red

Green

Blue



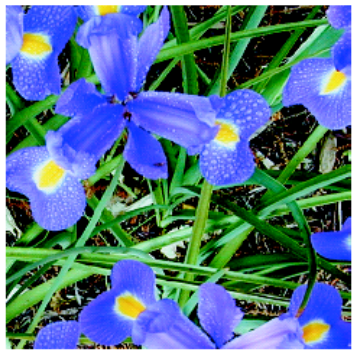
Hue

Saturation

Intensity

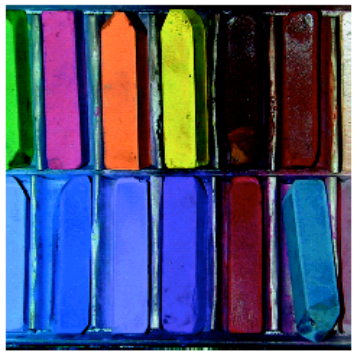
# Color Point Processing

Flat

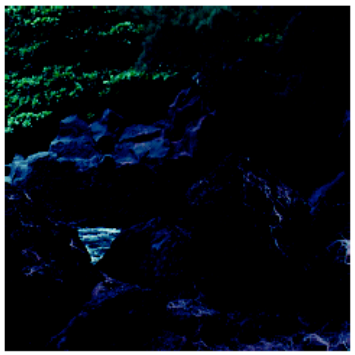


Corrected

Light



Corrected

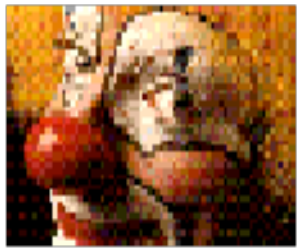


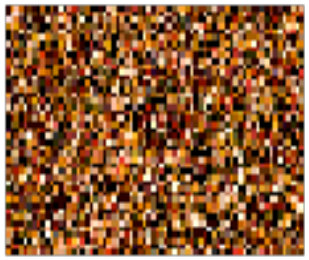
Dark

Corrected

# Neighborhood Processing (Filtering)

- Q: What happens if I reshuffle all pixels within the image?





- A: It's histogram won't change. No point processing will be affected…

- Need spatial information to capture this

# Filtering Noise

- How can we “smooth” away noise in an image?

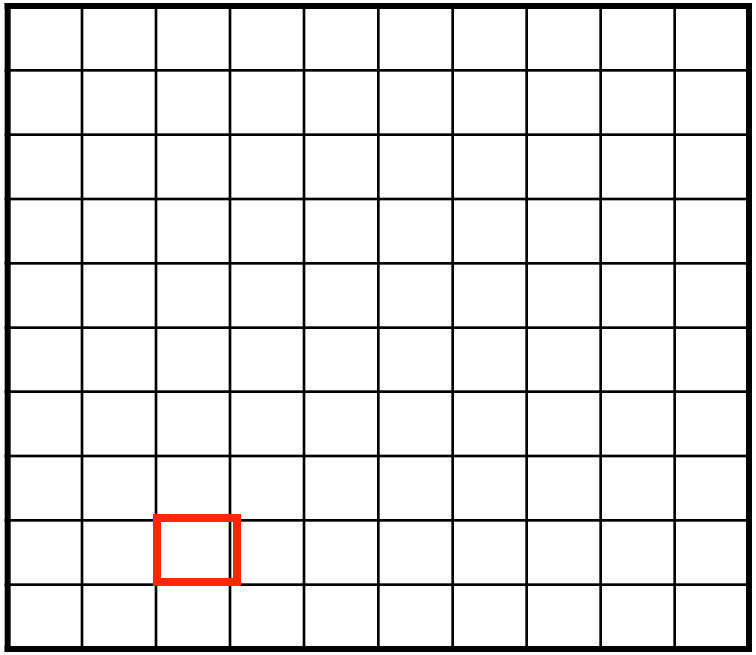
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 100 | 130 | 110 | 120 | 110 | 0 | 0 |
| 0 | 0 | 0 | 110 | 90 | 100 | 90 | 100 | 0 | 0 |
| 0 | 0 | 0 | 130 | 100 | 90 | 130 | 110 | 0 | 0 |
| 0 | 0 | 0 | 120 | 100 | 130 | 110 | 120 | 0 | 0 |
| 0 | 0 | 0 | 90 | 110 | 80 | 120 | 100 | 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 | 0 |

# Mean Filtering

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$F[x, y]$

$G[x, y]$



# Mean Filtering

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$F[x, y]$

$G[x, y]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | 30 | 20 | 10 | |
| | 0 | 20 | 40 | 60 | 60 | 60 | 40 | 20 | |
| | 0 | 30 | 60 | 90 | 90 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 20 | 30 | 50 | 50 | 60 | 40 | 20 | |
| | 10 | 20 | 30 | 30 | 30 | 30 | 20 | 10 | |
| | 10 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | |
| | | | | | | | | | |

# Cross-Correlation Filtering

- Let's write this down as an equation. Assume the averaging window is (2k+1)x(2k+1):

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

- We can generalize this idea by allowing different weights for different neighboring pixels:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

- This is called a **cross-correlation** operation and written:

$$G = H \otimes F$$

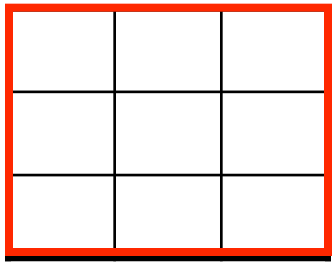
- H is called the "filter," "kernel," or "mask."
- The above allows negative filter indices. When you implement need to use: H[u+k,v+k] instead of H[u,v]

# Mean Kernel

- What's the kernel for a 3x3 mean filter?

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$F[x, y]$



$H[u, v]$

When can taking an unweighted mean be bad idea?

# Gaussian Filtering

- A Gaussian kernel gives less weight to pixels further from the center of the window

$F[x, y]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

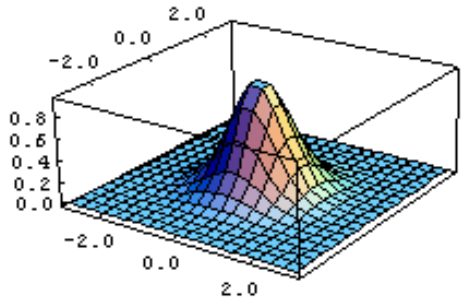
$\frac{1}{16}$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

$H[u, v]$

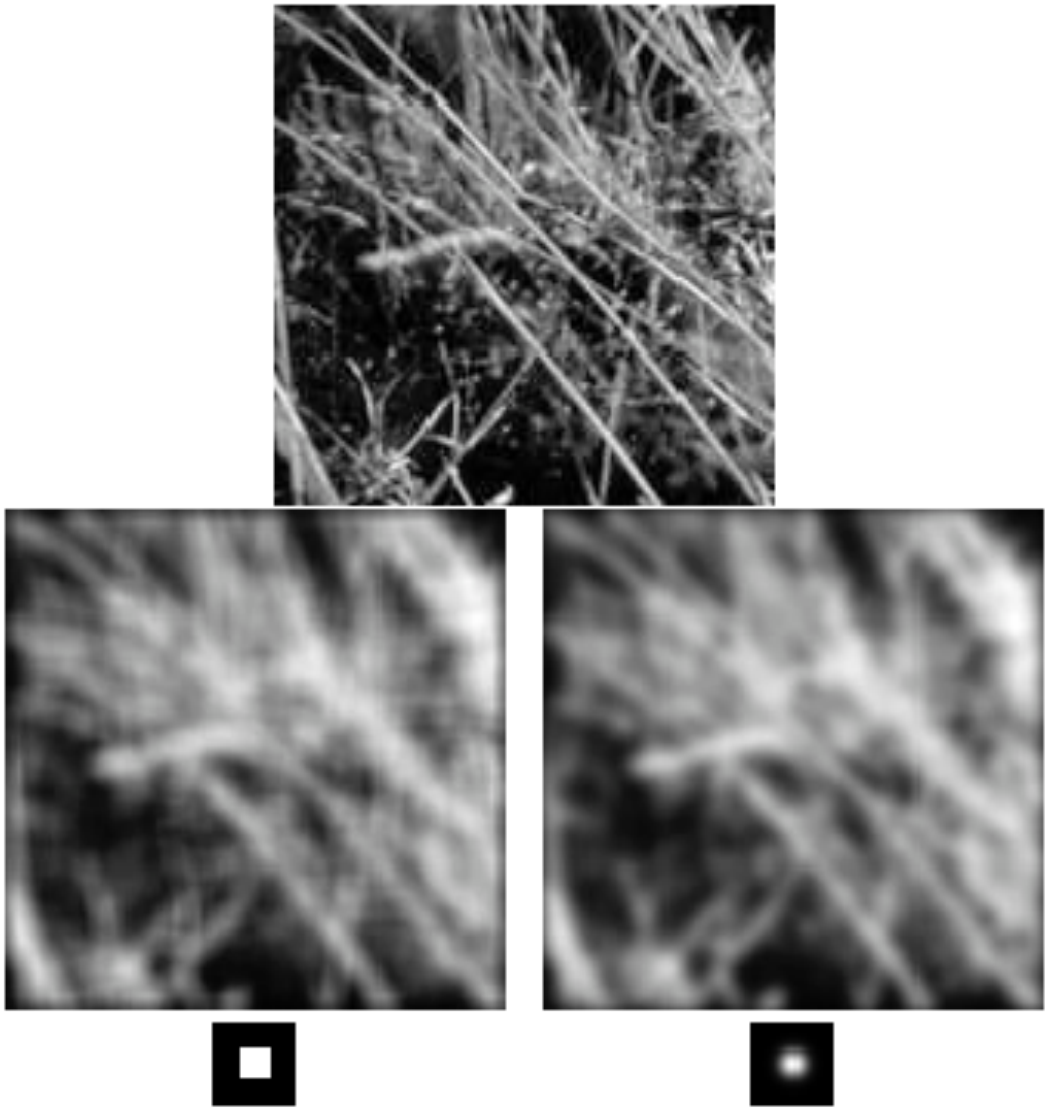
- This kernel is an approximation of a Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$




- What happens if you increase σ ?

# Mean vs. Gaussian Filtering

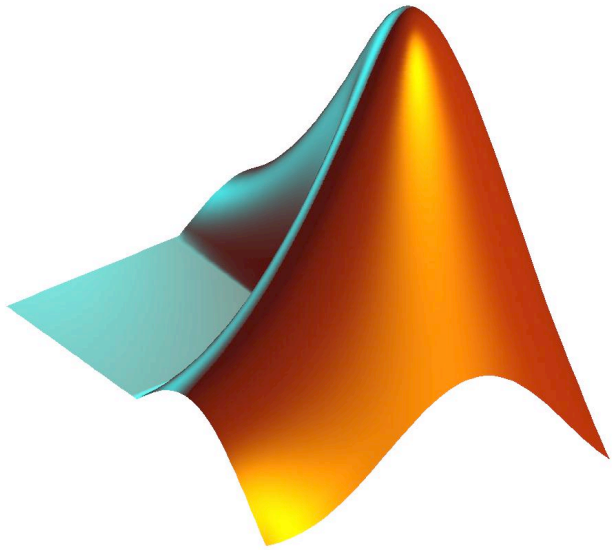


# Project 0.5

- Basic Image Processing
  - Color image to gray-scale image
  - Positive image to negative image
  - Gamma Correction
  - Mean Filtering
  - Gaussian Filtering
- Due next 1/31 Saturday midnight
- See the assignment page for details

# A Quick Matlab Tutorial

# Matlab Primer



# Matlab

- **Matlab** short for MATrix LABoratory
  - Designed for calculations used by scientists and engineers
- Interactive system
  - Contains a complete programming language
- Extended by numerous toolboxes
  - i.e., Image processing toolbox

# Running Matlab

- Available on computer science department machines (**tux.cs.drexel.edu**)
  - For an account, see the department webpage

- Runs in two modes, graphical and text
  - Graphical mode: **$> matlab**
  - Text mode: **$> matlab –nojvm**
  - **($> matlab –nodesktop)**

# Options for running Matlab

- Computing lab
  - Graphical mode
- Personal machine
  1. Open **Xwin32**
  2. SSH to **tux.cs.drexel.edu**
     - Xwin32/SSH available from **software.drexel.edu**
  3. On tux, set DISPLAY environment variable to your home IP
     - **$> export DISPLAY=<YOUR HOME IP>:0**
  4. Run Matlab in text mode

Or buy a Mac…

# Matlab Basics

- Variable assignment

```
>> X = 5
X = 5
>> X = 5;
>>
```

- Displaying output

```
>> disp(sprintf('Hello World'))
Hello World
>>
>> fprintf(1, ‘Hello World\n’);
```

# Matrices

- Various matrix operations

```
>> A = rand(2,2)
A =
    0.9501    0.6068
    0.2311    0.4860
>> size(A)
ans =
     2     2
>> [rows, columns] = size(A);
>> A(2,1)
ans = 0.2311
>>
```

# Conditional statements

- If statements

```
if x==2,
  fprintf(1, ‘x is 2\n’);
else
  fprintf(1, ‘Nope its not 2\n’);
end
```

# Relational Operators

| | |
|---|---|
| < | less than |
| > | greater than |
| <= | less than or equal |
| >= | greater than or equal |
| == | equal |
| ~= | not equal. |

# Loops

- For loops

```
>> for i=1:5,
                f(i);
   end
```

- While loop

```
>> while i<5,
                    f(i);
     end
```

# Creating a Function

- In the file myfunc.m

  function output = myfunc(input1,input2)

  .

  .

  .

  output = . .

  - Input and output can be a single variable, vector, matrix, etc.

- Calling the function from within matlab

  A = myfunc(x1,x2);

# Matlab and Images

- Reading in an image

  >> img = imread('circle.png');

  >> imshow(img)

  Also look up imagesc

- An image is stored as a matrix

  >> img

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 0 | 0 | 0 |
| 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 0 | 0 |
| 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 0 | 0 |
| 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 0 | 0 |
| 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 0 | 0 |
| 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 0 | 0 |
| 0 | 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 255 | 255 | 255 | 255 | 255 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Matlab and Images

```
>> img = imread('dog.png');
```

```
>> imshow(img);
>> ginput(1);
ans = 300    100
```

# Matlab and Images

```
>> sub_img = img(1:300,300:500,1:3);
>> imshow(sub_img);
>> imwrite(sub_img,'dog_head.jpg');
```

# Improving Matlab Performance

- Traditional loop:

```
>> dx = pi/30;
>> nx = 1 + 2*pi/dx;
>> for i = 1:nx
       x(i) = (i-1)*dx;
       y(i) = sin(3*x(i));
   end
```

- Vectorized Loop:

```
>> x = 0:pi/30:2*pi;
>> y = sin(3*x);
```

http://web.cecs.pdx.edu/~gerry/MATLAB/programming/performance.html

# Octave

- Octave is an open source Matlab alternative
  - Designed to have the same functionality and syntax, but there are some differences.
- **NOTE:** Homeworks will be graded using Matlab
- References:
  - http://www.gnu.org/software/octave/
  - http://octave.sourceforge.net/

# References

- Matlab primer
  - http://www.glue.umd.edu/~nsw/ench250/primer.htm
- Matlab programming tutorial
  - http://www.math.ufl.edu/help/matlab-tutorial/
- Tutorial scripts
  - http://www.cs.drexel.edu/~kon/compphoto/matlabtut/matlab_tutorial.tar.gz