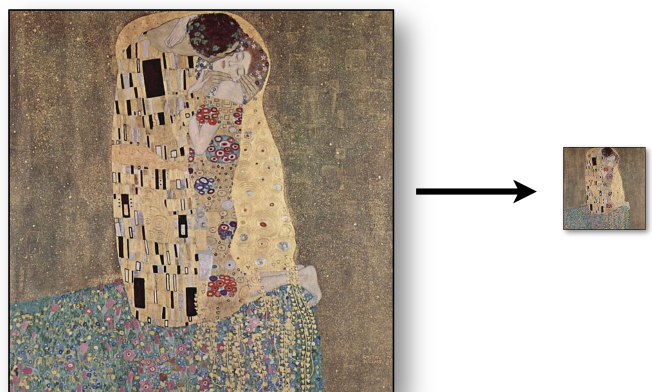
Tone Reproduction

Today's Agenda

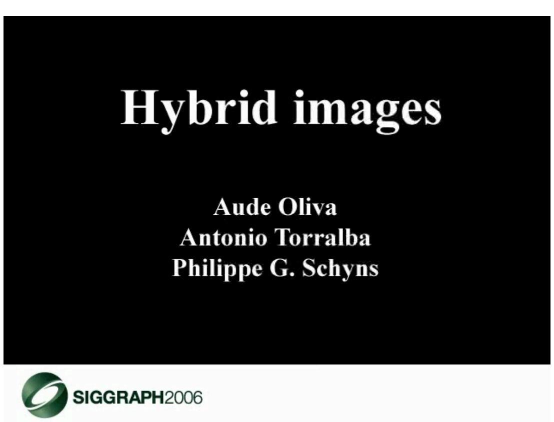
- Reminders:
  - A02 questions?
- Goals for today:
  - Wrap up filtering
  - Introduction point processing and tone reproduction

Warm-up Questions

Why does a lower resolution image still make sense to us? What do we lose?



Why/How does the technique of hybrid images work?



http://cvcl.mit.edu/hybridimage.htm

Types of Filters: Smoothing

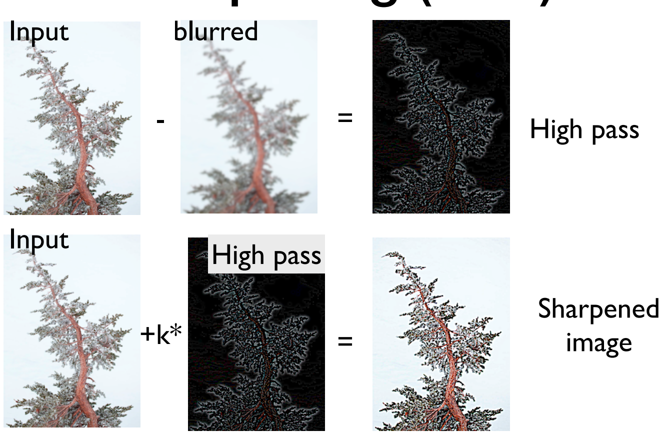
Smoothing Comparison

(a) Source image. (b) 17 × 17 Box. (c) 17 × 17 Gaussian.

Figure 6.10. Smoothing examples.

Types of Filters: Sharpening

Sharpening (Idea)



Sharpening is a Convolution

$$
\begin{aligned}
I_{\text{sharp}} &= (1+\alpha)I - \alpha(I \bigstar f_{g,\sigma}) \\
&= I \bigstar ((1+\alpha)d - \alpha f_{g,\sigma}) \\
&= I \bigstar f_{\text{sharp}}(\sigma, \alpha),
\end{aligned}
$$

Note: could also define d as $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$$
d = \frac{1}{9} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix},
$$

$$
f_{g,\sigma} = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},
$$

$$
\left((1+\alpha)d - \alpha f_{g,\sigma}\right) = \frac{1}{9} \times \begin{bmatrix} -\alpha & -\alpha & -\alpha \\ -\alpha & (9+8\alpha) & -\alpha \\ -\alpha & -\alpha & -\alpha \end{bmatrix}
$$

Edges

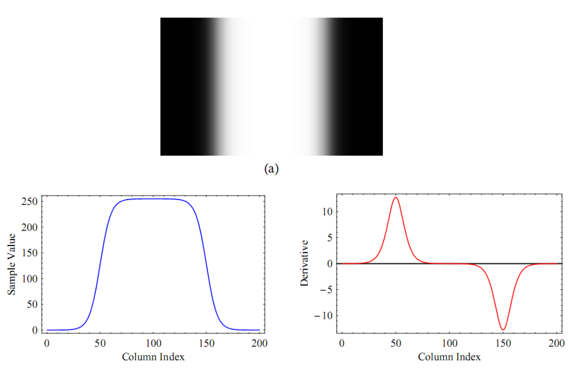


Figure 6.11. (a) A grayscale image with two edges, (b) row profile, and (c) first derivative.

(Review?) Derivatives via Finite Differences

- We can approximate the derivative with a kernel w:

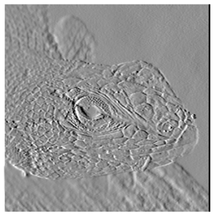
$$
\frac{\partial f}{\partial x} \approx \frac{1}{2h}\left(f(x+1,y) - f(x-1,y)\right)
$$

$$
\frac{\partial f}{\partial x} \approx w_{dx} \circ f \qquad w_{dx} = \boxed{-\tfrac{1}{2}} \boxed{0} \boxed{\tfrac{1}{2}}
$$

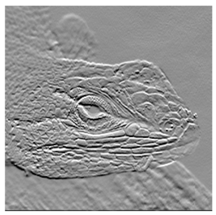
$$
\frac{\partial f}{\partial y} \approx w_{dy} \circ f \qquad w_{dy} = \begin{array}{|c|} \hline -\tfrac{1}{2} \\ \hline 0 \\ \hline \tfrac{1}{2} \\ \hline \end{array}
$$

Taking Derivatives with Convolution

$$\begin{matrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$$



$$\begin{matrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{matrix}$$



Gradients with Finite Differences

- These partial derivatives approximate the image gradient, ∇I.
- Gradients are the unique direction where the image is changing the most rapidly, like a slope in high dimensions
- We can separate them into components kernels G_x, G_y. $\nabla I = (G_x, G_y)$

$$\nabla I(x, y) = \begin{pmatrix} \delta I(x,y)/\delta x \\ \delta I(x,y)/\delta y \end{pmatrix}.$$

$$G_x = [1, 0, -1] \qquad G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix};$$

$$\nabla I = \begin{pmatrix} \delta I/\delta x \\ \delta I/\delta y \end{pmatrix} \simeq \begin{pmatrix} I \otimes G_x \\ I \otimes G_y \end{pmatrix}.$$

Figure 6.12. Image gradient (partial).

128	187	210	238	251
76	121	193	225	219
66	91	110	165	205
47	81	83	119	157
41	59	63	75	125

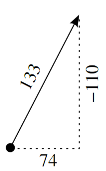
(a) Source Image.

	117	104	26	
	44	74	95	
	36	38	74	

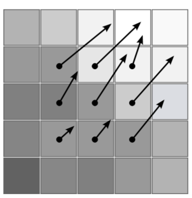
(b) $\delta I/\delta x$.

	−96	−100	−73	
	−40	−110	−106	
	−32	−47	−90	

(c) $\delta I/\delta y$.



(d) Center sample gradient.



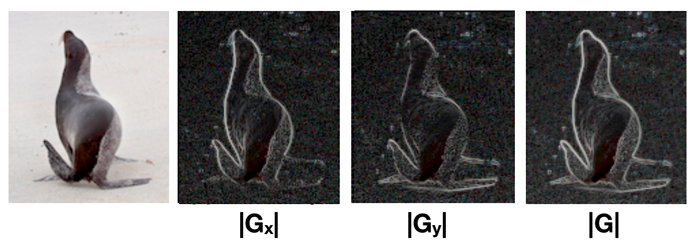
(e) Gradient.

	151	144	77	
	59	133	142	
	48	60	117	

(f) Magnitude of gradient.

Figure 6.14. Numeric example of an image gradient.

Gradients G_x, G_y



$$|G| = \sqrt{(G_x^2 + G_y^2)}$$

Second Derivatives (Sharpening, almost)

- Partial derivatives in x and y lead to two kernels:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

and, similarly, in the y-direction we have

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

Compare with Sharpening filter: unbalanced counts!

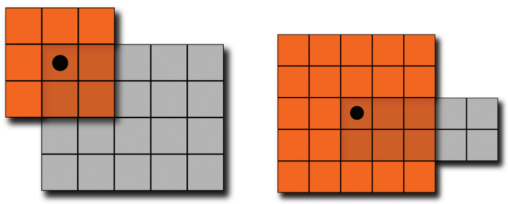
$$\begin{bmatrix} -\alpha & -\alpha & -\alpha \\ -\alpha & (9+8\alpha) & -\alpha \\ -\alpha & -\alpha & -\alpha \end{bmatrix}$$

1	1	1
1	-9	1
1	1	1

Boundaries

Handling Image Boundaries

- What should be done if the kernel falls off of the boundary of the source image as shown in the illustrations below?



(a) Kernel at $I(0,0)$. (b) Kernel larger than the source.

Figure 6.4. Illustration of the edge handling problem.

Handling Image Boundaries

- When pixels are near the edge of the image, neighborhoods become tricky to define
- Choices:
  1. Shrink the output image (ignore pixels near the boundary)
  2. Expanding the input image (padding to create values near the boundary which are “meaningful”)
  3. Shrink the kernel (skip values that are outside the boundary, and reweigh accordingly)

Boundary Padding

- When one pads, they pretend the image is large and either produce a constant (e.g. zero), or use circular / reflected indexing to tile the image:

(a) (b) (c)

Figure 6.5. (a) Zero padding, (b) circular indexing, and (c) reflected indexing.

Point Processing

Recall: Operations on Images

- Point Operations:

- Affect only the range of the image (e.g. brightness)
- Each pixel is processed separately, only depending on the color

Point Processing (Algorithm)

```
//given input: greyscale image
//produces processing image: output

for (let row = 0, row < H; row++) {
   for (let col = 0; col < W; col++) {
      let i = row*W + col;
      output[i] = some_func(input[i]);
   }
}
```

- This basic, but simple algorithm can be extended in lots of ways, depending on the function that we apply to each pixel and each color channel

First Example: Linear Rescaling

- **Rescaling** is a point processing technique that alters the **contrast** and/or **brightness** of an image.
- In photography, **exposure** is a measure of how much light is projected onto the imaging sensor.
  - **Overexposure**: more light than what the sensor can measure.
  - **Underexposure**: sensor is unable to detect the light.
- Images which are underexposed or overexposed can frequently be improved by brightening or darkening them.
- The contrast of an image can be altered to bring out the internal structure of the image.

Rescaling Math

- Given a sample C_{in} of the source image, rescaling computes the output sample, C_{out}, using the scaling function

$$C_{out} = \alpha C_{in} + \beta$$

- α is a real-valued scaling factor known as **gain**
- β is a real-valued scaling factor known as **bias**

Effects of Rescaling

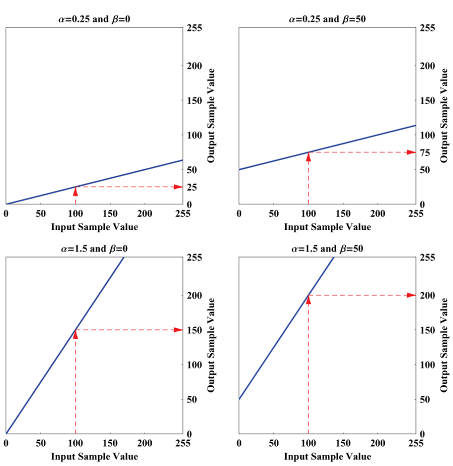


Figure 5.2. Graph of the linear scaling function with various gain and bias settings.

Why Use Both α, β?

- Consider two rescaled source samples of S rescaled to S'.

$$\begin{aligned} S_1' &= \alpha S_1 + \beta, \\ S_2' &= \alpha S_2 + \beta. \end{aligned}$$

- Calculate the **contrast** (the absolute difference) between the source and destination, called ΔS and $\Delta S'$.

$$\begin{aligned} \triangle S' &= |S_1' - S_2'|, \\ \triangle S &= |S_1 - S_2|. \end{aligned}$$

- Now consider the relative change in contrast between the source and destination.

$$\frac{\triangle S'}{\triangle S} = \frac{|S_1' - S_2'|}{|S_1 - S_2|}.$$

Why Use Both α, β?

- The relative change in contrast can be simplified as

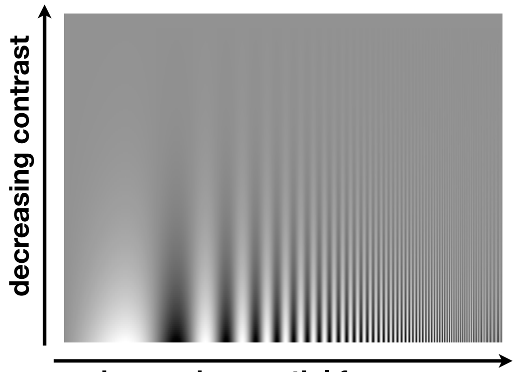
$$\begin{aligned} \frac{\triangle S'}{\triangle S} &= \frac{|(\alpha S_1 + \beta) - (\alpha S_2 + \beta)|}{|S_1 - S_2|} \\ &= \frac{|\alpha| \cdot |S_1 - S_2|}{|S_1 - S_2|} \\ &= |\alpha|. \end{aligned}$$

- Thus, gain (α) controls the change in contrast.
- Whereas bias (β) *does not* affect the contrast
- Bias, however, controls the final **brightness** of the rescaled image. Negative bias darkens and positive bias brightens the image

Sidebar: Relating Contrast Sensitivities to Signal Processing

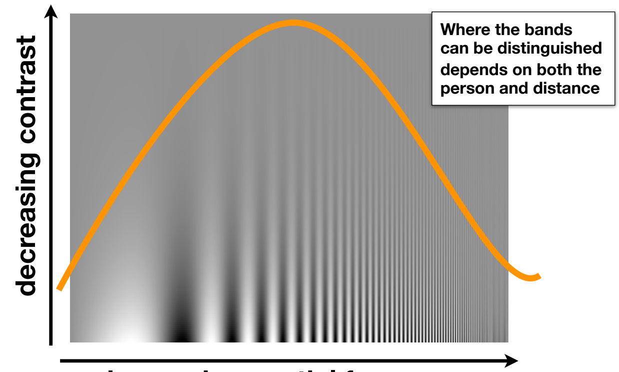
Contrast Sensitivity Function

Campbell-Robson Chart



Contrast Sensitivity Function

Campbell-Robson Chart



Contrast Sensitivities Vary by Channel

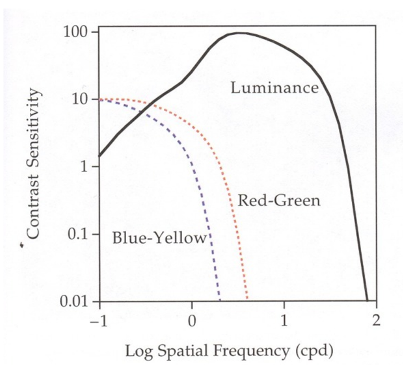
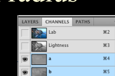


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

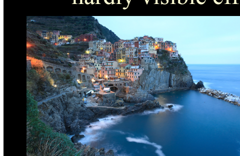
Photoshop demo

- Image > Mode > Lab color
- Go to channel panel, select Lightness

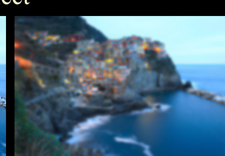
- Filter > Blur > Gaussian Blur , e.g. 4 pixel radius
  - very noticeable



- Undo, then select a & b channels
- Filter > Blur > Gaussian Blur , same radius
  - hardly visible effect



Original



Blur Lightness

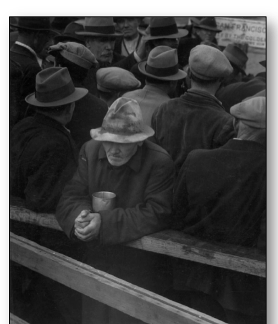
Blur a & b

Important: Clamping

- Rescaling may produce samples that lie outside of the output images (e.g. below 0 or above 255 in 8-bit images)
- **Clamping** the output values ensures that the output samples are truncated to the 8-bit dynamic range limit
- Note that clamping does 'lose' information, since it truncates.

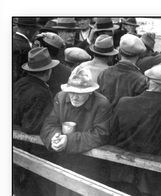
$$clamp(x, min, max) = \begin{cases} \text{min} & \text{if } \lfloor x \rceil \leq \text{min}, \\ \text{max} & \text{if } \lfloor x \rceil \geq \text{max}, \\ \lfloor x \rceil & \text{otherwise.} \end{cases}$$

Rescaling Examples



gain = 1, bias = 55

gain = 1, bias = -55



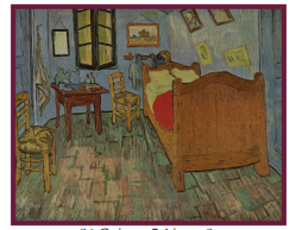
gain = 2, bias=0

gain = .5, bias=0

Rescaling Color Images

- Often, it is desirable to apply different gain and bias values to each channel of a color image separately
  - Example: A color image that utilizes the HSB color model. Since all color information is contained in the H and S channels, it may be useful to adjust ONLY the brightness, encoded in channel B, without altering the color of the image in any way.
- Rescaling the channels of a color image in a non-uniform manner is also possible rescaling each color channel separately.

(a) Original.



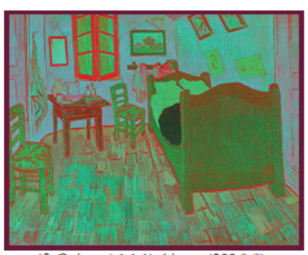
(b) Gain = .7, bias = 0.

(c) Gain = 1.2, bias = 40.

(d) Gain = -1, bias = 255.

Rescaling Channels Separately

(e) Gain = (1,0,0), bias = (0,0,0).



(f) Gain = (-1,1,1), bias = (255,0,0).

Gamma Correction

Gamma Correction

- **Gamma correction** is an image enhancement operation that seeks to maintain perceptually uniform sample values
- Gamma correction seeks to eliminate the nonlinear distortions introduced by the first (acquisition) and the final (display) phases of the image processing pipeline.

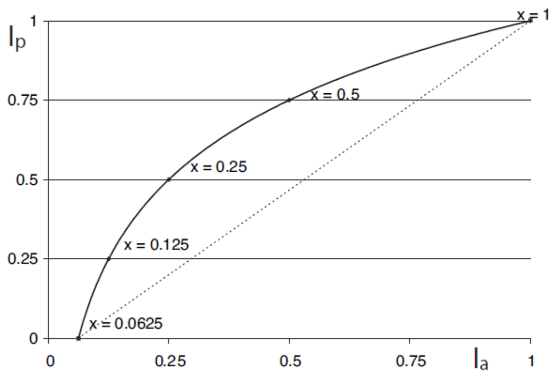
Human Perception

- Eye distinguishes color intensities as a function of the ratio between intensities.
- Consider $I_1 < I_2 < I_3$, for the step between I_1 and I_2 to look like the step from I_2 to I_3, it must be that:

$$I_2 / I_1 = I_3 / I_2$$

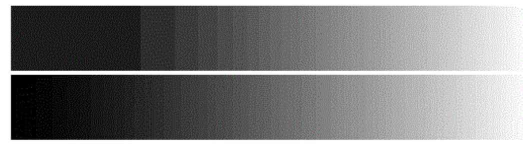
- As opposed to the differences in contrast! $I_2 - I_1 \neq I_3 - I_2$

Perceived (I_p) vs. Actual (I_a) Intensity



Perceived (I_p) vs. Actual (I_a) Intensity

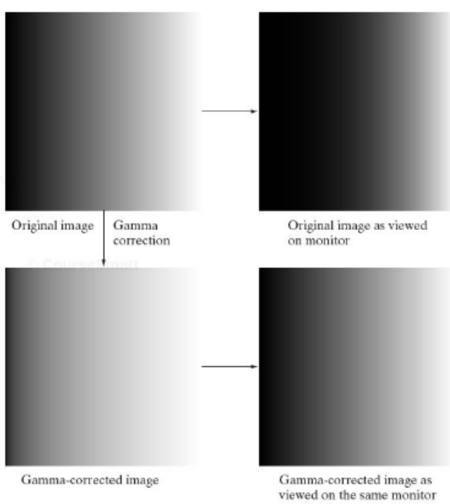
- Perceived light actually behaves like $I_p = (I_a)^\gamma$



http://www.anyhere.com/gward/hdrenc/

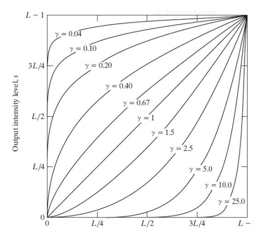
a b
c d

FIGURE 3.7
(a) Intensity ramp image. (b) Image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma-corrected image. (d) Corrected image as viewed on the same monitor. Compare (d) and (a).



Example: Gamma Correction

$s = r^\gamma$



a b
c d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0$, 4.0, and 5.0, respectively. (Original image for this example courtesy of NASA.)

Gamma Correction vs. Scaling with Gain/Bias Adjustments

- Gamma changes curve instead of sliding it (bias) or changing just slope (gain)

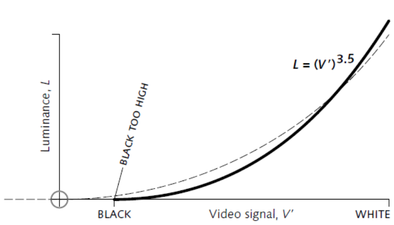


Figure 4 **Gamma 3.5** A naive approach to the measurement of CRT nonlinearity is to model the response as $L = (V')^\gamma$, and to find the exponent of the power function that is the best fit to the voltage-to-intensity transfer function of a particular CRT. However, if this measurement is undertaken with BRIGHTNESS set too high, an unrealistically large value of gamma results from the modelled curve being "pegged" at the origin.

http://www.poynton.com/PDFs/Rehabilitation_of_gamma.pdf

Putting it all together: Gain, Bias, and Gamma

- $C_{out} = (\alpha C_{in} + \beta)^\gamma$
- α is known as **gain** (exposure)
- β is known as **bias** (offset)
- γ maps to a non-linear curve (**gamma** correction)

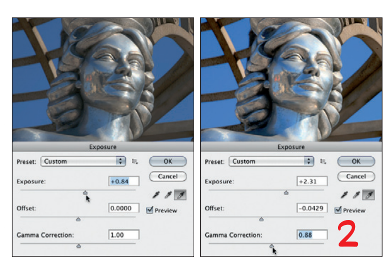


Image of Photoshop from Christian Bloch - The HDRI Handbook 2.0

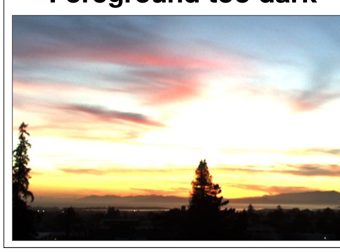
Dynamic Range

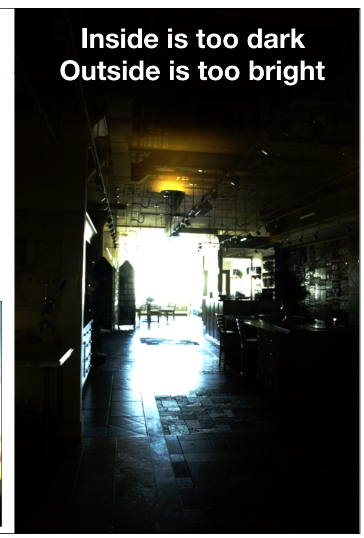
The World is a High Dynamic Range (HDR)

The World is a High Dynamic Range (HDR)

Examples

Sun overexposed
Foreground too dark



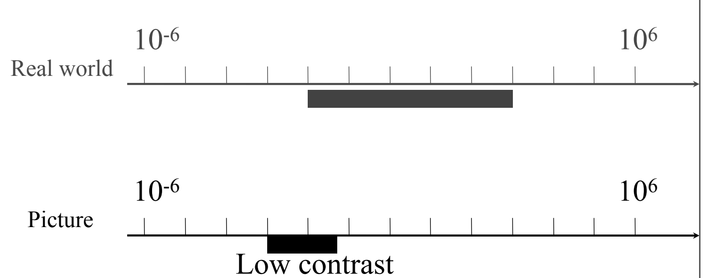


Dynamic Range in Displays?

- Range of pure black vs. pure white?

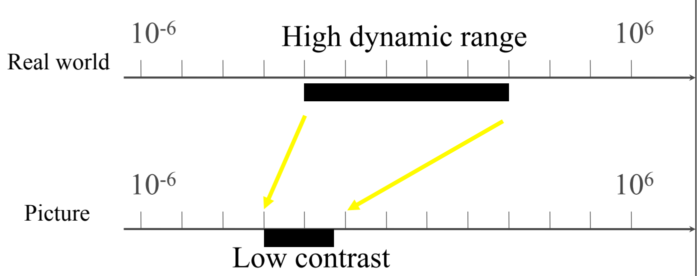
Dynamic Range in Displays?

- **Typically 1: 20 or 1:50**
  - Black is ~ 50x darker than white
- **Max 1:500**

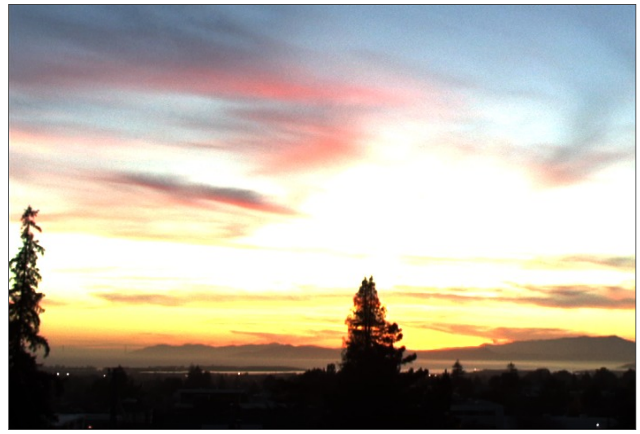


Problem: Displaying the Information

- Problem: How should we map scene radiances (up to 1:100,000) to display radiances (only around 1:100) to produce a satisfactory image?
- Goal: match limited contrast of the display medium while preserving details
- Solution: **Tone Mapping**



Without HDR + Tone Mapping

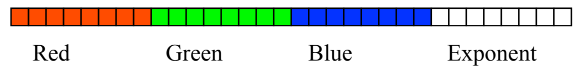


With HDR + Tone Mapping

Tone Mapping

Radiance RGBE Format (.hdr)

32 bits/pixel



(145, 215, 87, 149) =

(145, 215, 87) * 2^(149-128) =

1190000 1760000 713000

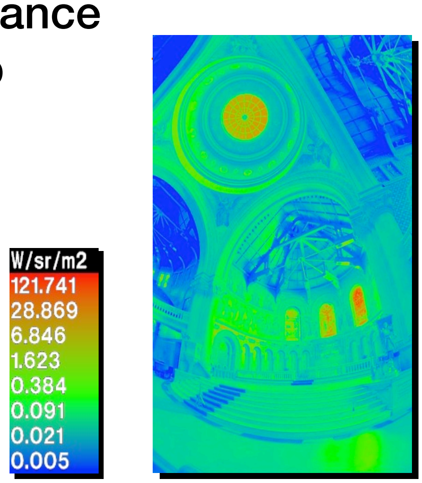
(145, 215, 87, 103) =

(145, 215, 87) * 2^(103-128) =

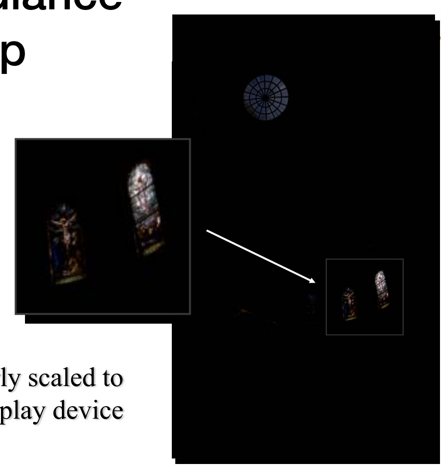
0.00000432 0.00000641 0.00000259

Ward, Greg. "Real Pixels," in Graphics Gems IV, edited by James Arvo, Academic Press, 1994

The Radiance Map

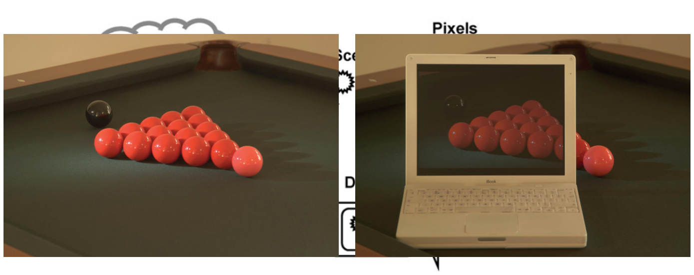


The Radiance Map



Linearly scaled to display device

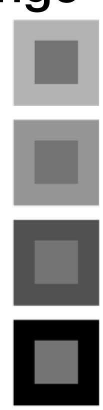
Approach: Visual Matching



- We do not need to reproduce the true radiance as long as it gives us a visual match.

Eyes and Dynamic Range

- We're sensitive to change (multiplicative)
  - A ratio of 1:2 is perceived as the same contrast as a ratio of 100 to 200
  - Use the log domain as much as possible
- But, eyes are **not** photometers
  - Dynamic adaptation (very local in retina)
  - Different sensitivity to spatial frequencies



Headlights are ON in both photos

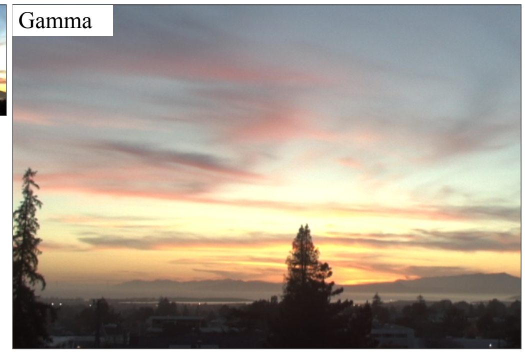
Can we just scale? Maybe!

- For a color image, try to convert the input (world) luminance L_w to a target display luminance L_d
- This type of scaling works (sometimes). In particular, it works best in the log and/or exponential domains

$$\begin{bmatrix} R_d \\ G_d \\ B_d \end{bmatrix} = \begin{bmatrix} L_d \dfrac{R_w}{L_w} \\ L_d \dfrac{G_w}{L_w} \\ L_d \dfrac{B_w}{L_w} \end{bmatrix}$$

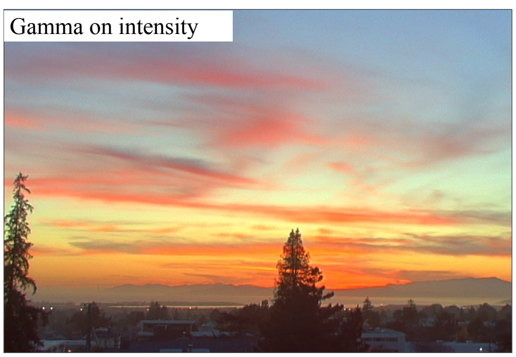
What scale value to use? How about Gamma compression

- $C_{out} = C_{in}^{\gamma}$, where $0 < \gamma < 1$ applied to each R,G,B channel
- Colors are washed out, why?



Gamma compression on Intensity

- Colors ok, but details in intensity are blurry

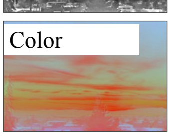


Bilateral Filtering

(slides from Frédo Durand)

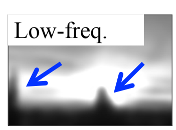
Oppenheim 1968, Chiu et al. 1993

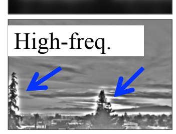
- Reduce contrast of low-frequencies
- Keep mid and high frequencies

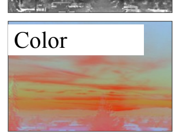


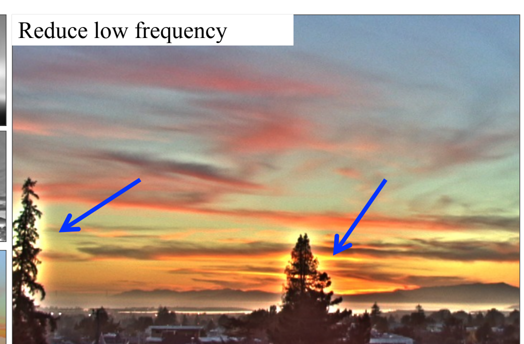
The halo nightmare

- **For strong edges**
- **Because they contain high frequency**



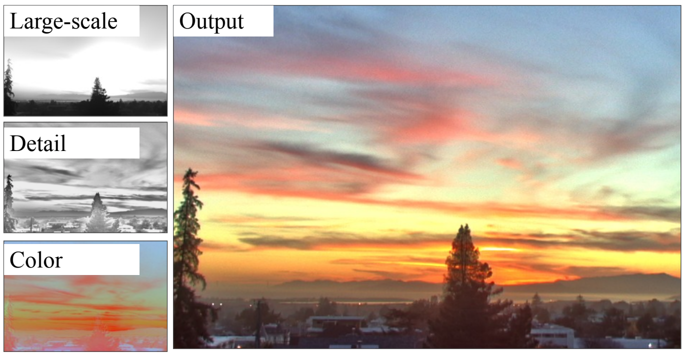






Our approach

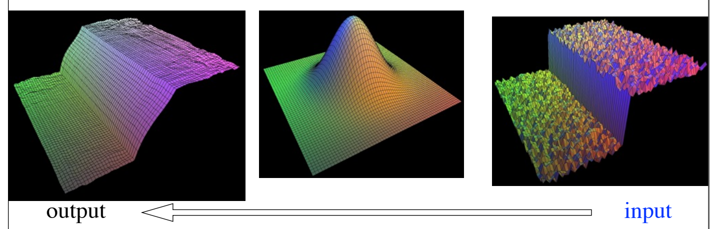
- Do not blur across edges
- Non-linear filtering



Start with Gaussian filtering

- Here, input is a step function + noise

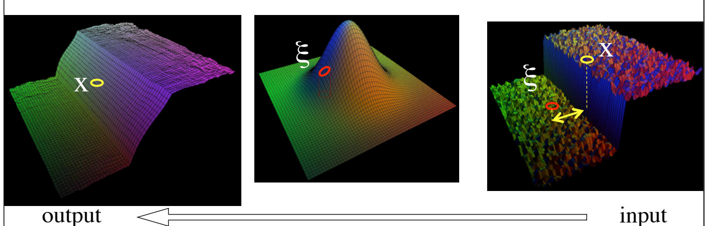
$$J = f \otimes I$$



Gaussian filter as weighted average

- Weight of ξ depends on distance to x

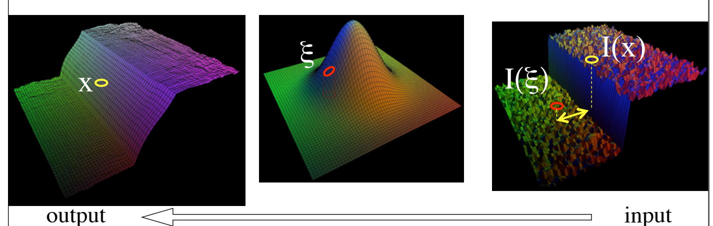
$$J(x) = \sum_{\xi} f(x,\xi) \quad I(\xi)$$



The problem of edges

- Here, $I(\xi)$ "pollutes" our estimate J(x)
- It is too different

$$J(x) = \sum_{\xi} f(x,\xi) \quad I(\xi)$$



Principle of Bilateral filtering

[Tomasi and Manduchi 1998]

- Penalty g on the intensity difference

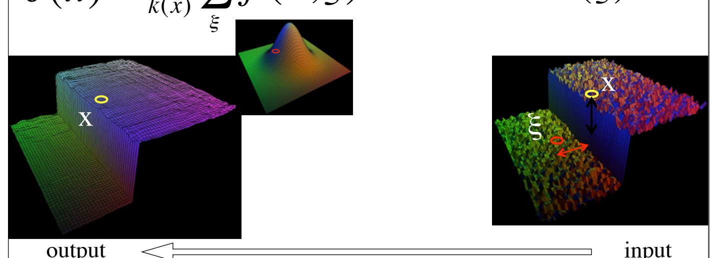
$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$

Bilateral filtering

[Tomasi and Manduchi 1998]

- Spatial Gaussian f

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

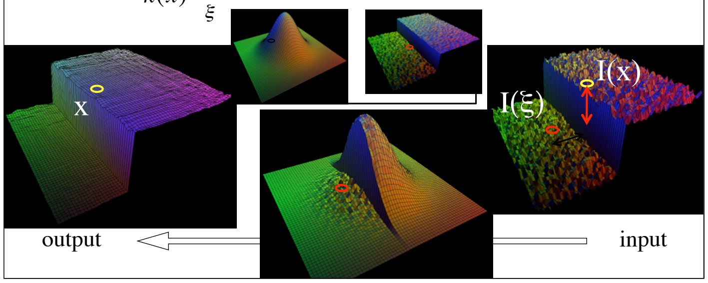


Bilateral filtering

[Tomasi and Manduchi 1998]

- Spatial Gaussian f
- Gaussian g on the intensity difference

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

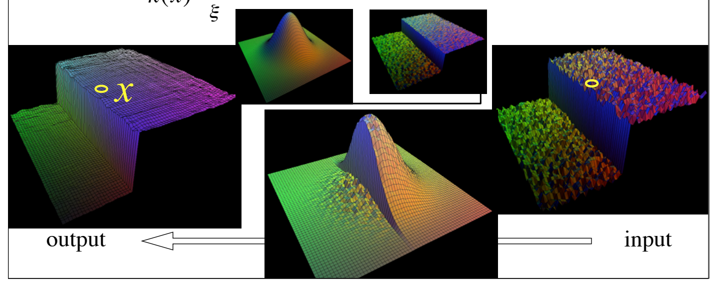


Normalization factor

[Tomasi and Manduchi 1998]

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

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

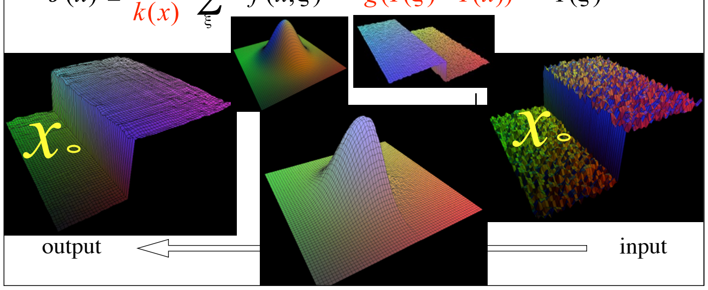


Bilateral filtering is non-linear

[Tomasi and Manduchi 1998]

- The weights are different for each output pixel

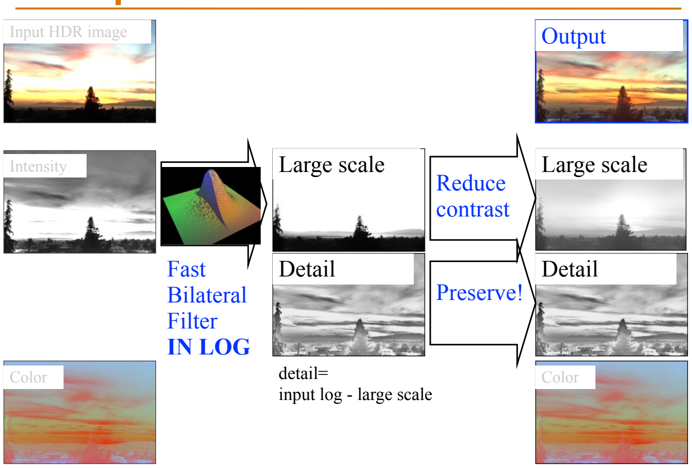
$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi) - I(x)) \quad I(\xi)$$

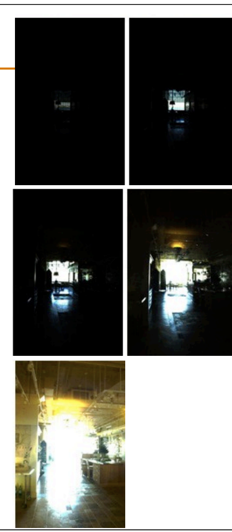


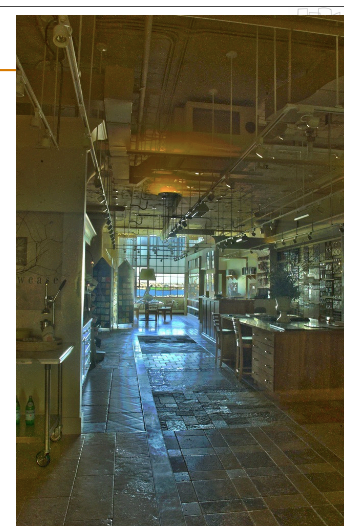
Log domain

- Very important to work in the log domain
- Recall: humans are sensitive to multiplicative contrast
- With log domain, our notion of "strong edge" always corresponds to the same contrast

Recap







Reading For next Lecture

- FOCG, Ch. 3.4, 20.1-20.2
- Compositing Digital Images. Thomas Porter and Tom Duff, SIGGRAPH Comput. Graph. 18(3): 253-259, 1984.
- See also optional readings!