Computational Photography

Week 3

Instructor: Lou Kratz

Image Filtering

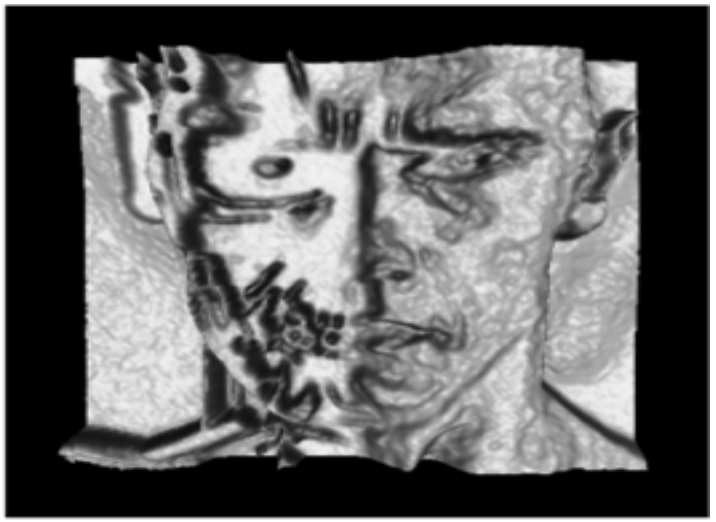
What is an image?

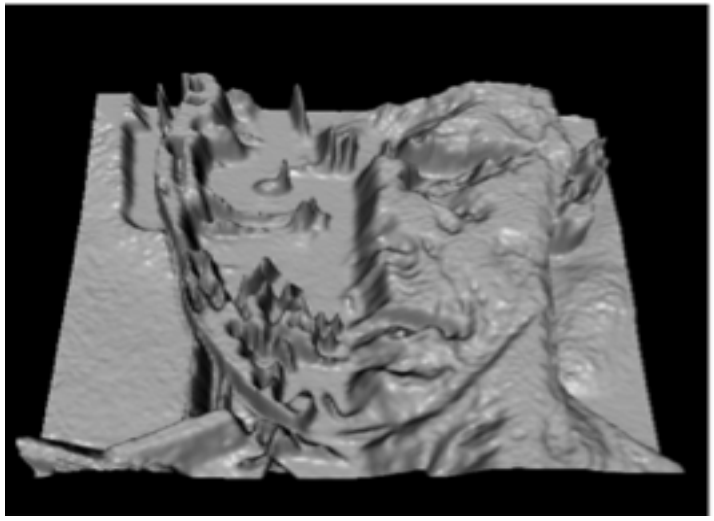
- We can think of an **image** as a function, f, from R^2 to R:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - f: $[a,b]$x$[c,d]$ → $[0,1]$

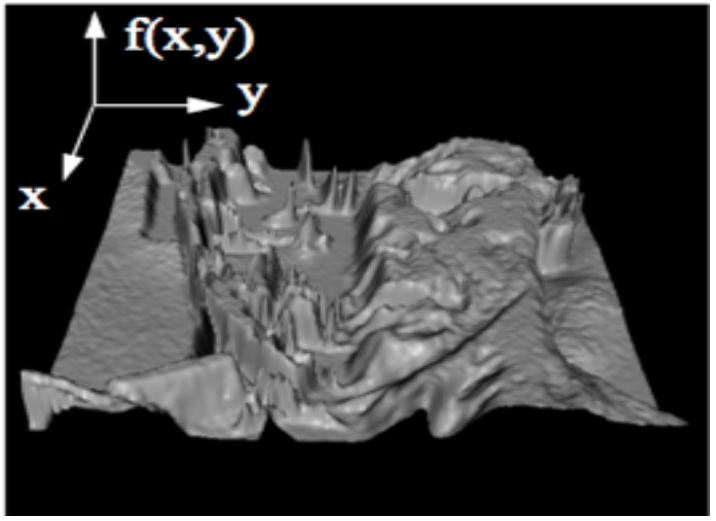
- A color image is just three functions pasted together. We can write this as a "vector-valued" function:

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

Images as Functions





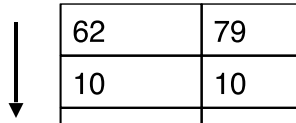


What is a digital image?

- 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 Δ 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?

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

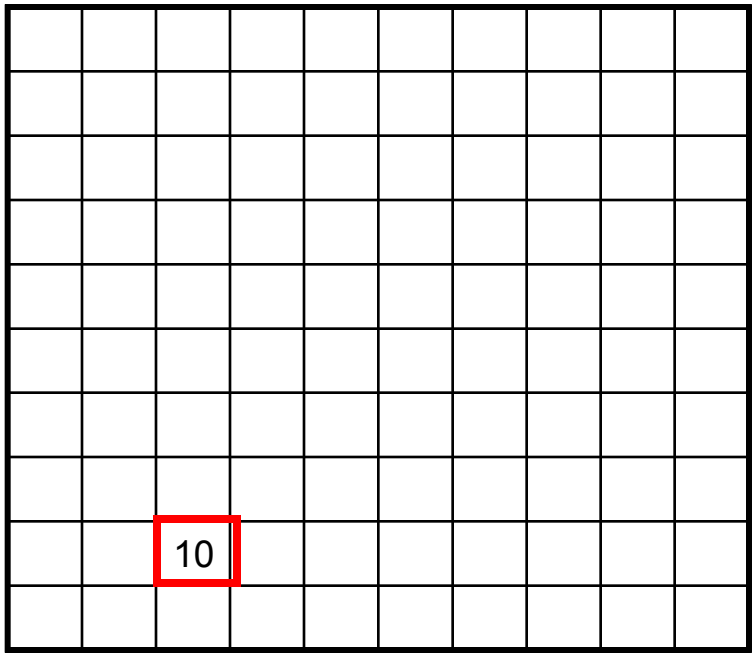
Mean Filtering

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

$F[x, y]$

Replace each pixel with the average of a kxk window around it

$G[x, y]$



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

$F[x, y]$

Replace each pixel with the average of a kxk window around it

$G[x, y]$

	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:

$$G = H \otimes F$$

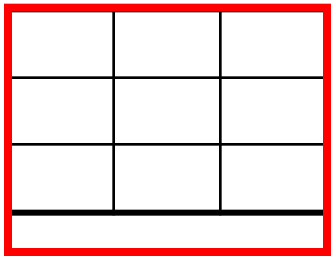
- H is called the "filter," "kernel," or "mask."
- The above allows negative filter indices. When you implement need to use: H[u+k,v+k] instead of H[u,v] OR be careful going out of bounds

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

$F[x, y]$



$H[u, v]$

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

$F[x, y]$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$H[u, v]$

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

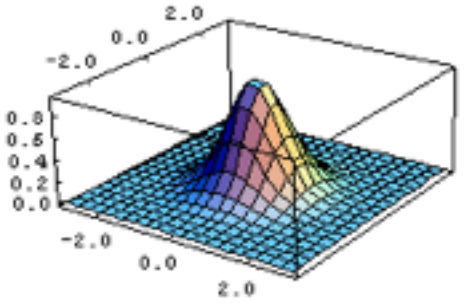
$\frac{1}{16}$

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 ?

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

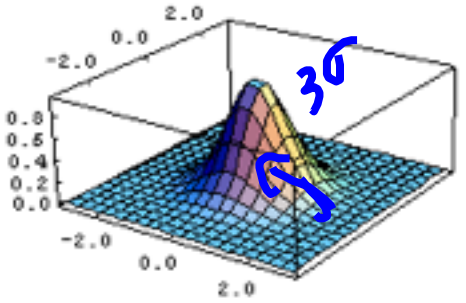
$H[u, v]$

① kernel size k×k

② Gaussian variance

- This kernel is an approximation of a Gaussian function:

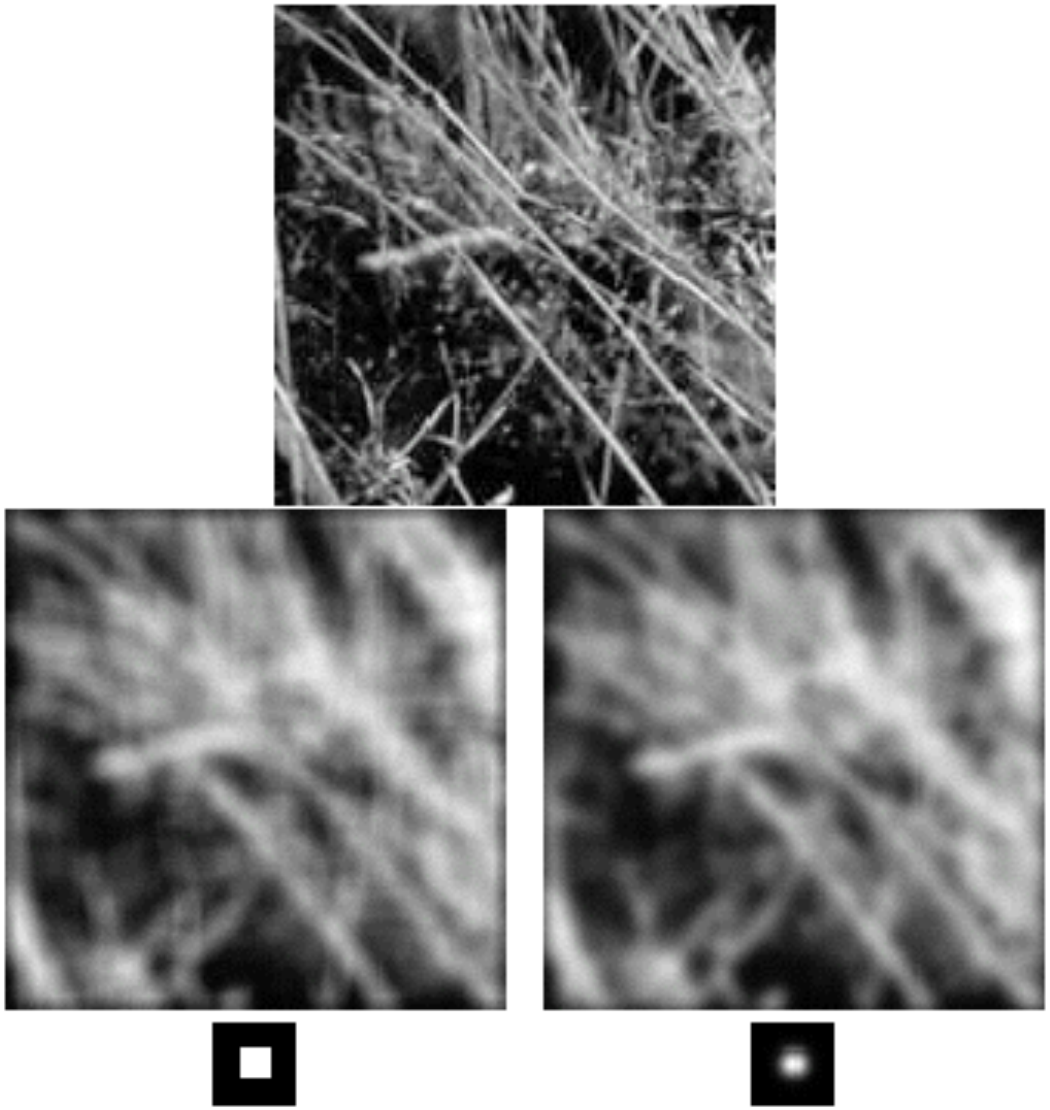
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



- What happens if you increase σ ?

Note that there are TWO sizes to think about

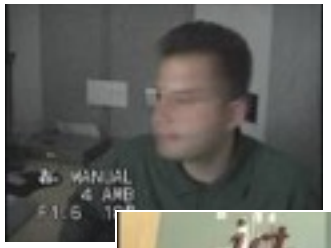
Mean vs. Gaussian Filtering



Capturing Light

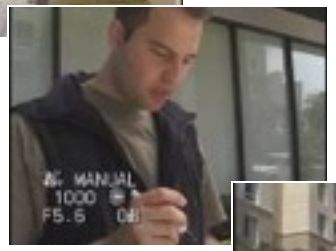
Let's use our techniques to overcome a camera limitation....

Problem: Dynamic Range

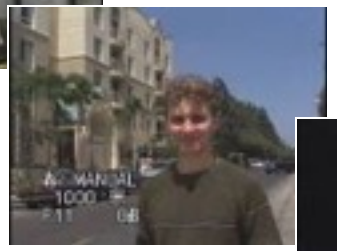


1

1500



25,000

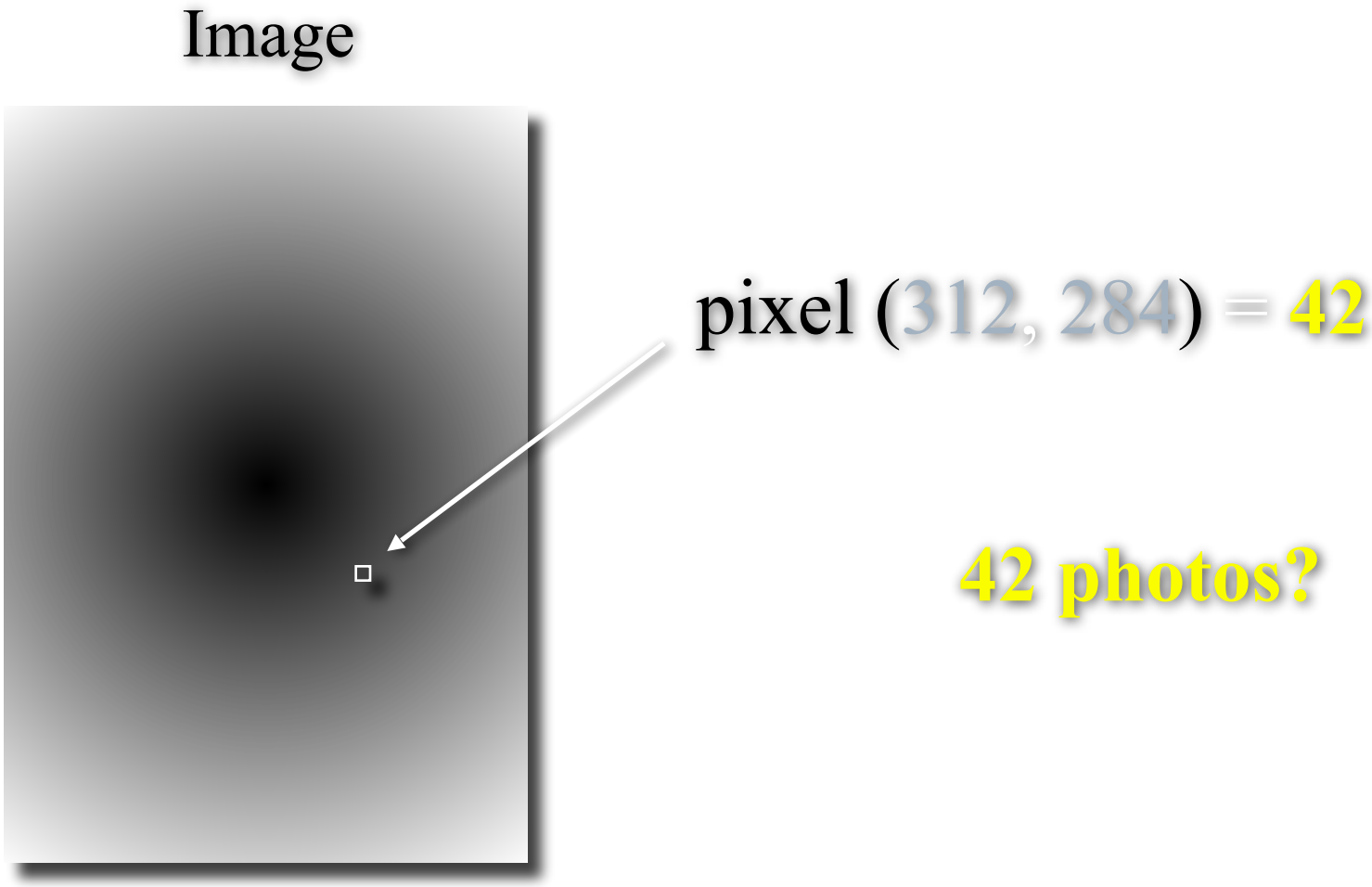


400,000

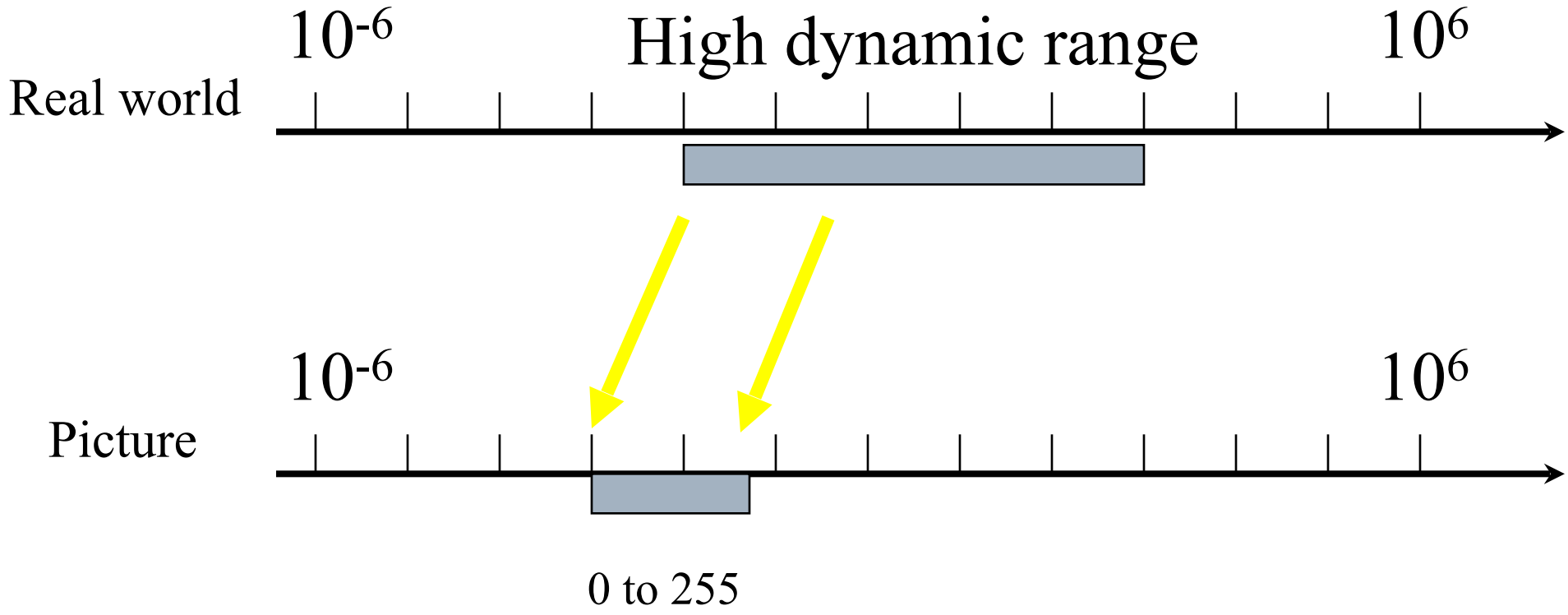
2,000,000,000

The real world is high dynamic range

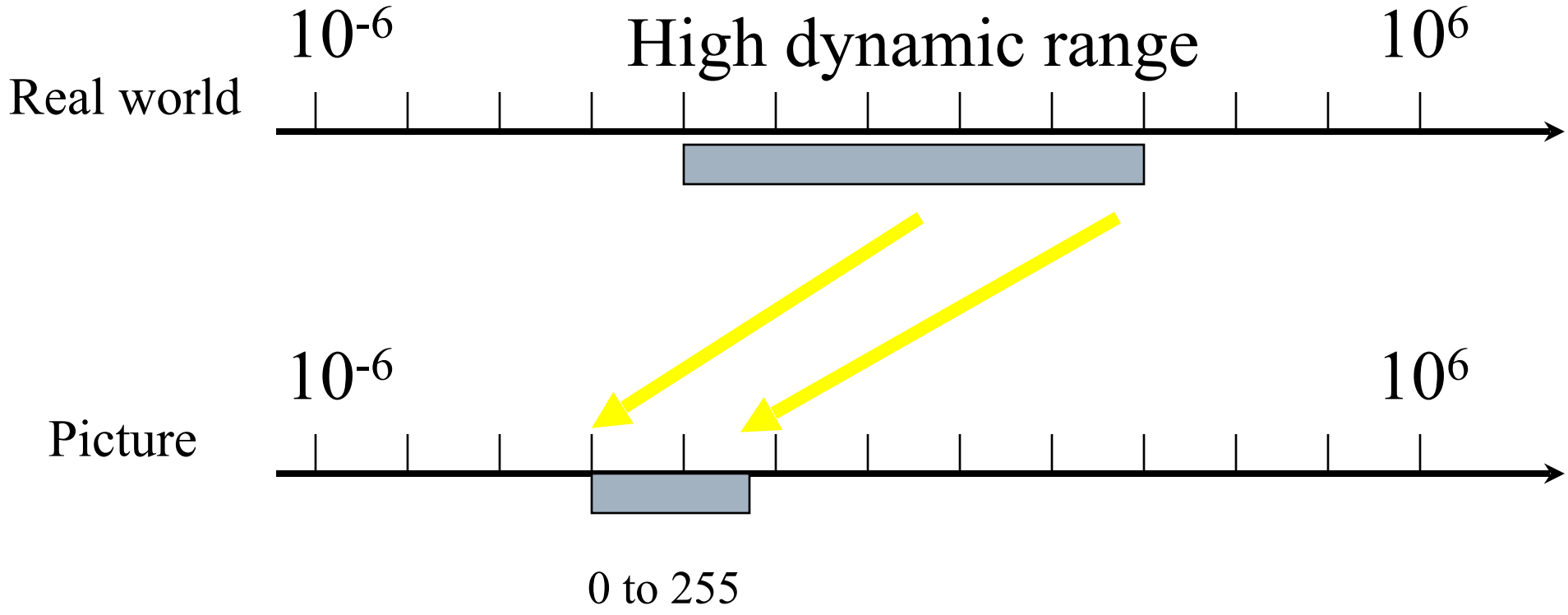
Is camera a photometer?



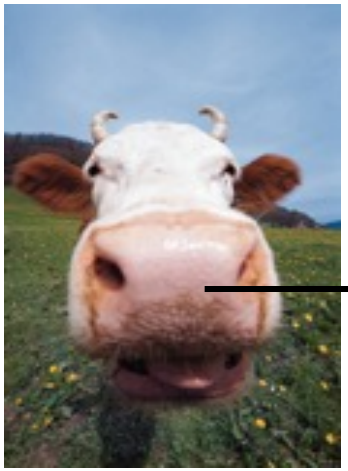
Long Exposure

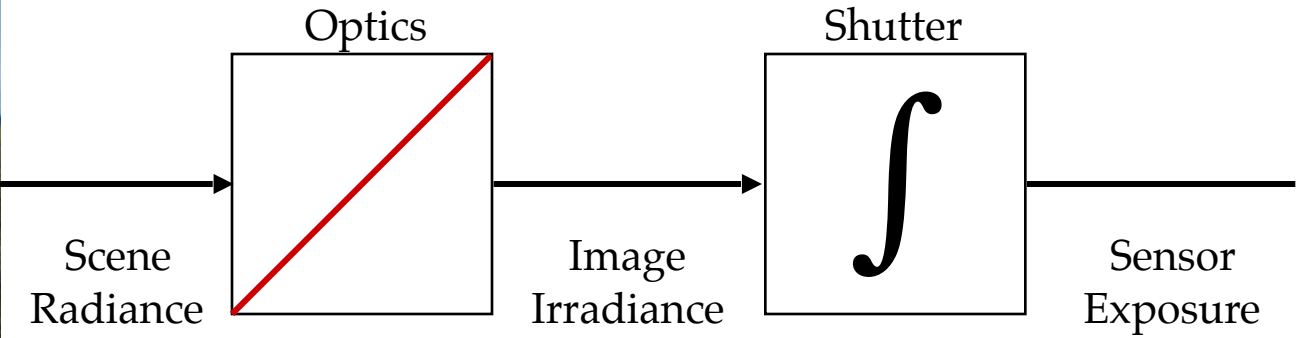


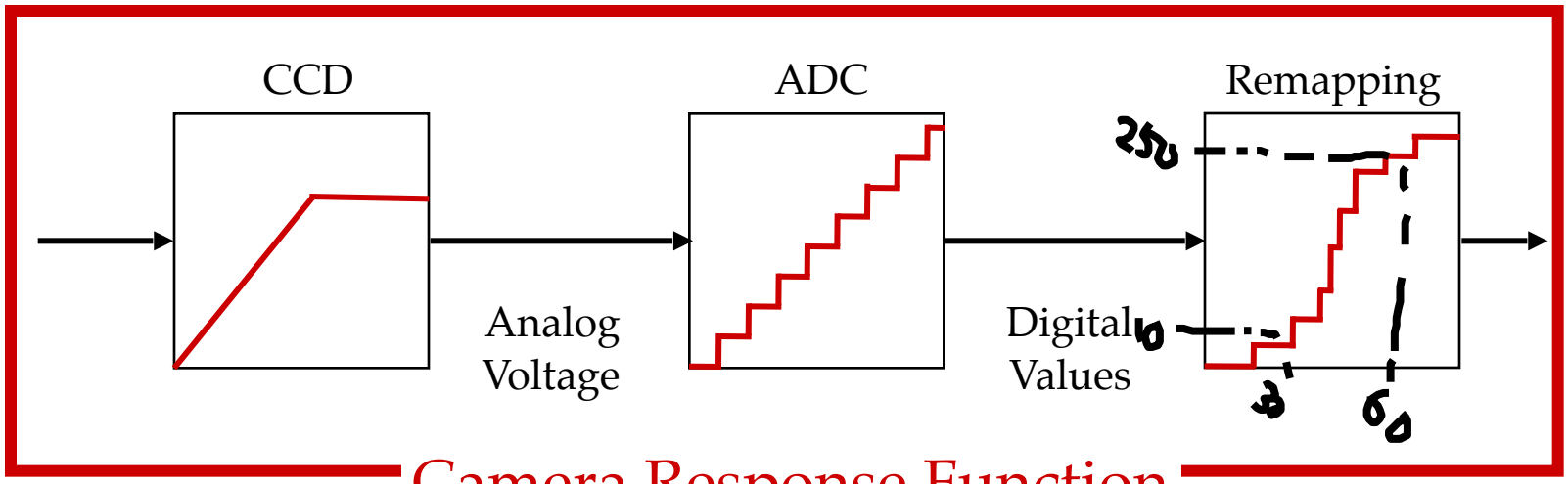
Short Exposure



Radiometric Pipeline

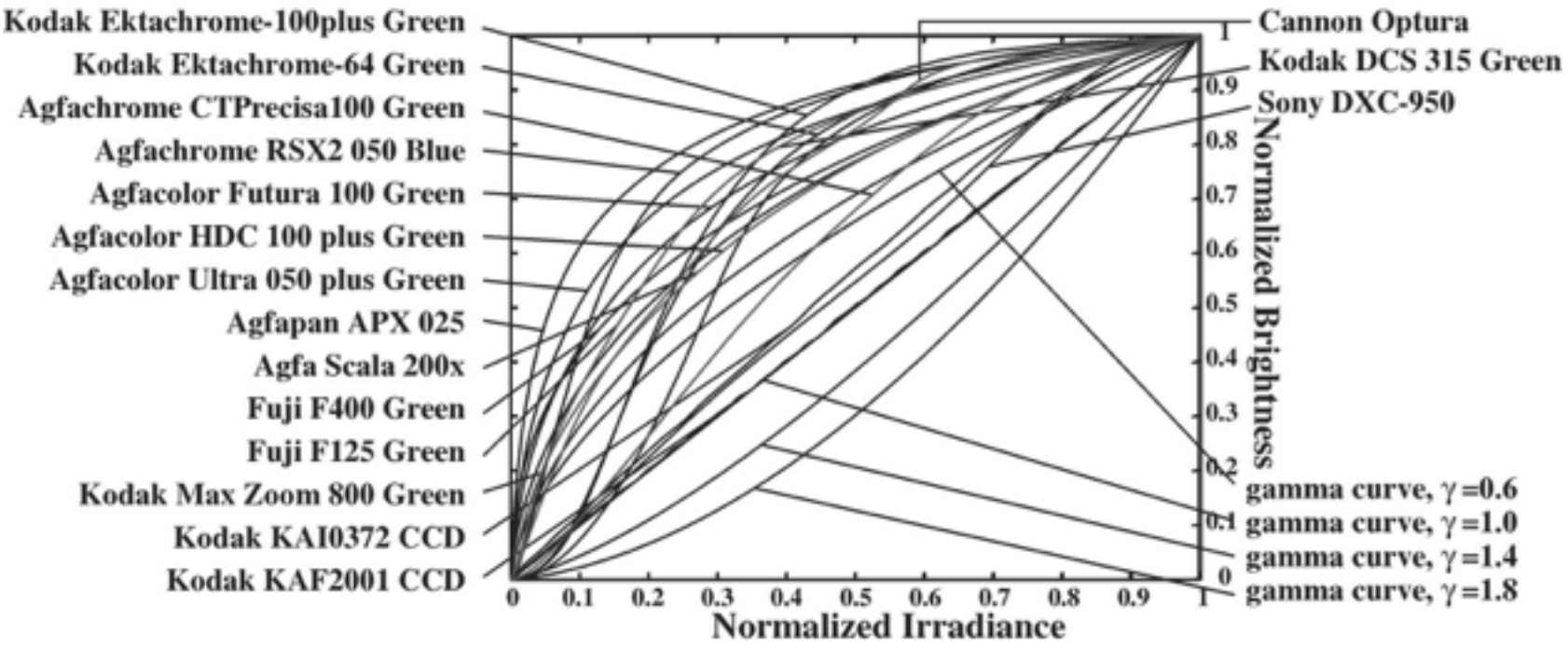






Camera Response Function

Camera Response Function



by M. Grossberg and S.K. Nayar

Dynamic Range

P. Debevec and J. Malik, SIGGRAPH 1997

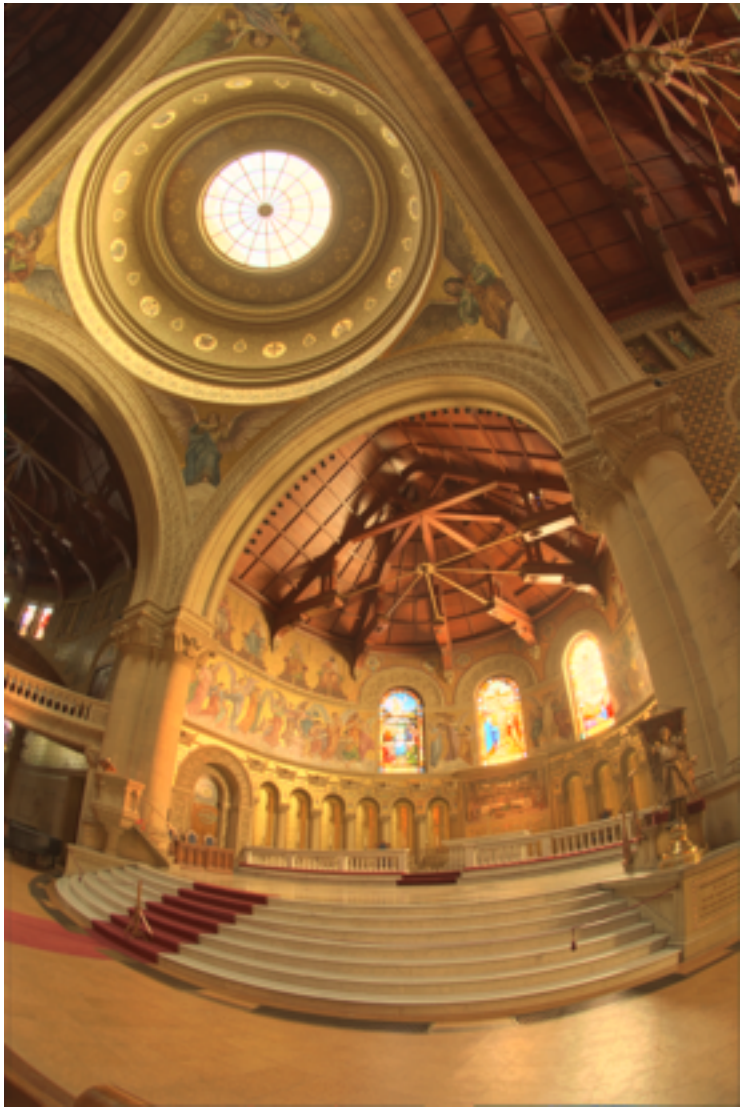
Extending the Dynamic Range

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!

High Dynamic Range Image

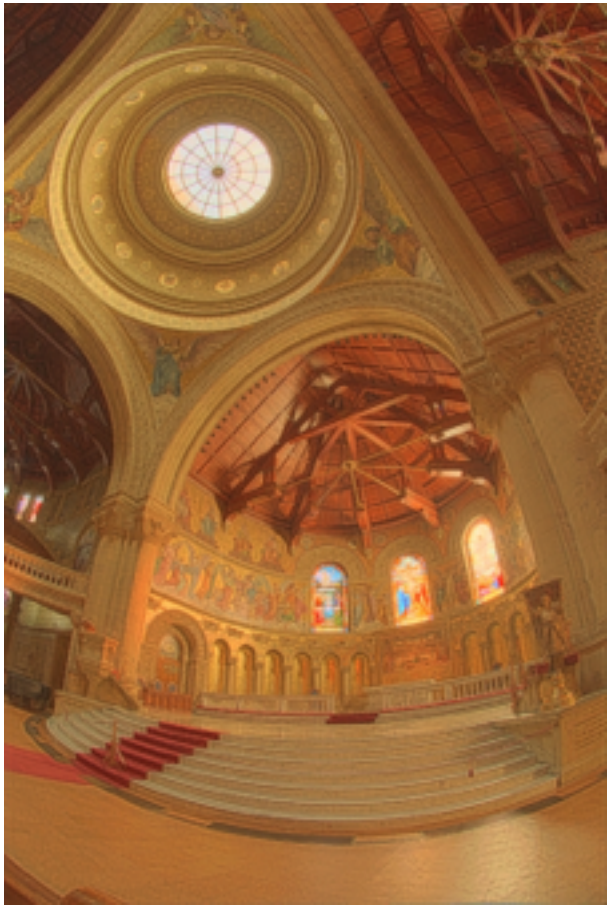


Tone Mapping

- HDR to LDR!

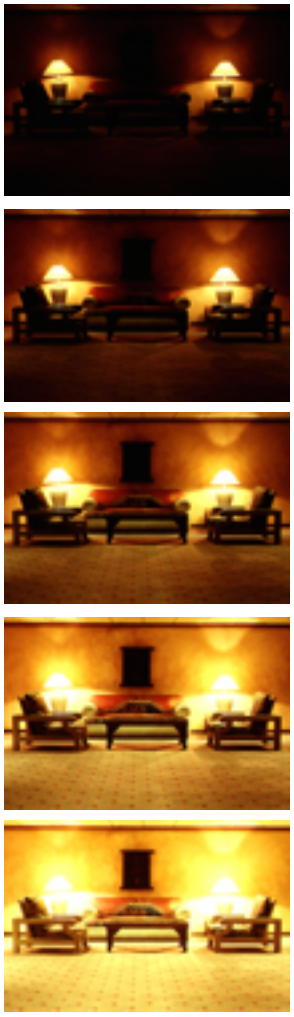
Ward et al. 98

Fattal et al. 02

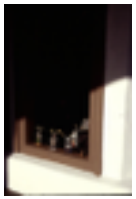


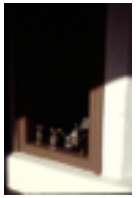
Tumblin et al. 99

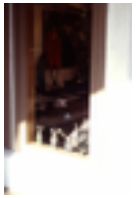
LDR to HDR to LDR

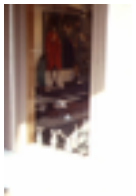


LDR to HDR to LDR









HDR Displays

BrightSide Technologies

Dolby

Recovering High Dynamic Range Radiance Maps from Photographs

Paul Debevec
Jitendra Malik

Computer Science Division
University of California at Berkeley

August 1997

Ways to vary exposure

- Shutter Speed (*)
- F/stop (aperture, iris)
- Neutral Density (ND) Filters

Shutter Speed

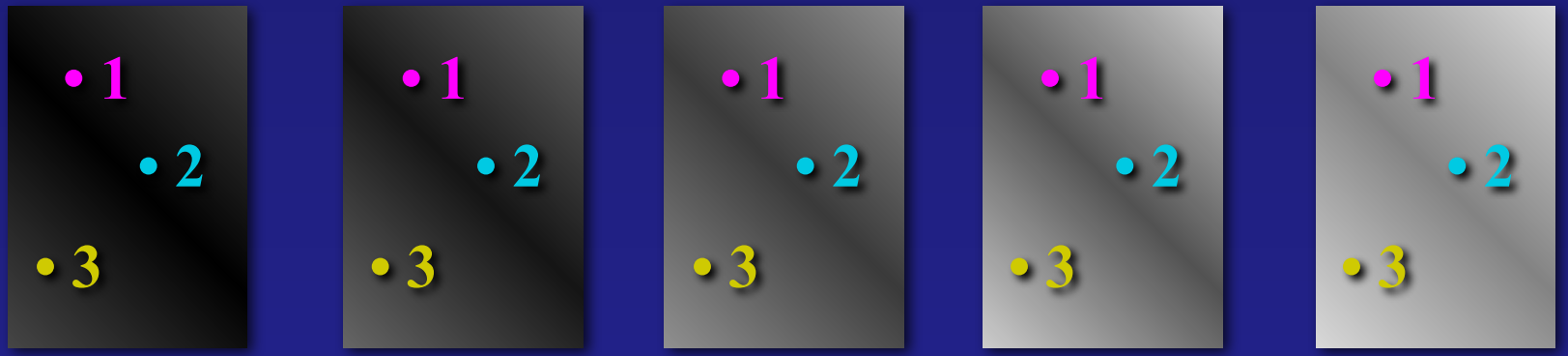
- **Ranges: Canon D30: 30 to 1/4,000 sec.**
- **Sony VX2000: ¼ to 1/10,000 sec.**
- **Pros:**
- **Directly varies the exposure**
- **Usually accurate and repeatable**
- **Issues:**
- **Noise in long exposures**

Shutter Speed

- **Note: shutter times usually obey a power series – each "stop" is a factor of 2**
- **¼, 1/8, 1/15, 1/30, 1/60, 1/125, 1/250, 1/500, 1/1000 sec**
- **Usually really is:**
- **¼, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512, 1/1024 sec**

The Algorithm

Image series



Δt =	Δt =	Δt =	Δt =	Δt =
1/64 sec	**1/16 sec**	**1/4 sec**	**1 sec**	**4 sec**

True scene brightness

Pixel Value Z = f(Exposure)

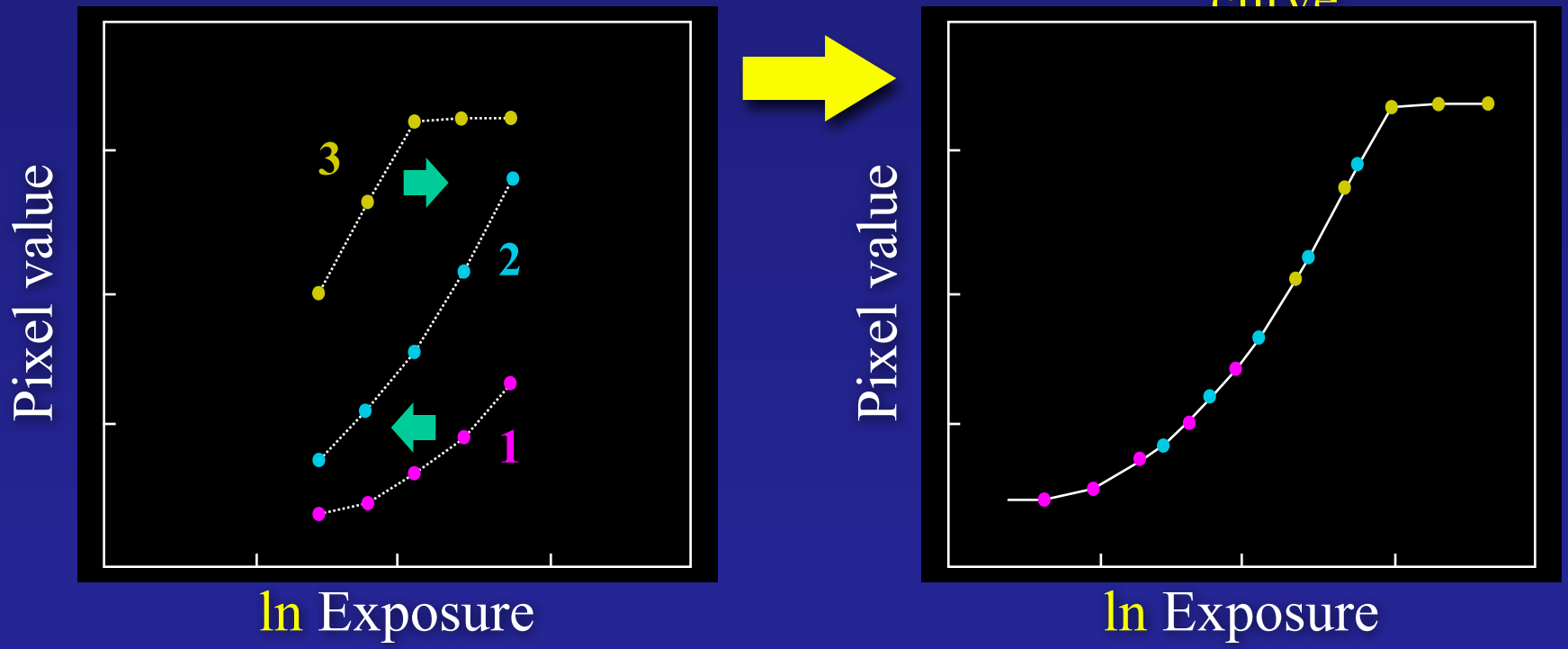
Exposure = Radiance × Δt

log Exposure = log Radiance + log Δt

Response Curve

Assuming unit radiance for each pixel

After adjusting radiances to obtain a smooth response curve



$$Z_{ij} = f(Radiance_i \times \Delta t_j)$$

$$f^{-1}(Z_{ij}) = Radiance_i \times \Delta t_j$$

$$\ln f^{-1}(Z_{ij}) = \ln Radiance_i + \ln \Delta t_j$$

$$g = \ln f^{-1}$$

$$g(Z_{ij}) = \ln Radiance_i + \ln \Delta t_j$$

N Number of pixels
i Which pixel we are on
P Number of pictures
j Which picture we are on
Z_{ij} Value of pixel i on image j
Δt_j Exposure time of image j
Ri True scene brightness at pixel i

Find the radiance for some pixels while estimating the inverse response function g

The Math

- Let *g(z)* be the *discrete* inverse response function
- For each pixel site *i* in each image *j*, want:

$$\ln Radiance_i + \ln \Delta t_j = g(Z_{ij})$$

- Solve the overdetermined linear system:

discretized

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\left[\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})\right]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} g''(z)^2$$

fitting term (data term)

smoothness term

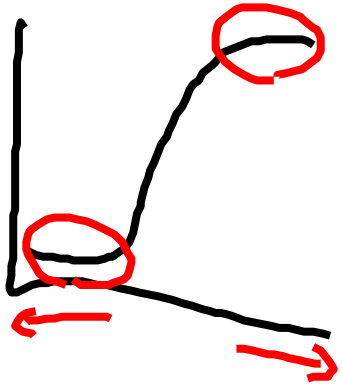
How do we make g smooth?

- Set the second derivative equal to zero.
- First derivative:
  - $g'(z) = g(z + t) - g(z - t)$
  - t is some delta (typically 1), but can be anything we can measure.
- Second derivative:
  - $g''(z) = g'(z + t) - g'(z - t)$
    $= g(z+2t) - g(z) - (g(z) - g(z - 2t))$
    $= g(z+2t) - 2g(z) + g(z - 2t)$
  - For convenience, let t=0.5:
    - $g''(z) = g'(z + 1) - 2g(z) + g'(z - 1)$

But not perfectly straight....

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\left[\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})\right]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} g''(z)^2$$

The response function is less smooth at the ends (low and high radiance, so weight the error and smoothness!



$$w(z) = \begin{cases} z - Z_{\min} & for \quad z \le \frac{1}{2}(Z_{\min} + Z_{\max}) \\ Z_{\max} - z & for \quad z > \frac{1}{2}(Z_{\min} + Z_{\max}) \end{cases}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\left[w(Z_{ij})\left[\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})\right]\right]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} \left[w(z)g''(z)\right]^2$$

How many samples do we need?

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\left[w(Z_{ij})\left[\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})\right]\right]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}\left[w(z)g''(z)\right]^2$$

- **g** requires Z_{max} - Z_{min} values (256 for us!)
- P different pictures (i.e., shutter speeds)
- The number of pixel samples Z_{ij} (i.e., NP) should be larger than the number of unknowns:

$$NP > N + 256$$

$$N(P-1) > 256$$

N	Number of pixels
i	Which pixel we are on
P	Number of pictures
j	Which picture we are on
Z_{ij}	Value of pixel i on image j
Δt_j	Exposure time of image j

One more gotcha....

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\left[w(Z_{ij})\left[\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})\right]\right]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}\left[w(z)g''(z)\right]^2$$

- The Radiance values and response function can be scaled by the same amount and still result in the same error, (there's a scale ambiguity)
- Handle this by fixing the middle value to be 1!

$$g\left(\frac{Z_{\min} + Z_{\max}}{2}\right) = \log 1 = 0$$

So we know....

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\left[w(Z_{ij})\left[\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})\right]\right]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}\left[w(z)g''(z)\right]^2$$

- Need N(P-1) > 256
- g(128) = 0

How can we solve for *Radiance* and ***g***?

Least Squares

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\Big[w(Z_{ij})\big[\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})\big]\Big]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}\big[w(z)g''(z)\big]^2$$

fitting term (data term)

smoothness term

- We have:
  - NP equations in data term.
  - $Z_{max} - Z_{min} - 2$ equations in smoothness term
  - 1 extra equation: g(128) = 0
- Construct and solve the system:
  Ax = b

N Number of pixels
i Which pixel we are on
P Number of pictures
j Which picture we are on
Z_{ij} Value of pixel i on image j
Δt_j Exposure time of image j

What are our unknowns?

- Our unknowns are the 256 values for $\boldsymbol{g}$ and the N values for *Radiance*.
- Let:
  $x = \begin{bmatrix} \boldsymbol{g} \\ Radiance \end{bmatrix}$
- g is 256 x 1.
- Radiance is N x 1.
- So x is (256 + N) x 1 (i.e., a column vector)

N	Number of pixels
i	Which pixel we are on
P	Number of pictures
j	Which picture we are on
Z_{ij}	Value of pixel i on image j
Δt_j	Exposure time of image j
Ri	True scene brightness at pixel i

Least Squares: Ax = b

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\Big[w(Z_{ij})\big[\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})\big]\Big]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}\big[w(z)g''(z)\big]^2$$

fitting term (data term)

smoothness term

- We have:
  - NP equations in data term.
  - $Z_{max} - Z_{min} - 2$ equations in smoothness term
  - 1 extra equation: $g(128) = 0$
- So:
  - x is (256 + N) x 1
  - A is (NP + 256 - 2 + 1) x (256 + N)
  - b is (NP + 256 - 2 + 1) x 1

N	Number of pixels
i	Which pixel we are on
P	Number of pictures
j	Which picture we are on
Z_{ij}	Value of pixel i on image j
Δt_j	Exposure time of image j

What are our equations?

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\Big[w(Z_{ij})\Big[\ln Radiance_i + \ln\Delta t_j - g(Z_{ij})\Big]\Big]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}\big[w(z)g''(z)\big]^2$$

fitting term (data term)

smoothness term

- Let's look at a single data term (i.e., k = iN+j)

  $$w(Z_{ij})\Big[\ln Radiance_i + \ln\Delta t_j - g(Z_{ij})\Big]$$

  - A[k, Zij] = -w(Zij)
  - A[k, 256 + j] = w(Zij)
  - b[k] = -w(Zij) $\ln\Delta t_j$
- Fills the first NP rows of A!

N Number of pixels
i Which pixel we are on
P Number of pictures
j Which picture we are on
Z_{ij} Value of pixel i on image j
Δt_j Exposure time of image j

What are our equations?

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\Big[w(Z_{ij})\big[\underbrace{\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})}_{\text{fitting term (data term)}}\big]\Big]^2 + \lambda \underbrace{\sum_{z=Z_{min}+1}^{Z_{max}-1}\big[w(z)g''(z)\big]^2}_{\text{smoothness term}}$$

- How about the smoothing terms?

$$\lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}\big[w(z)g''(z)\big]^2$$

- k = Z_{min}-1 ... Z_{max}-1
  - A[k + NP, k-1] = λw(k-1)
  - A[k + NP, k] = -λ2w(k)
  - A[k + NP, k+1] =λw(k+1)
  - b[k + NP] = 0
- Fills the next Z_{max}- Z_{min} -2 rows of A!

N	Number of pixels
i	Which pixel we are on
P	Number of pictures
j	Which picture we are on
Z_{ij}	Value of pixel i on image j
Δt_j	Exposure time of image j

What are our equations?

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\Big[w(Z_{ij})\big[\underbrace{\ln Radiance_i + \ln \Delta t_j - g(Z_{ij})}_{\text{fitting term (data term)}}\big]\Big]^2 + \lambda \underbrace{\sum_{z=Z_{min}+1}^{Z_{max}-1}\big[w(z)g''(z)\big]^2}_{\text{smoothness term}}$$

- And finally:
  - 1 extra equation: g(128) = 0
  - A[NP + 256 - 2] =0

N	Number of pixels
i	Which pixel we are on
P	Number of pictures
j	Which picture we are on
Z_{ij}	Value of pixel i on image j
Δt_j	Exposure time of image j

Solving for Ax=b

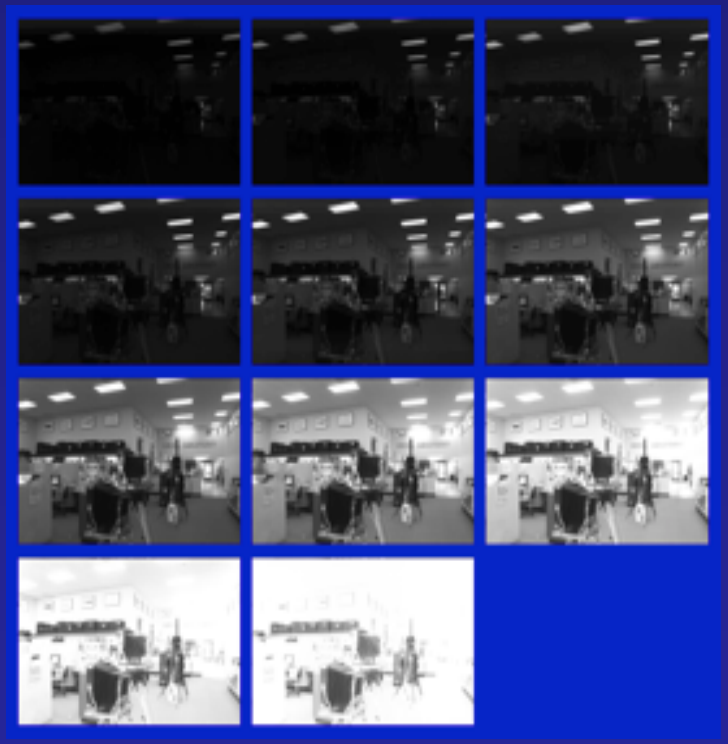
- Classic linear algebra!
- Typical approach assumes positive semidefinite gram matrix:
  - $Ax = b$
  - $A^TAx = A^Tb$
  - $(A^TA)^{-1}A^TAx = (A^TA)^{-1}A^Tb$
  - $x = (A^TA)^{-1}A^Tb$
- Hard part: inverting A^TA!
  - Sometimes not full rank.
  - Sometimes REALLY big!

Matlab Code

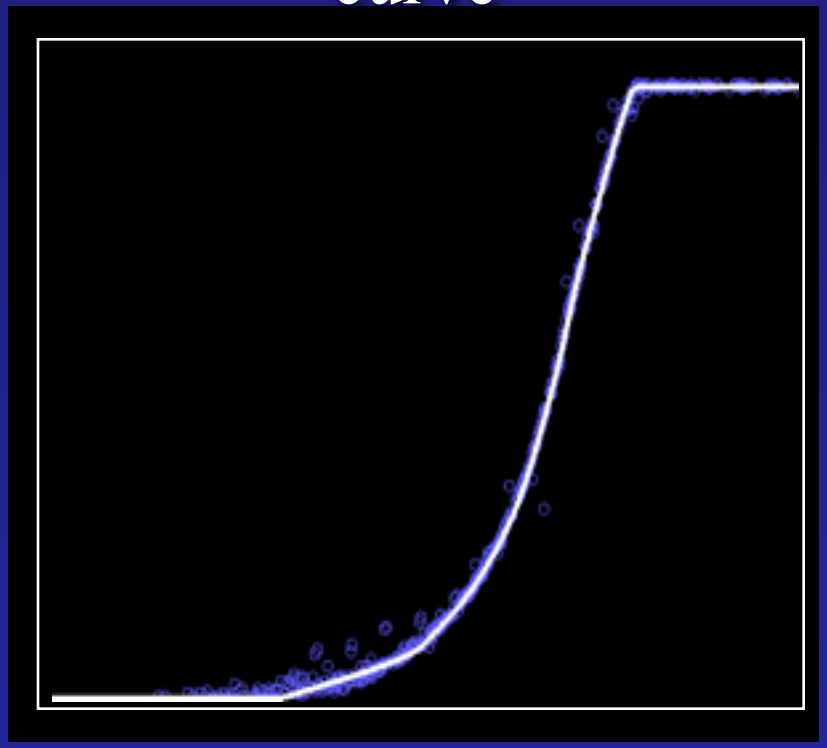
```
function [g,lE]=gsolve(Z,B,l,w)

n = 256;
A = zeros(size(Z,1)*size(Z,2)+n+1,n+size(Z,1));
b = zeros(size(A,1),1);

k = 1;                  %% Include the data-fitting equations
for i=1:size(Z,1)
  for j=1:size(Z,2)
    wij = w(Z(i,j)+1);
    A(k,Z(i,j)+1) = wij; A(k,n+i) = -wij; b(k,1) = wij * B(i,j);
    k=k+1;
  end
end

A(k,129) = 1;           %% Fix the curve by setting its middle value to
k=k+1;

for i=1:n-2             %% Include the smoothness equations
  A(k,i)=l*w(i+1); A(k,i+1)=-2*l*w(i+1); A(k,i+2)=l*w(i+1);
  k=k+1;
end

x = A\b;                %% Solve the system using SVD

g = x(1:n);
lE = x(n+1:size(x,1));
```

Results: Digital Camera

Kodak DCS460
1/30 to 30 sec



Recovered response curve

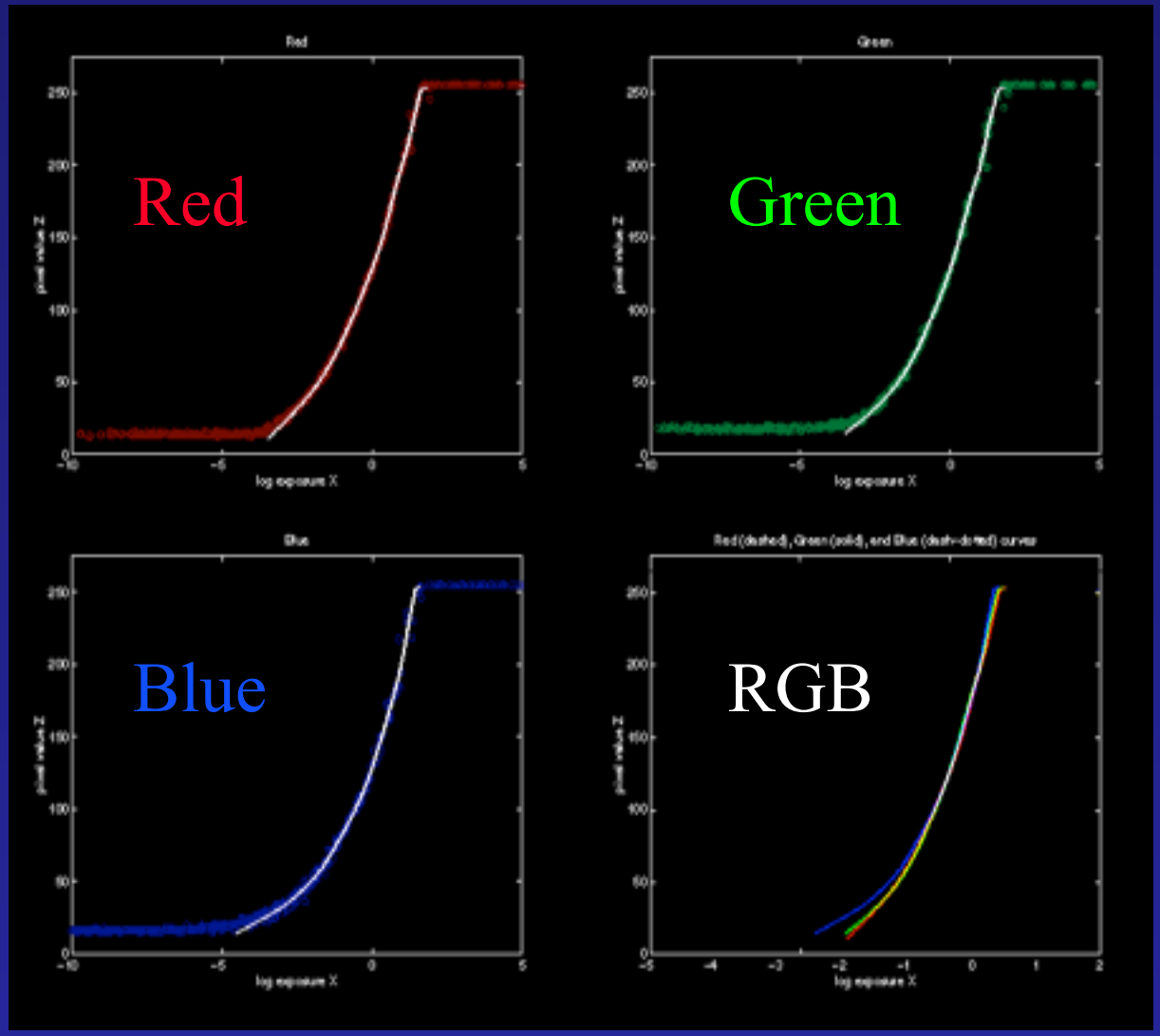


Reconstructed radiance map

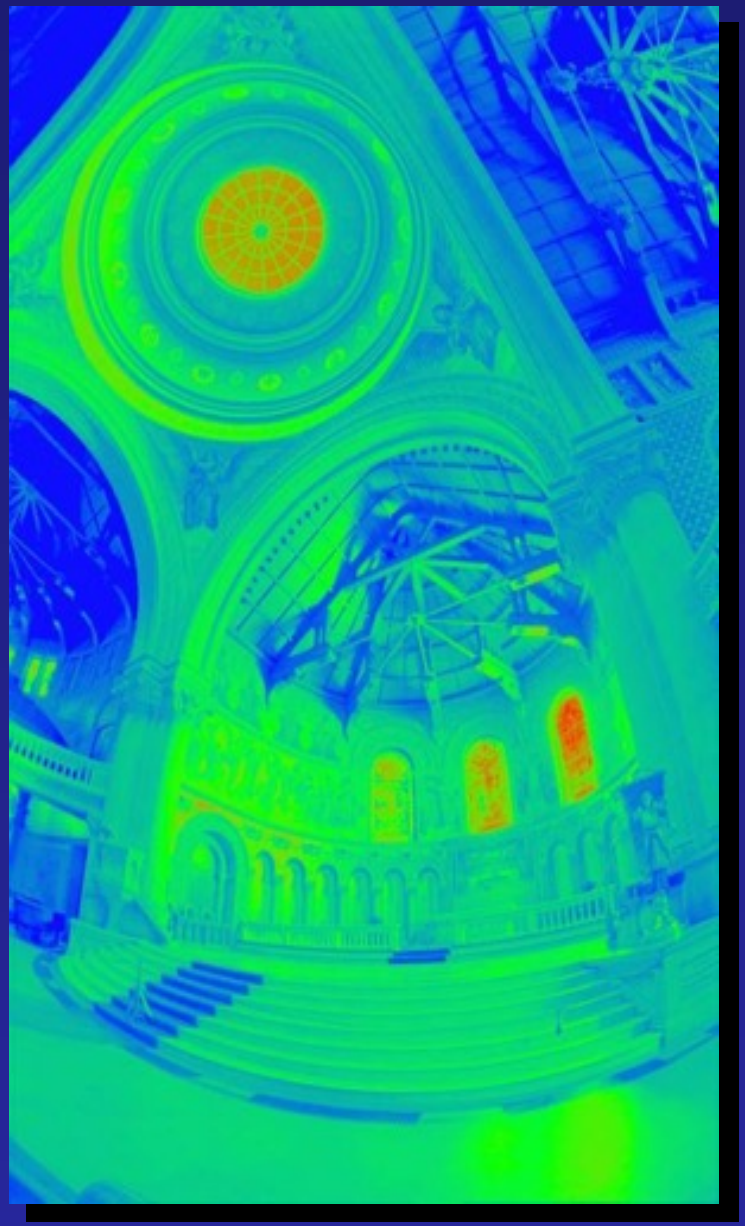
Results: Color Film

- Kodak Gold ASA 100, PhotoCD

Recovered Response Curves

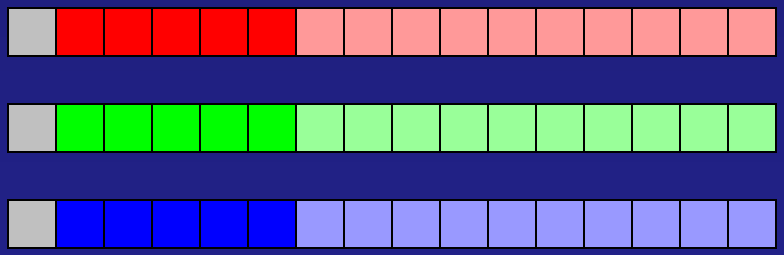


Result: The Radiance Map



ILM's OpenEXR (.exr)

- 6 bytes per pixel, 2 for each channel, compressed



sign exponent mantissa

- Several lossless compression options, 2:1 typical
- Compatible with the "half" datatype in NVidia's Cg
- Supported natively on GeForce FX and Quadro FX

- Available at http://www.openexr.net/

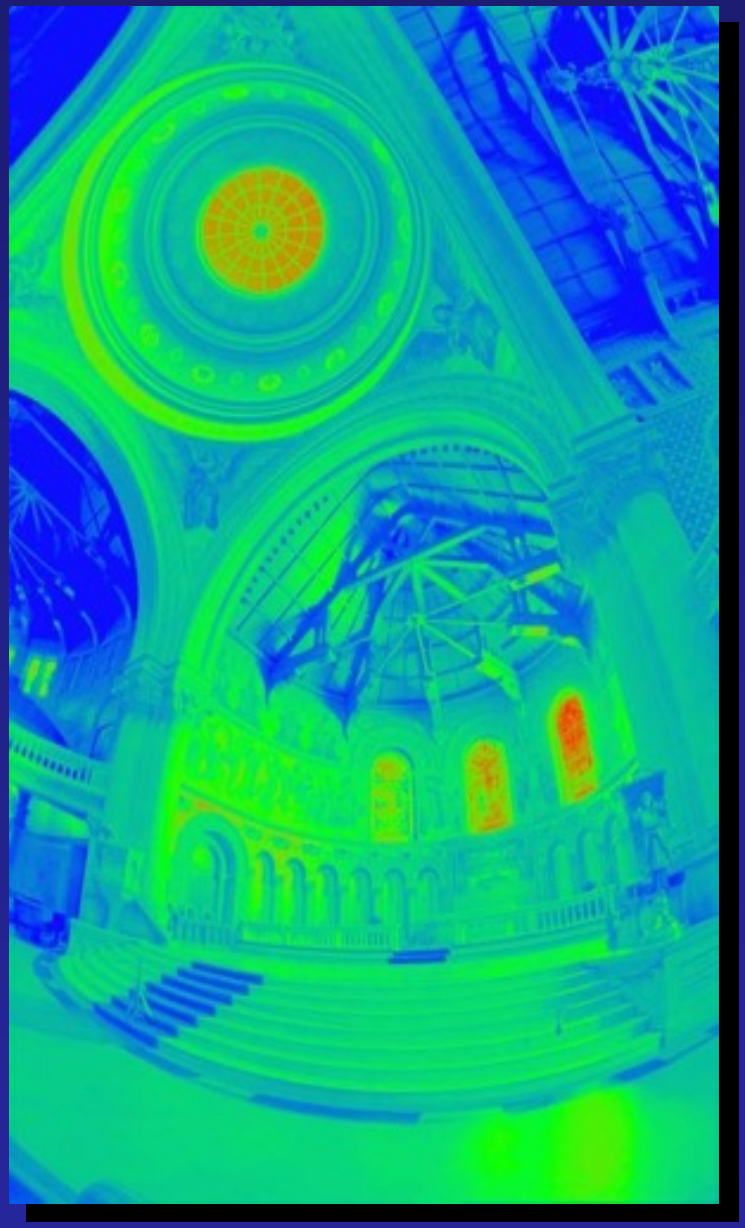
Review

- Load P images taken with exposure times.
- Compute weights vector w(z)
- For each channel R, G, B:
  - Estimate response function g
  - Compute log radiance map for each pixel i:

$$\ln E_i = \frac{\sum_{j=1}^{P} w(Z_{ij})(g(Z_{ij}) - \ln \Delta t_j)}{\sum_{j=1}^{P} w(Z_{ij})}$$

  - Take exp to get Ei
- Combine Ei from RGB images and save to EXR.

Now What?

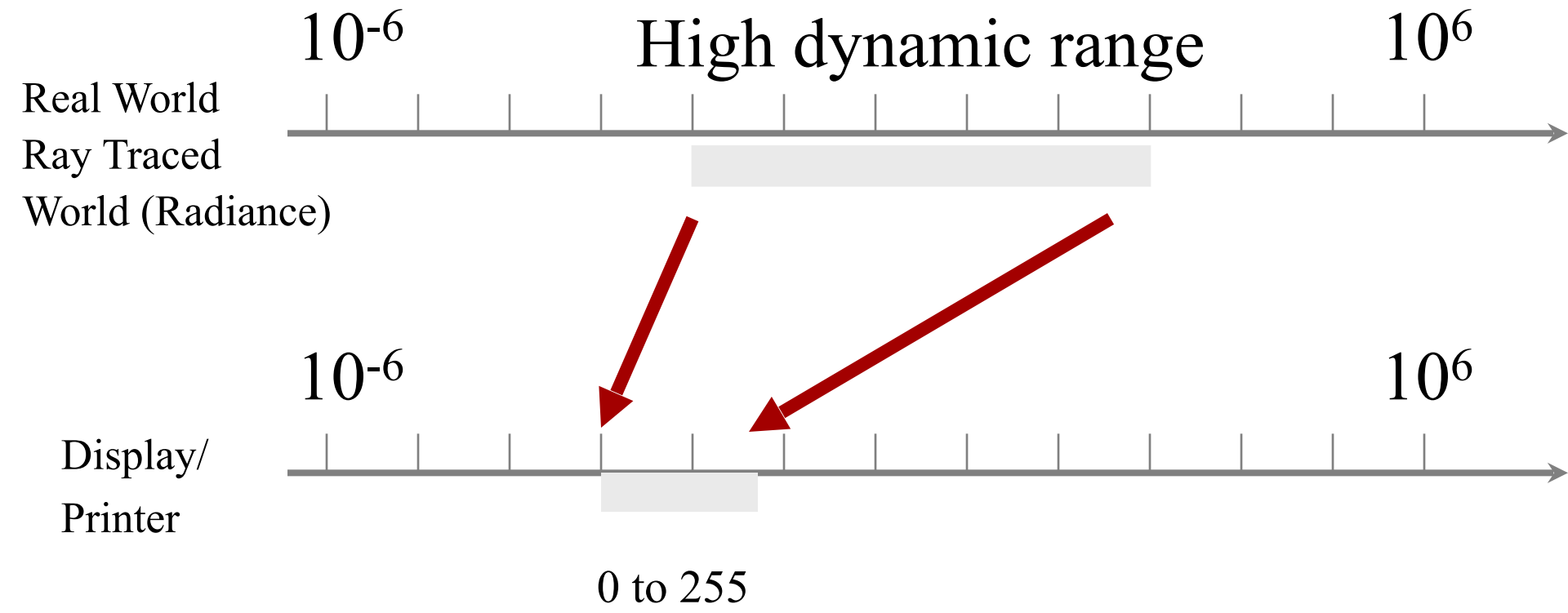


Tone Mapping

Tone Mapping

- How can we do this?

  Linear scaling?, thresholding? Suggestions?

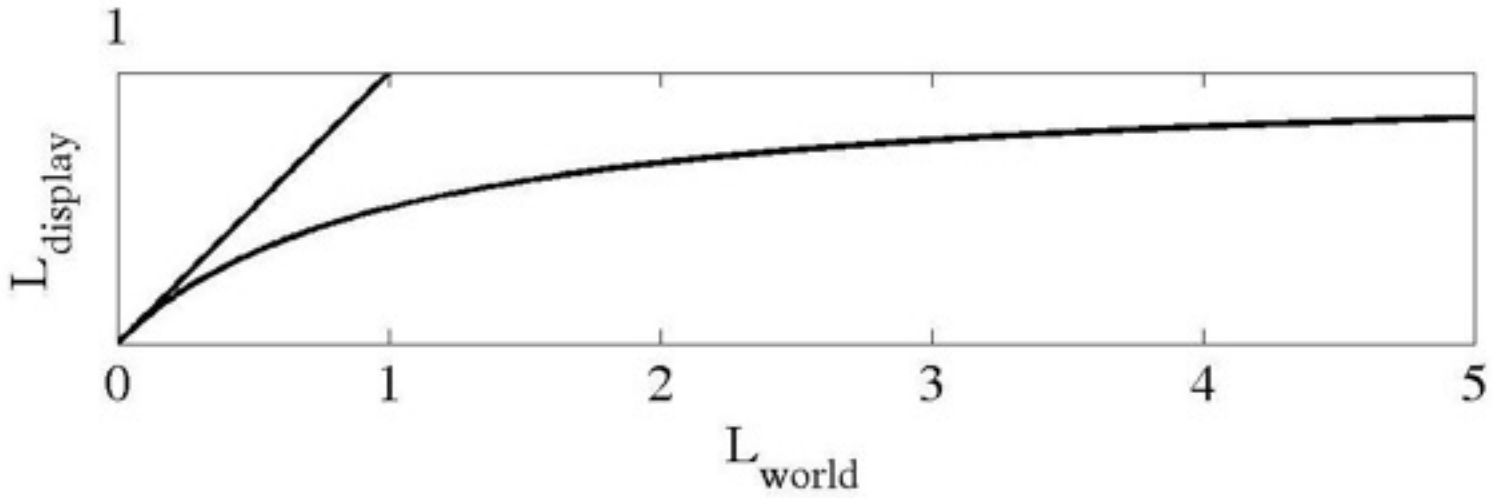


Simple Global Operator

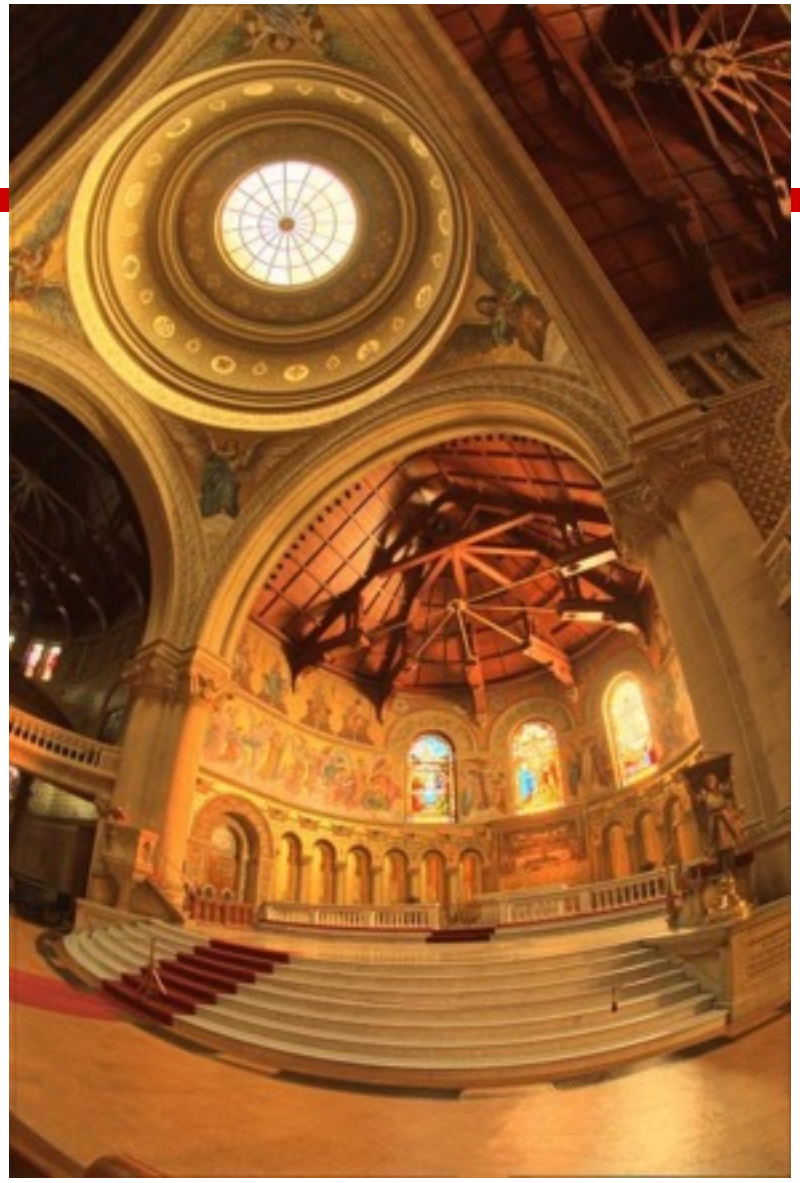
- Compression curve needs to
  - Bring everything within range
  - Leave dark areas alone
- In other words
  - Asymptote at 255
  - Derivative of 1 at 0

Global Operator (Reinhart et al)

$$L_{display} = \frac{L_{world}}{1 + L_{world}}$$



Global Operator Results



Reinhart Operator

Darkest 0.1% scaled to display device

All you need to understand is sections 3.3 and 6 of the paper

Fast Bilateral Filtering for the Display of High-Dynamic-Range Images

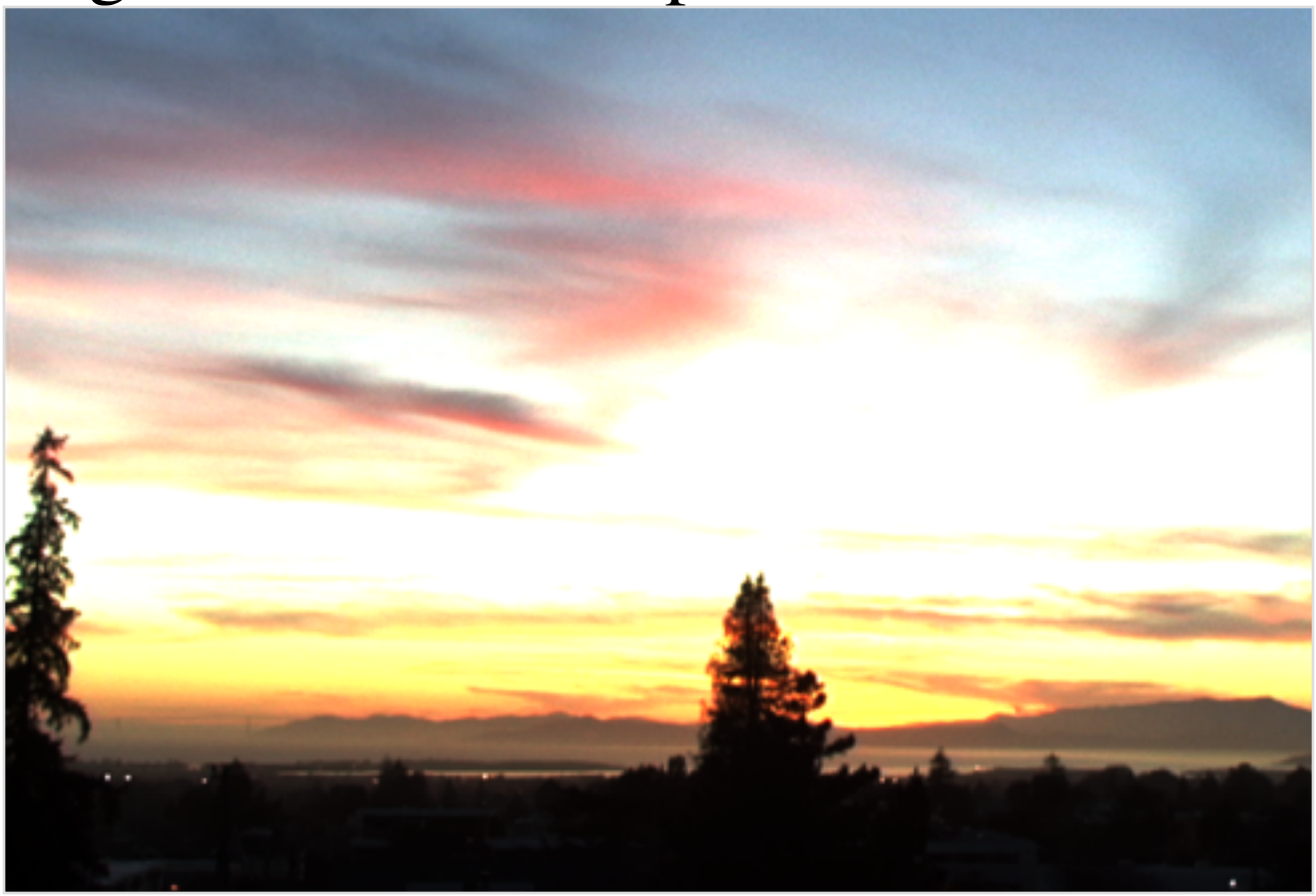
Frédo Durand & Julie Dorsey
Laboratory for Computer Science
Massachusetts Institute of Technology

Contributions

- Contrast reduction for HDR images
  - Local tone mapping
  - Preserves details
  - No halo
  - Fast
- Edge-preserving filter

A typical photo

- Sun is overexposed
- Foreground is underexposed



Gamma compression

- $X \rightarrow X^{\gamma}$
- Colors are washed-out

Input

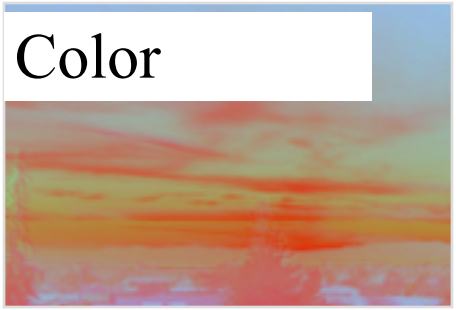
Gamma

Gamma compression on intensity

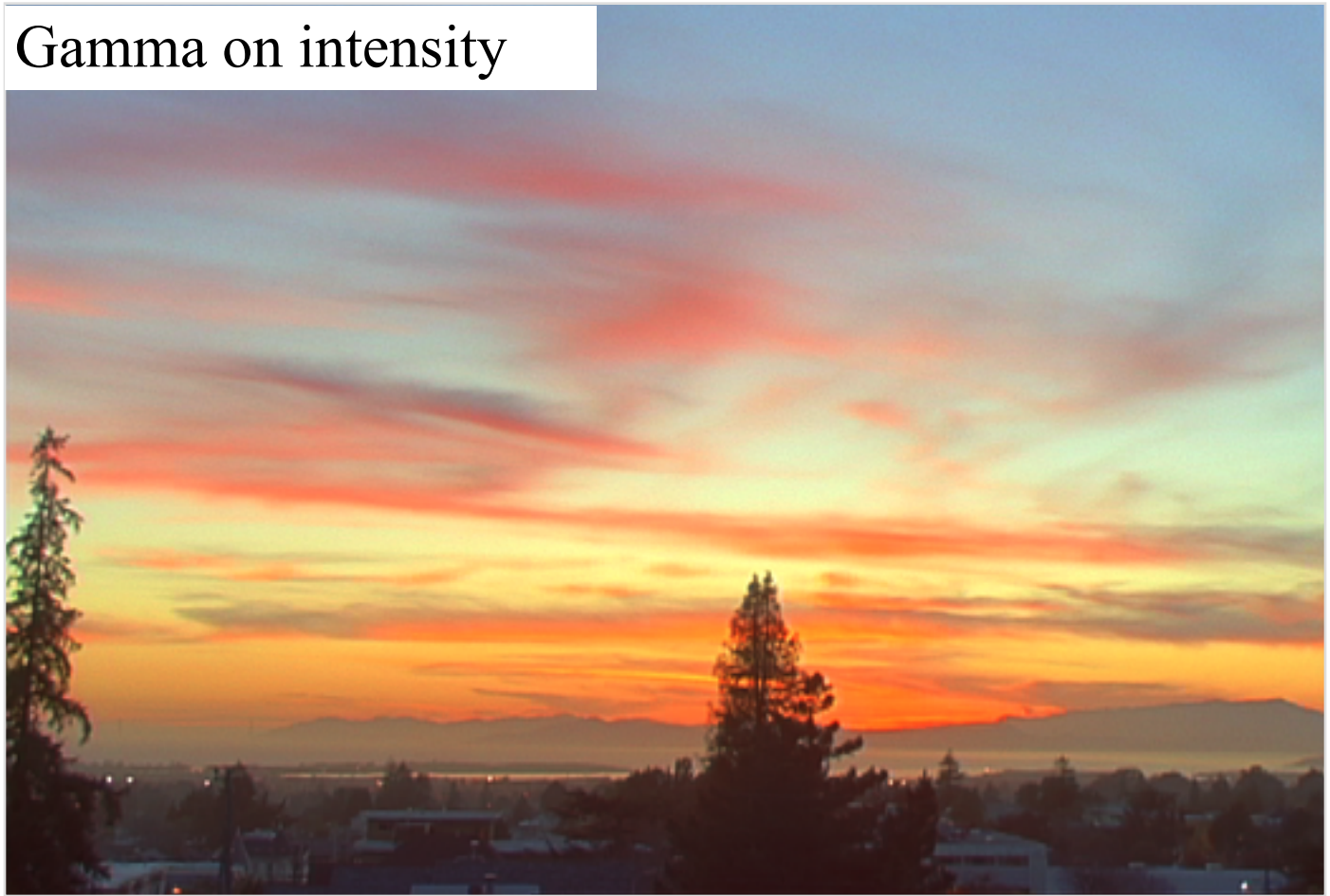
- Colors are OK, but details (intensity high-frequency) are blurred

Intensity

Color

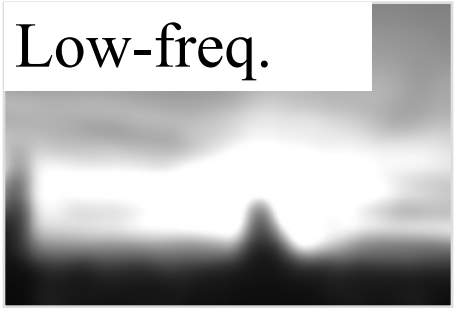


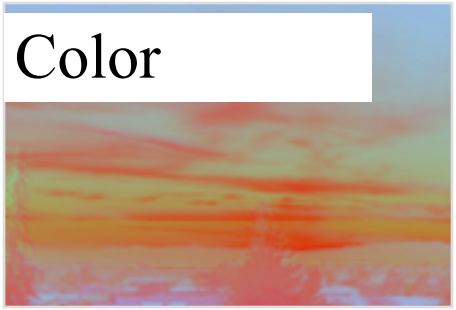
Gamma on intensity



Chiu et al. 1993

- Reduce contrast of low-frequencies
- Keep high frequencies

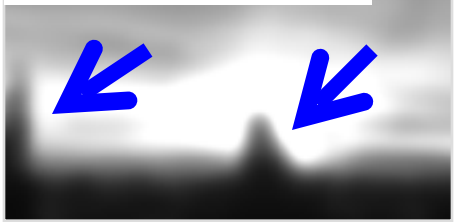




The halo nightmare

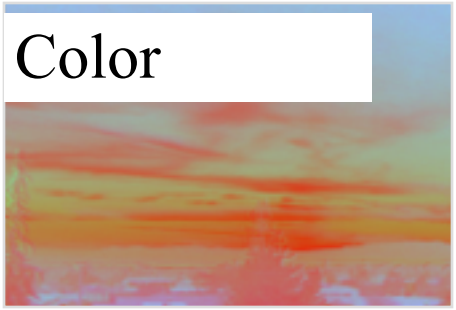
- For strong edges
- Because they contain high frequency

Low-freq.

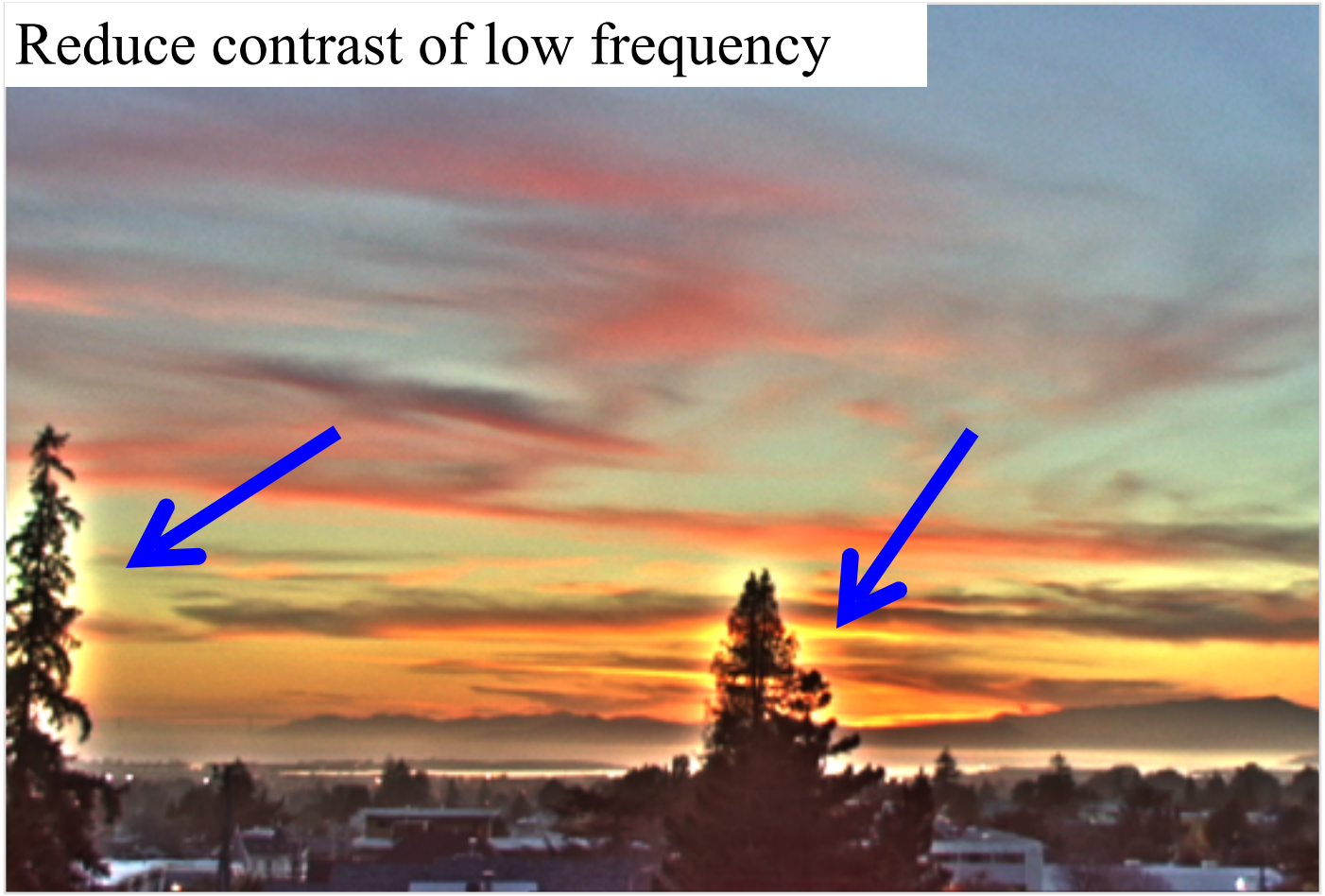


High-freq.

Color



Reduce contrast of low frequency



Durand & Dorsey

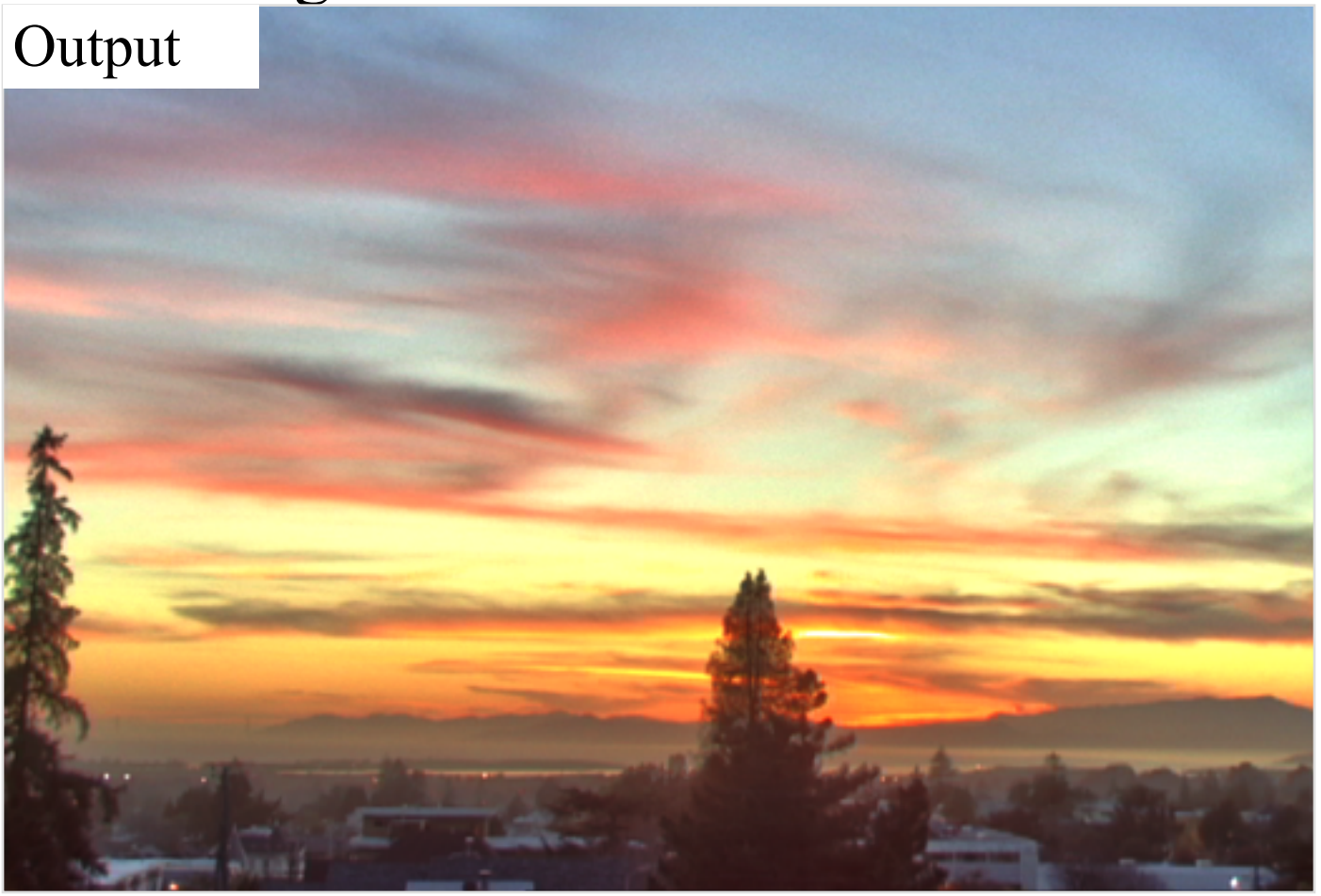
- Do not blur across edges
- Non-linear filtering

Large-scale

Detail

Color

Output



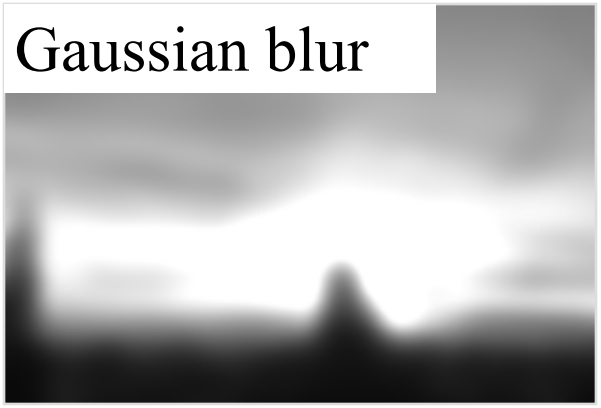
General Approach

- Split Image into *color* and *intensity*
- Separate the detail from intensity image (call this a 'base' image)
- Reduce contrast on the base image.
- Add detail back in.
- Add color back in.

How do we separate detail?

Edge-preserving filtering

- Blur, but not across edges

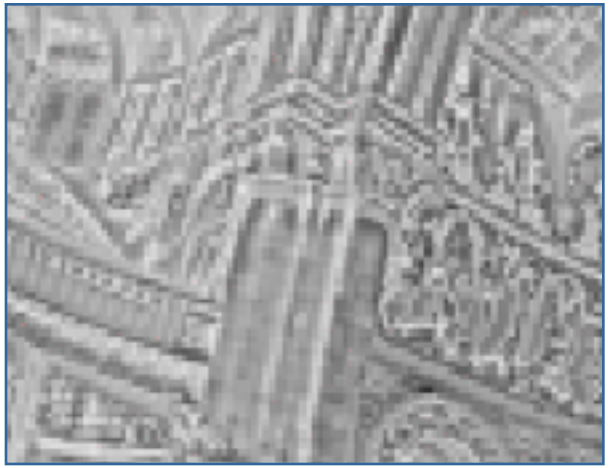


- Anisotropic diffusion [Perona & Malik 90]
  - Blurring as heat flow
  - LCIS [Tumblin & Turk]
- Bilateral filtering [Tomasi & Manduci, 98]

Edge-preserving filtering & LCIS

- [Tumblin & Turk 1999]
- Multiscale decomposition using LCIS (anisotropic diffusion)

Simplified
(at multiple scales)
Compressed

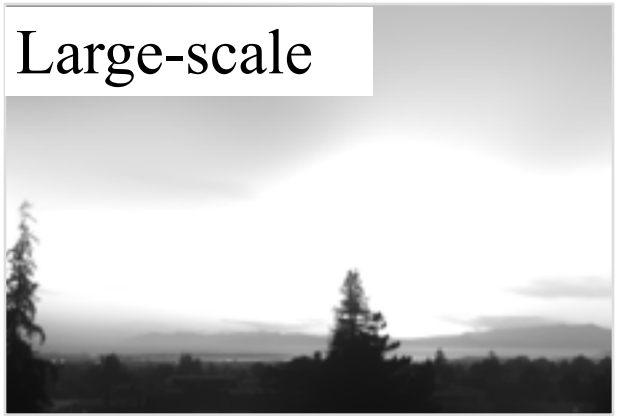


Details

Output

Comparison with Durand & Dorsey

- We use only 2 scales
- Can be seen as illumination and reflectance
- Different edge-preserving filter from LCIS



Compressed

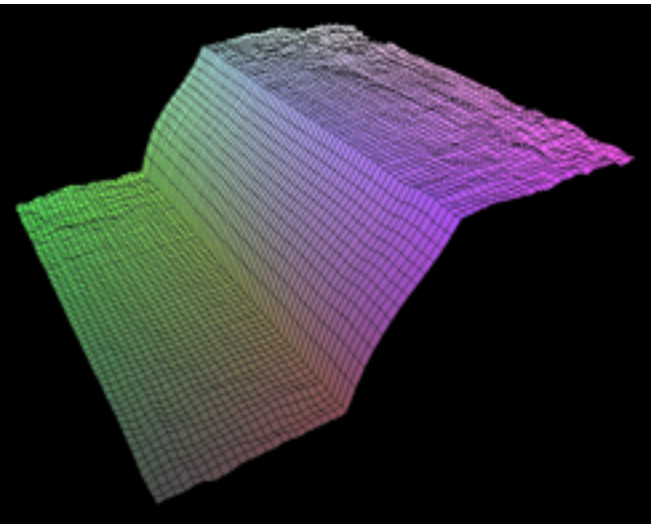
Plan

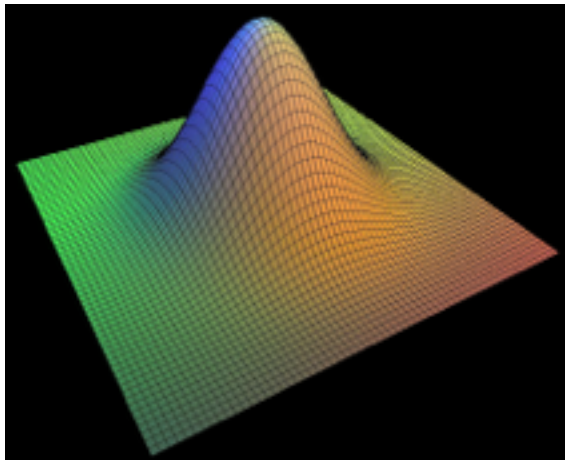
- Review of bilateral filtering [Tomasi and Manduchi 1998]
- Use for contrast reduction
- Theoretical framework
- Acceleration
- Handling uncertainty

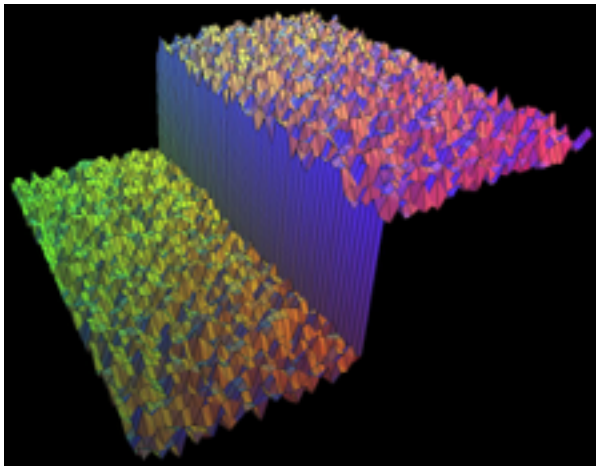
Start with Gaussian filtering

- Spatial Gaussian f
- Result is blurred

$$J = f \otimes I$$







output

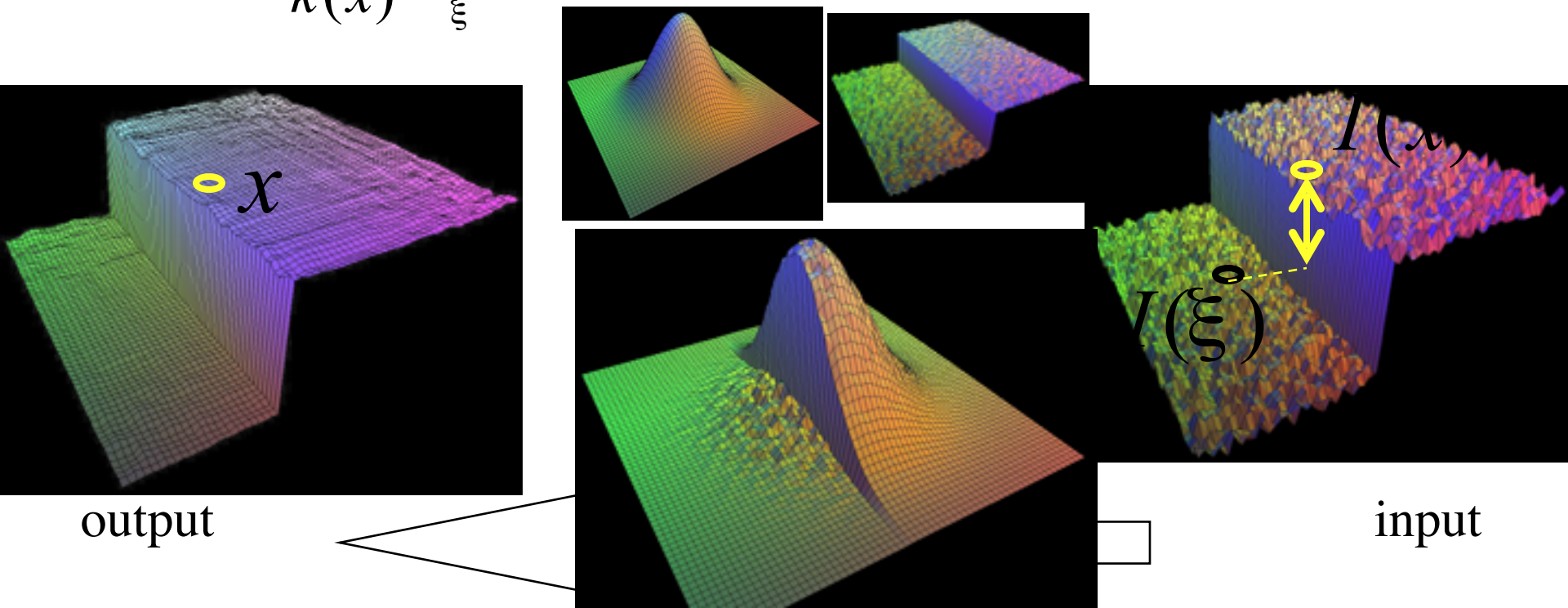
input

Principle of Bilateral filtering

[Tomasi and Manduchi 1998]

- Penalty g on the intensity difference

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$

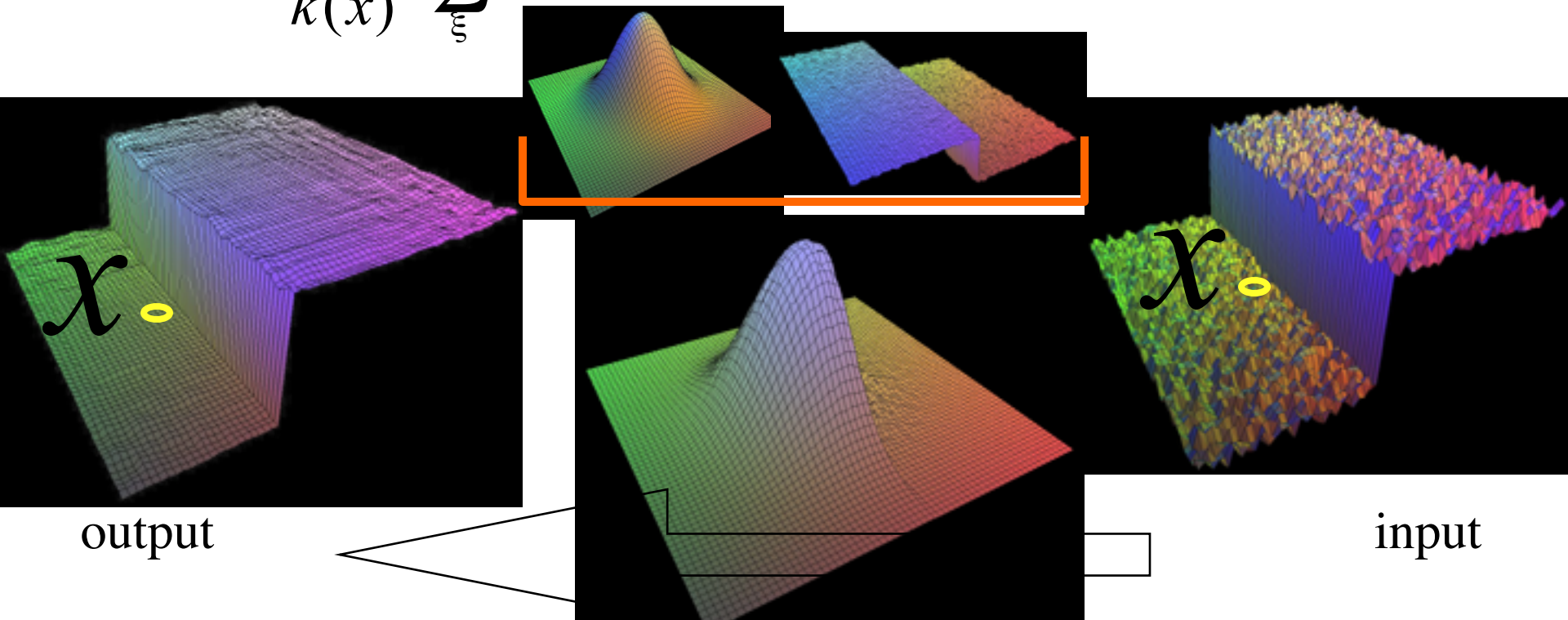


Bilateral filtering is non-linear

[Tomasi and Manduchi 1998]

- The weights are different for each output pixel

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$



Plan

- Review of bilateral filtering [Tomasi and Manduchi 1998]
- Use for contrast reduction
- Theoretical framework
- Acceleration
- Handling uncertainty

Contrast reduction

Input HDR image

Contrast
too high!

Contrast reduction

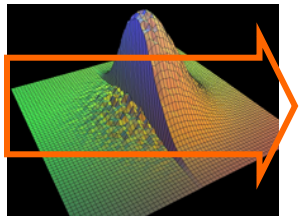
Input HDR image

$$I_i = \frac{R_i + G_i + B_i}{3} \qquad L_i = \log_2(I_i)$$

$$\left(\frac{R_i}{I_i}, \frac{G_i}{I_i}, \frac{B_i}{I_i} \right)$$

Contrast reduction

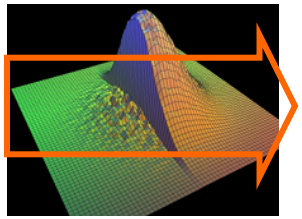
Input HDR image



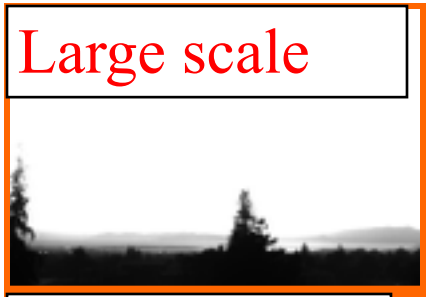
$$B = bilateral(L)$$

Fast
Bilateral
Filter

Contrast reduction



Fast
Bilateral
Filter

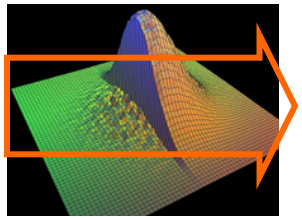


$$B = bilateral(L)$$

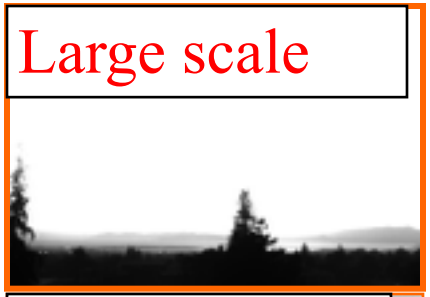
$$D = L - B$$

Contrast reduction

$$B' = (B - \max(B)) \times s$$



Fast
Bilateral
Filter

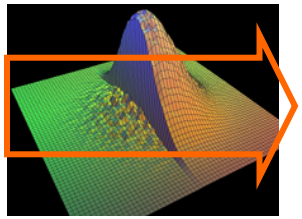


Reduce
contrast

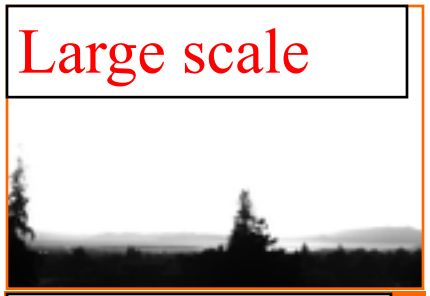
$$s = \frac{dR}{\max(B) - \min(B)}$$

$$2 \le dR \le 8$$

Contrast reduction



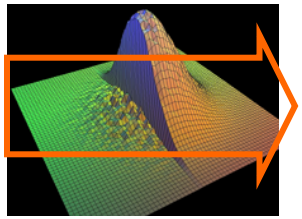
Fast
Bilateral
Filter



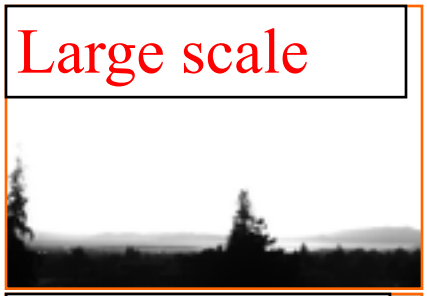
Reduce
contrast

Preserve!

Contrast reduction



Fast
Bilateral
Filter

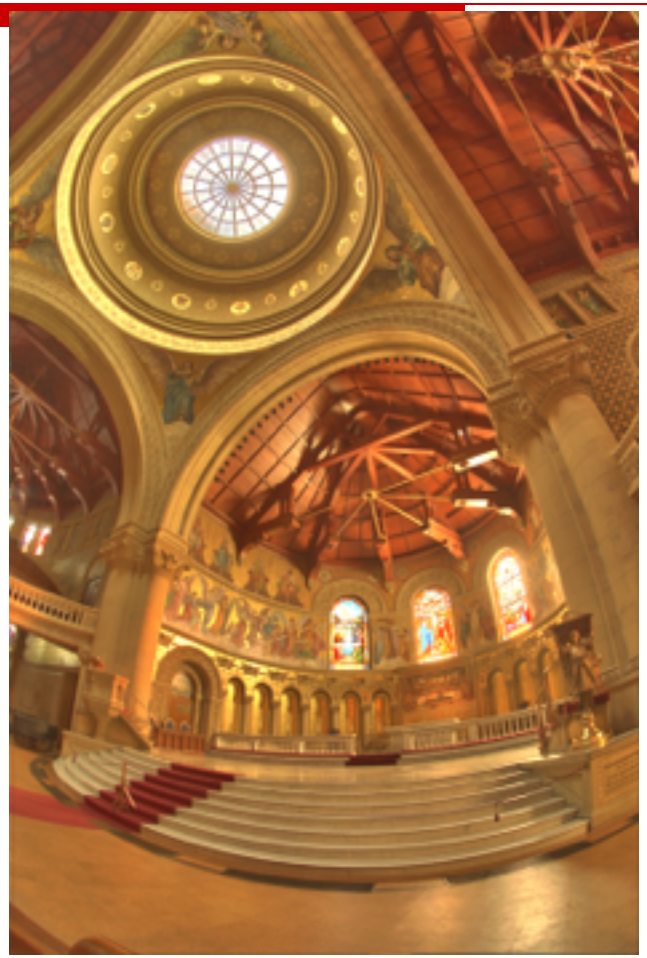


Reduce
contrast

Preserve!

Informal comparison

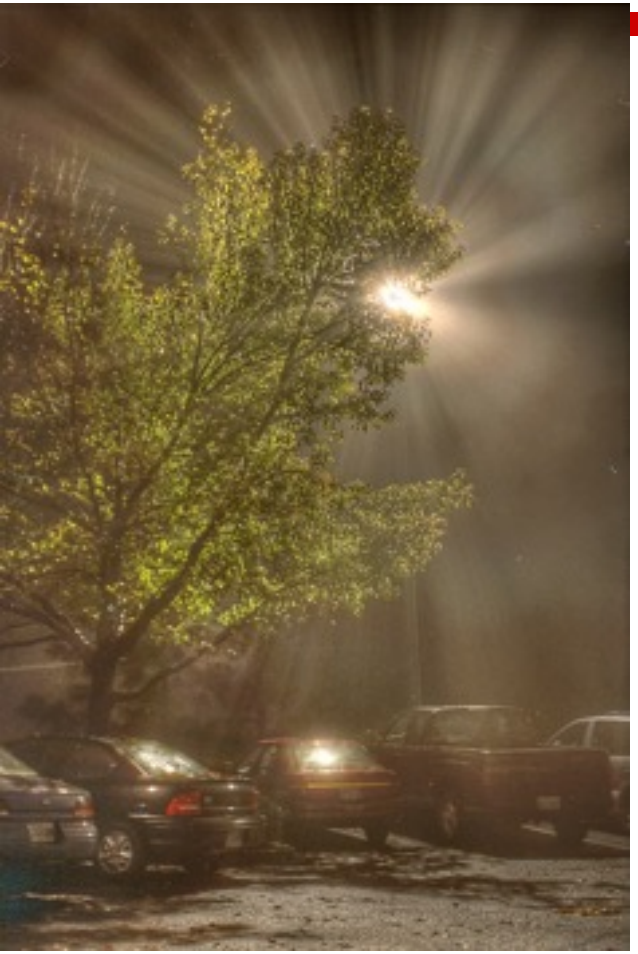
Gradient-space
[Fattal et al.]



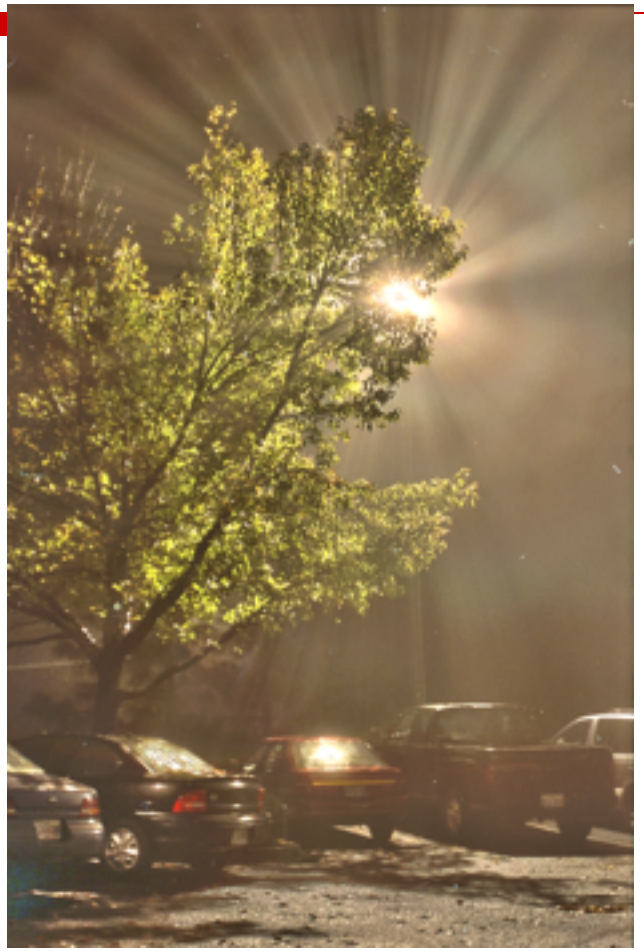
Bilateral
[Durand et al.]

Photographic
[Reinhard et al.]

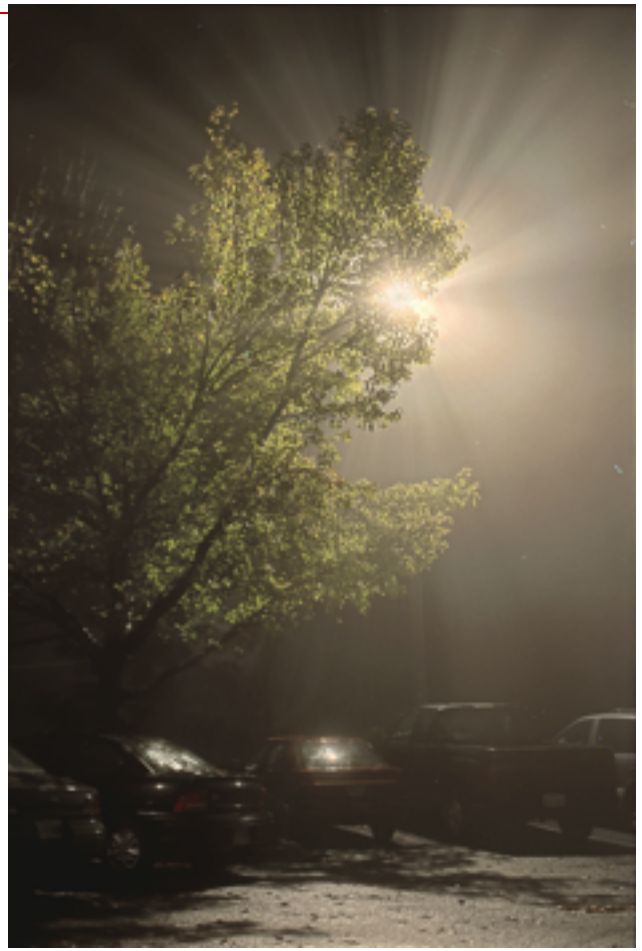
Informal comparison



Gradient-space
[Fattal et al.]



Bilateral
[Durand et al.]



Photographic
[Reinhard et al.]

Game-Plan for Homework 1

- Given RGB in HDR image:
- Compute mean intensity I = (R + G + B) / 3
- Compute the log intensity image L = log2(I)
- Filter w/ a bilateral filter B = bilateral_filter(L)
- Compute detail layer D = L - B
- Apply an offset and scale to B, B' = (B - o) * s
  - o = max(B)
  - s is set so that the output has dR stops of dynamic range (s = dR / (max(B) - min(B))). Values for dR between 2 and 8 are appropriate.
- Reconstruct the log intensity, O = exp(B' + D)
- Reconstruct the colors, (R',G',B') = O * (R/I, G/I, B/I)
- Apply gamma compression, R'' = R'^0.5
- Rescale each channel to fit in an LDR image [0..255]

Theoretical framework

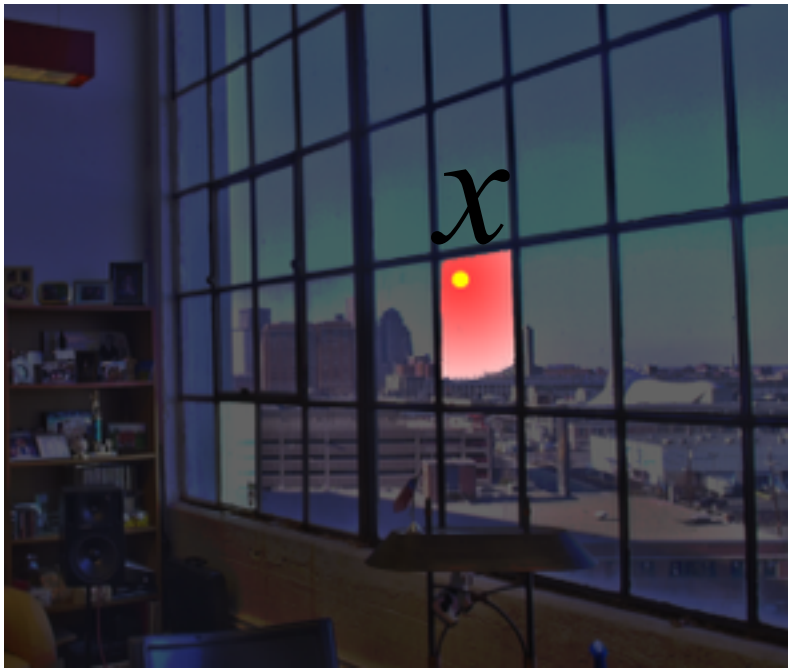
- Framework of robust statistics
  - Output = estimator at each pixel
  - Less influence to outliers (because of g)
- Unification with anisotropic diffusion
  - Mostly equivalent
  - Some differences

- Details and other insights in paper

Spatial support

Spatial support

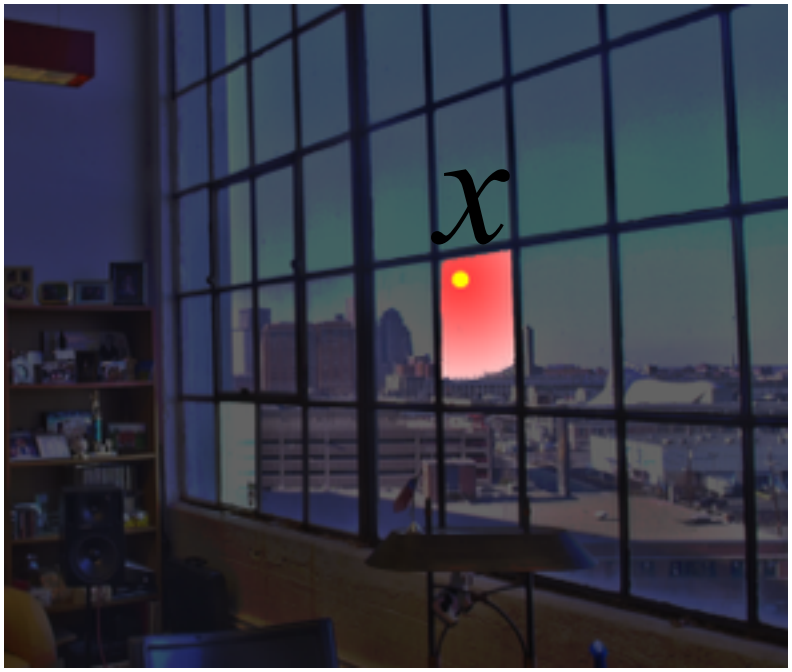
- Anisotropic diffusion cannot diffuse across edges



Support of anisotropic diffusion

Spatial support

- Anisotropic diffusion cannot diffuse across edges
- Bilateral filtering can
- Larger support => more reliable estimator



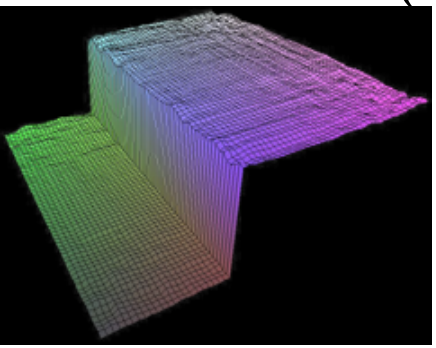
Support of anisotropic diffusion

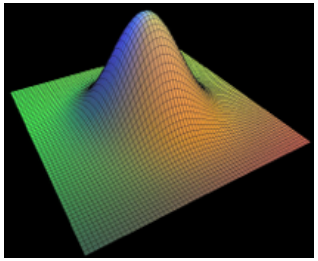
Support of bilateral

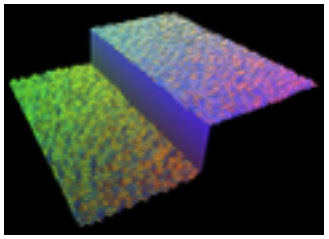
Acceleration

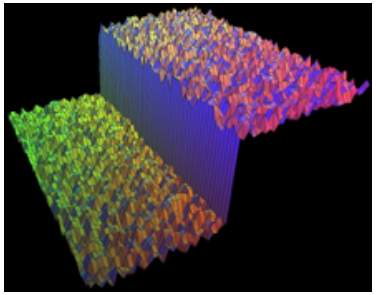
- Non-linear because of g

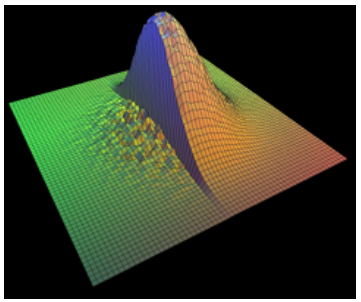
$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$







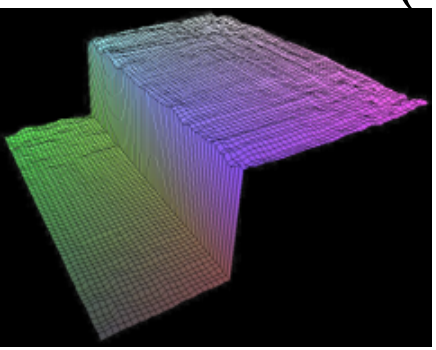


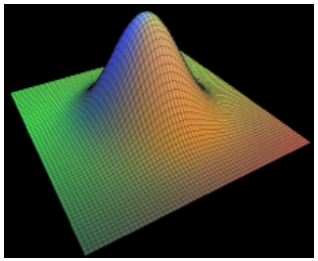


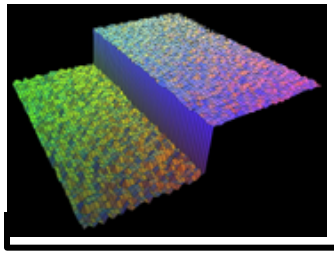
Acceleration

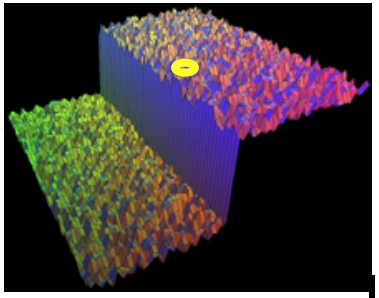
- Linear for a given value of I(x)
- Convolution of g I by Gaussian f

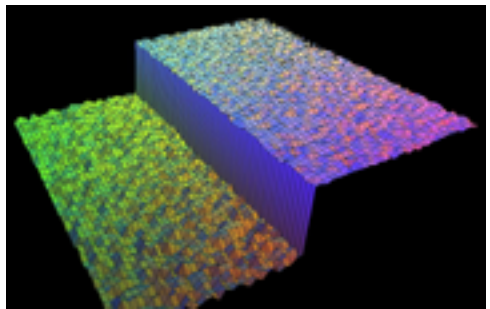
$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$







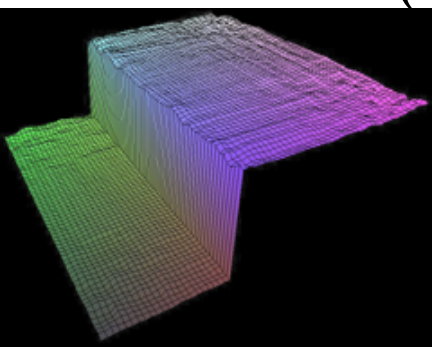
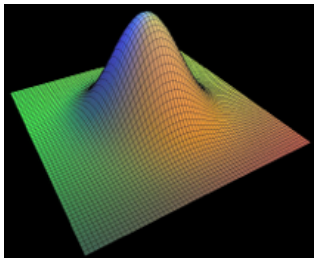
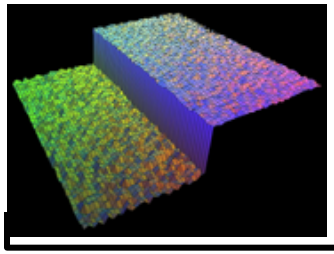
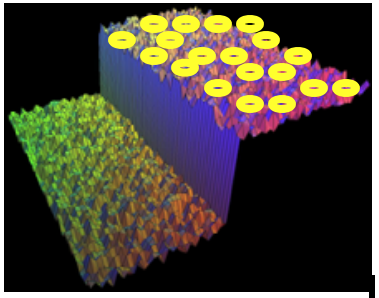
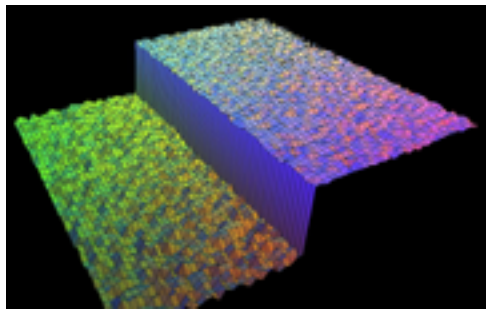




Acceleration

- Linear for a given value of I(x)
- Convolution of g I by Gaussian f
- Valid for all x with same value I(x)

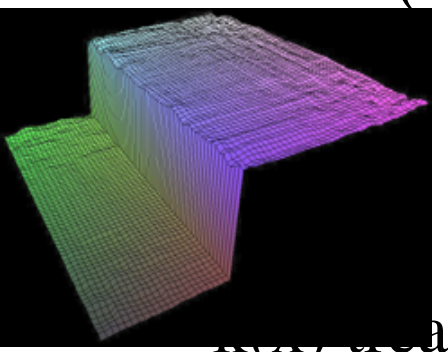
$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$

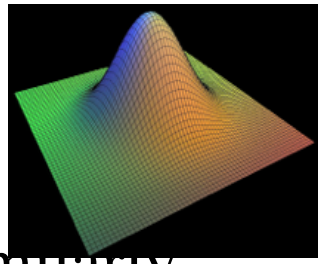






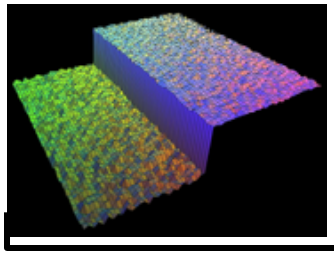
Acceleration

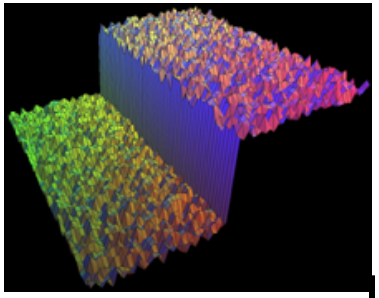
- Discretize the set of possible I(x)
- Perform linear Gaussian blur (FFT)
- Linear interpolation in between

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$

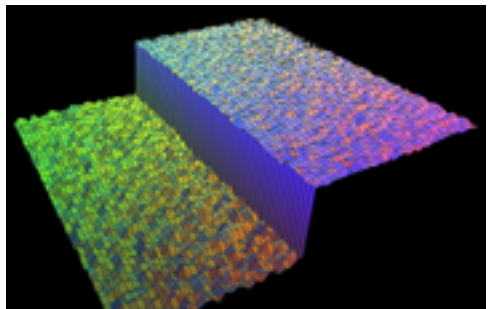








k(x) treated similarly



What does the eye see?

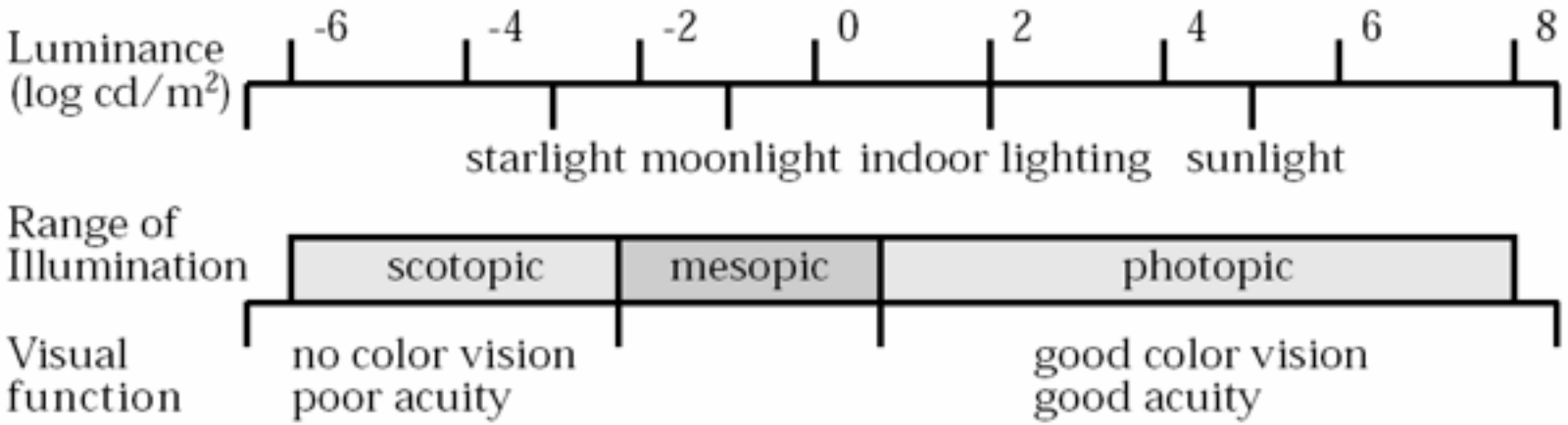


Figure 1: The range of luminances in the natural environment and associated visual parameters. After Hood (1986).

The eye has a huge dynamic range

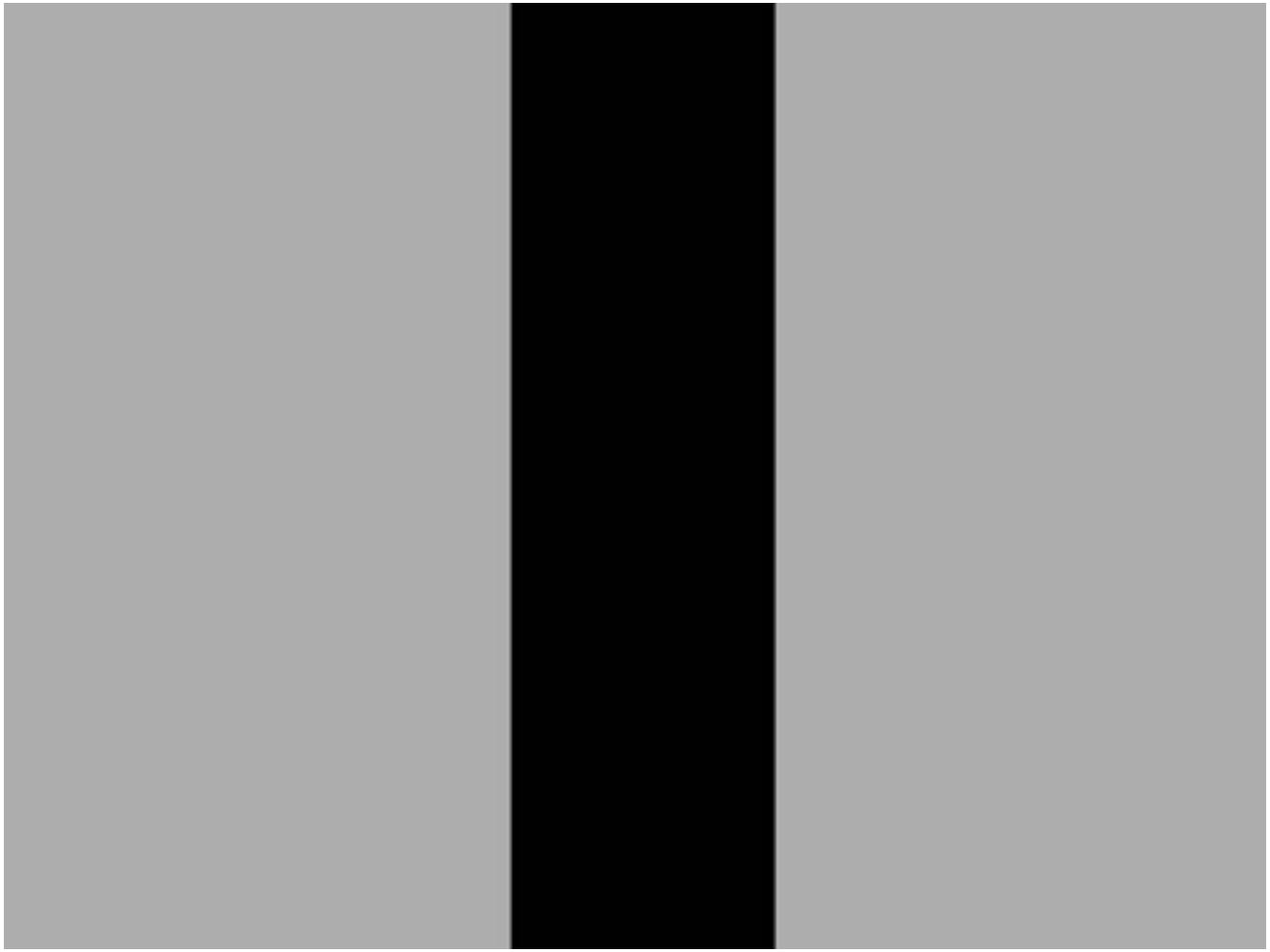
Do we see a true radiance map?

Eye is not a photometer!

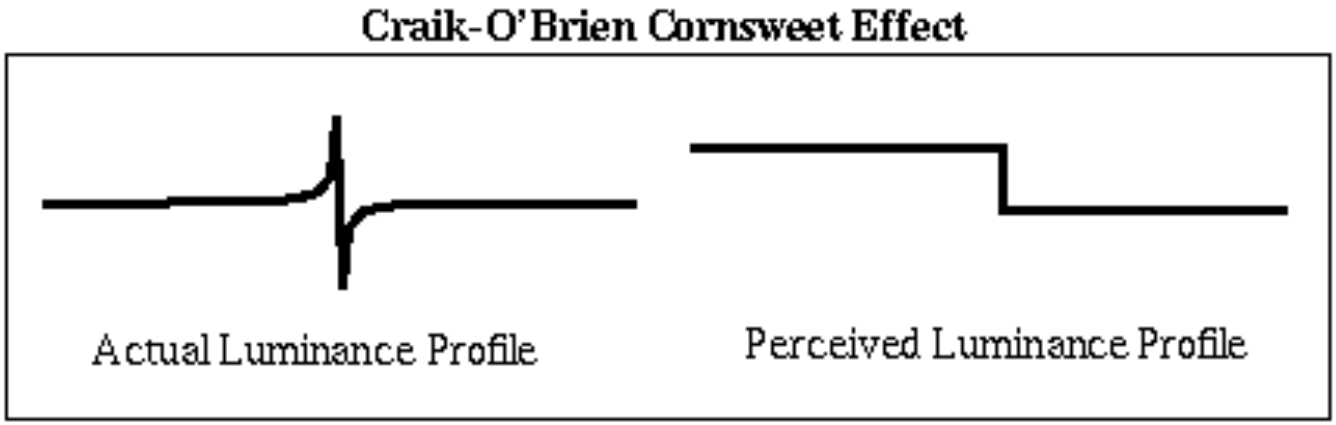
"Every light is a shade, compared to the higher lights, till you come to the sun; and every shade is a light, compared to the deeper shades, till you come to the night."

— John Ruskin, 1879

Cornsweet Illusion

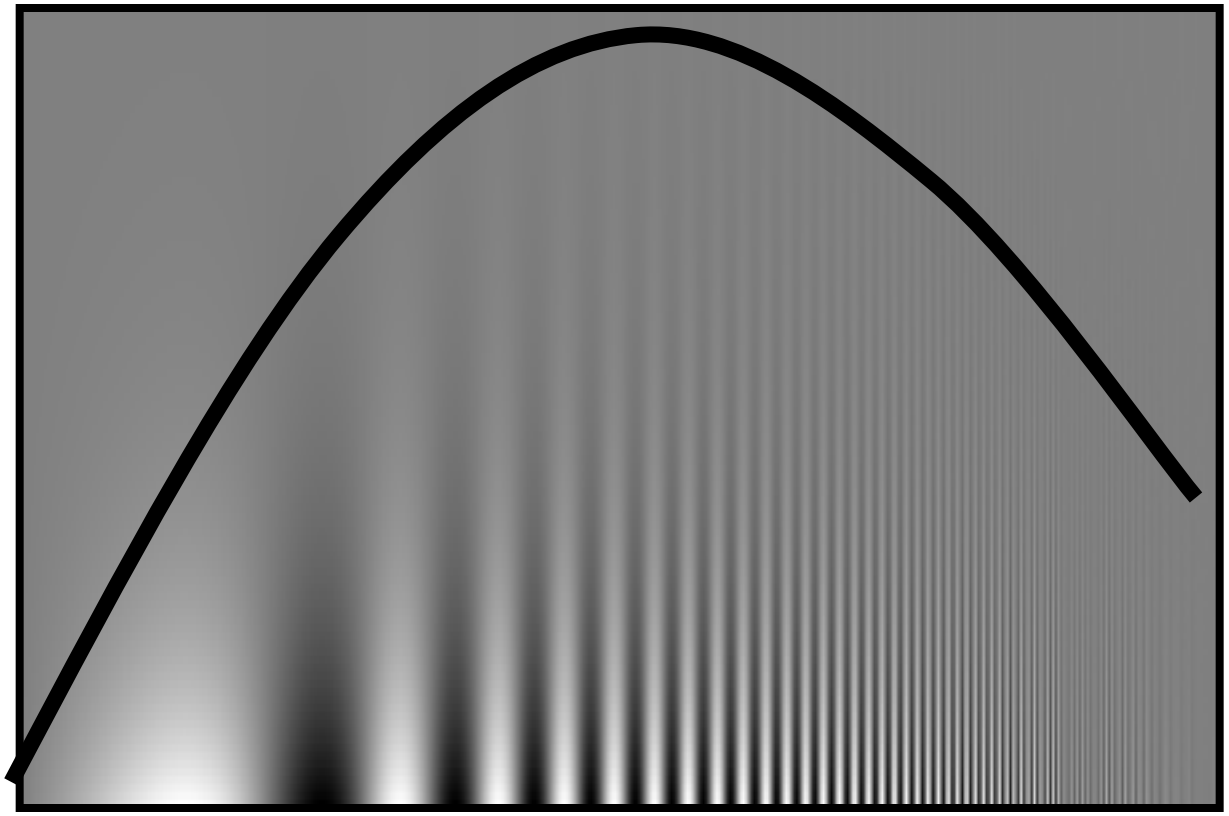


Metameric Failure



Eye is sensitive to changes

Sine Wave

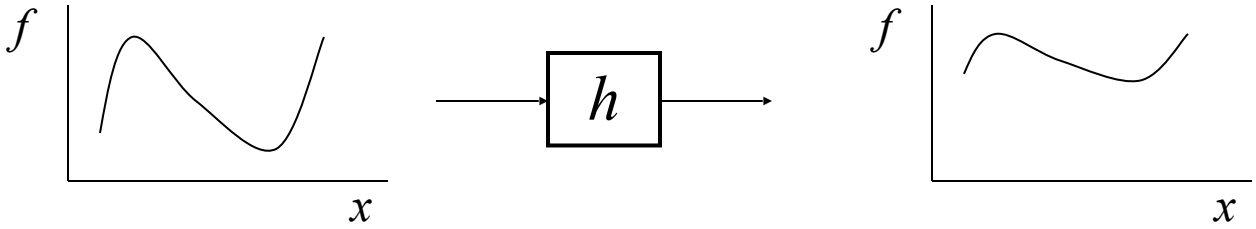


Campbell-Robson contrast sensitivity curve

Image Processing

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

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



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

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

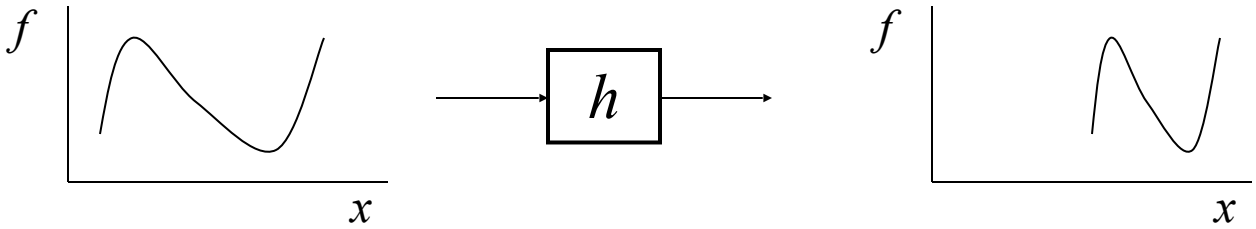
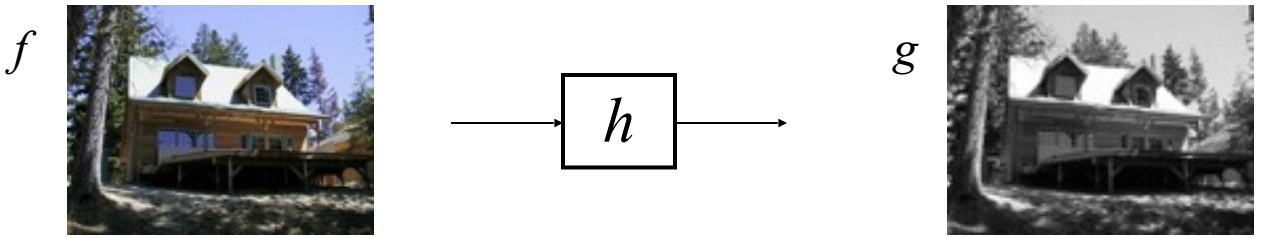


Image Processing

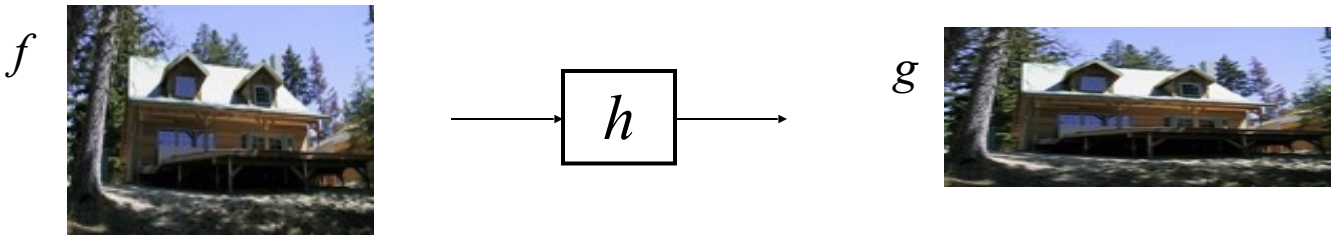
- image filtering: change ***range*** of image

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



- 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:

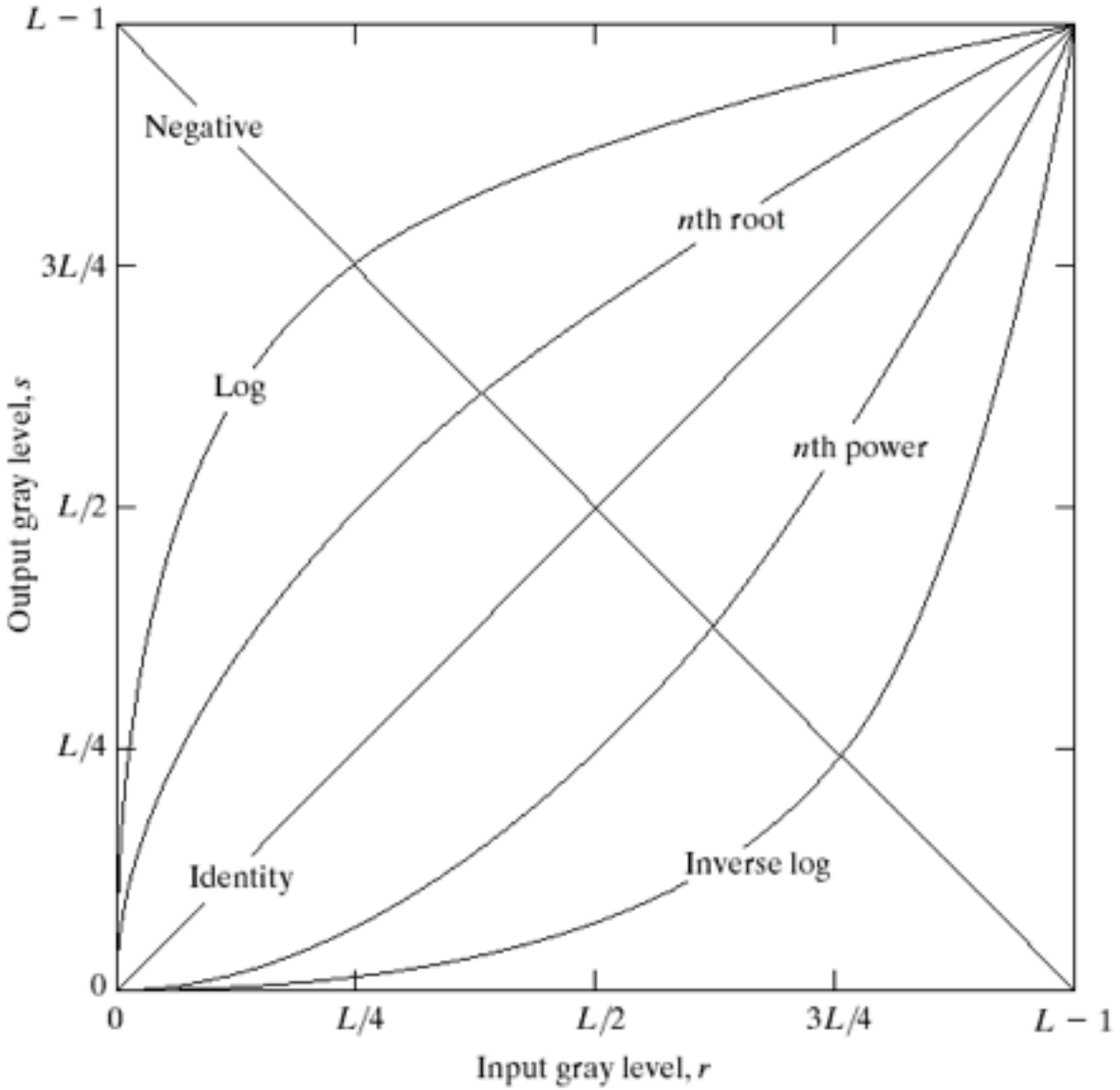
$$g = t(f)$$

- This is called point processing.

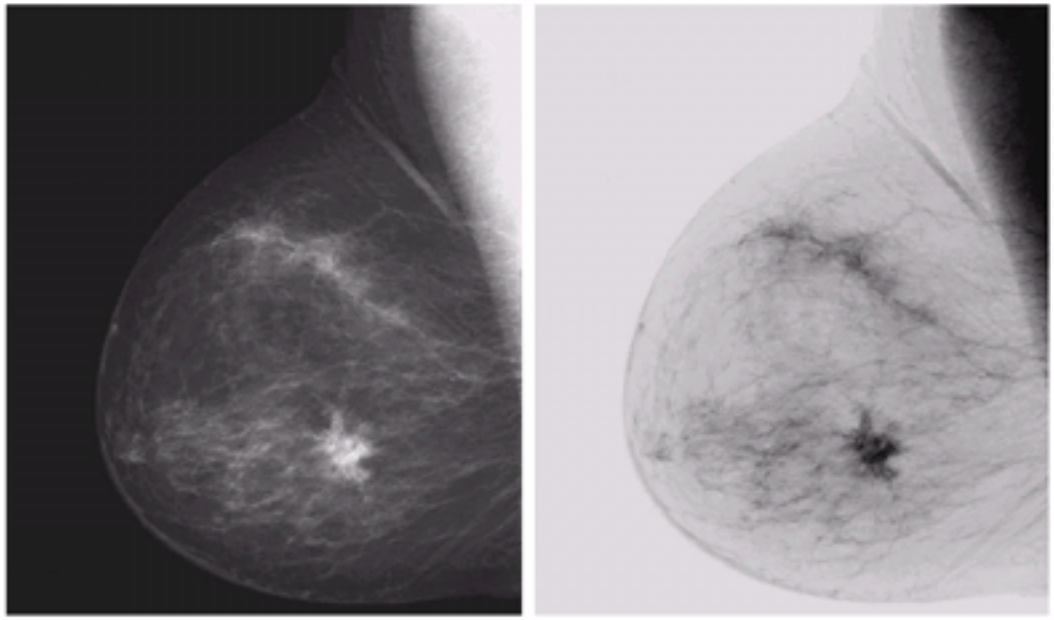
- What can they do?
- What's the form of *t*?

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

Basic Point Processing



Negative



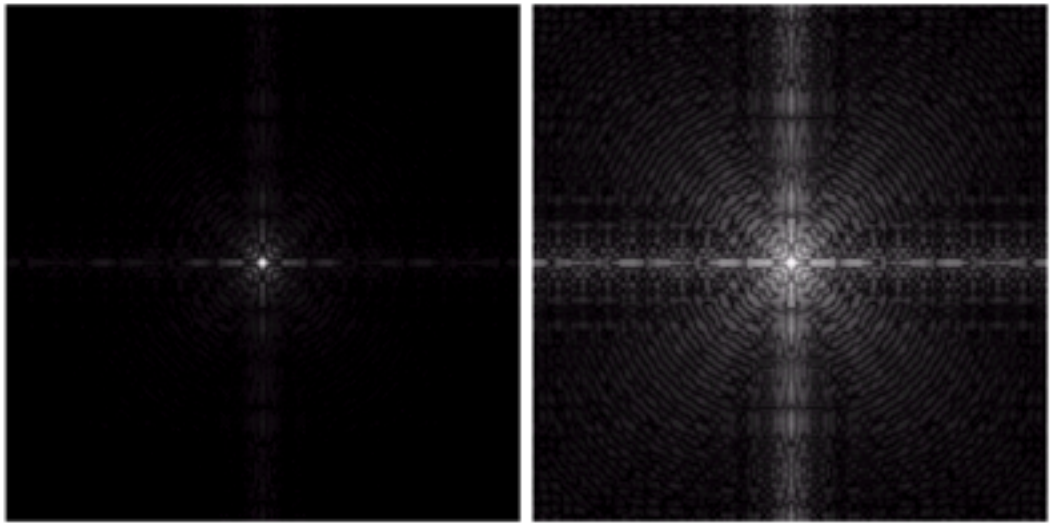
a b

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

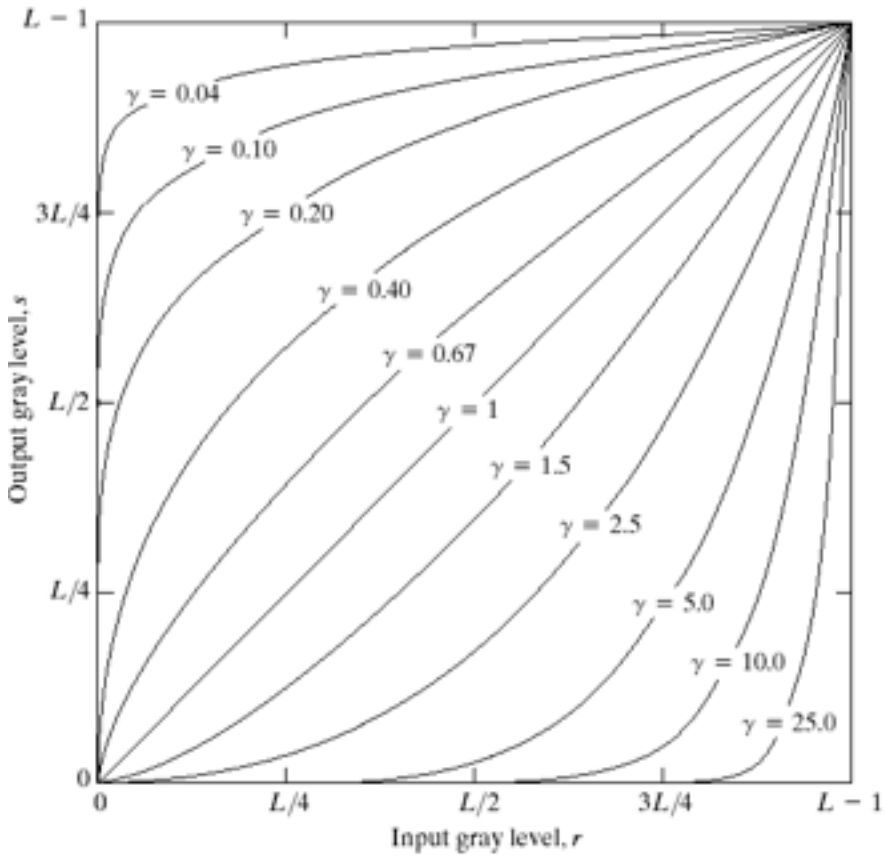


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

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

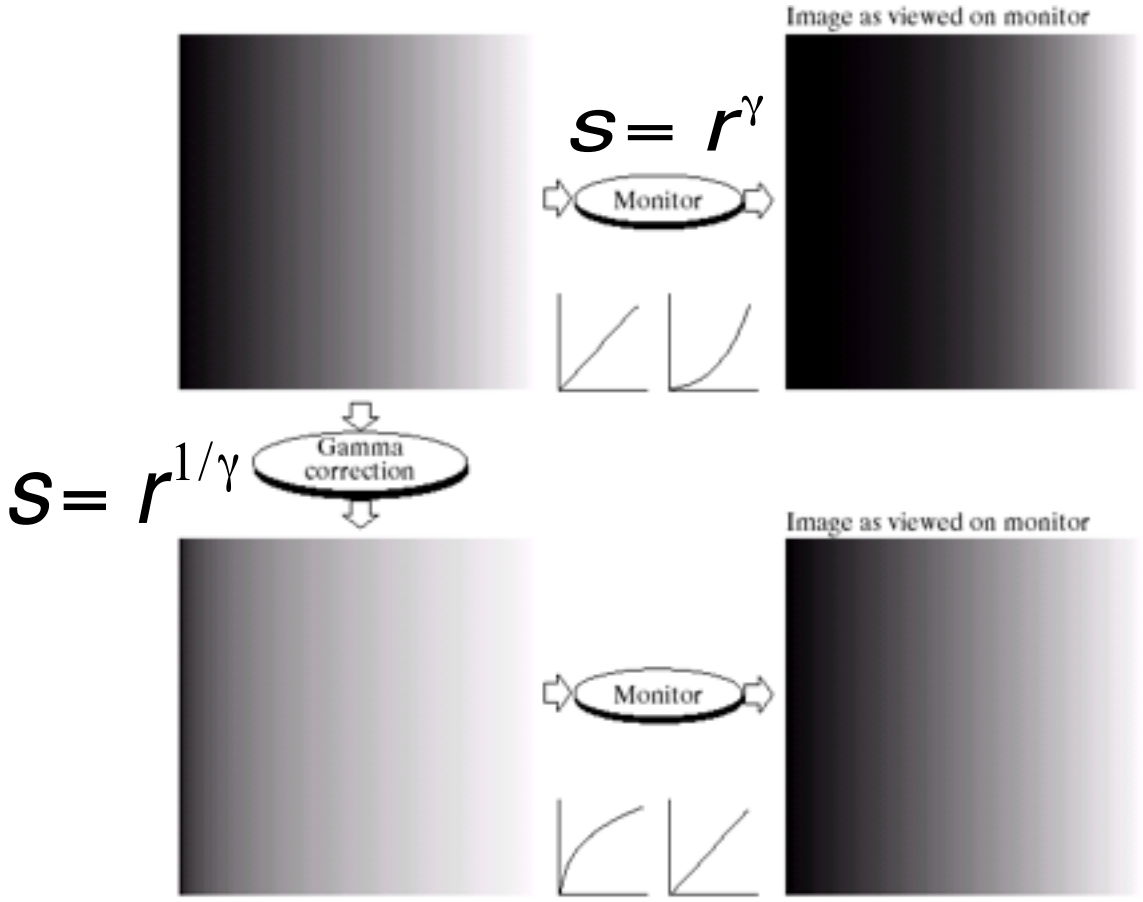
Image Enhancement

$\gamma = 3.0$

$\gamma = 4.0$

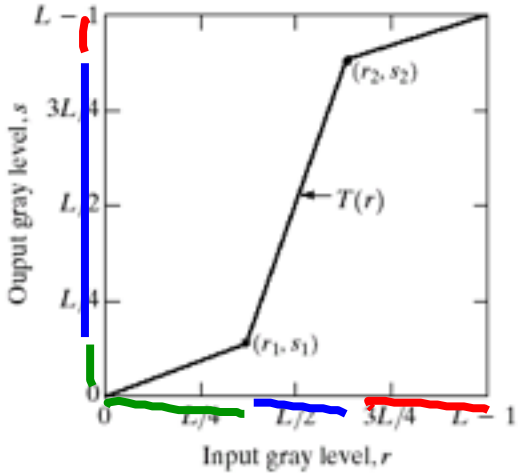
$\gamma = 5.0$

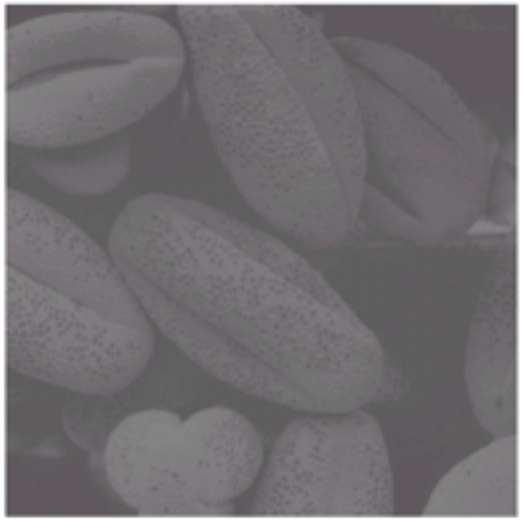
Example: Gamma Correction



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

Contrast Stretching





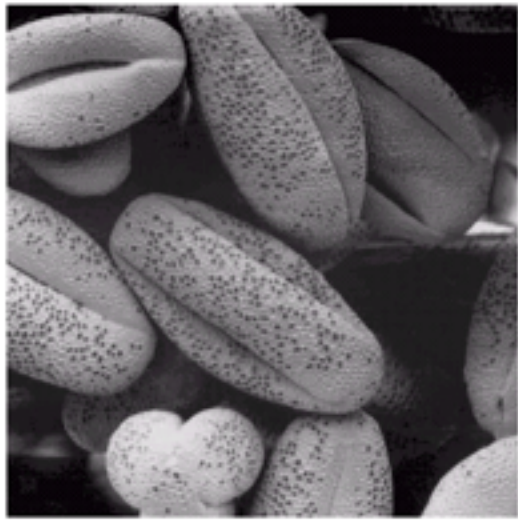
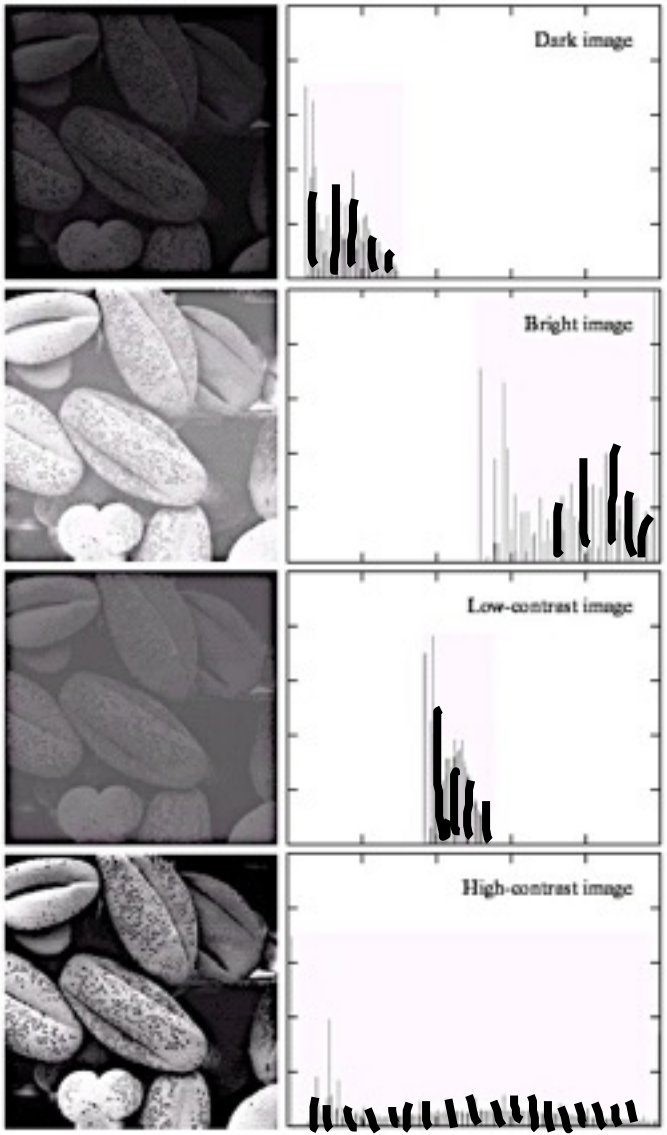
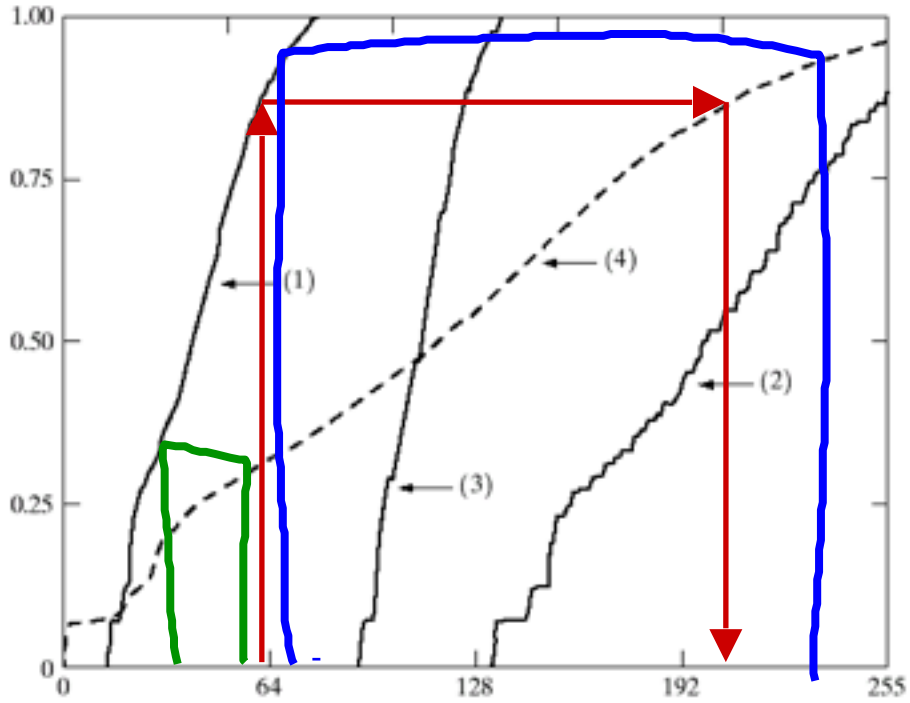


Image Histograms

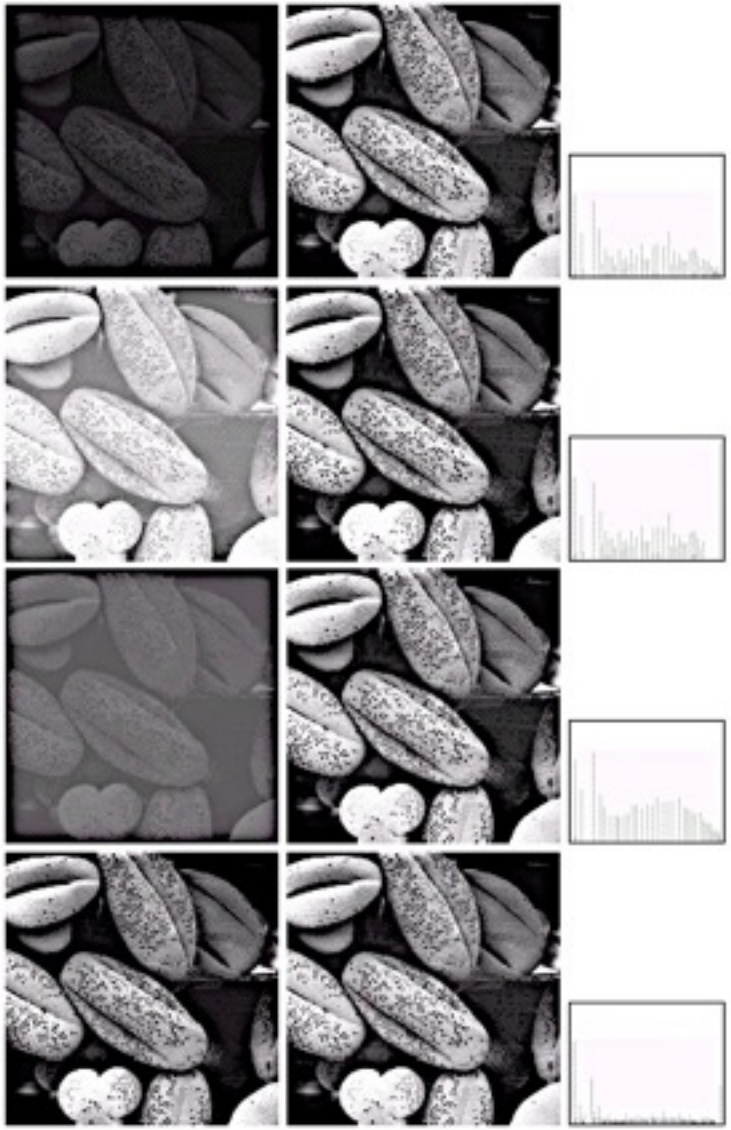




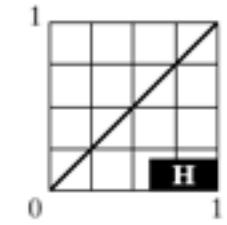
Cumulative Histograms

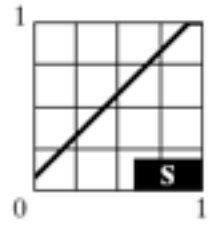
$s = T(r)$

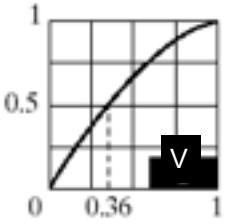
Histogram Equalization



Histogram Equalization



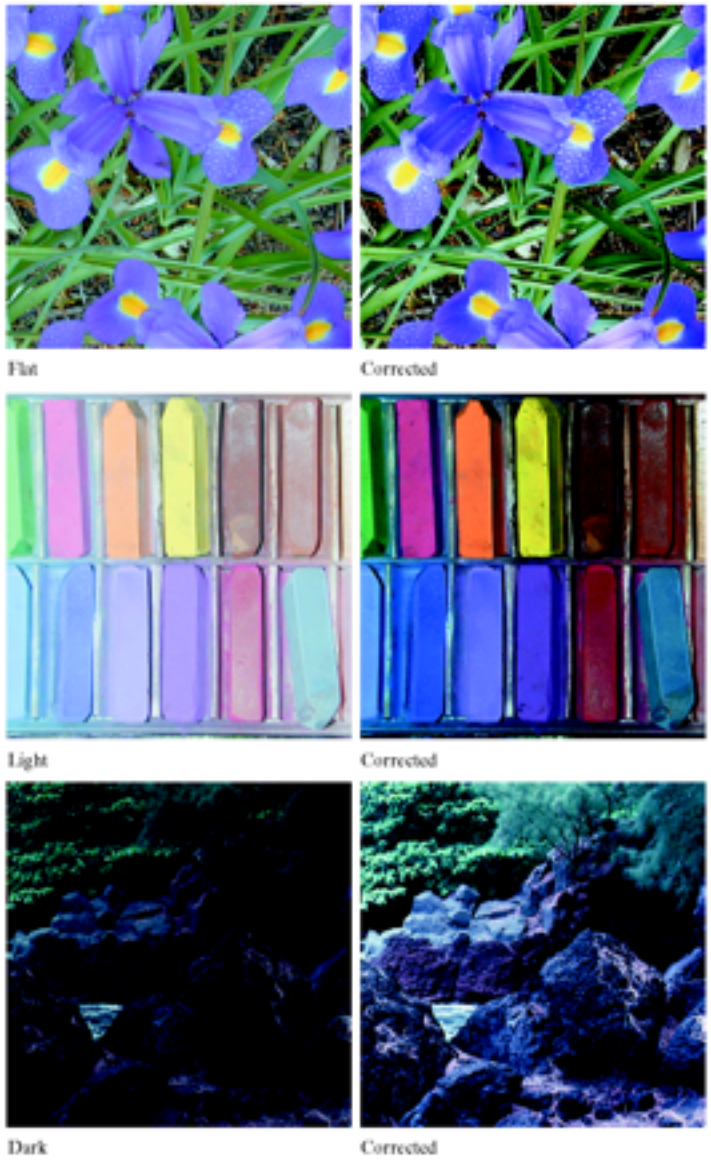




Histogram equalize the brightness (V)

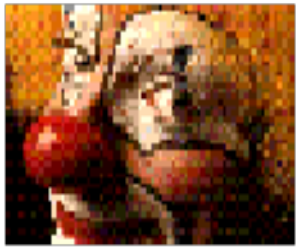
Color Channels

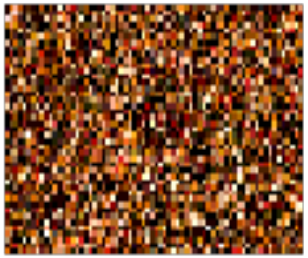
Color Point Processing



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