HDR and the Bilateral Grid

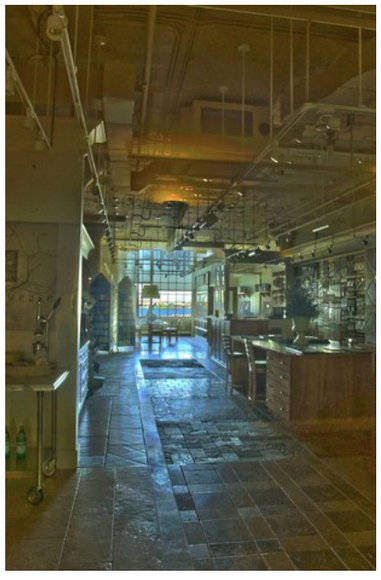
Fredo Durand
MIT EECS 6.8370/6.8371

Today

- Multiple-exposure High-Dynamic-Range imaging



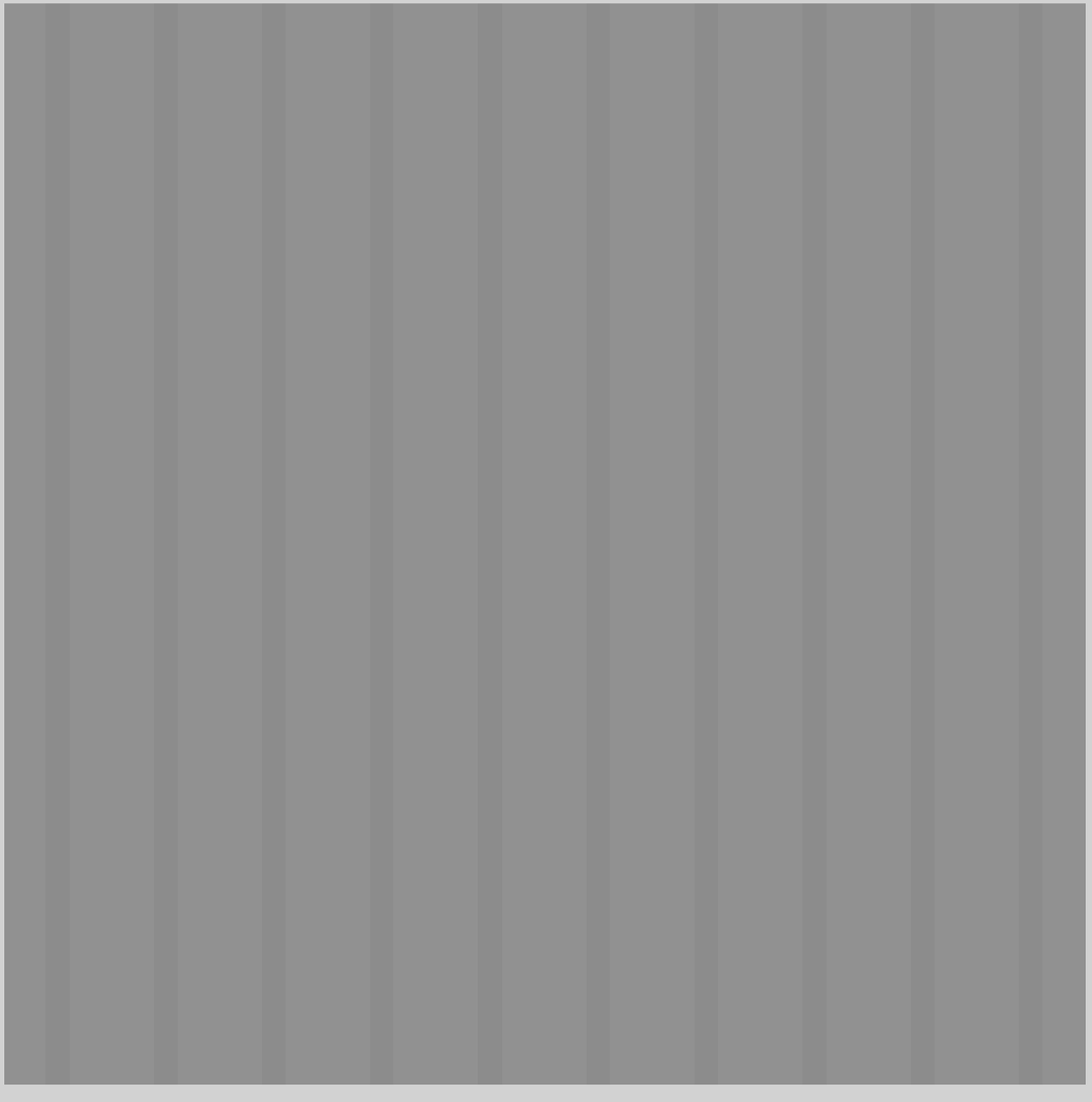
- **Tone mapping using the bilateral filter**

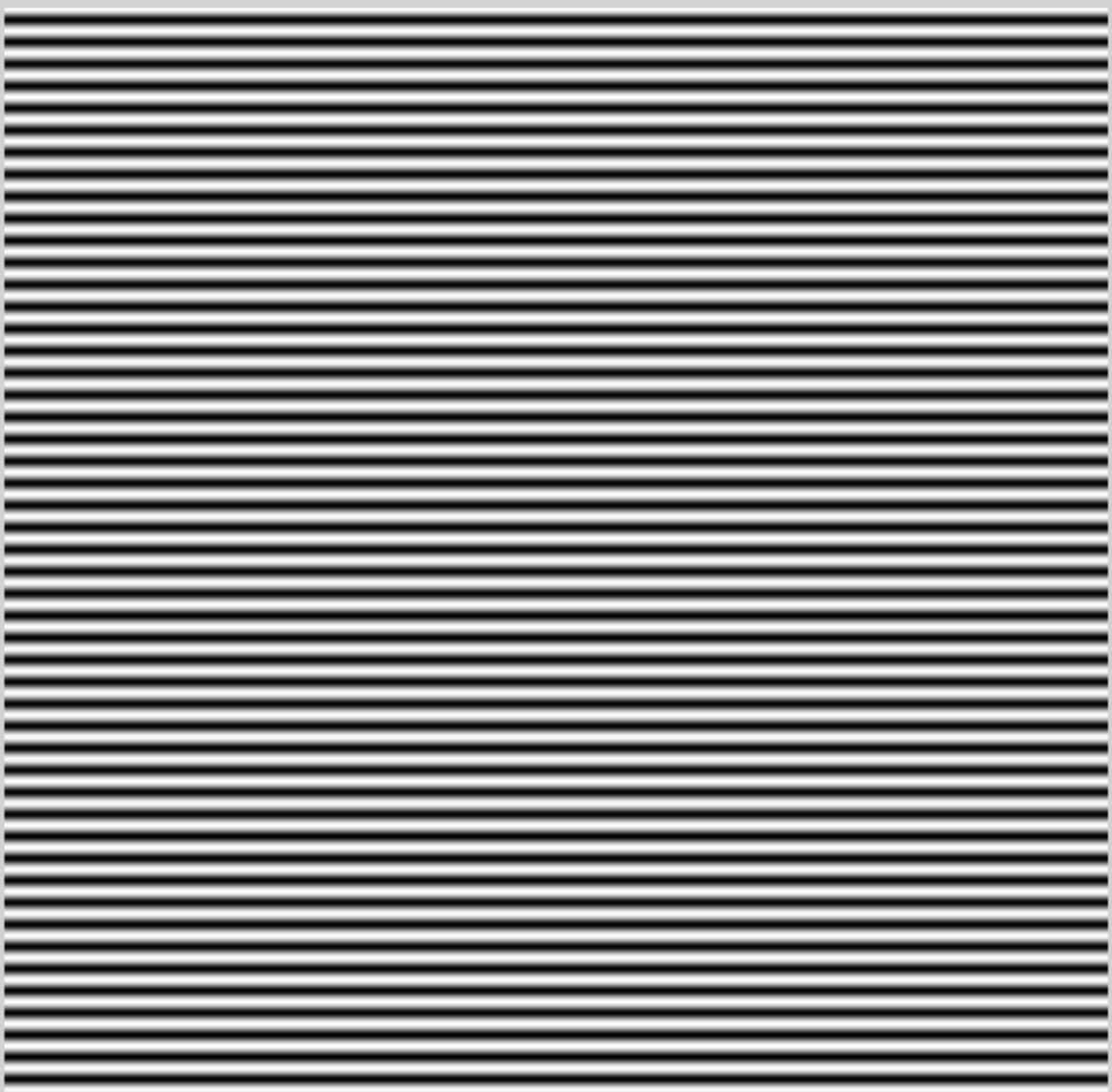


Spatial color vision

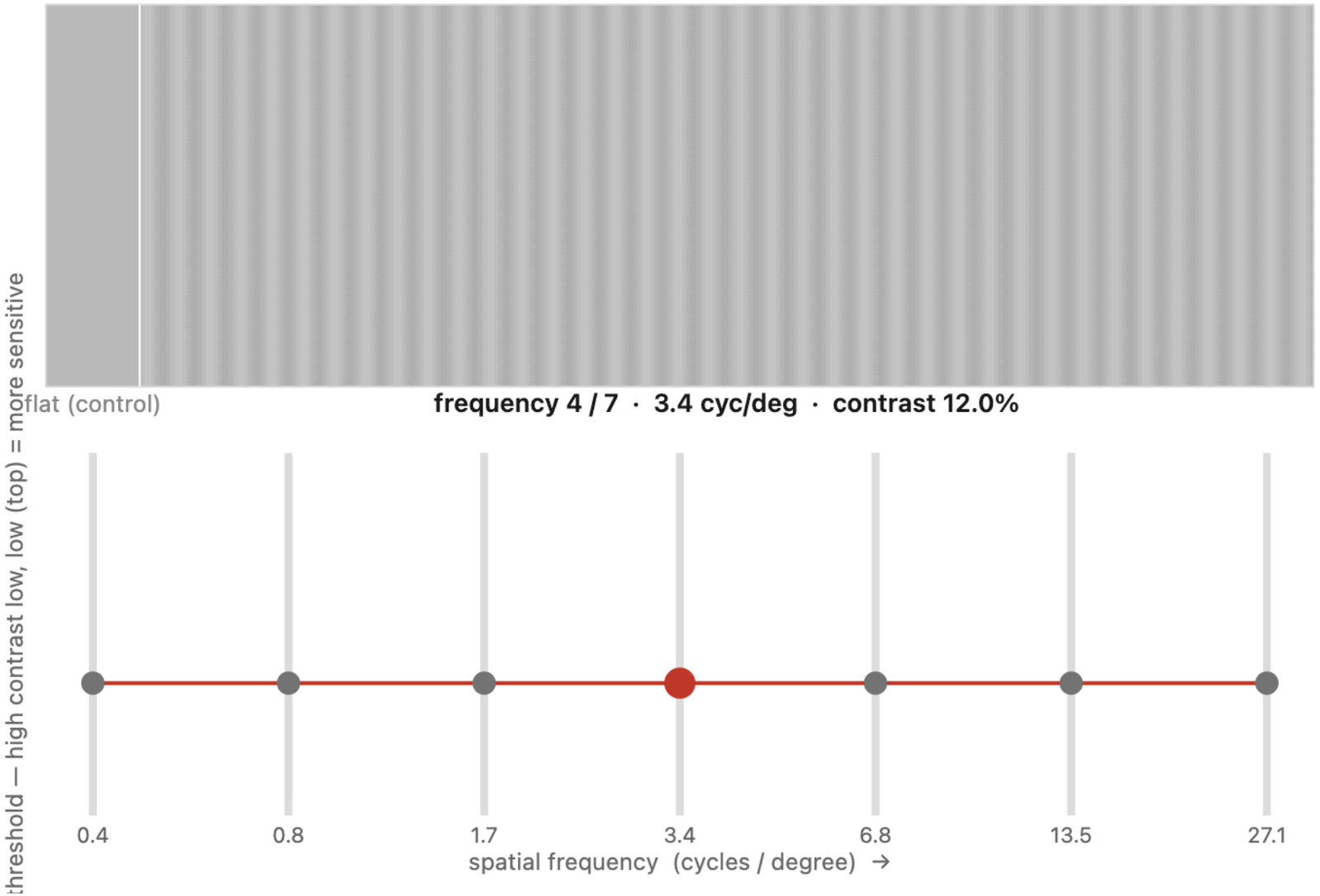
Contrast Sensitivity

- Sine Wave grating
- What contrast is necessary to make the grating visible?
- How does it change with spatial frequency



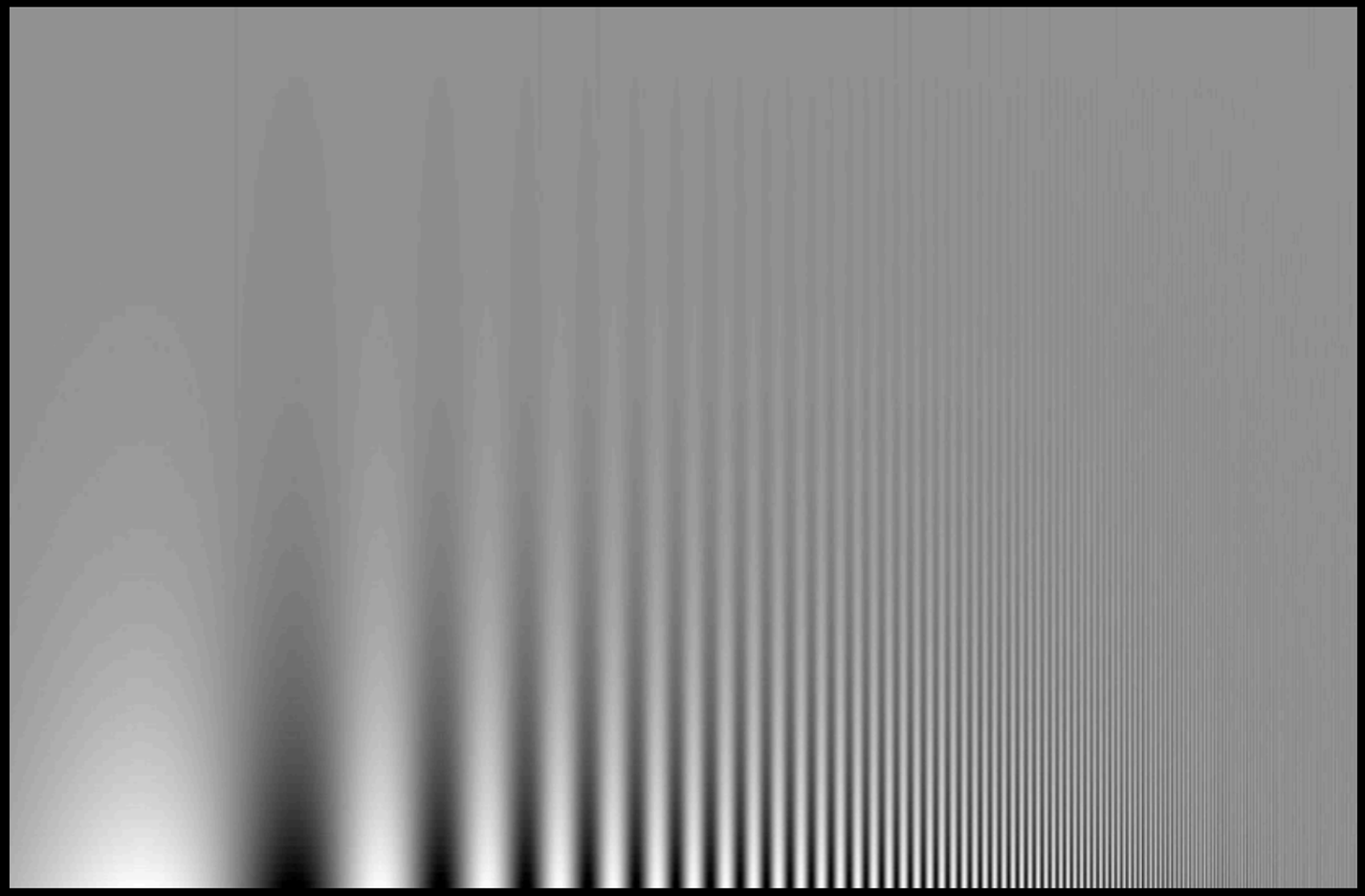


Demo: CSF by adjustment



Contrast Sensitivity Function

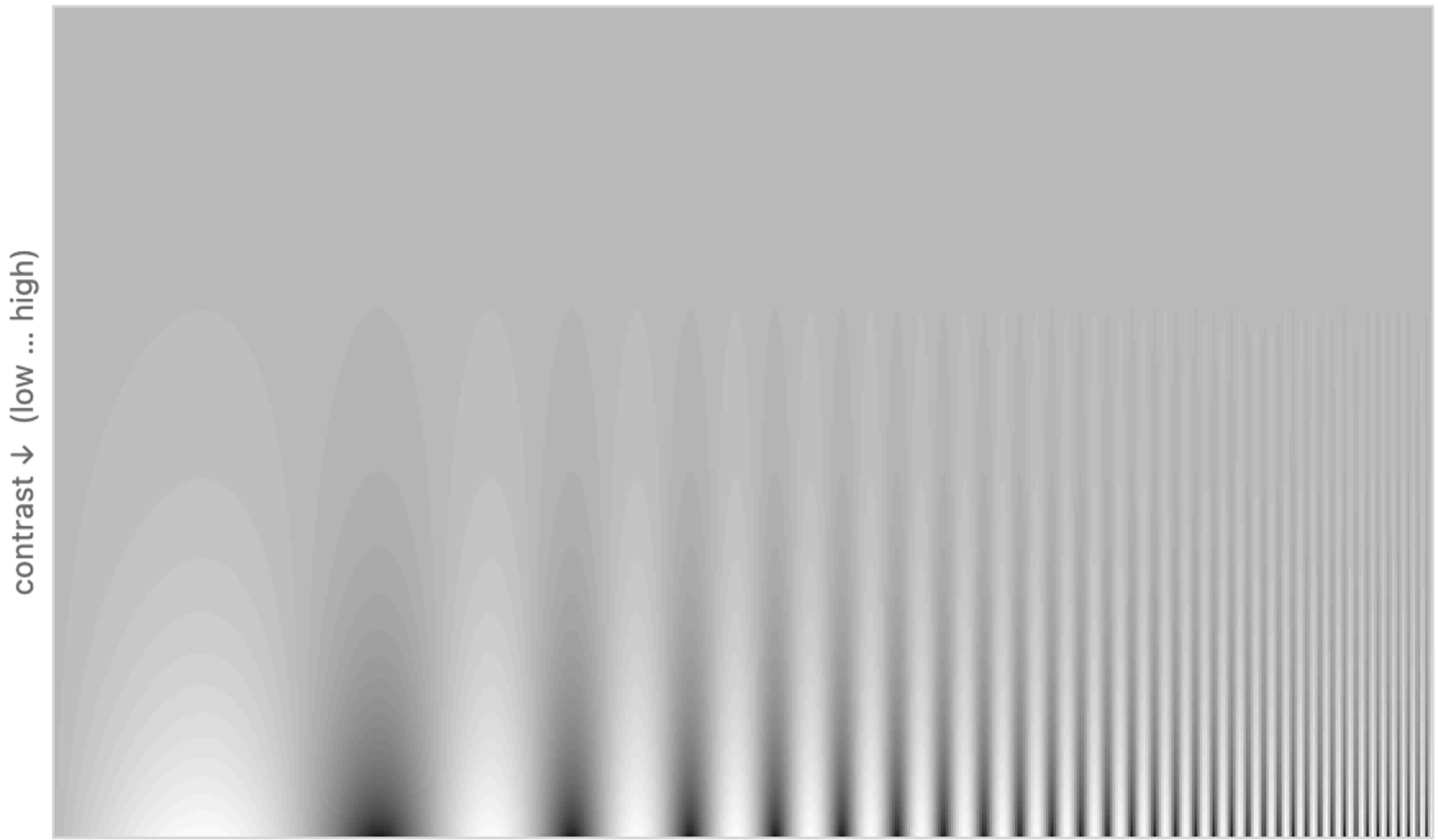
Campbell-Robson Chart



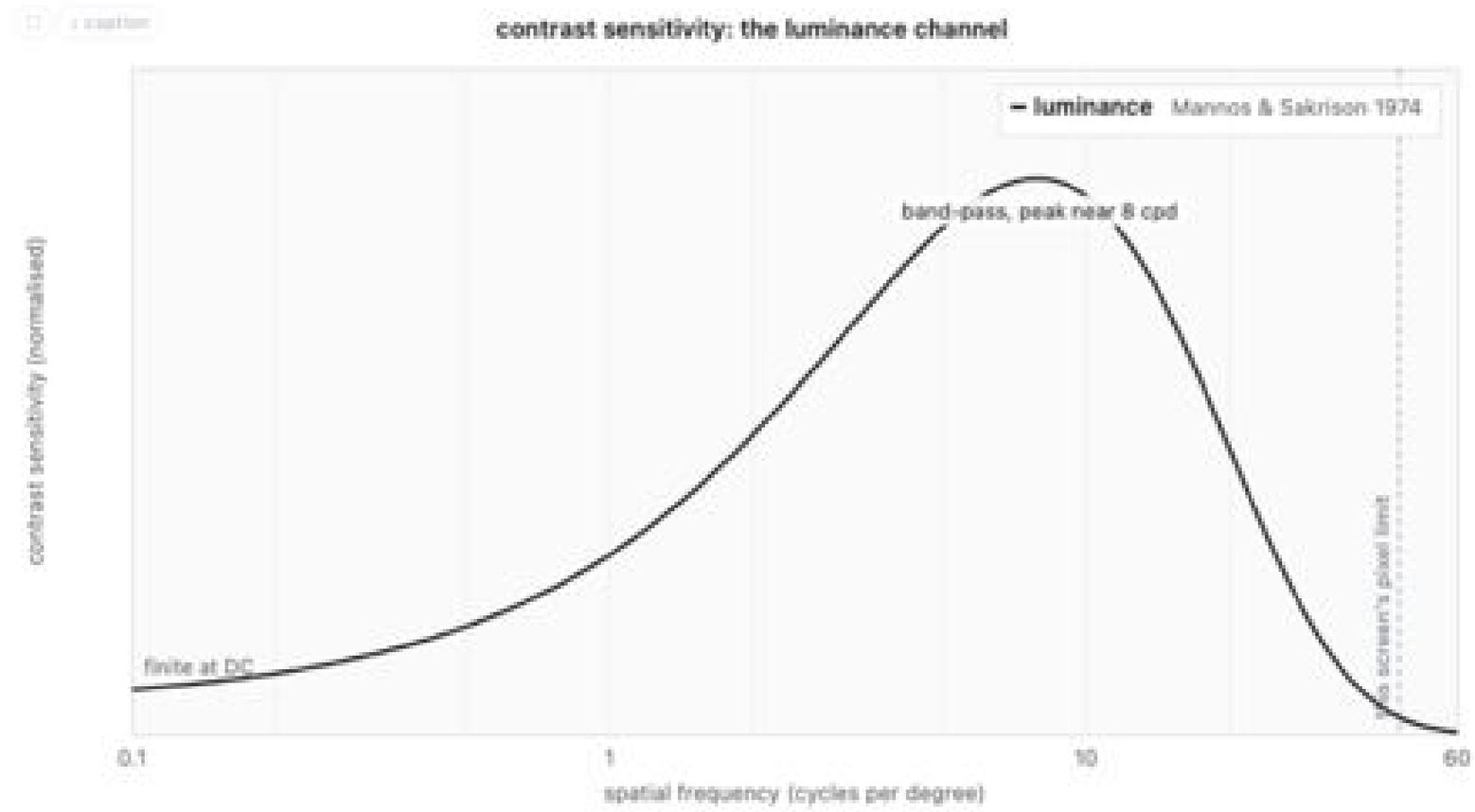
decreasing contrast

increasing spatial frequency

Demo: Campbell–Robson chart



Demo: The luminance contrast-sensitivity function, as a model: luminance

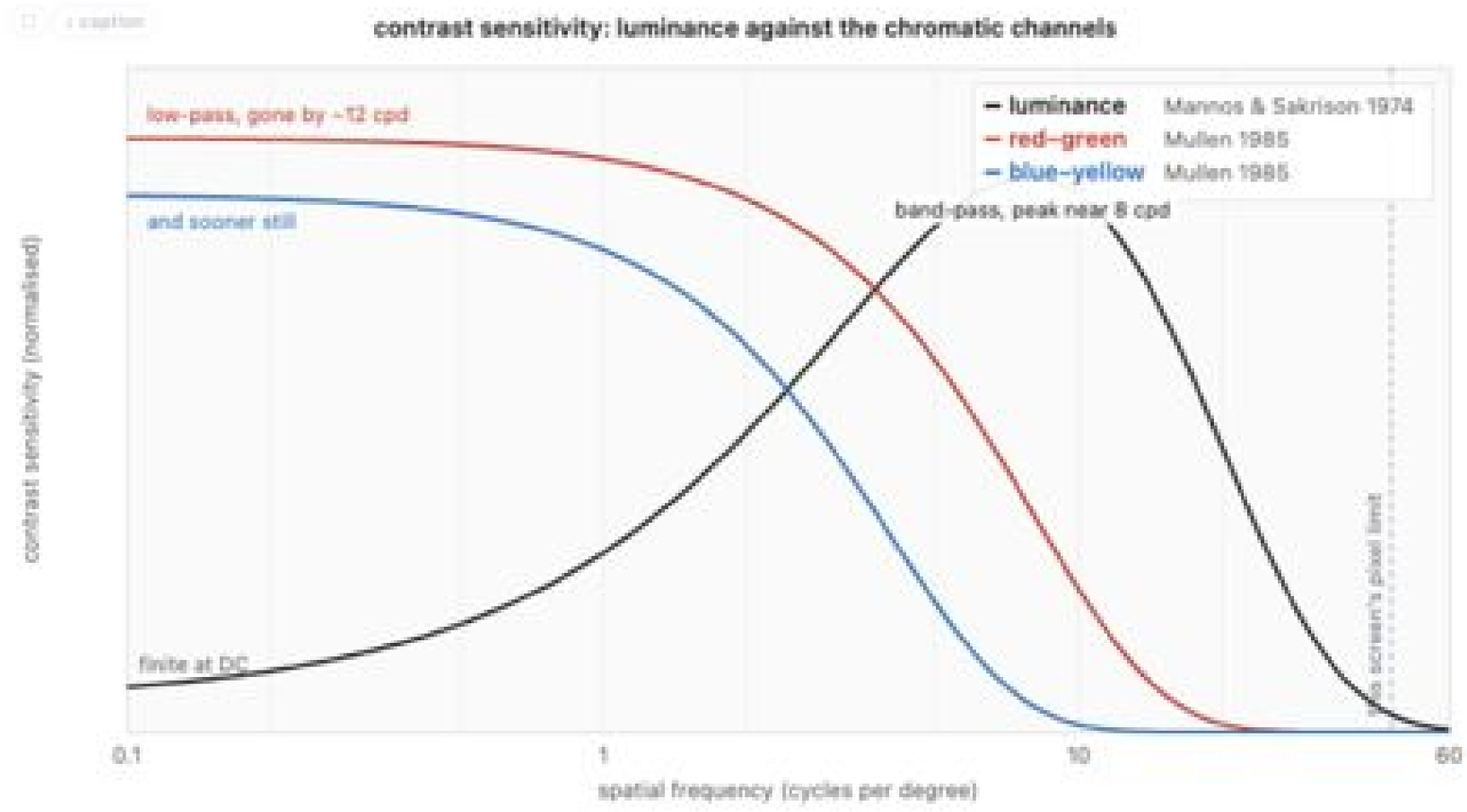


1 luminance | 2 + chromatic

viewing distance 0.60 m screen 220 ppi the landmarks reset

Demo courtesy of Frédo Durand, MIT.

Demo: The luminance contrast-sensitivity function, as a model: + chromatic



1 luminance | 2 + chromatic

viewing distance 0.60 m screen 220 ppi the landmarks reset

Demo courtesy of Frédo Durand, MIT.

Contrast Sensitivity Function (CSF)

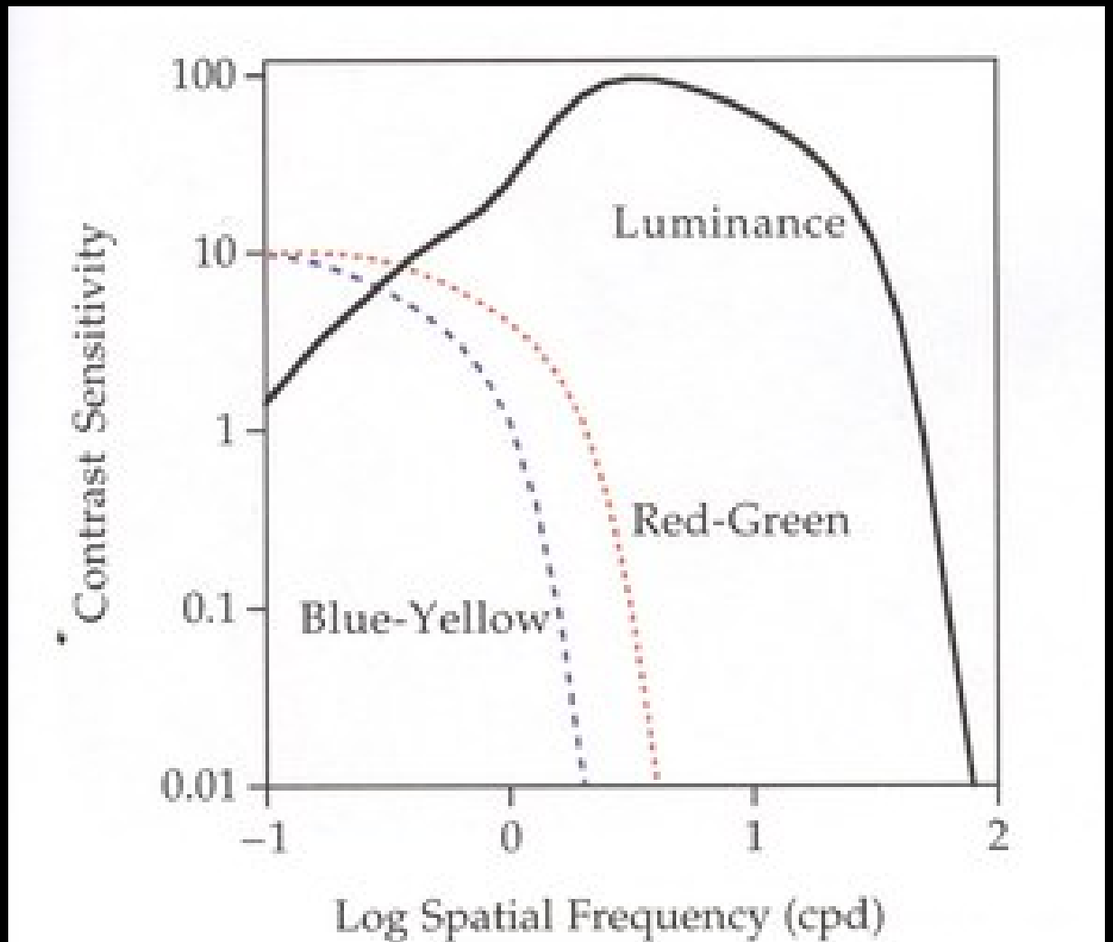


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

Photoshop demo



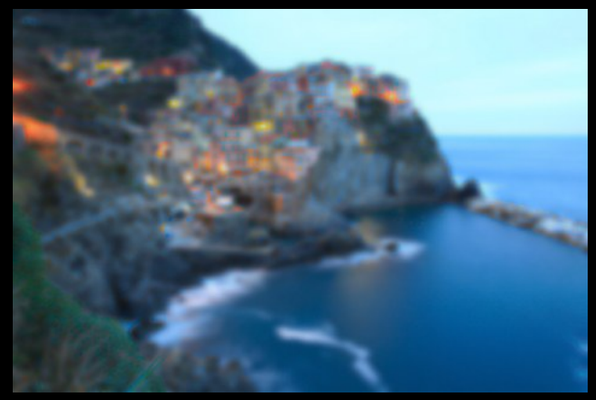
- Image > Mode > Lab color
- Go to channel panel, select Lightness
- Filter > Blur > Gaussian Blur , e.g. 4 pixel radius
  - very noticeable
- Undo, then select a & b channels



- Filter > Blur > Gaussian Blur , same radius
  - hardly visible effect



Original



Blur Lightness



Blur a & b

Demo: The gap, done to a photograph

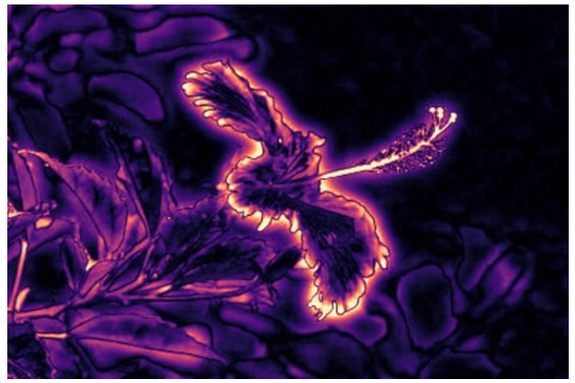
original



chroma blurred (σ=7), luma untouched



|difference| ×6



luma Y' (kept sharp)



chroma (original)



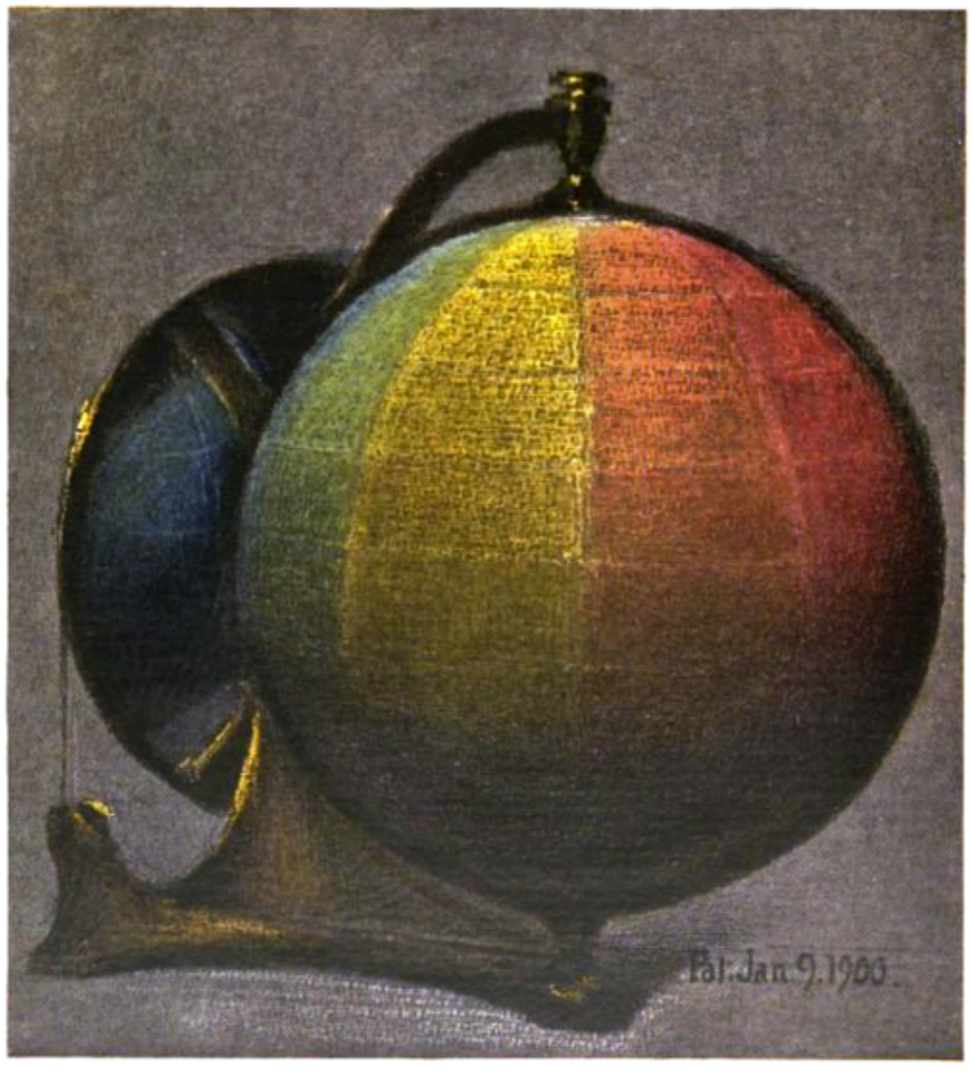
chroma (blurred σ=7)



Questions?



Questions?

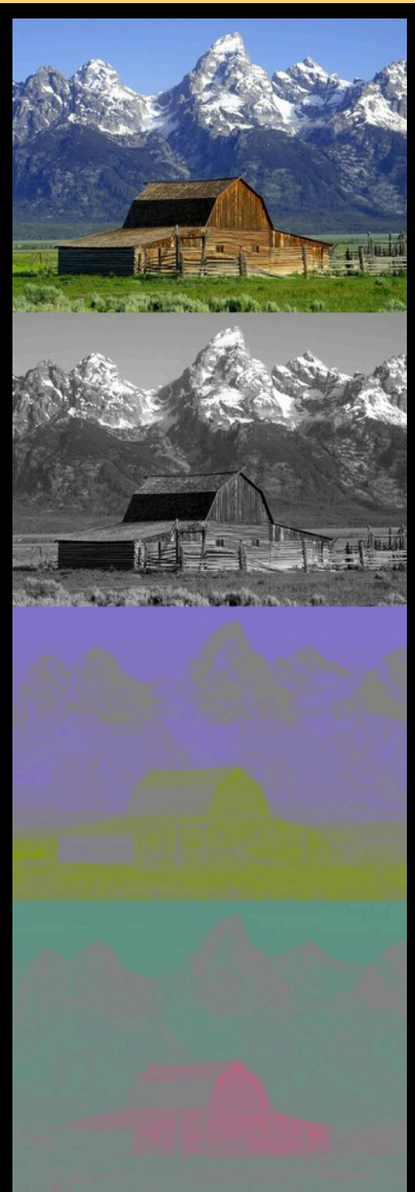


A BALANCED COLOR SPHERE

JPEG

Opponents and image compression

- JPG, MPG, television
- Color opponents instead of RGB
  - YCrCB, similar to YUV
- Compress color more than luminance
  - downsample by factor of two for jpeg
  - less bandwidth for TV
- Exploit contrast sensitivity function
  - Compress high frequencies more



Demo: Chroma subsampling

chroma subsampling — 4:2:0



original (4:4:4)



reconstructed (4:2:0)



Y′ (luma) (full-res)

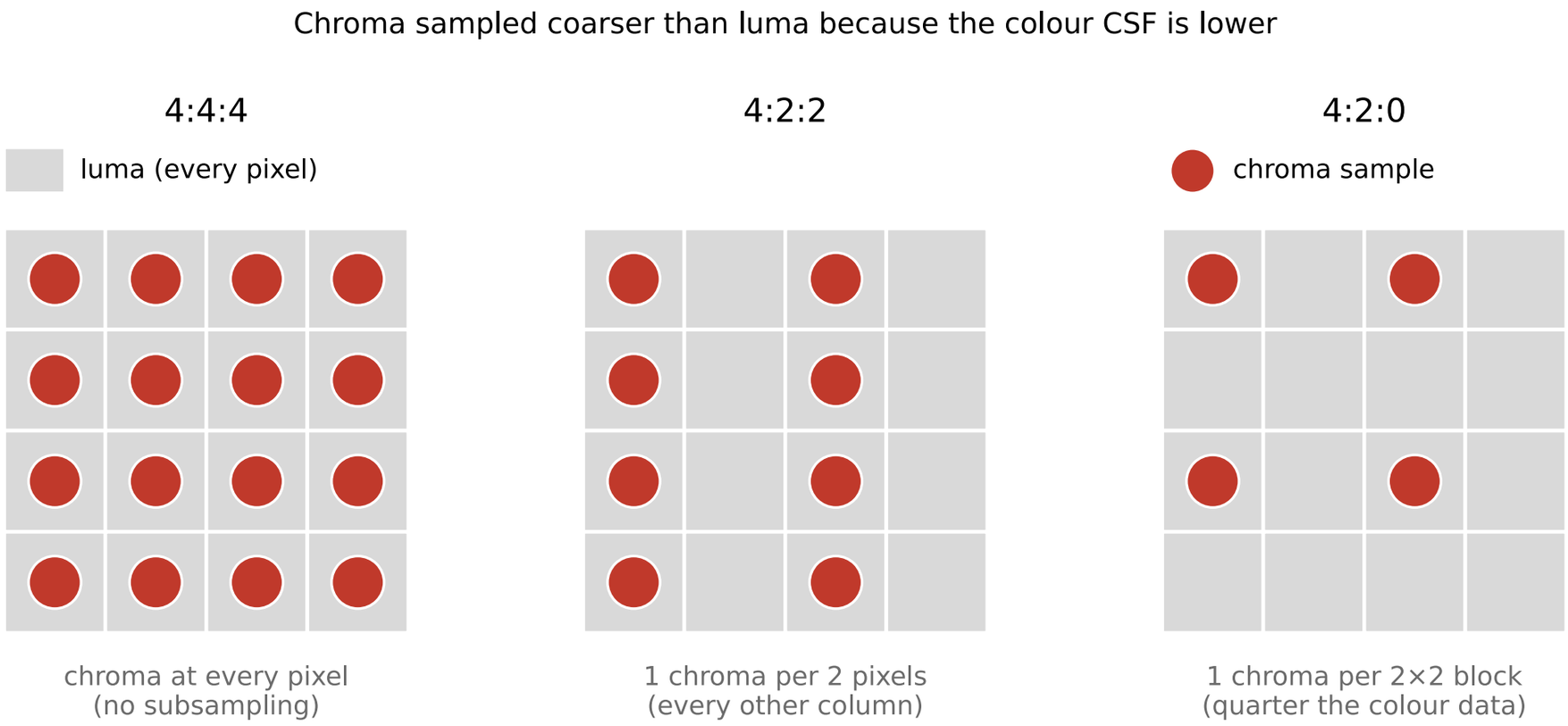


Cb (subsampled)



Cr (subsampled)

Figure 8.11.4: Chroma subsampling notation, on a 4×2 block of pixels



Chroma subsampling notation, on a 4×2 block of pixels. 4:4:4: every pixel keeps its own Cb and Cr (full color resolution). 4:2:2: chroma sampled every other column (half…

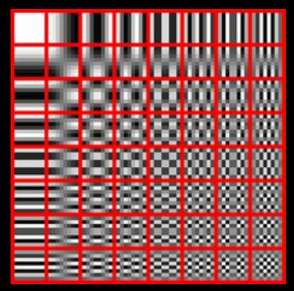
JPEG Compression

- convert to YCbCr
  - half the resolution for Cr & Cb
- Perform Discrete Cosine Transform
  to work in frequency space
  - Local DCT, 8x8 pixel blocks

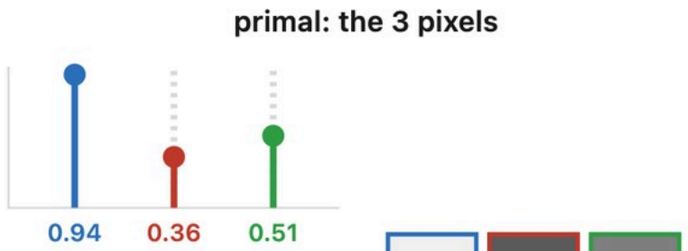




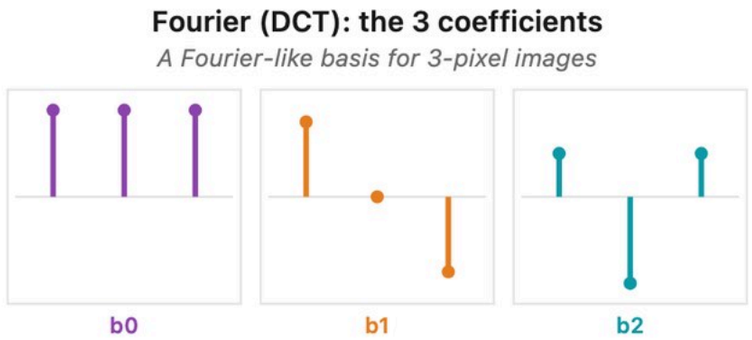




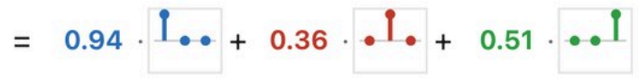
Demo: Fourier: three pixels (3-D)



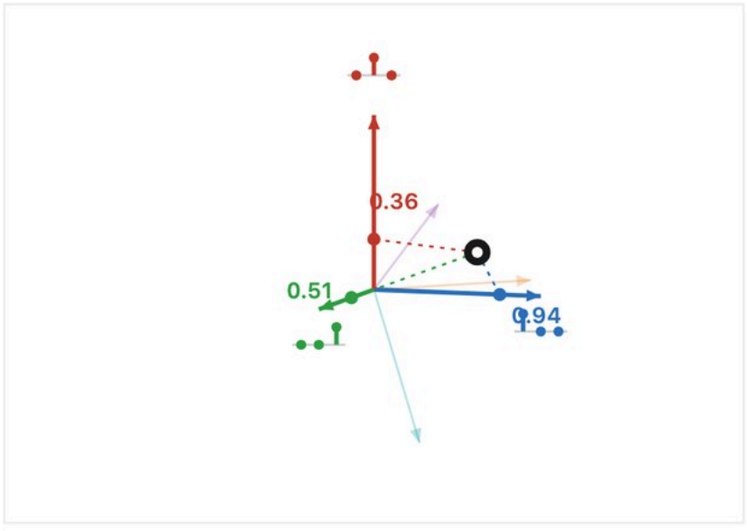


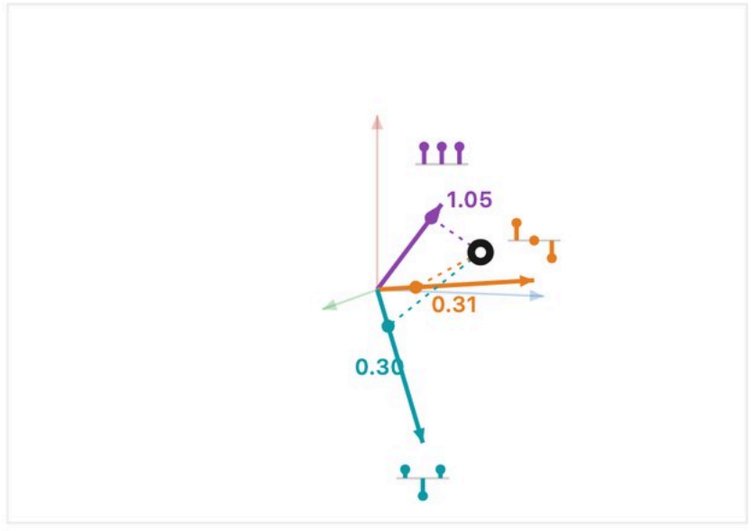


the same signal, written in the two bases:



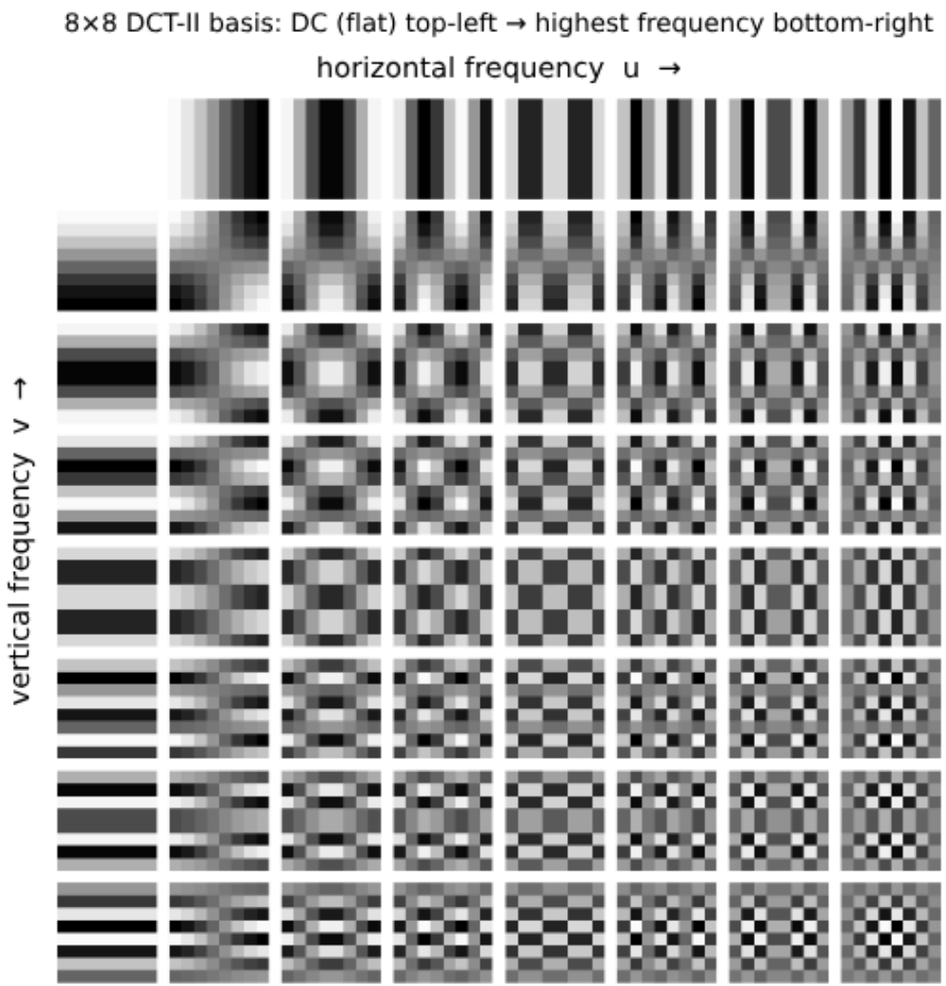






Drag a pixel bar or a DCT axis to change the signal; drag a view to orbit. Same viewpoint: the point holds still, the axes rotate.

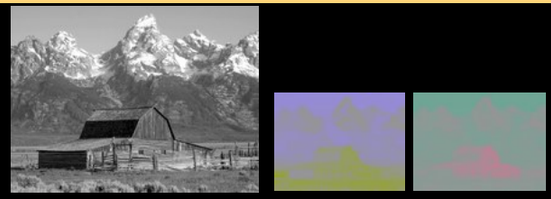
Figure 8.11.6: The 64 basis patterns of the 8×8 DCT, arranged in an 8×8 grid

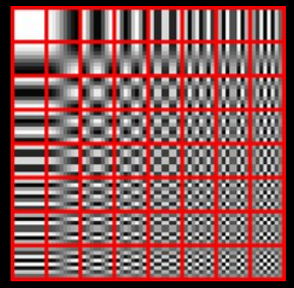


Top-left is the constant (DC) pattern, the block's average.

JPEG Compression

- convert to YCbCr
  - half the resolution for Cr & Cb
- Perform Discrete Cosine Transform
  to work in frequency space
  - Local DCT, 8x8 pixel blocks
- Use CSF for quantization
  - more bits for frequencies
    with more sensitivity (medium)
- Other usual coding tricks
  - entropy coding, smart order of blocks





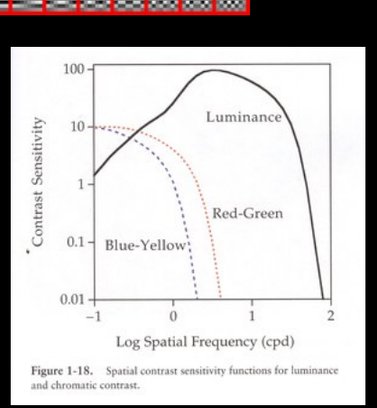
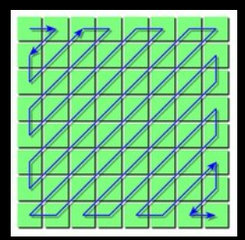
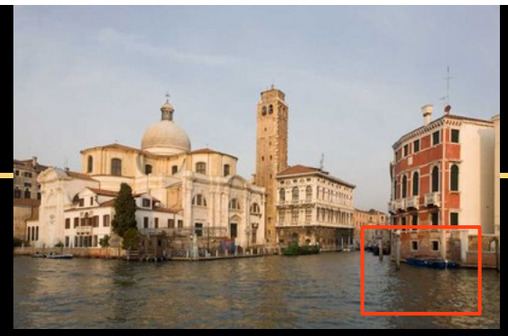


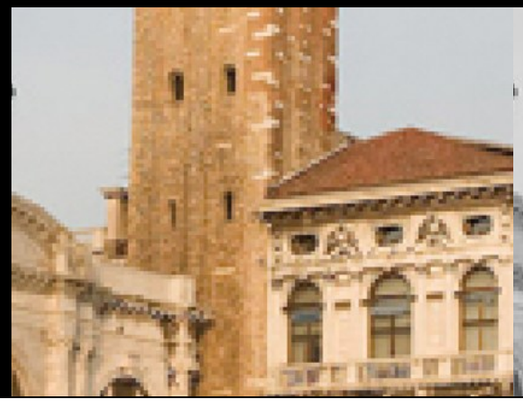
Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.



Example

- 800 x 533 image



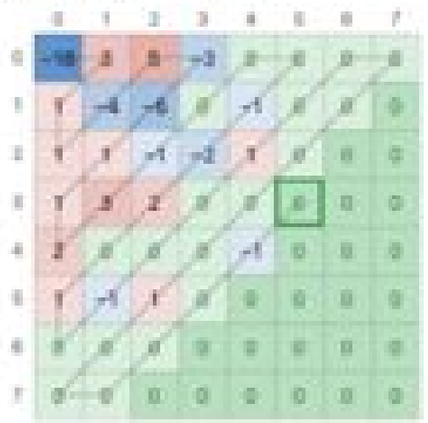
	RGB	Lightness	a
Low quality JPEG (0 in Photoshop, 172 KB)			
High quality JPEG (12 in Photoshop, 460 KB)			

Stage 5, **entropy coding: zig-zag, runs and Huffman codes**

input R′G′B′ | Y′ (luma) | Cb, 4:2:0 (16 × 16) | Cr, 4:2:0 (16 × 16) | decoded, quality 50



q_{uv} in zig-zag order



symbols, each with its bits (code + amplitude)

DC 6 −4 6 | (0, 3) 4 | (0, 1) 3 | (0, 1) 3 | (0, −4) 6 | (0, 5) 6 | (0, −3) 4 | (0, −6) 6 | (0, 1) 3 | (0, 1) 3 | (0, 2) 4 | (0, 3) 4 | (0, −1) 3 | (3, −1) 7 | (0, −2) 4 | (0, 2) 4 | (1, 1) 5 | (1, −1) 5 | (2, 1) 6 | (7, 1) 9 | (5, −1) 8 | EOB 4

Y′ block (1, 1): DC 6 + AC 97 + EOB 4 = 107 bits

the whole image: Y′ 1319 + Cb 134 + Cr 277 = 1730 bits = 1.69 bits per pixel, 14.2 : 1 against 24 bits per pixel

entropy-coded data only, headers not counted; decoded image: PSNR 21.9 dB

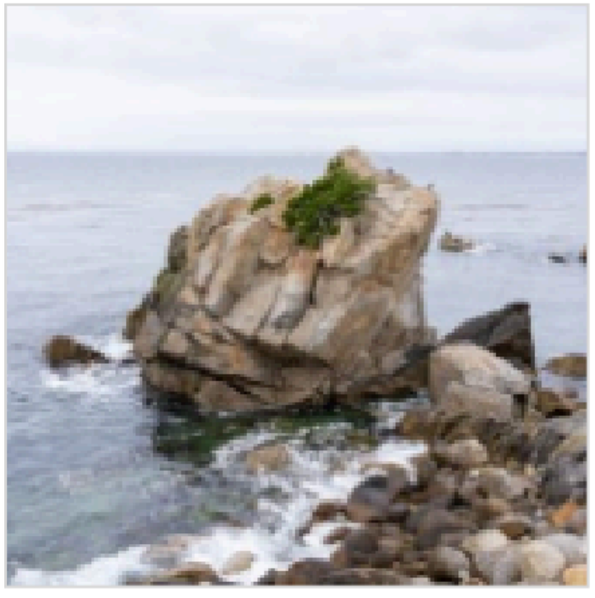
1 opponent color | 2 chroma subsampling | 3 8×8 DCT | 4 quantization | 5 entropy coding

input: hibiscus (photograph) | JPEG quality: 50 | reset

Demo courtesy of Frédo Durand, MIT.

Demo: Exposure round-trip (JPEG)

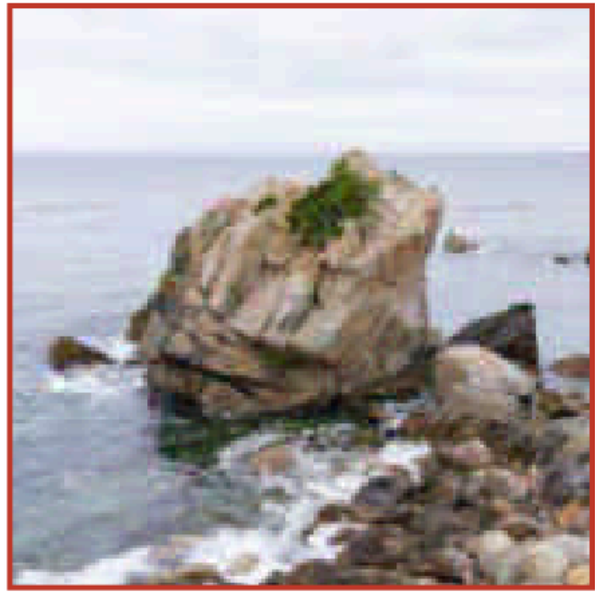
input (8-bit)



middle — ½ exposure (–1 EV)

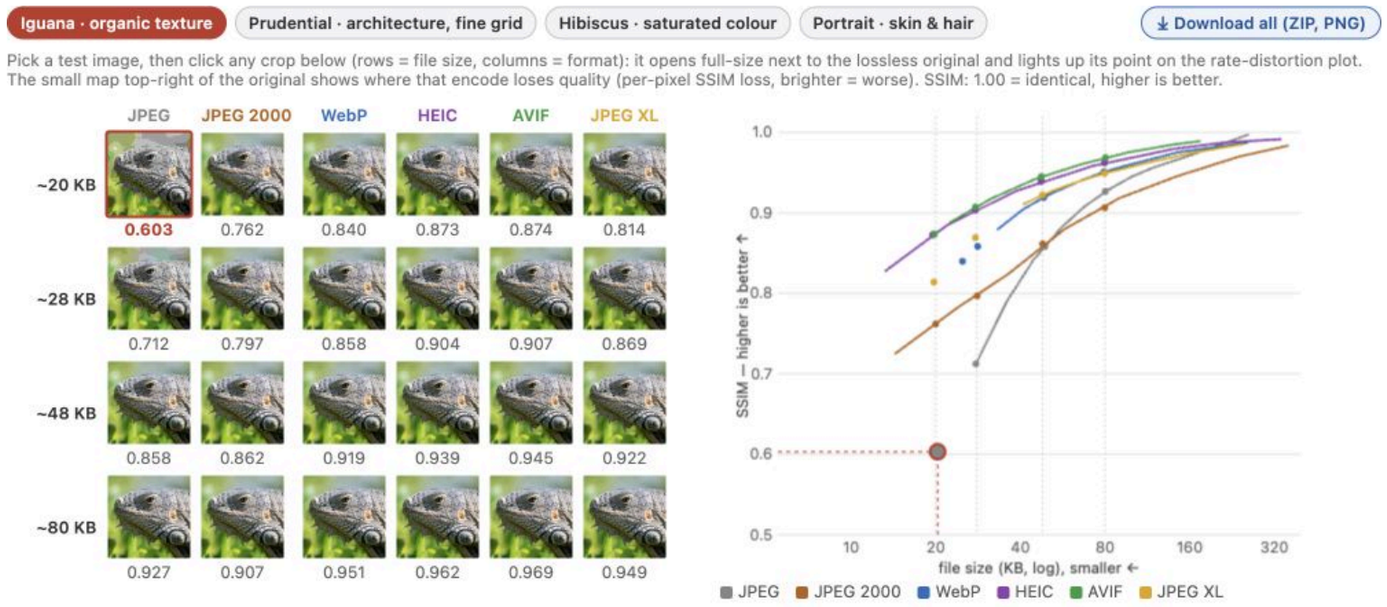


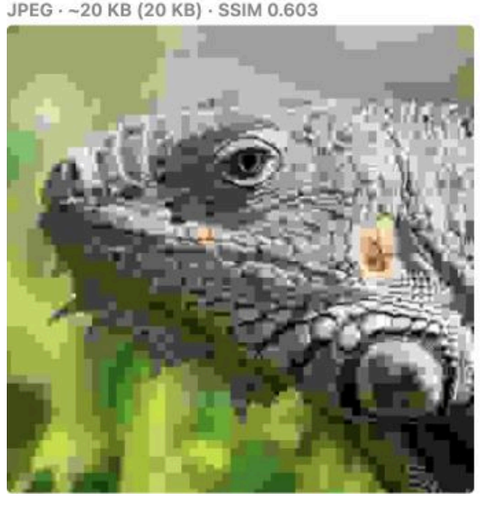
final — back to full



12 saves · 8-bit sRGB · RGB DCT quantization, quality 40%

Demo: Modern formats, measured







Questions?

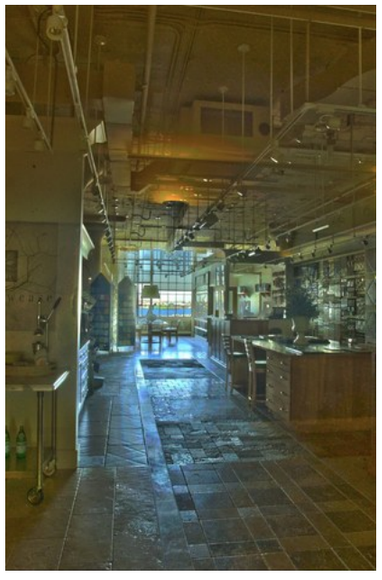
Tone mapping

Today

- Multiple-exposure High-Dynamic-Range imaging

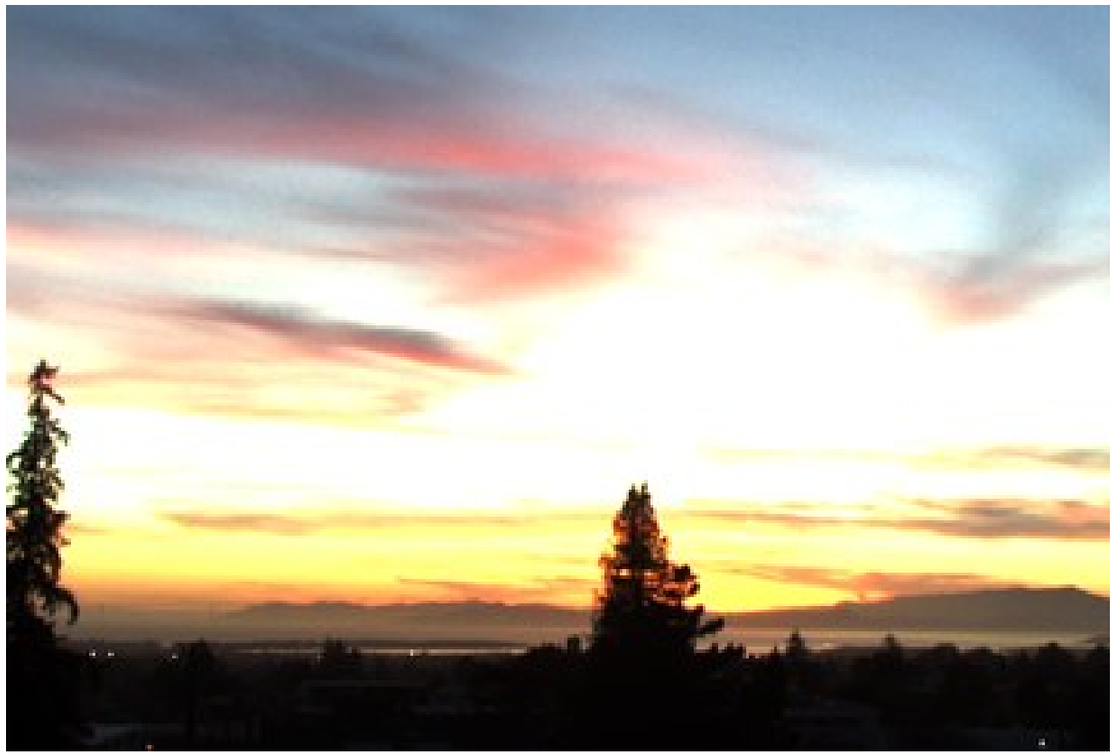


- **Tone mapping using the bilateral filter**



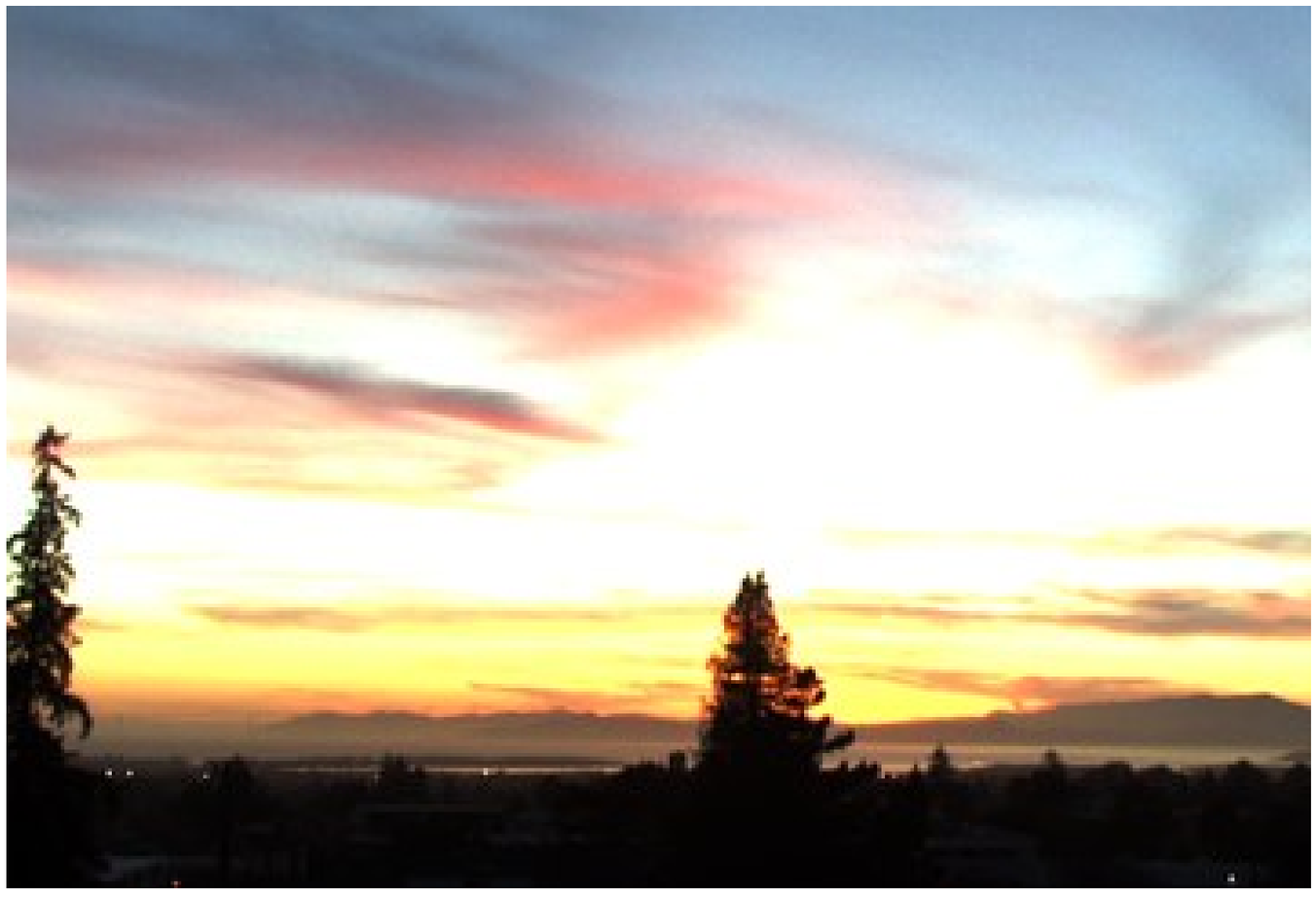
The second half: contrast reduction

- **Input: high-dynamic-range image**
  - (floating point per pixel)



Naïve technique

- **Scene has *1:10,000* contrast, display has *1:100***
- **Simplest contrast reduction?**



Naïve: Gamma compression

- $X \rightarrow X^{\gamma}$ **(where γ=0.5 in our case)**

applied independently on R, G & B

- **But… colors are washed-out. Why?**



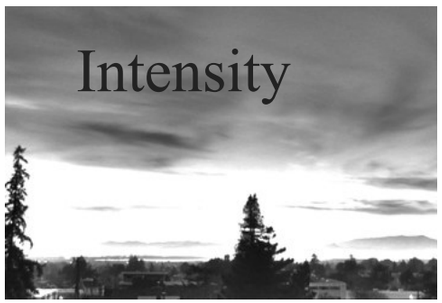


Careful: gamma is overloaded

Gamma encoding
vs Gamma correction
and other usages

Gamma compression on intensity

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







Contrast Sensitivity Function (CSF)

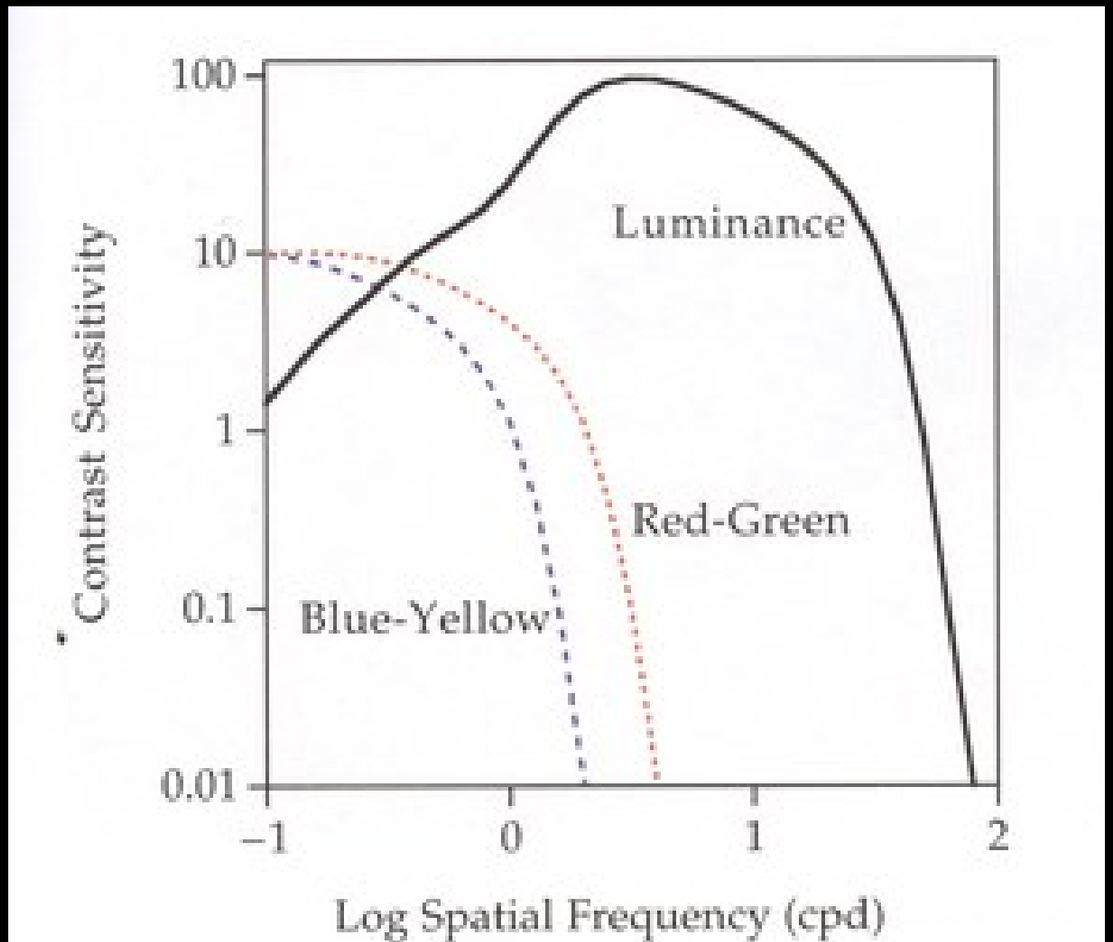
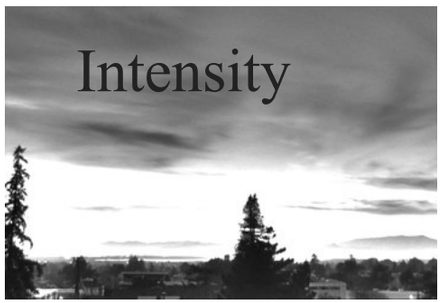


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

Gamma compression on intensity

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



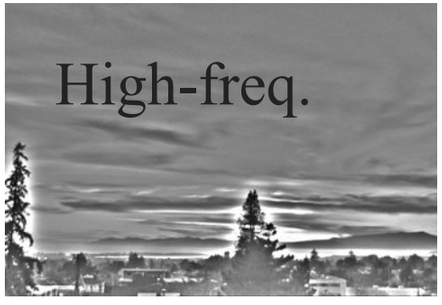




Oppenheim 1968, Chiu et al. 1993

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







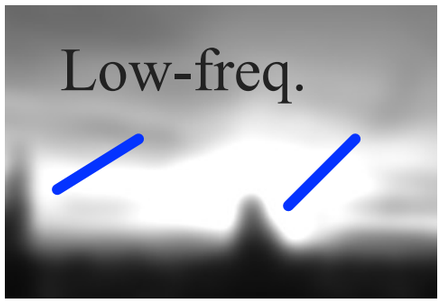


Homomorphic filtering

- **Oppenhein, in the sixties**
- **Images are the *product* of illumination and albedo**
  - Similarly, many sounds are the *product* of an envelope and a modulation
- **Illumination is usually slow-varying**
- **Perform albedo-illumination using low-pass filtering of the log image**
- **http://www.cs.sfu.ca/~stella/papers/blairthesis/main/node33.html**

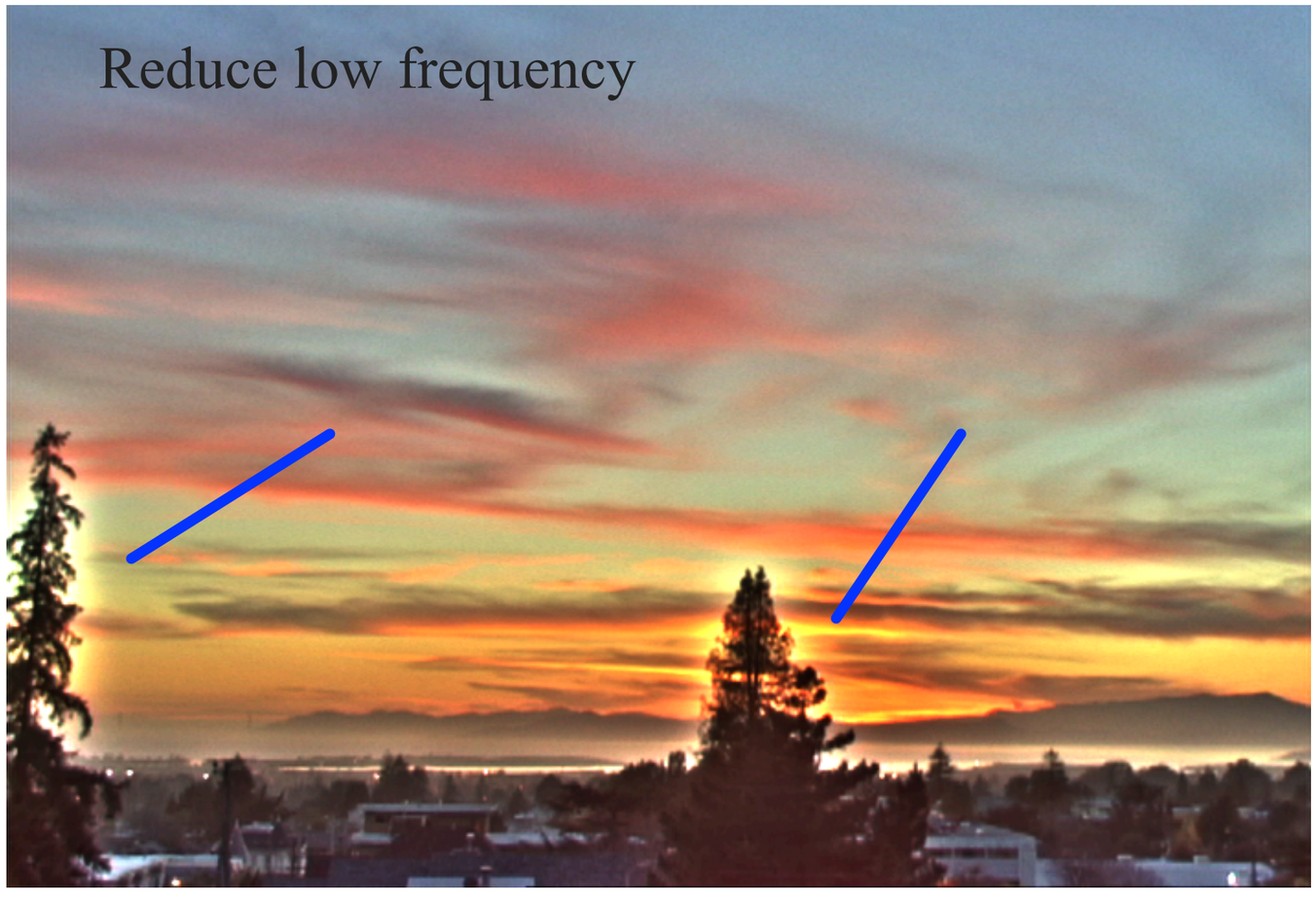
The halo nightmare

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









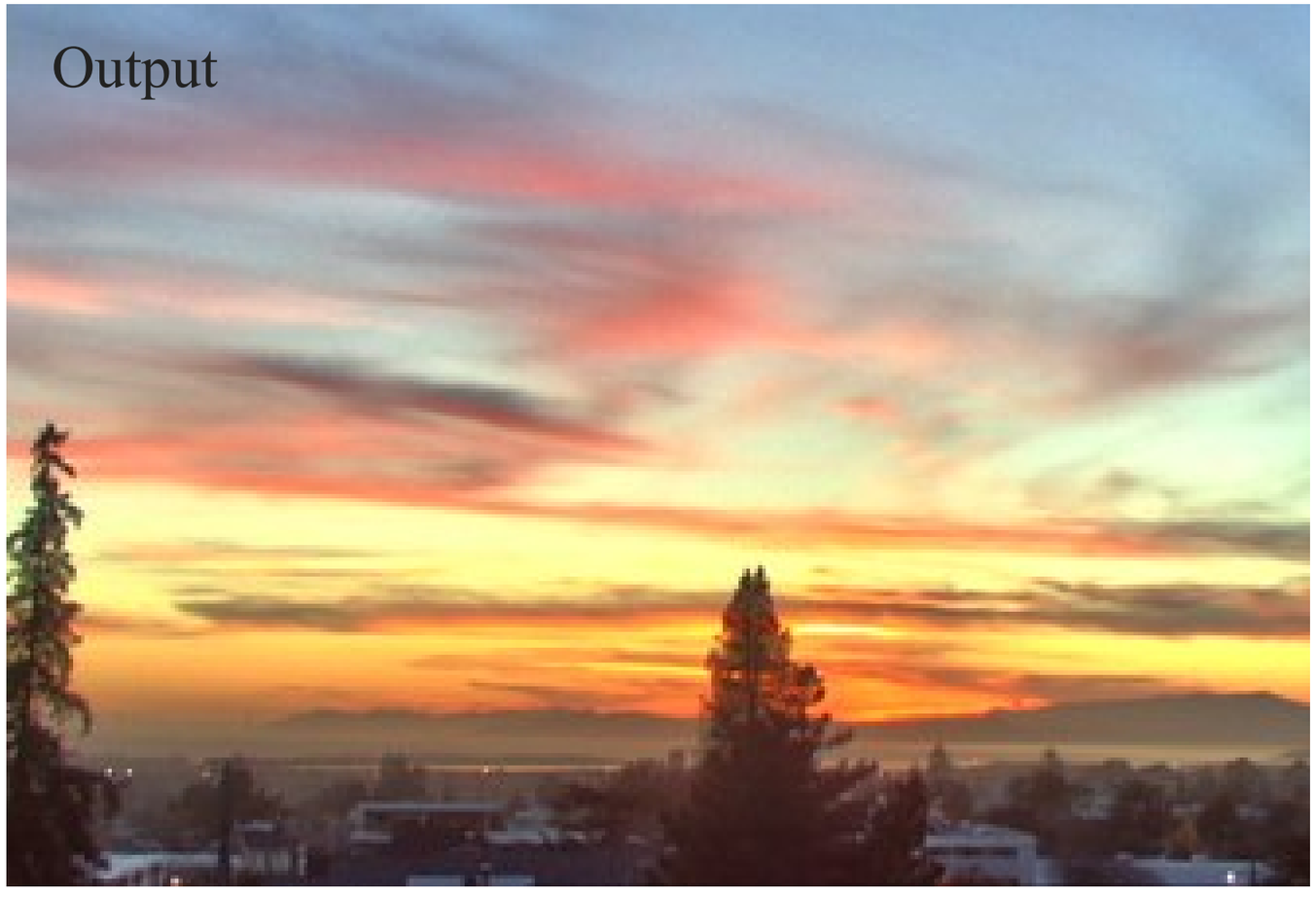
Our approach

- **Do not blur across edges**
- **Non-linear filtering**









Bilateral filter

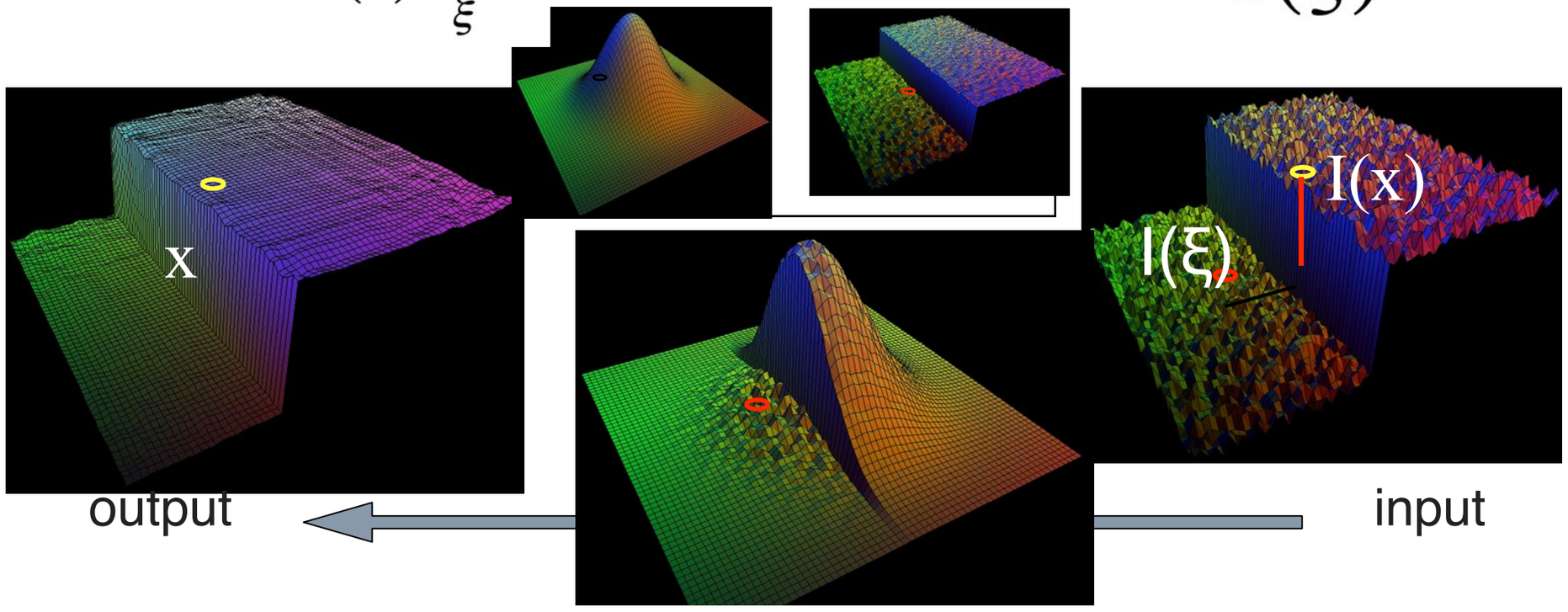
- **Tomasi and Manduci 1998]**
  - **http://www.cse.ucsc.edu/~manduchi/Papers/ICCV98.pdf**
- **Developed for denoising**
- **Related to**
  - SUSAN filter [Smith and Brady 95] **http://citeseer.ist.psu.edu/smith95susan.html**
  - Digital-TV [Chan, Osher and Chen 2001] **http://citeseer.ist.psu.edu/chan01digital.html**
  - sigma filter **http://www.geogr.ku.dk/CHIPS/Manual/f187.htm**
- **Full survey: http://people.csail.mit.edu/sparis/publi/2009/fntcgv/Paris_09_Bilateral_filtering.pdf**

Bilateral filtering

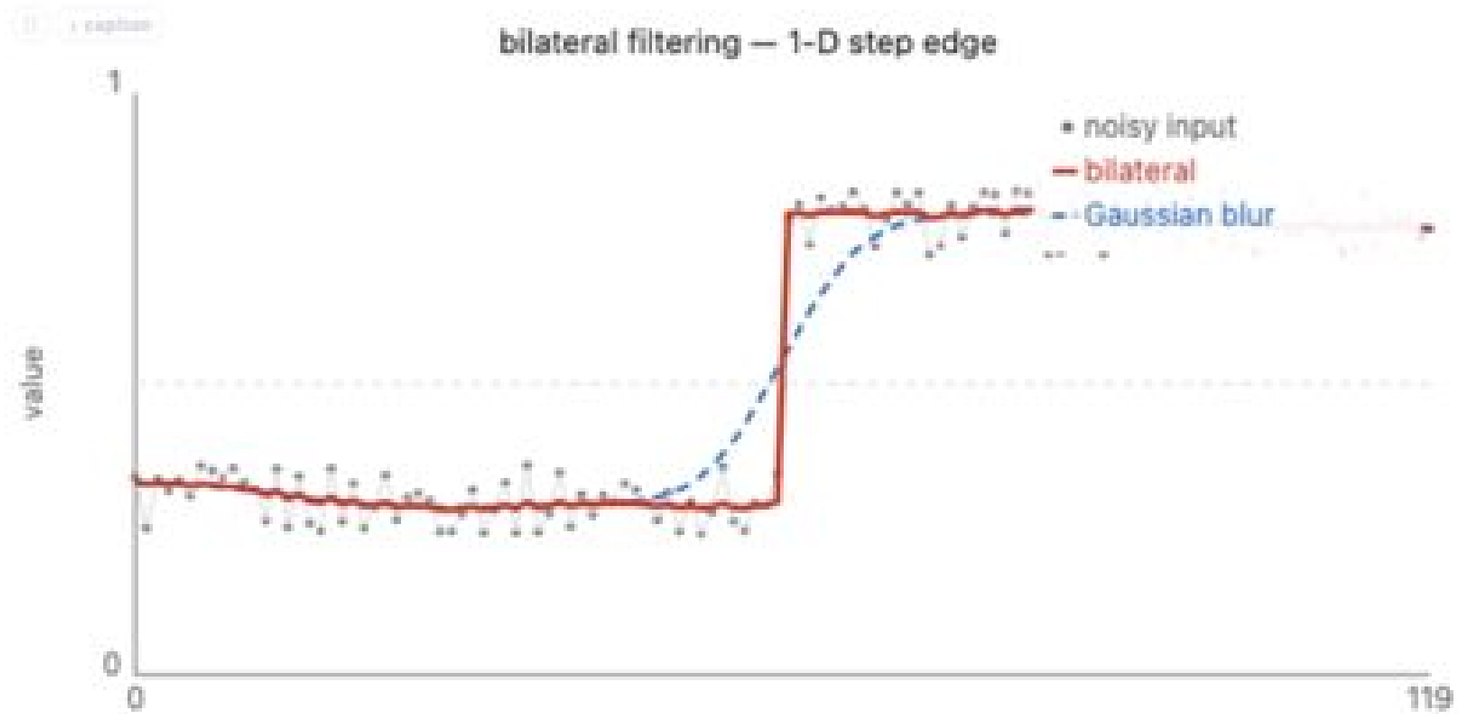
[Tomasi and Manduchi 1998]

- **Spatial Gaussian f**
- **Gaussian g on the intensity difference**

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



Demo: Bilateral 1-D: the bilateral filter



◀ 1 the noisy signal | 2 a Gaussian blur | 3 the bilateral filter ▶

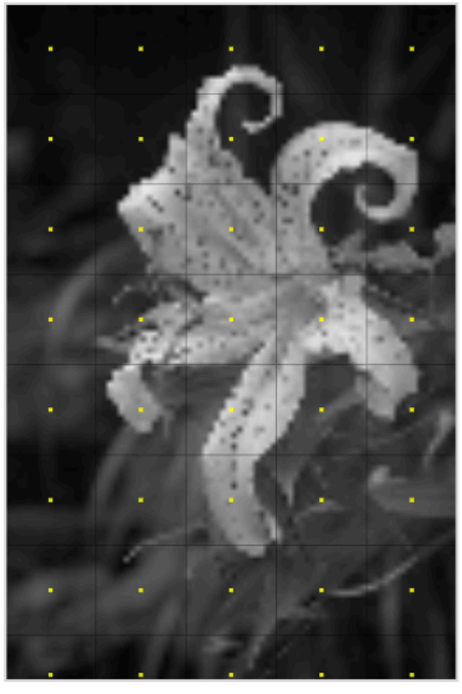
σ_s (spatial) 6.0 σ_r (range) 0.12

☑ overlay Gaussian blur reset

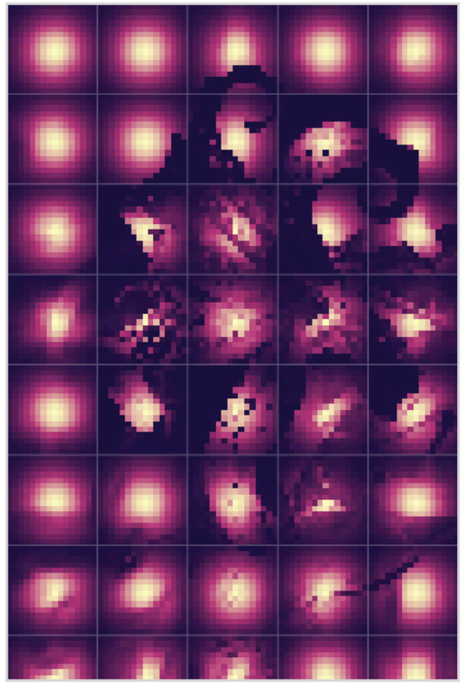
Demo courtesy of Frédo Durand, MIT.

Demo: Bilateral kernels

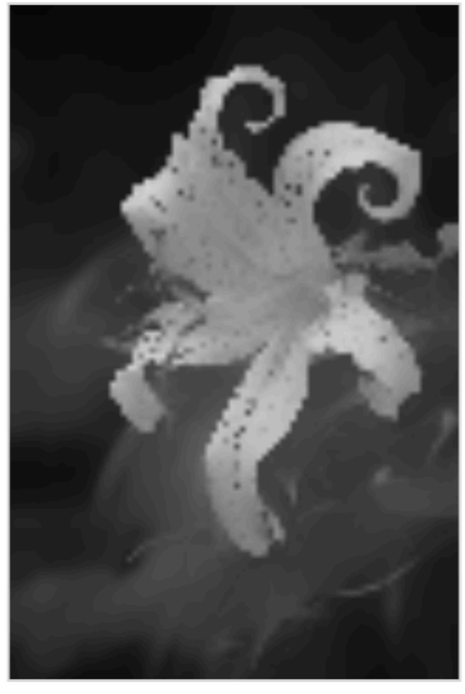
input



bilateral kernels



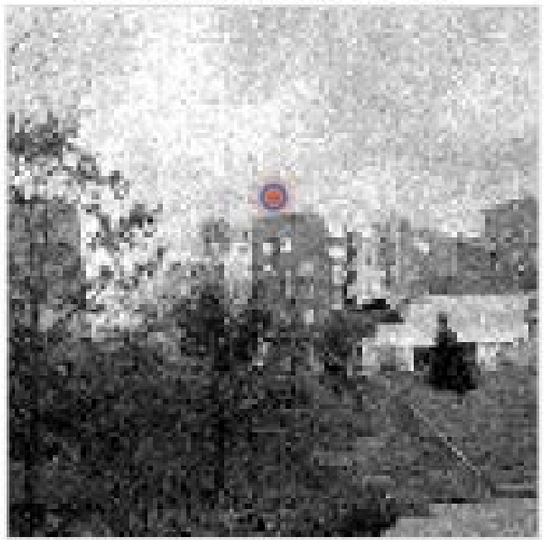
output



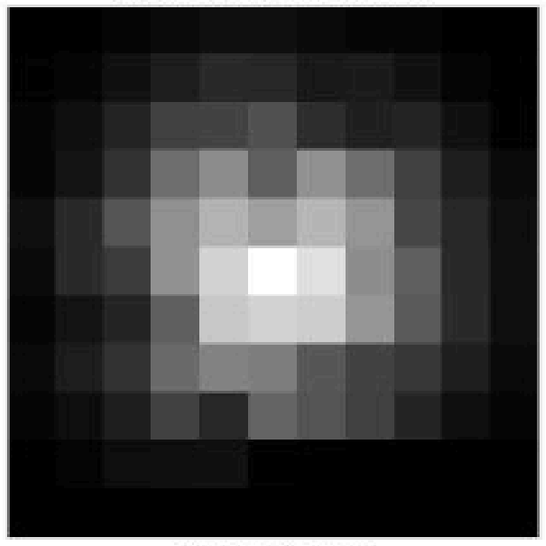
Demo: **Click any pixel and the panel draws the weight map the bilateral…**

click anywhere to move the probe

the weights, at the probe

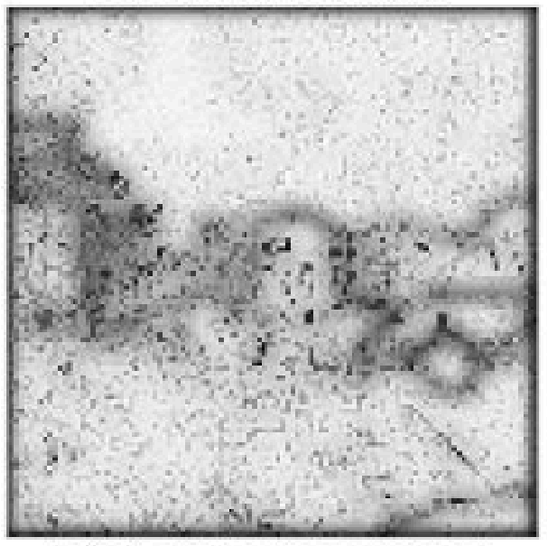


that kernel, magnified



11×11 window

effective N, every pixel



bright = more samples averaged

effective N at the probe: 51.4 pixels

$(\Sigma w)^2 / \Sigma w^2$ · the full window holds 121 · fixed-weight $\sqrt{N}$ = 7.17×

fixed-weight formula says σ 0.060 → 0.0084; measured over 48 draws: 0.0169

adaptive weights and clipping invalidate this fixed-weight, iid-noise prediction

An edge can reduce effective N. Clamped copies of the same pixel count as one sample.

Recall bilateral denoising



Noisy input



After bilateral filter

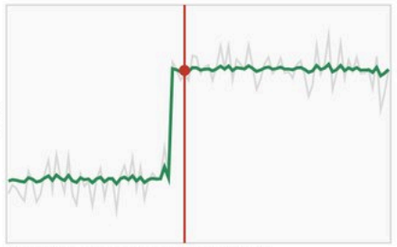
Demo: One filter, three views, and the equation it comes from

$$I_{out}(p) = 1 / k(p) \cdot \sum_q f(p - q) \cdot g(I(p) - I(q)) \times I(q)$$

$I_{out}(p)$ the output · $f(p - q) \cdot g(I(p) - I(q))$ the weights at p · $I(q)$ the neighbours

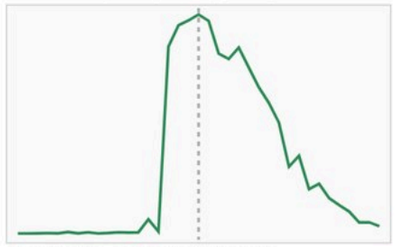
p = (44, 34) · k(p) = 195.55 · click a panel, or drag a height field to turn it

1-D



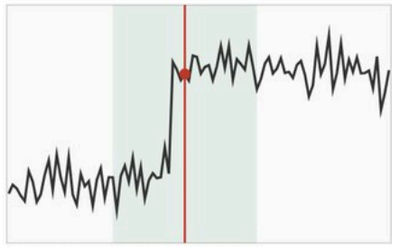
the filtered scanline (input faint)

$= 1 / k(p) \cdot \sum_q$



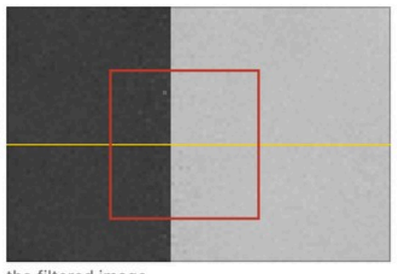
the weights, scaled to their peak

×



the scanline of the image below

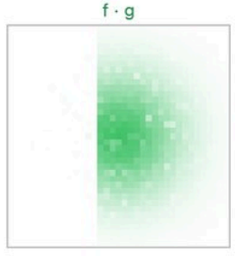
2-D



the filtered image

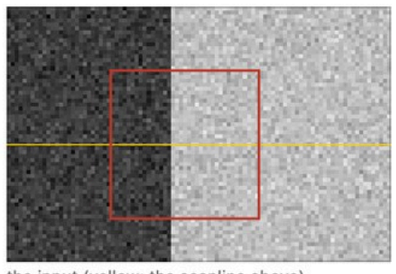
$= 1 / k(p) \cdot \sum_q$

f · g



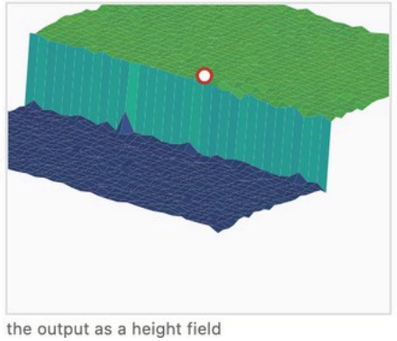
the window around p, 37 × 37

×



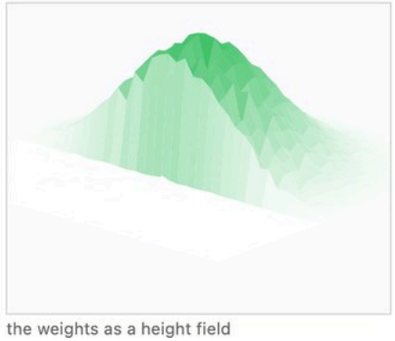
the input (yellow: the scanline above)

2.5-D



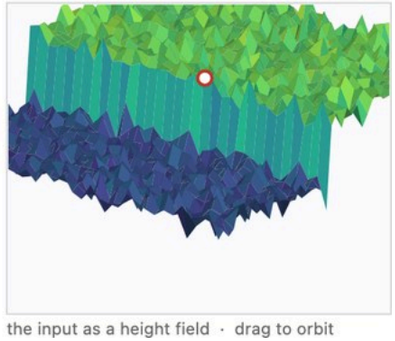
the output as a height field

$= 1 / k(p) \cdot \sum_q$



the weights as a height field

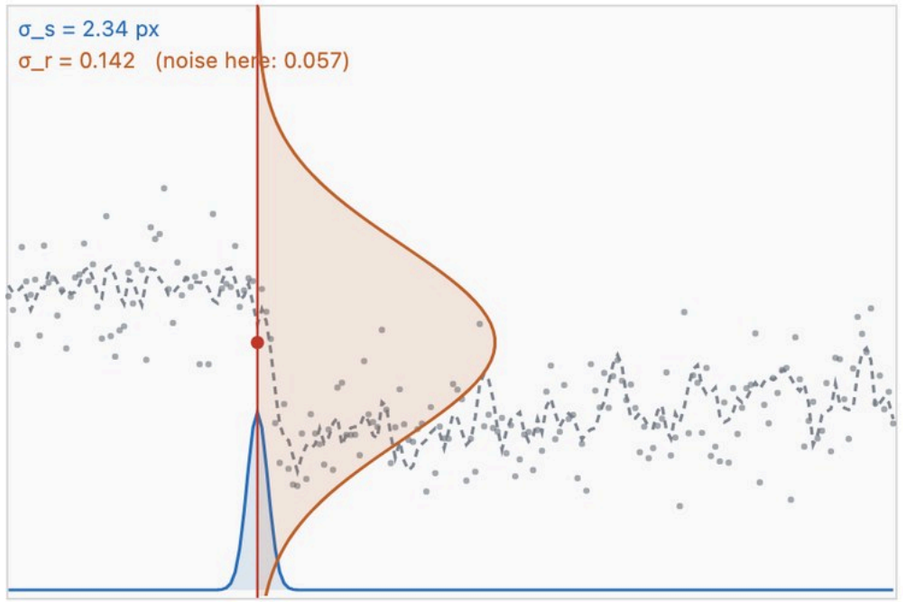
×



the input as a height field · drag to orbit

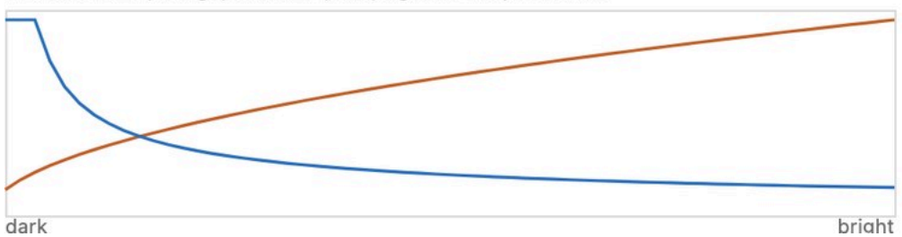
Demo: The other way: leave the values alone and move the widths…

the input: the signal, and one measurement of it

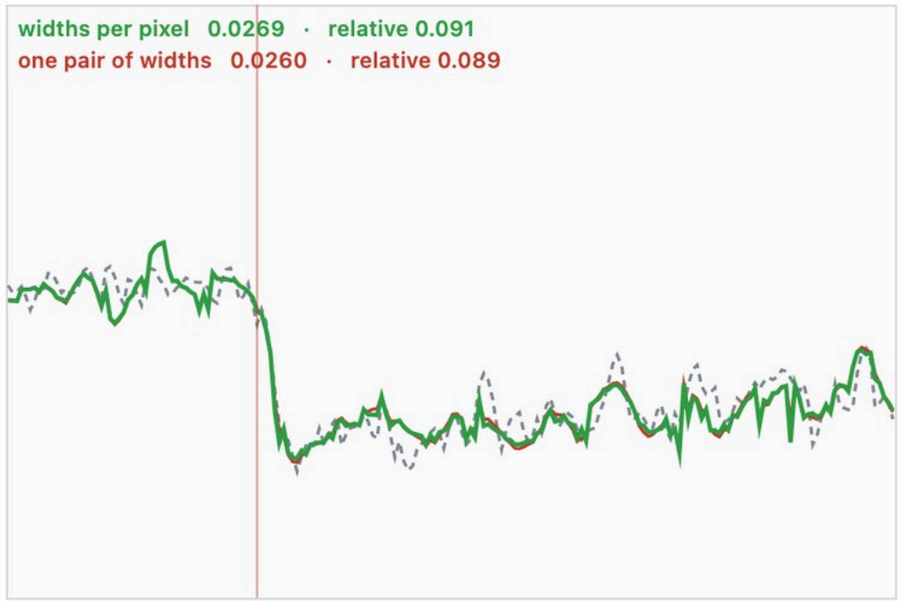


click anywhere to move the probe; the widths follow the value under it

the rules: σ_r (orange) and σ_s (blue) against the pixel value



the output, with the widths set pixel by pixel



absolute RMS error, then the error against the level each pixel sits on.
The measurement is 0.0523. A wider window in the shadows buys the second, not the first.

Contrast reduction



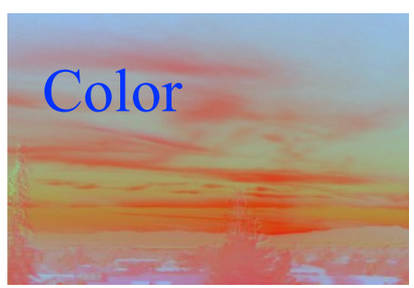
Contrast too high!

Contrast reduction





intensity
= (20R+40G+B)/61



R'=R/intensity
G'=G/intensity
B'=B/intensity

important to use ratios (makes it luminance invariant)

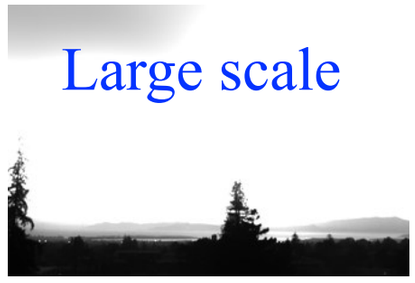
Contrast reduction

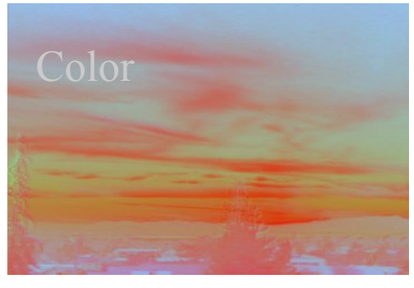




Bilateral Filter

in log





Spatial sigma: 2 to 5% image size
Range sigma: 0.4 (in log 10)

What is large scale

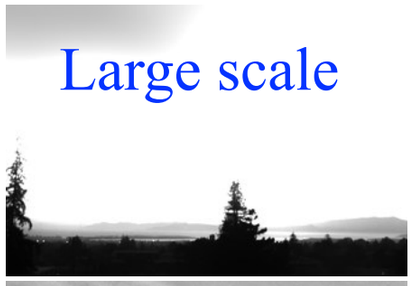
- **Some notion of average luminancxe around a pixel**
- **previously: Gaussian-blurred log luminance**
- **Now bilaterally-filtered log luminance**

Contrast reduction

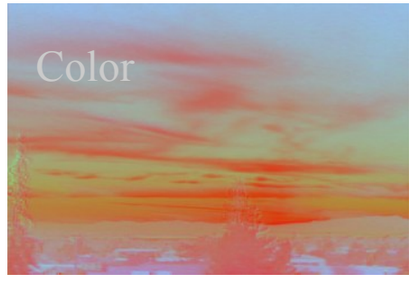




Bilateral Filter in log







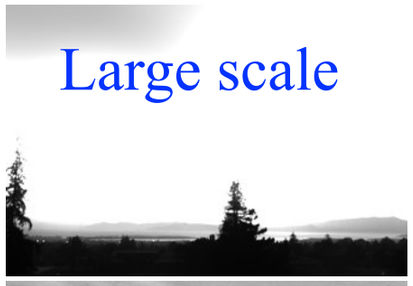
Detail = log intensity - large scale (residual)

Contrast reduction





Bilateral Filter

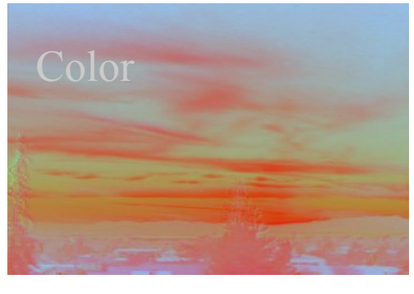


Detail in log



Reduce contrast





Contrast reduction





Fast Bilateral Filter



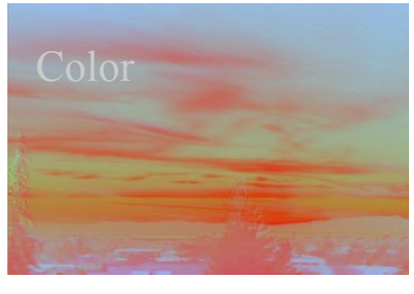


Reduce contrast

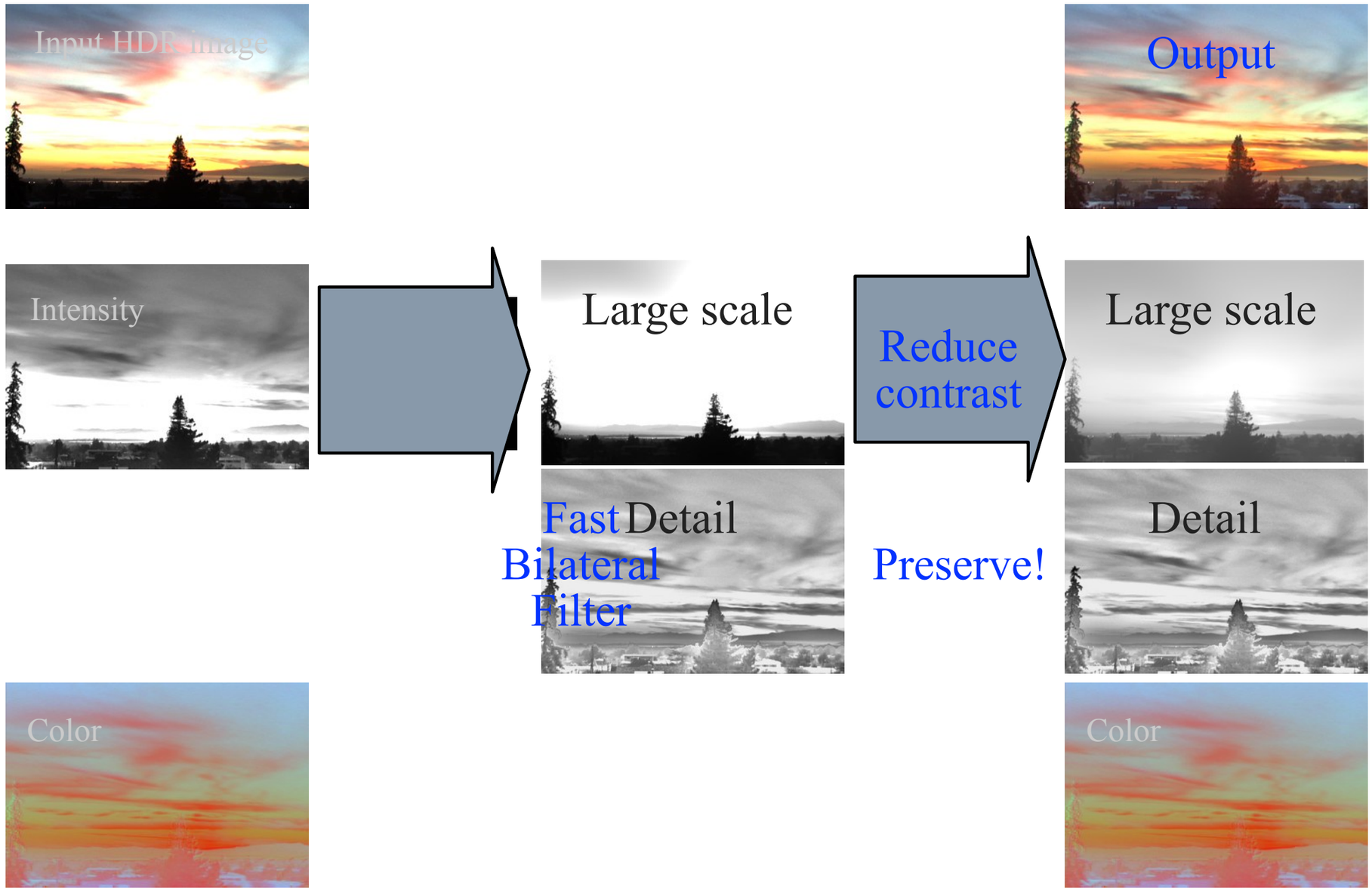
Preserve!







Contrast reduction



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**

Contrast reduction

- **Could target full image contrast/brightnes**
- **In practice easier to focus on large scale**

Contrast reduction in log domain

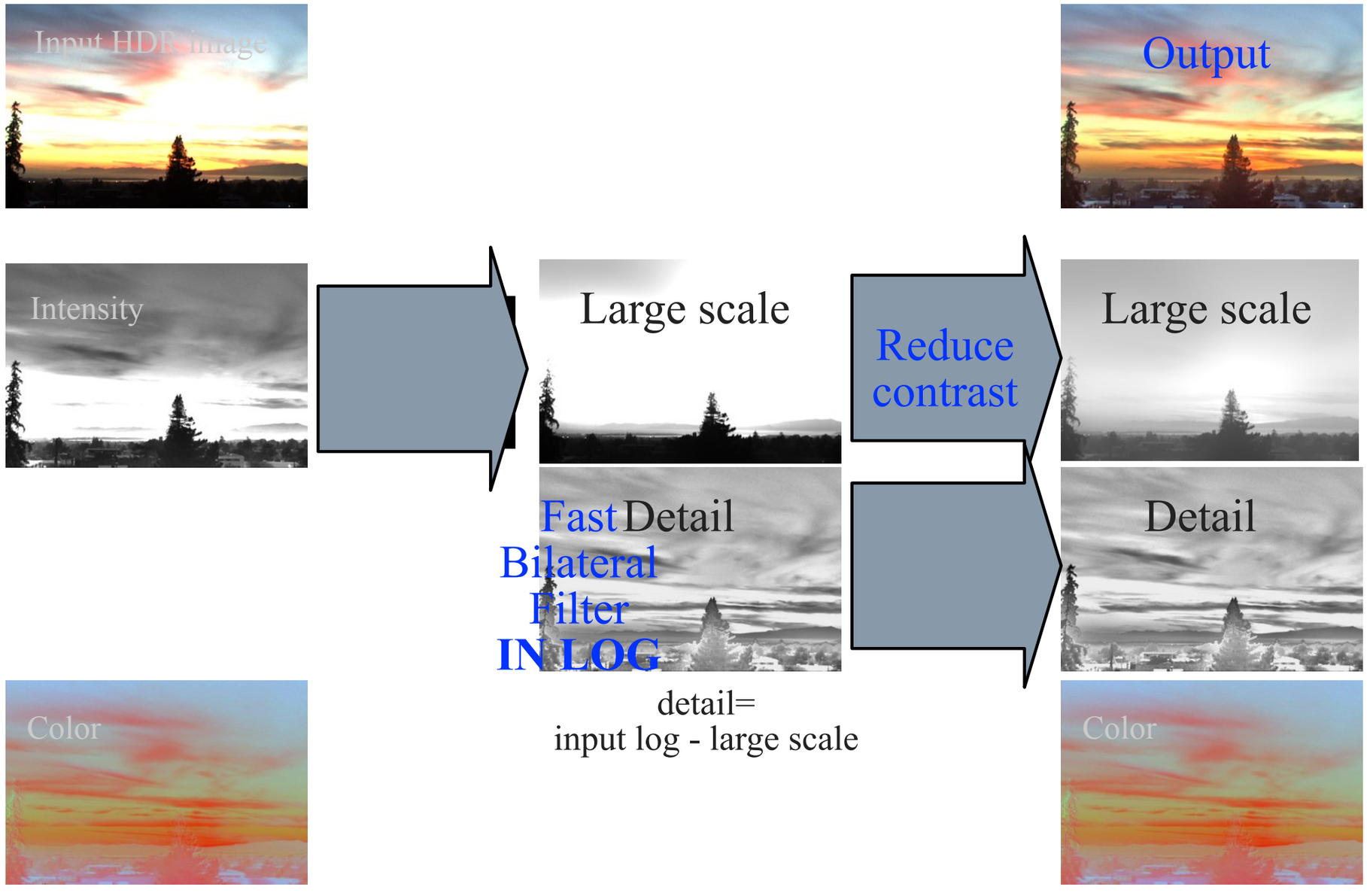
- **Set target large-scale contrast (e.g. $\log_{10}$ 100)**
  - i.e. in **linear** output,
    we want 1:100 contrast for large scale
- **Compute range of input large scale layer:**
  - largeRange = max(inLogLarge) – min (inLogLarge)
- **Scale factor *k*= $\log_{10}$ (100) / largeRange**
- **Normalize so that the biggest value is 0 in log**
- **Optional: amplify detail by detailAmp**

outLog= detailAmp*inLogDetail +
k(inLogLarge – max(inLogLarge))

Final output

- From last slide:
  outLog= detailAmp*inLogDetail +
  k(inLogLarge – max(inLogLarge))
- outIntensity=10^{outLog}
- Recall that R’,G’,B’ is the intensity-normalized RGB color
- outR=outIntensity * R’
- outG=outIntensity * G’
- outB=outIntensity * B’

Recap


Input HDR image
Intensity
Large scale
Fast Bilateral Filter IN LOG
Detail
Reduce contrast
Large scale
Detail
Output
Color
Color
detail=
input log - large scale

Demo: From a global curve to two-scale tone mapping: input: the radiance map

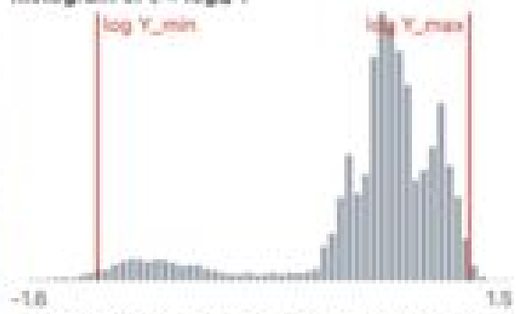


probe pixel, R/B

scene

R/B = 8.08

histogram of $t = \log_{10} Y$



1st to 99th percentile: 2.29 $\log_{10}$, 7.6 stops, 1 : 196

at display white: Y_max × 2^0; 1.0% clipped to white, 0% below the first code value

radiance map: Paul Debevec

1 input: the radiance map | 2 global curve: per channel | 3 global curve: luminance + ratios | 4 two-scale: Gaussian base | 5 two-scale: bilateral base

radiance map: vine sunset

exposure: 0 stops from Y_max

beauty curve (the book's display tone curve) reset

Demo courtesy of Frédo Durand, MIT.

Demo: From a global curve to two-scale tone mapping: global curve: per channel



probe pixel, R/B

scene
R/B = 8.08

per channel
R/B = 3.27 (8.08^0.57 = 3.27)

f spans 2.29 $\log_{10}$ (1st to 99th percentile): c = $\log_{10}$ DR / 2.29 = 0.567

radiance map: Paul Debevec

1 input: the radiance map | 2 global curve: per channel | 3 global curve: luminance + ratios | 4 two-scale: Gaussian base | 5 two-scale: bilateral base

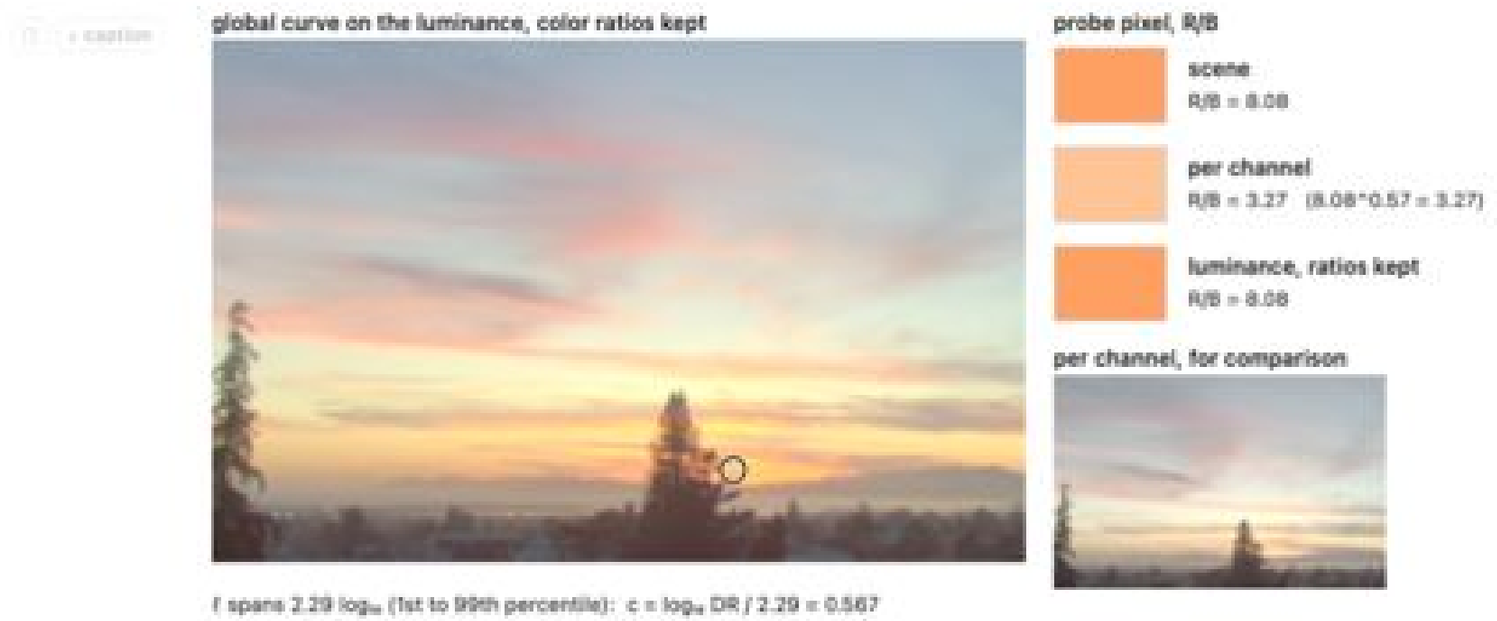
radiance map: vine sunset

target contrast: 1 : 20

beauty curve (the book's display tone curve) reset

Demo courtesy of Frédo Durand, MIT.

Demo: From a global curve to two-scale tone mapping: global curve: luminance + ratios

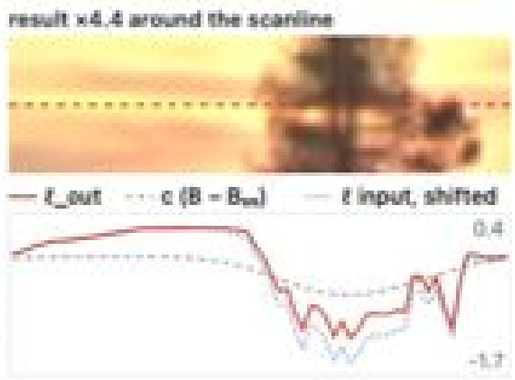


Demo courtesy of Frédo Durand, MIT.

Demo: From a global curve to two-scale tone mapping: two-scale: Gaussian base







B spans ΔB = 2.14 log₁₀ (1st to 99th percentile): c = log₁₀ DR / ΔB = 0.608

probe R/B: scene 8.08, per channel 3.27, here 8.08

radiance map: Paul Debevec

◀ 1 input: the radiance map | 2 global curve: per channel | 3 global curve: luminance + ratios | 4 two-scale: Gaussian base | 5 two-scale: bilateral base ▶

radiance map: vine sunset

target contrast: 1 : 20

spatial scale σs: 2.0% · 6.4 px

☐ profile in linear

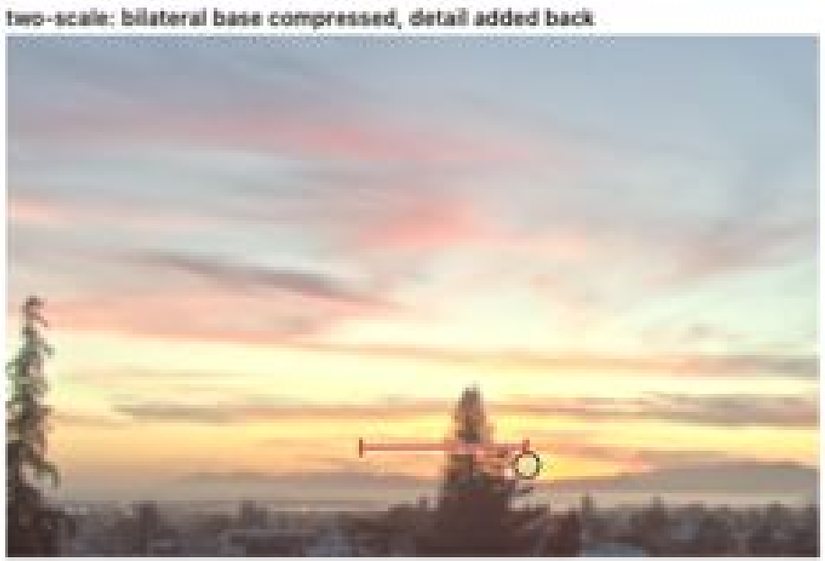
☐ beauty curve (the book's display tone curve)

detail amplification a: 1.00

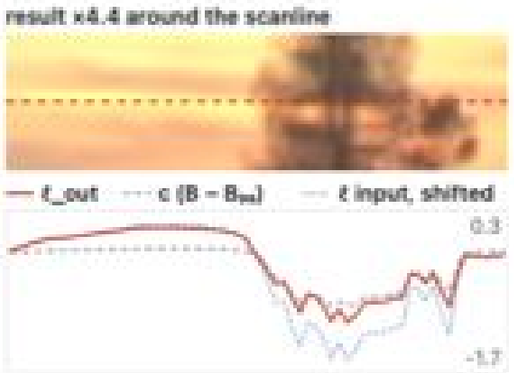
reset

Demo courtesy of Frédo Durand, MIT.

Demo: From a global curve to two-scale tone mapping: two-scale: bilateral base







B spans ΔB = 2.19 $\log_{10}$ (1st to 99th percentile): c = $\log_{10}$ DR / ΔB = 0.595

probe R/B: scene 8.08, per channel 3.27, here 8.08

radiance map: Paul Debevec

1 input: the radiance map | 2 global curve: per channel | 3 global curve: luminance + ratios | 4 two-scale: Gaussian base | 5 two-scale: bilateral base

radiance map: vine sunset

target contrast 1 : 20

spatial scale σs 2.0% · 6.4 px

range scale σr ($\log_{10}$) 0.40

profile in linear

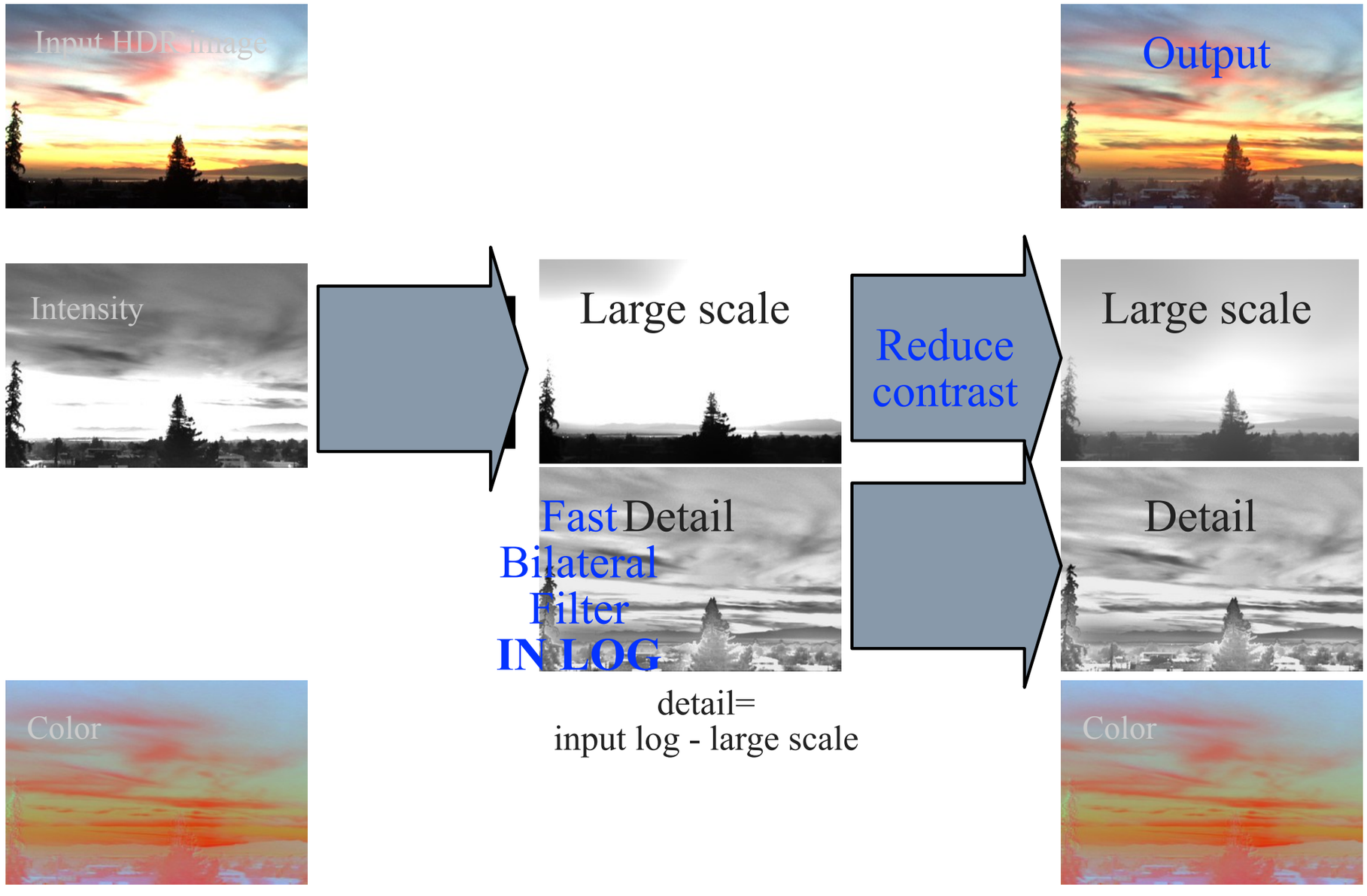
beauty curve (the book's display tone curve)

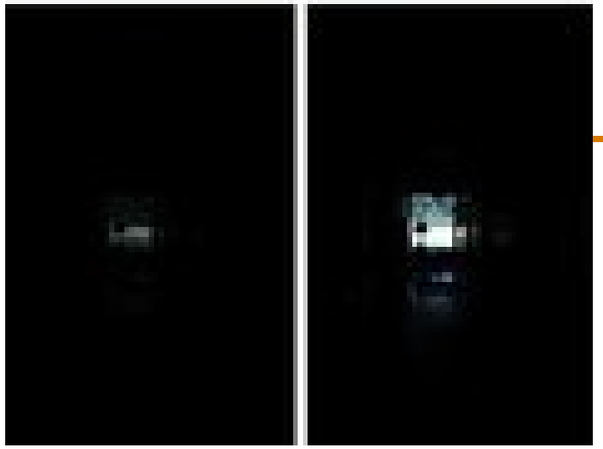
detail amplification a 1.00

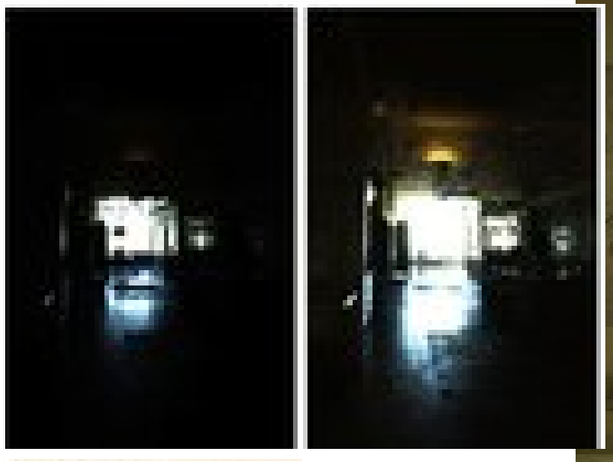
reset

Demo courtesy of Frédo Durand, MIT.

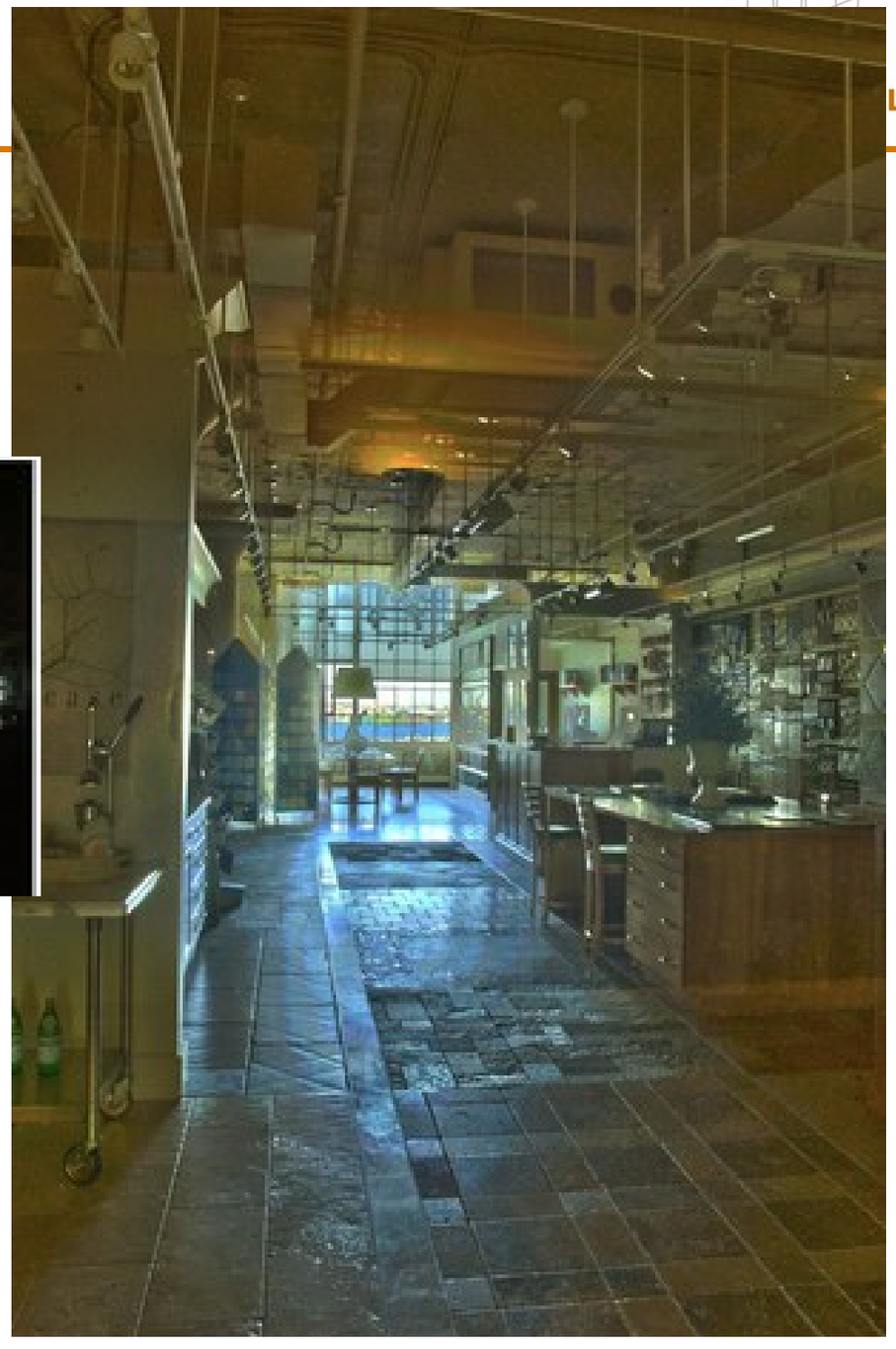
Bells and whistles: increase detail









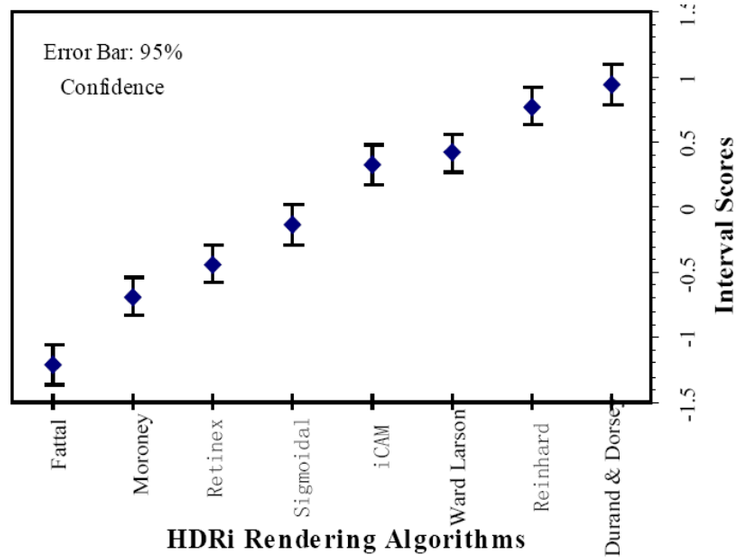


What matters

- **Spatial sigma: not very important**
- **Range sigma: quite important**
- **Use of the log domain for range: critical**
  - Because HDR and because perception sensitive to multiplicative contrast
  - CIELab might be better for other applications
- **Luminance computation**
  - Not critical, but has influence
  - see our Flash/no-flash paper [Eisemann 2004] for smarter function

Tone mapping evaluation

- **User experiments to evaluate competing tone mapping**
  - Ledda et al. 2005
    http://www.cs.bris.ac.uk/Publications/Papers/2000255.pdf
  - Kuang et al. 2004
    http://www.cis.rit.edu/fairchild/PDFs/PRO22.pdf
- **Interestingly, the former concludes bilateral is the worst, the latter that it is the best!**
  - They choose to test a different criterion: fidelity vs. preference
- **More importantly, they focus on algorithm and ignore parameters**



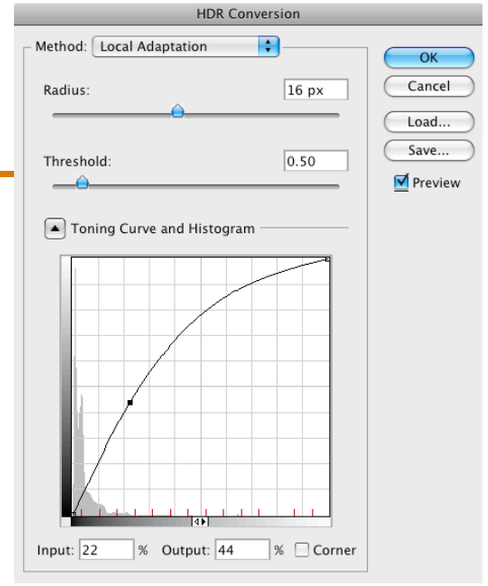
From Kuang et al.

	1st	2nd	3rd	4th	5th	6th
Scene 1	*P*	*B*	*A*	*H*	*I*	*L*
Scene 2	*I*	*P*	*H*	*A*	*B*	*L*
Scene 3	*P*	*I*	*A*	*H*	*L*	*B*
Scene 4	*P*	*L*	*I*	*A*	*H*	*B*
Scene 5	*I*	*H*	*A*	*P*	*L*	*B*
Scene 6	*I*	*H*	*A*	*P*	*L*	*B*
Scene 7	*I*	*A*	*P*	*H*	*B*	*L*
Scene 8	*I*	*P*	*A*	*H*	*L*	*B*
Scene 9	*P*	*A*	*L*	*H*	*B*	*I*

Adapted from Ledda et al.

Related tools

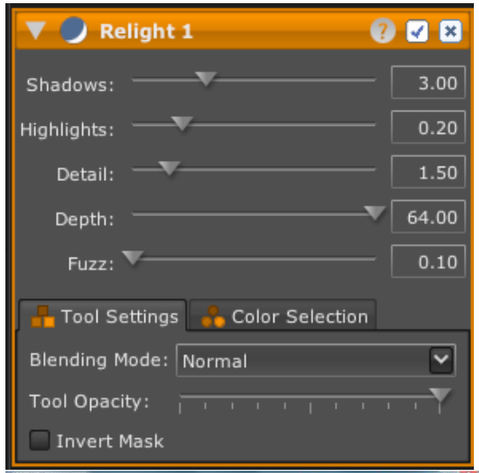
- **Photoshop “Local adaptation”**



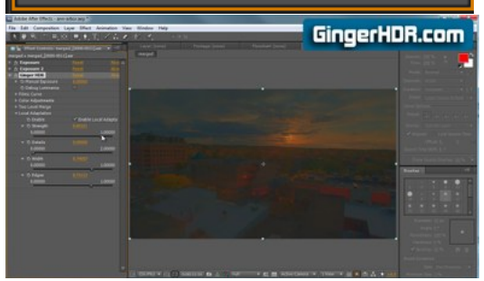
- **Lightroom Fill Light**
  - or shadow



- **Lightzone Relight**
  - RIP



- **Ginger HDR**
  - **http://vimeo.com/32901501**



NB: useful for single image

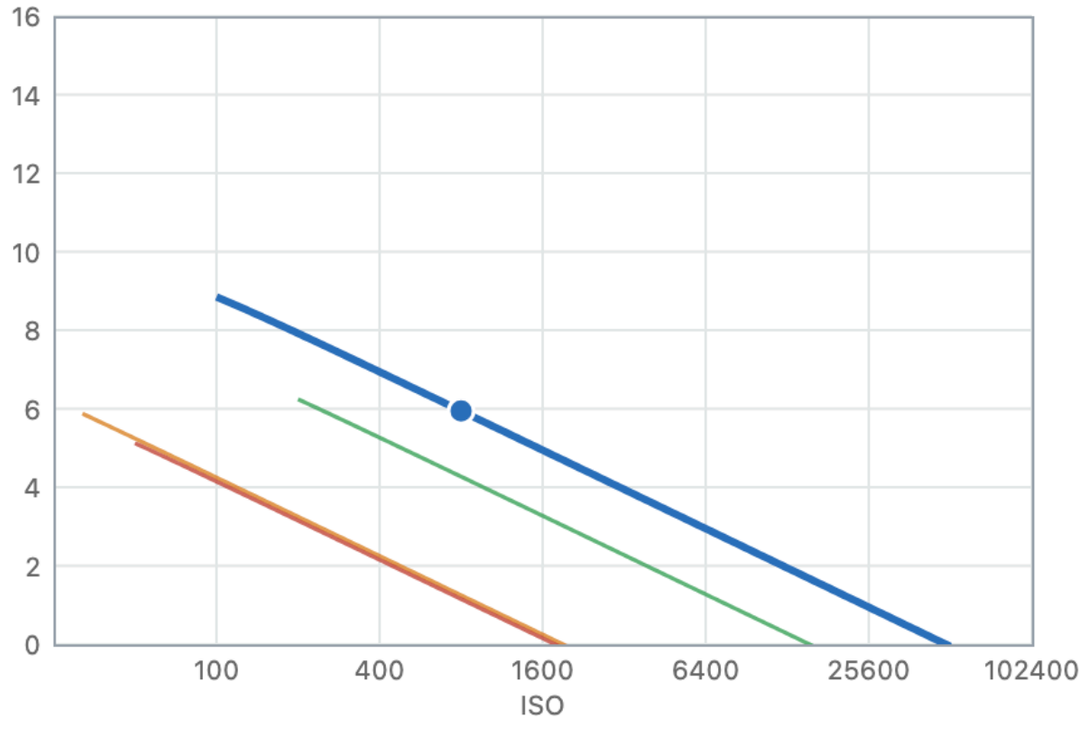
- **Modern SLRs give you 13 stops of dynamic range**
- **Can benefit from tone mapping**



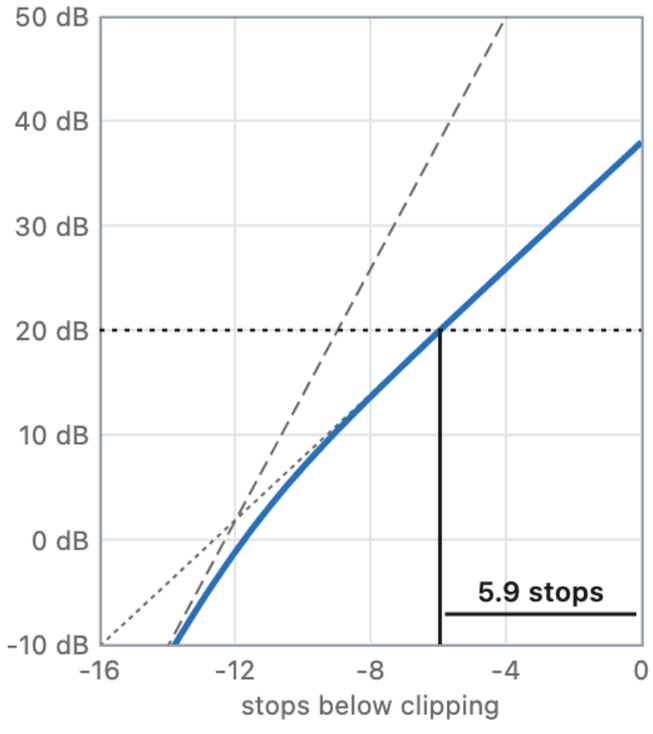


Demo: Dynamic range against ISO from the noise model, for four typical…

dynamic range (stops) against ISO, floor at 20 dB

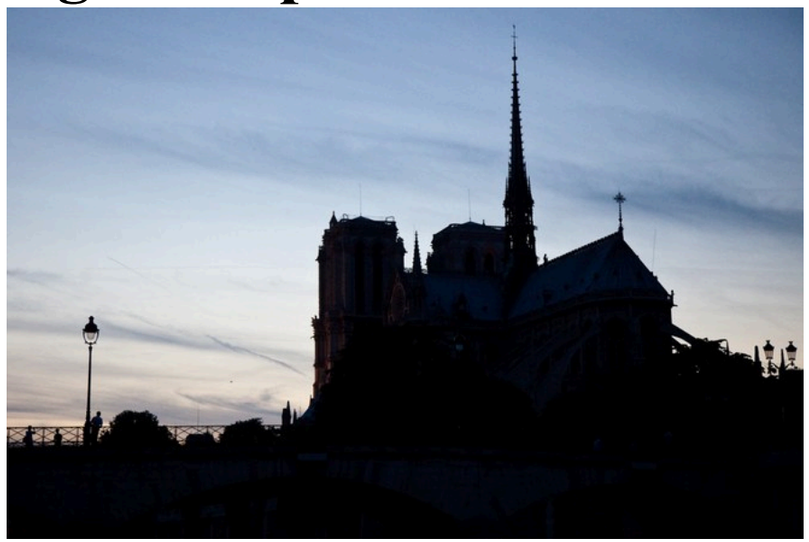


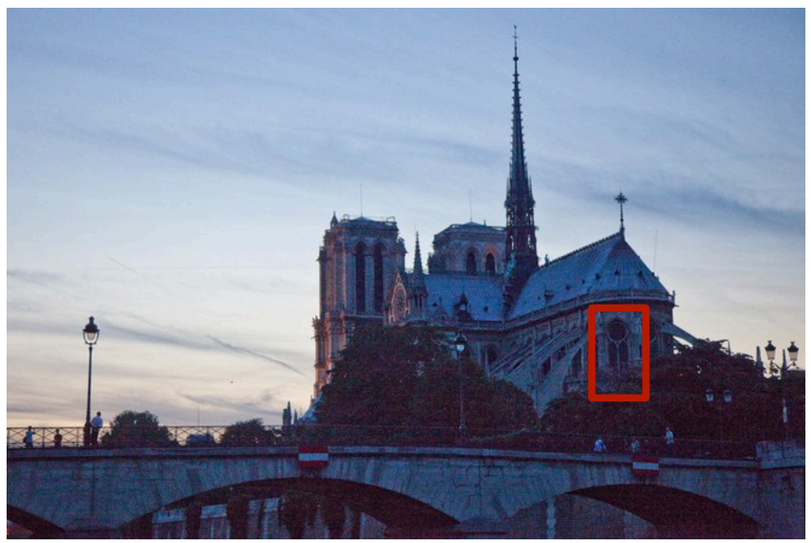
SNR, modern full frame at ISO 800



Up to a point

- **Noise gets amplified in the shadows**



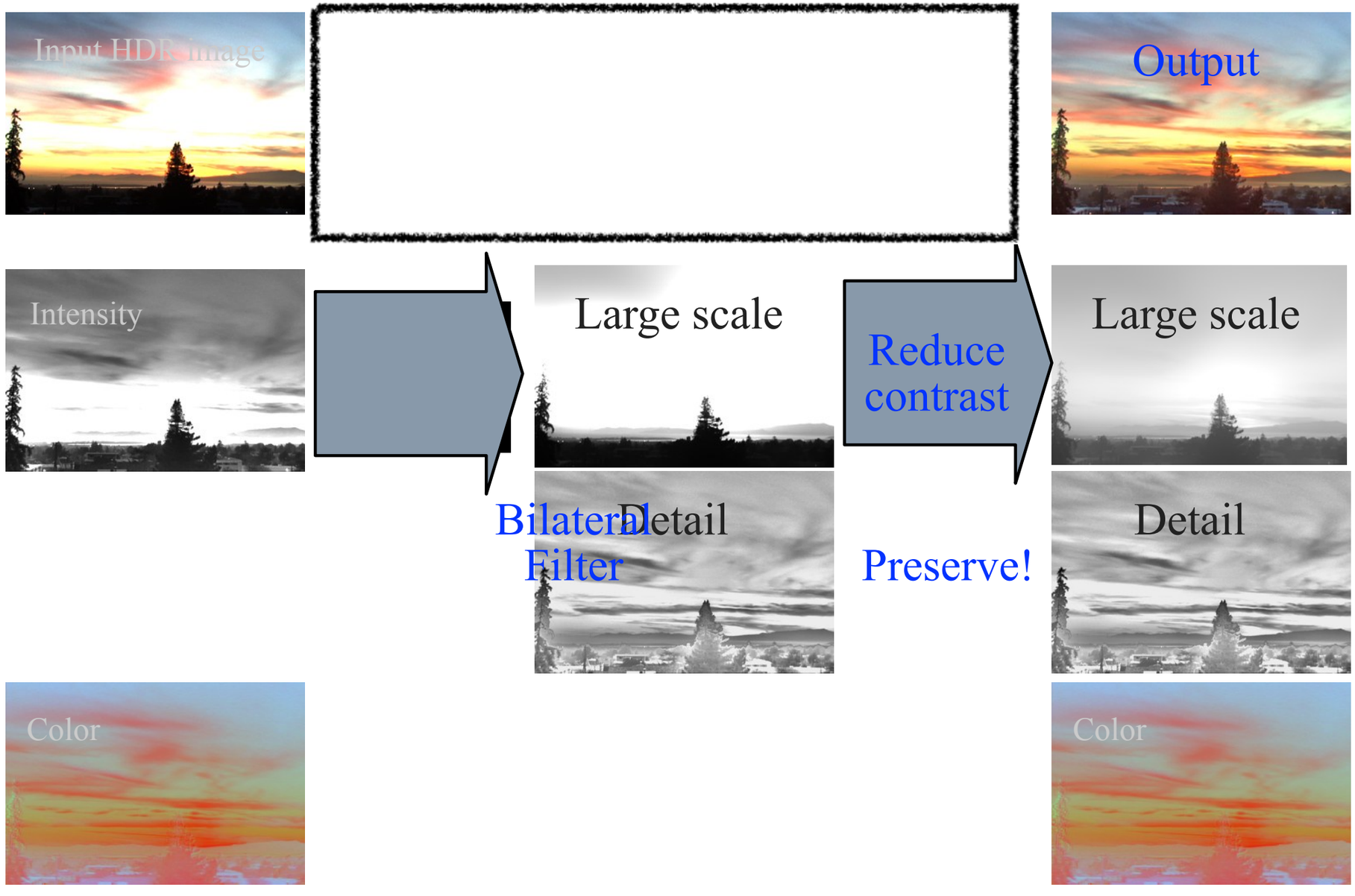




Questions?

Bilateral Grid

Back to bilateral tone mapping



Speed

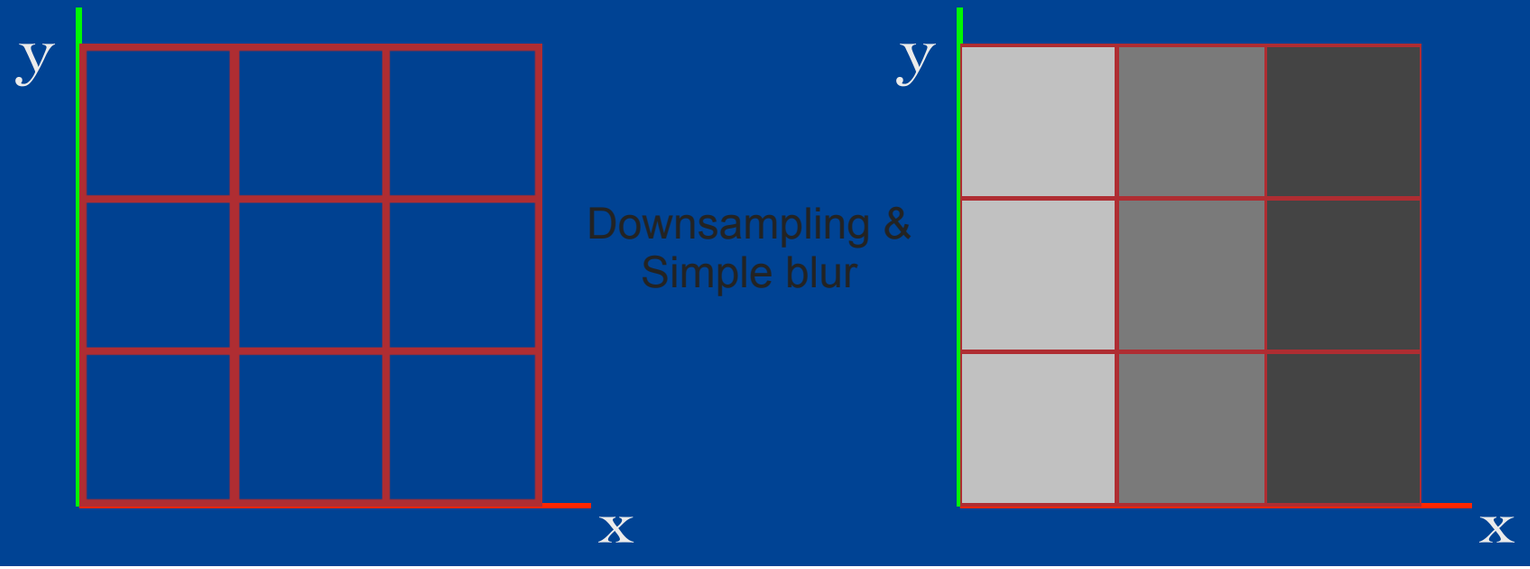
- **Direct bilateral filtering is slow (minutes)**

- **Fast algorithm: bilateral grid (in a few slides**
  - **http://groups.csail.mit.edu/graphics/bilagrid/**
  - **http://people.csail.mit.edu/sparis/publi/2009/ijcv/Paris_09_Fast_Approximation.pdf**
  - **http://graphics.stanford.edu/papers/gkdtrees/**

Bilateral Grid: basic motivation

[Paris and Durand 06, Chen et al. 07]

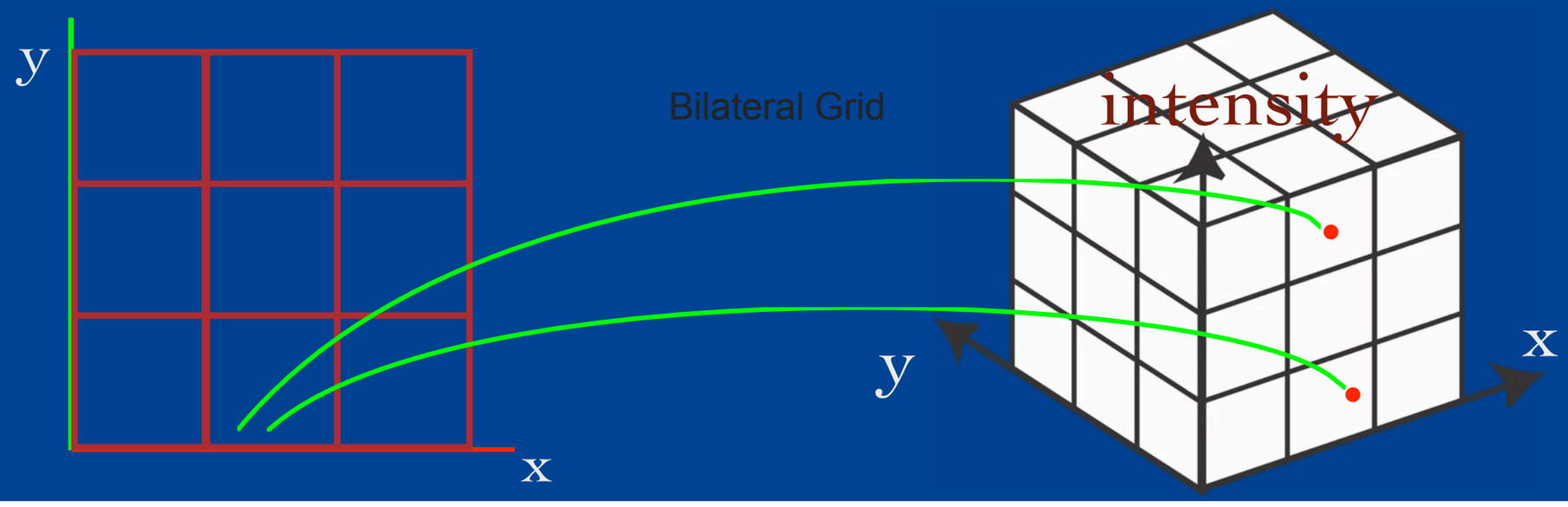
- When we smooth, we reduce complexity of image => we should be able to do it at a lower resolution



Bilateral Grid: basic motivation

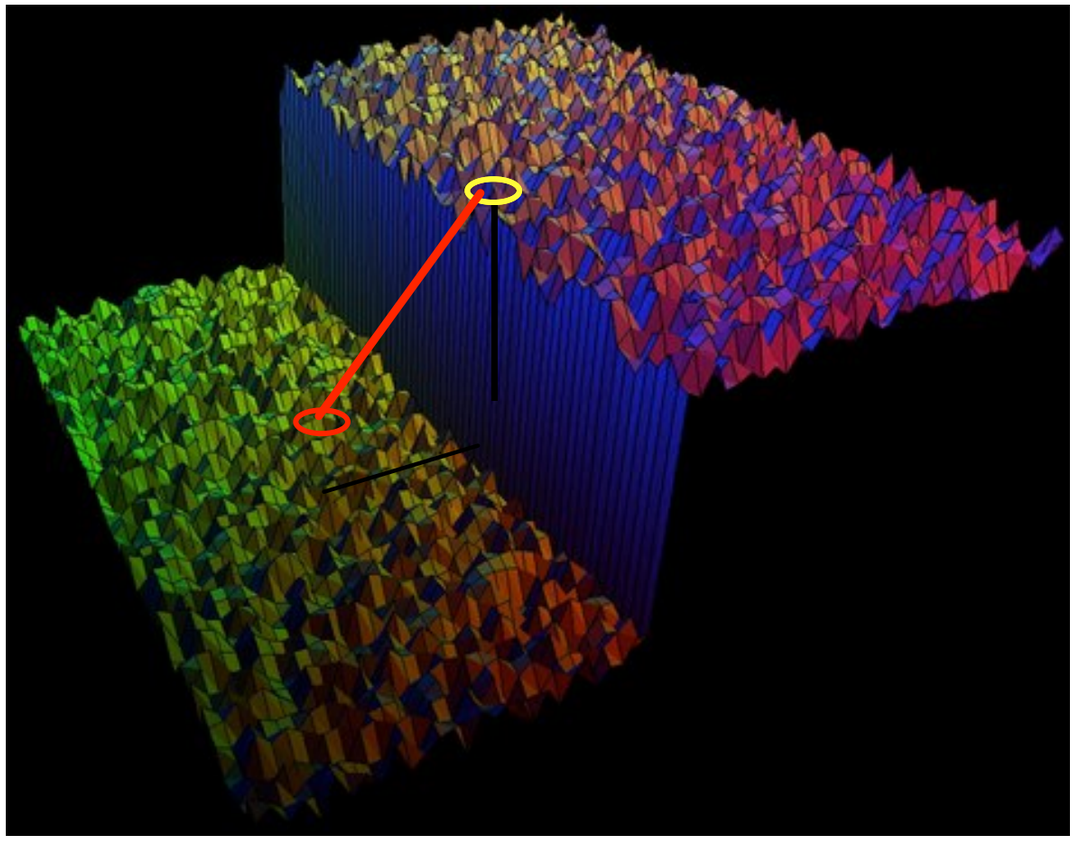
[Paris and Durand 06, Chen et al. 07]

- When we smooth, we reduce complexity of image => we should be able to do it at a lower resolution
- However, the bilateral filter preserves sharp edges and a low resolution image does not
- Idea: add a 3rd dimension to the image so that intensity difference are handled well



3D gaussian

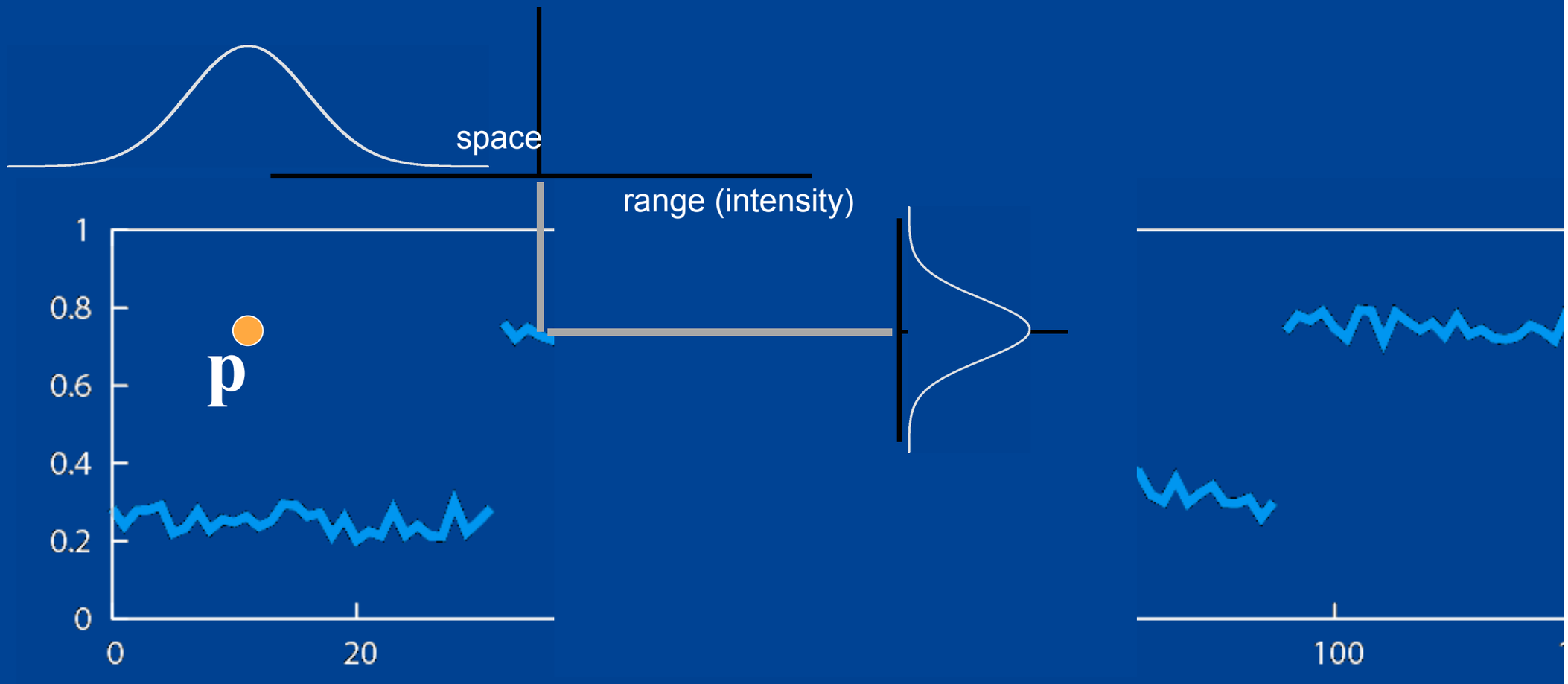
- **The bilateral filter uses the 3D distance**
- **With the bilateral grid, this becomes a 3D blur**



Bilateral Grid

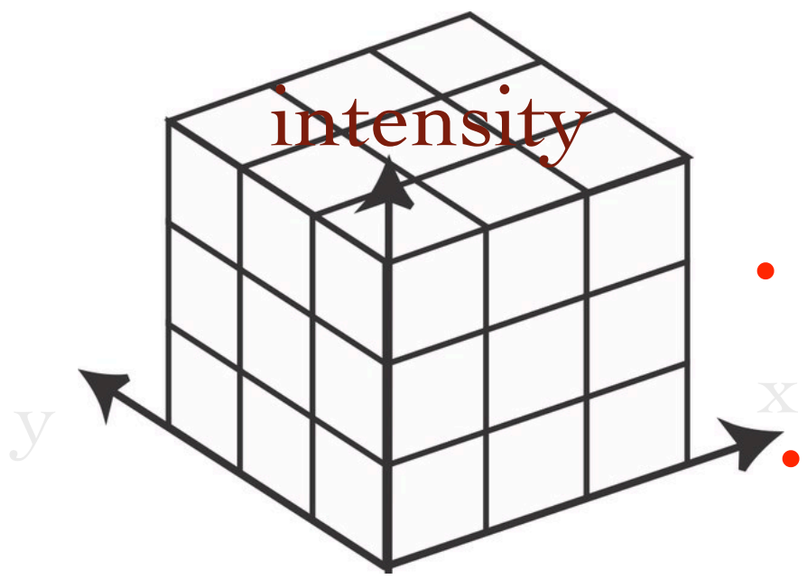
[Paris and Durand 06, Chen et al. 07]

- Idea 2.0: The product of spatial and intensity Gaussian defines a 3D Gaussian in x, y, I

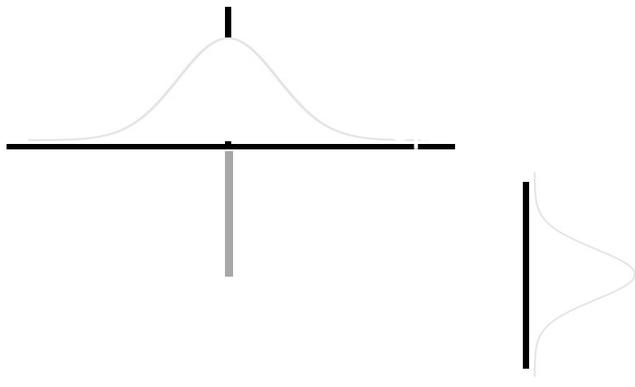


Fast bilateral filter idea

- **Represent image in low-resolution 3D grid**



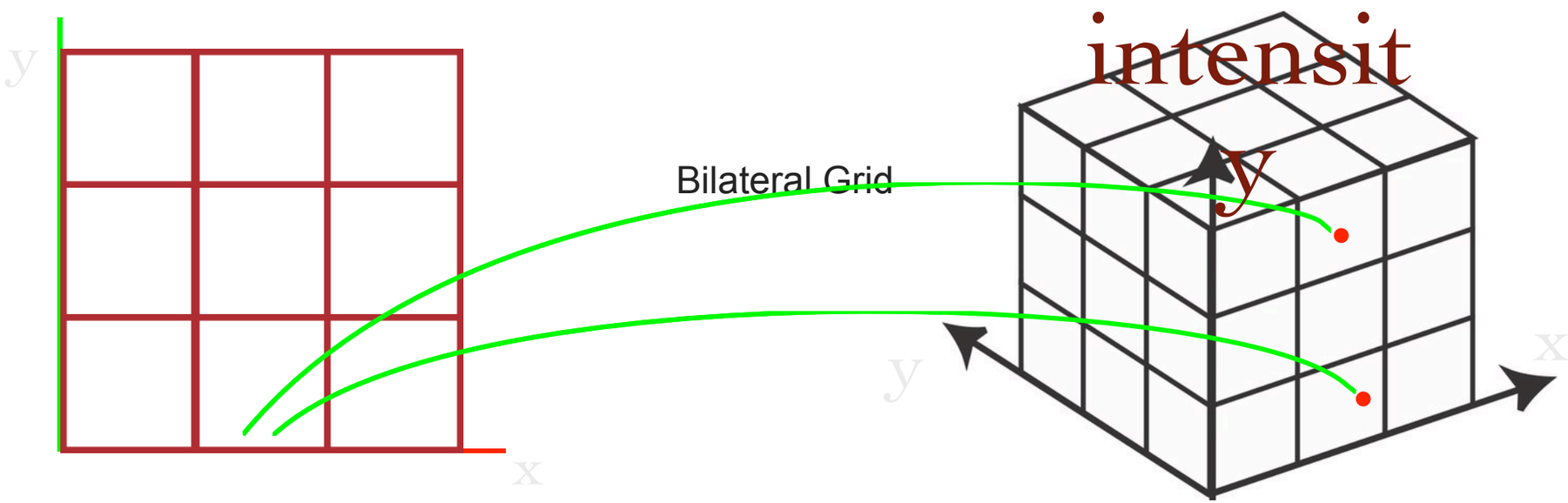
- **3D blur combines space and intensity terms**



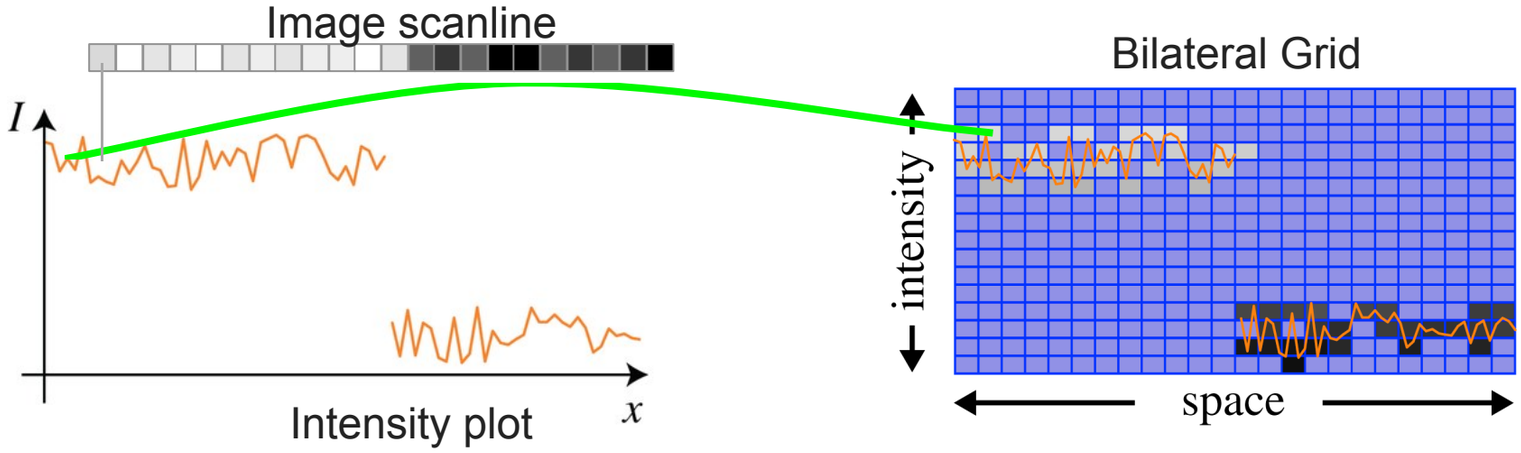
Questions?

Overview

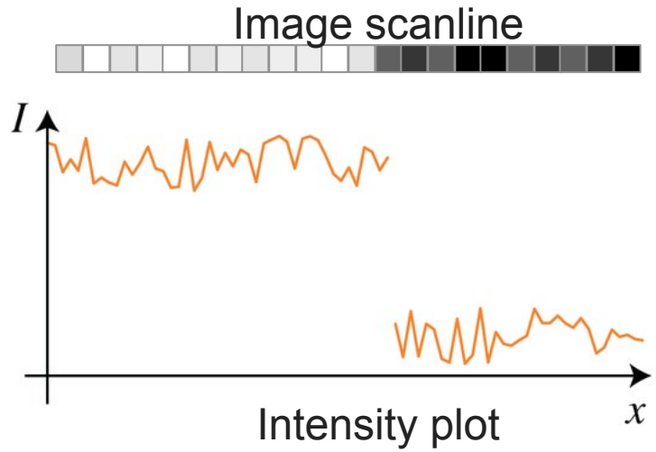
- **Convert image to bilateral grid**
  - (x, y) pixel goes to x, y, I(x,y)
- **Blur the grid**
  - 3D Gaussian combines 2D x,y term f and I term g
- **Convert back to 2D image space**

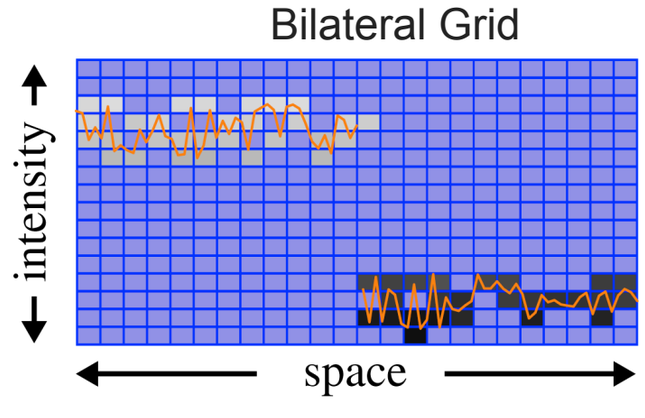


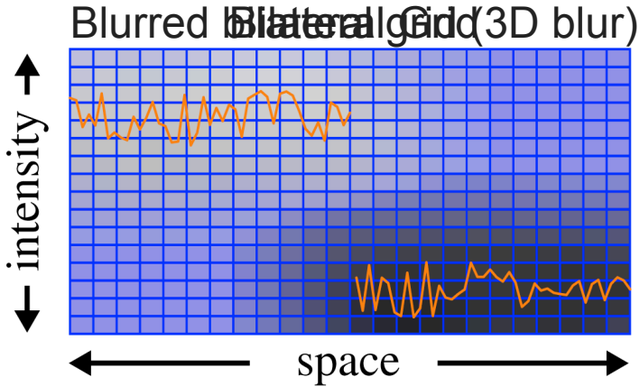
Bilateral Filter on the Bilateral Grid

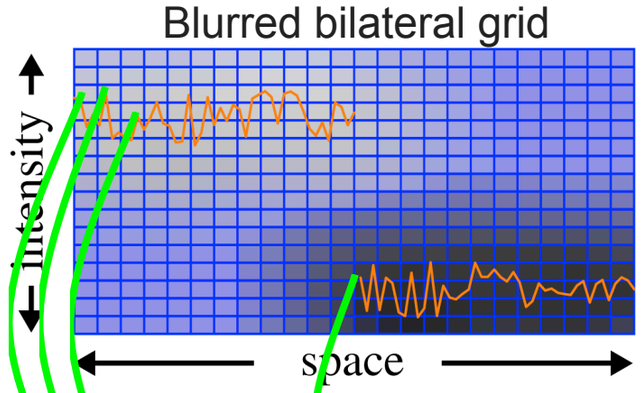


Bilateral Filter on the Bilateral Grid









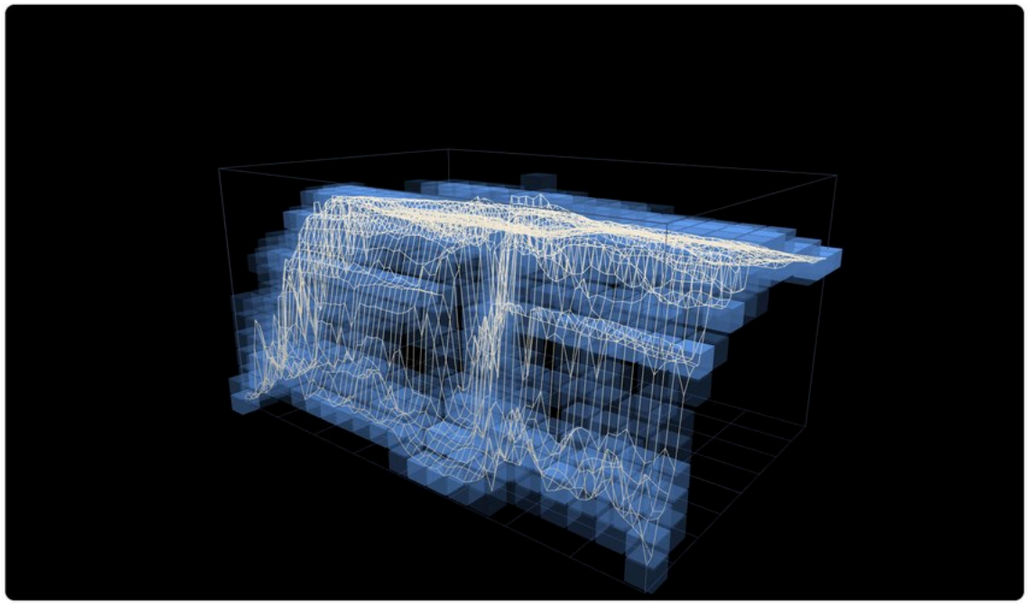
Query grid with input image

Filtered scanline

Demo: Bilateral grid (3-D)

The bilateral grid, in 3-D

A greyscale image is **splatted** into a 3-D grid whose axes are image *x*, image *y*, and **intensity** (height). Each cell's opacity shows the **sum of weights** that landed in it. Blurring the grid with a plain 3-D Gaussian and slicing it back out is exactly the bilateral filter — smoothing that refuses to cross intensity edges. The **real HDR scenes** (vine sunset, Antelope Canyon, design center) use the height axis the way HDR tone mapping does — **log luminance** — so the grid layers are spaced in stops. Drag to orbit; scroll to zoom.



input (log-luminance)



hover the image → the matching 3-D grid point lights up

Input | Output

image

vine sunset (HDR)

upload

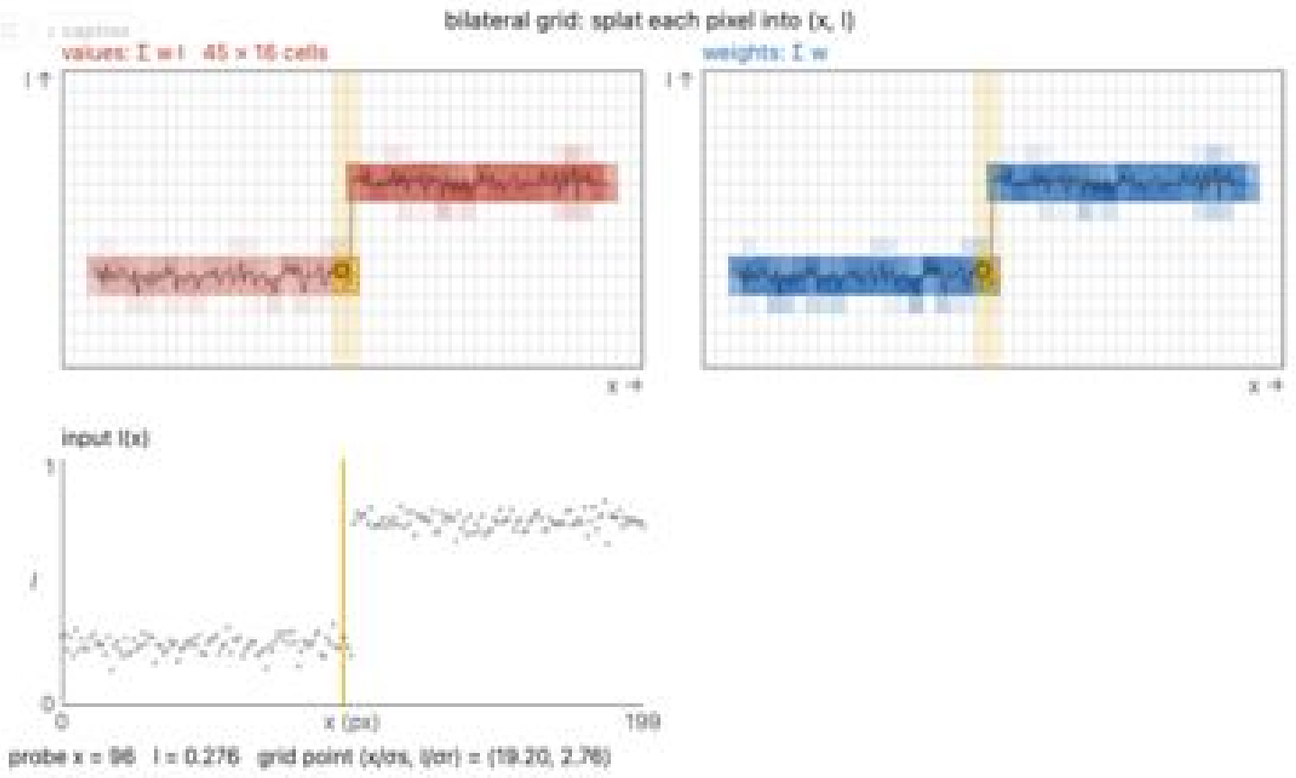
grid | Original (splat) | 3-D Gaussian blurred | overlay input as height field

spatial cells σ_s 22×15 | intensity bins σ_r 14 | reset view

bilateral grid — opacity ∝ Σ weights in the cell | input as a height-field wireframe (height = intensity) | hovered point

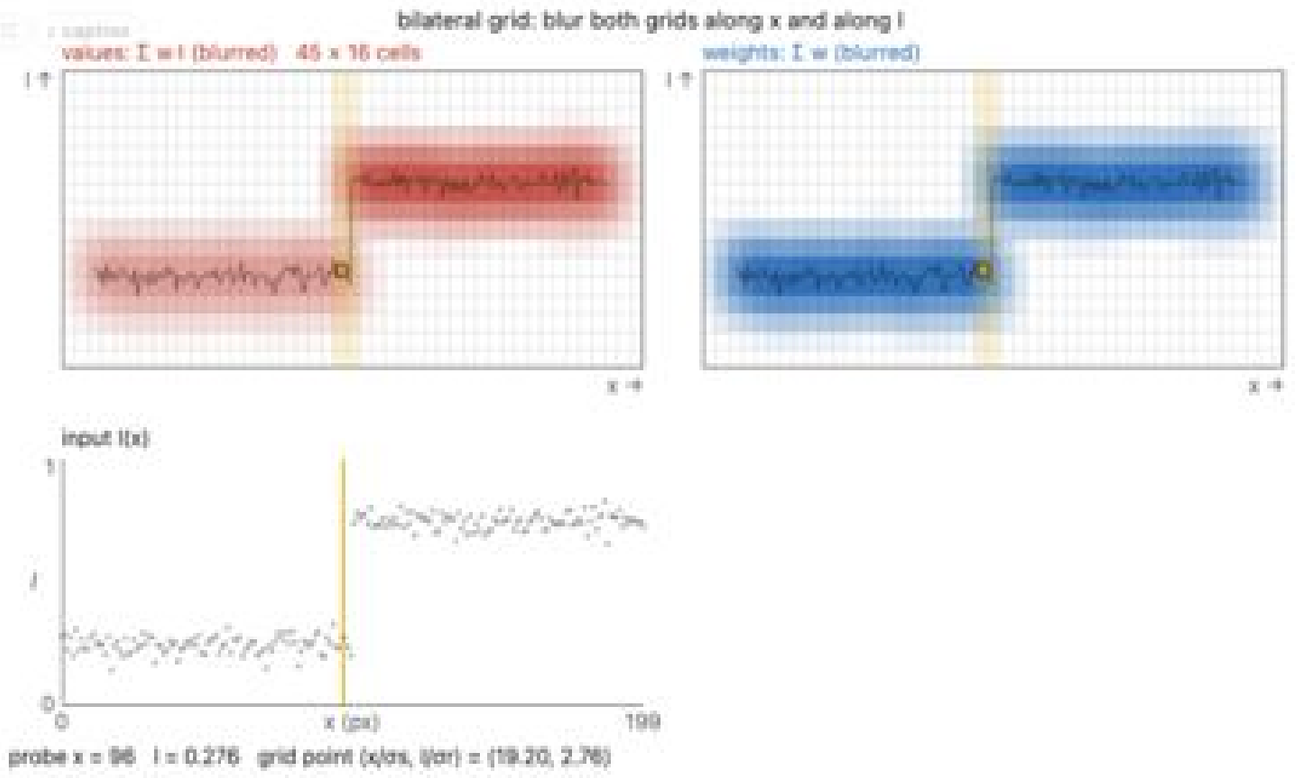
What you are seeing. Every pixel drops a vote into the grid cell at its (x, y, intensity); a flat region piles many votes into one column of cells while an edge splits its votes between a low and a high intensity layer, leaving the cells *between* them empty. That gap is why a 3-D Gaussian blur — which freely mixes neighbours *within* a layer but barely reaches across the empty gap — smooths textures yet preserves edges. Switch to the **blurred** grid to watch the cells spread and soften; the empty edge-gap survives. The transparent cubes are drawn strictly **back-to-front** so the layering reads correctly from any angle; the input is shown as a light **wireframe terrain** (height = intensity) you can see the grid through.

Demo: The bilateral grid in 1-D: splat



Demo courtesy of Frédo Durand, MIT.

Demo: The bilateral grid in 1-D: blur



1 bilateral grid: splat | 2 bilateral grid: blur | 3 bilateral grid: slice | 4 bilateral grid: divide

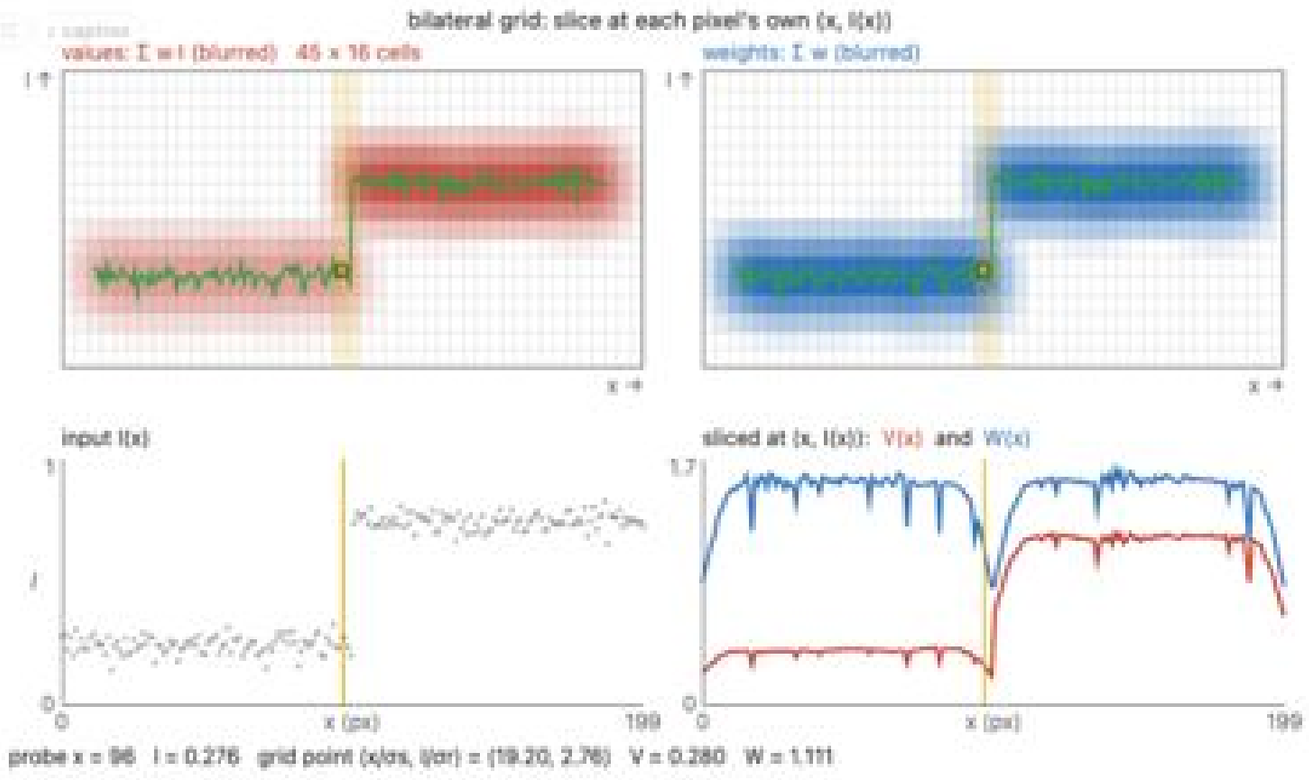
scanline: noisy step

σ_s = one cell in x 5.0 px

σ_r = one cell in I 0.10 reset

Demo courtesy of Frédo Durand, MIT.

Demo: The bilateral grid in 1-D: slice



1 bilateral grid: splat | 2 bilateral grid: blur | 3 bilateral grid: slice | 4 bilateral grid: divide

scanline: noisy step

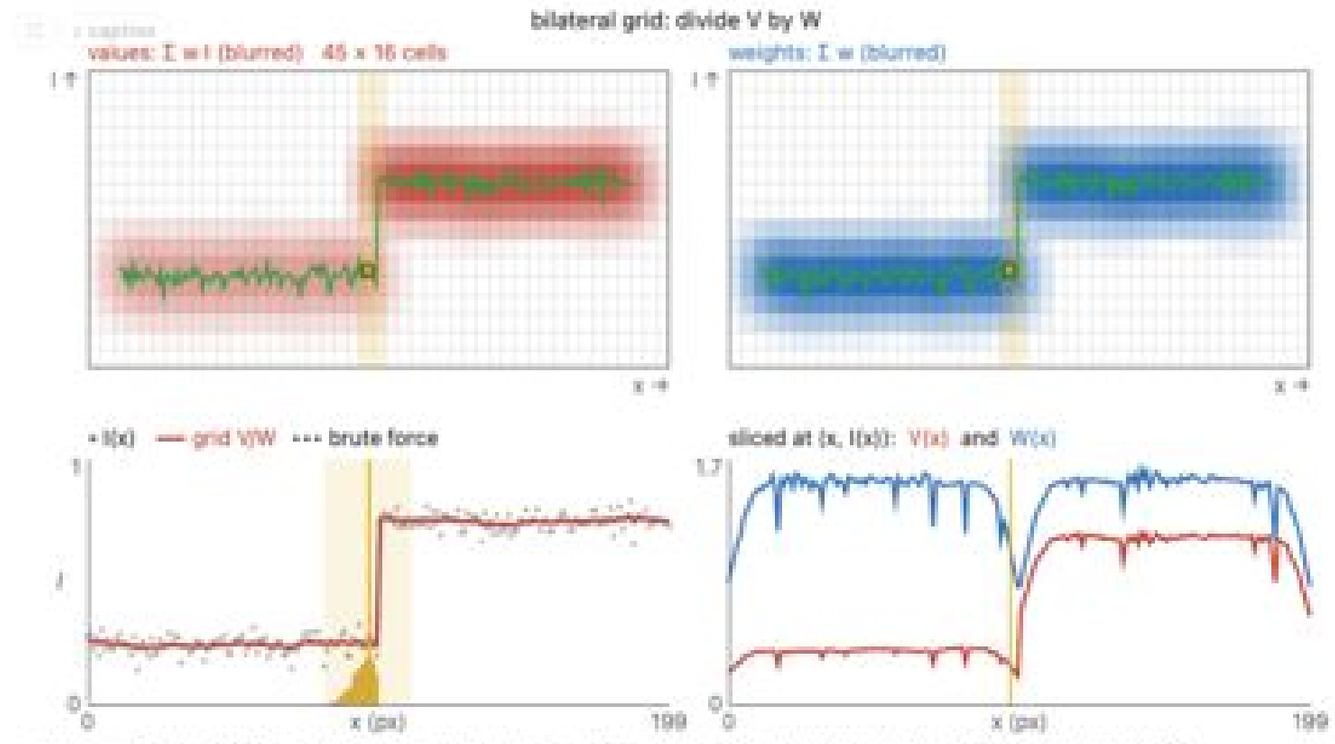
σ_s = one cell in x: 5.0 px

σ_r = one cell in I: 0.10

reset

Demo courtesy of Frédo Durand, MIT.

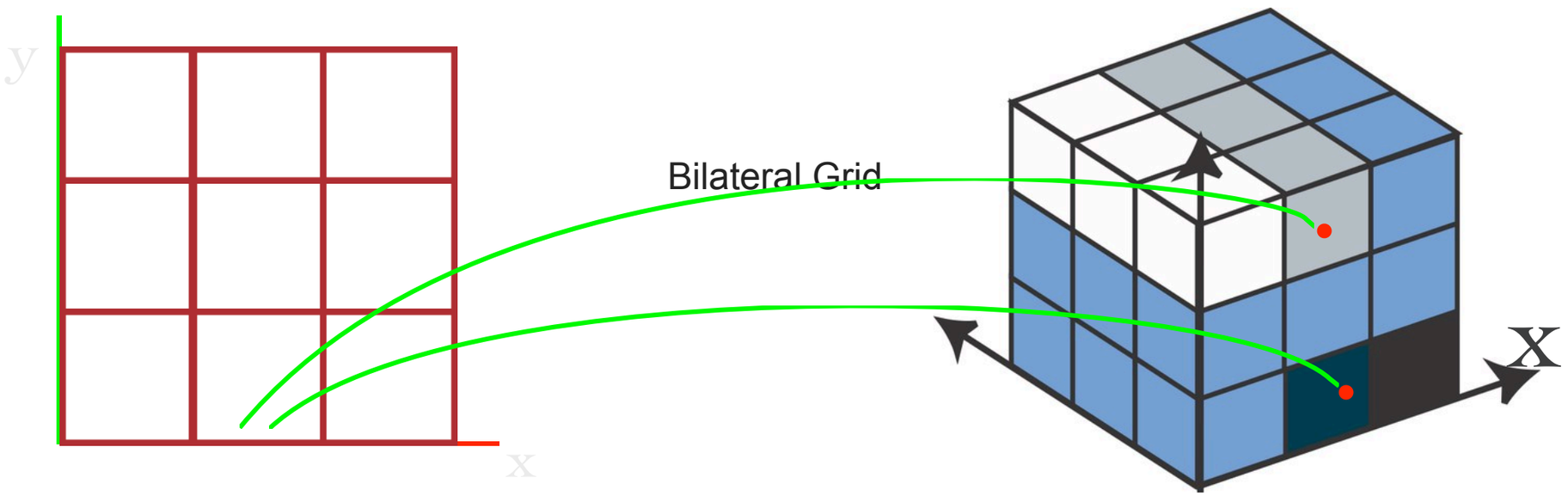
Demo: The bilateral grid in 1-D: divide



Demo courtesy of Frédo Durand, MIT.

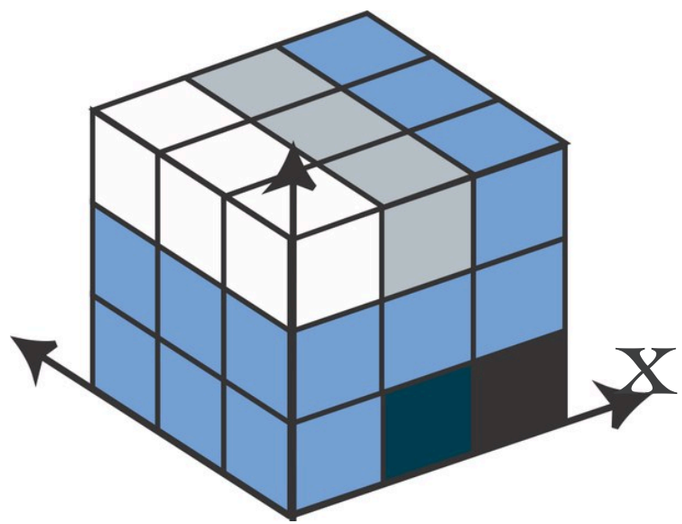
Grid creation

- **Convert image to bilateral grid**
  - (x, y) pixel goes to x, y, I(x,y)
- **Note that not all grid cells receive the same # of pixels**
  - empty cells shown in blue here
  - store a weight to keep track of #pixel
  - will give us normalization factor k in bilateral filter



Bilateral Grid data structure

- **3D array indexed by x, y, intensity**
- **Each cell stores**
  - a value (either RGB or just intensity)
  - a weight (keeps track of #pixels)
- **Resolution depends on application**
  - For bilateral filter, depends on σ_s and σ_r (σ should be ~ the width of a cell)



Implementation details

- **Probably a good idea to have helper functions to index grid directly from image x, y and I**
  - i.e. do the downsampling with appropriate scale factors (here σ_s and σ_r)

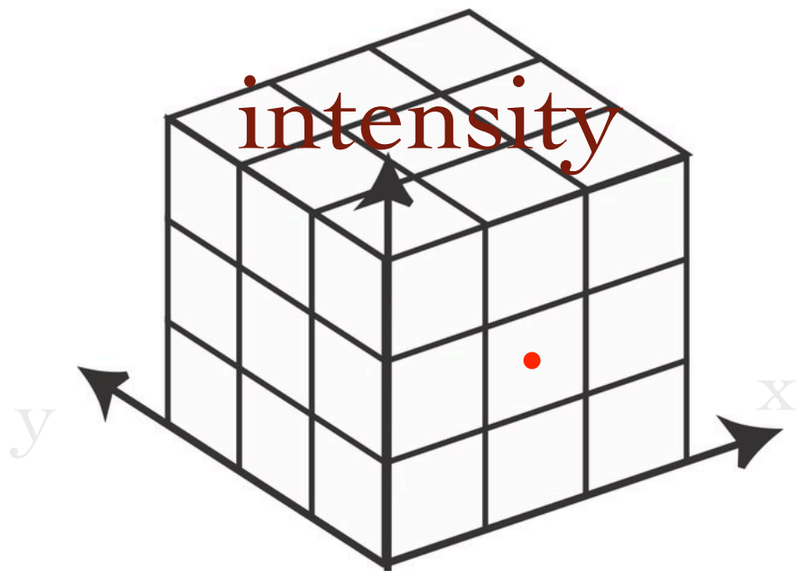
$$x, y, I \rightarrow \left[\frac{x}{\sigma_s}\right], \left[\frac{y}{\sigma_s}\right], \left[\frac{I}{\sigma_r}\right]$$

where [] denotes
integer truncation

Blurring the grid

- **Same as in 2D**
- **Each cell replaced by Gaussian-weighted average of neighbors**
  - if v is the value in grid, the blurred output b is:

$$b(x,y,i) = \sum_{x',y',i'} G_{\sigma_s}(x-x')\, G_{\sigma_s}(y-y')\, G_{\sigma_r}(i-i')\, v(x',y',i')$$

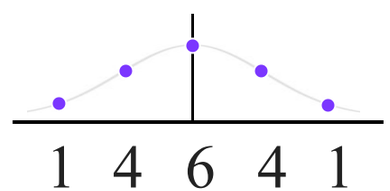


Even smarter: separable

- **Blur one axis at a time**
- **works because our blurring kernel is separable (defined as product along axes)**
- **e.g. blur along x axis:**

$$b(x, y, i) = \sum_{x'} G_{\sigma_s}(x - x')\, v(x', y, i)$$

- **If we have chosen sf = 1 cell width, then the Gaussian is simply [1 4 6 4 1]/16**



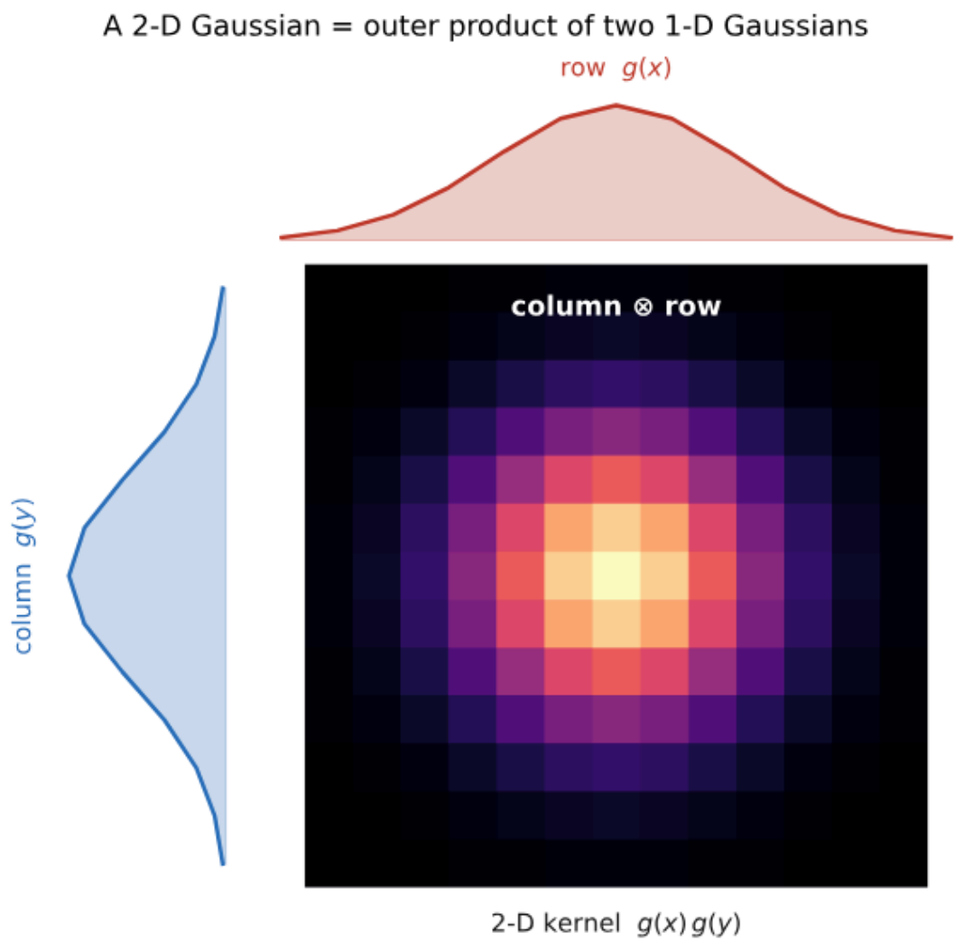
Why are Gaussians separable?

- **e.g. product of a Gaussian in x and one in y**

$$e^{-\frac{y^2}{2\sigma^2}} \times e^{-\frac{x^2}{2\sigma^2}} = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- **gives you a radial Gaussian**

Figure 6.2.15: Separability: a 2-D Gaussian = 1-D ⊗ 1-D



The 2-D Gaussian kernel (center) is the outer product of a horizontal 1-D Gaussian and a vertical 1-D Gaussian, so blurring the image is equivalent to blurring the rows…

Blurring

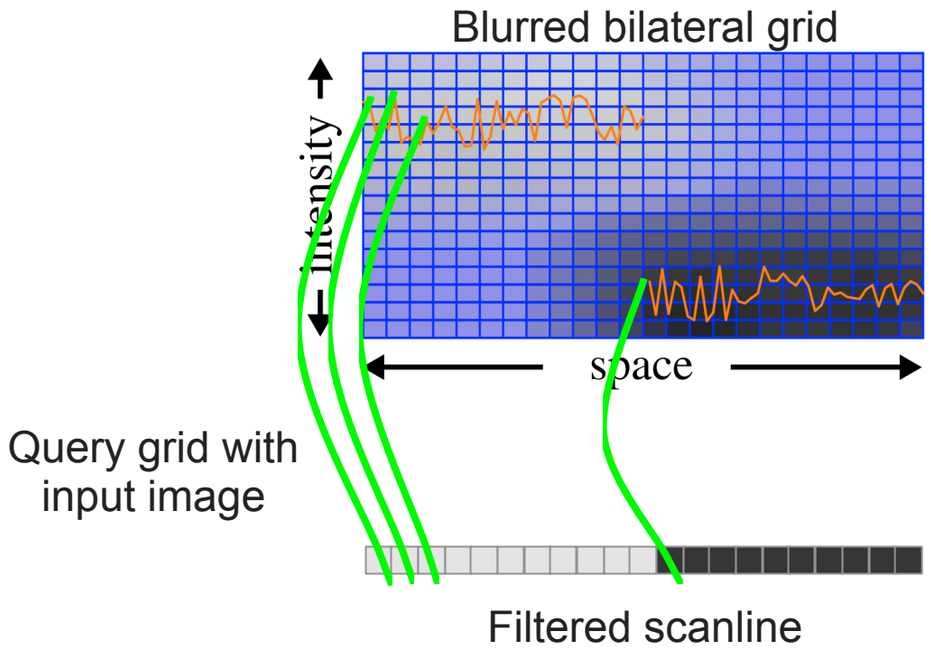
- **Blur BOTH the values and the weights**
- **Recall original bilateral filter formulas**

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

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

Slicing: critical step

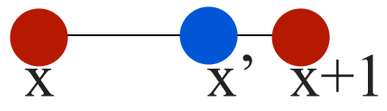
- **Read the grid at locations specified by input image**
  - Output at pixel x, y, is read from grid cell x, y, I(x, y)
- **Trilinear reconstruction**
  - because the grid is downsampled



Linear reconstruction

- **Say we only have values v at integer x**
- **We want to reconstruct at real-valued x'**
- **Linear reconstruction:**

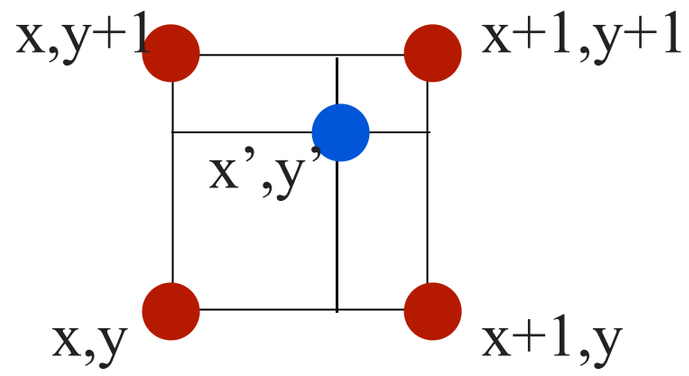
$$(1-x'+x)v(x, y)+(x'-x)v(x+1, y)]$$



BiLinear reconstruction

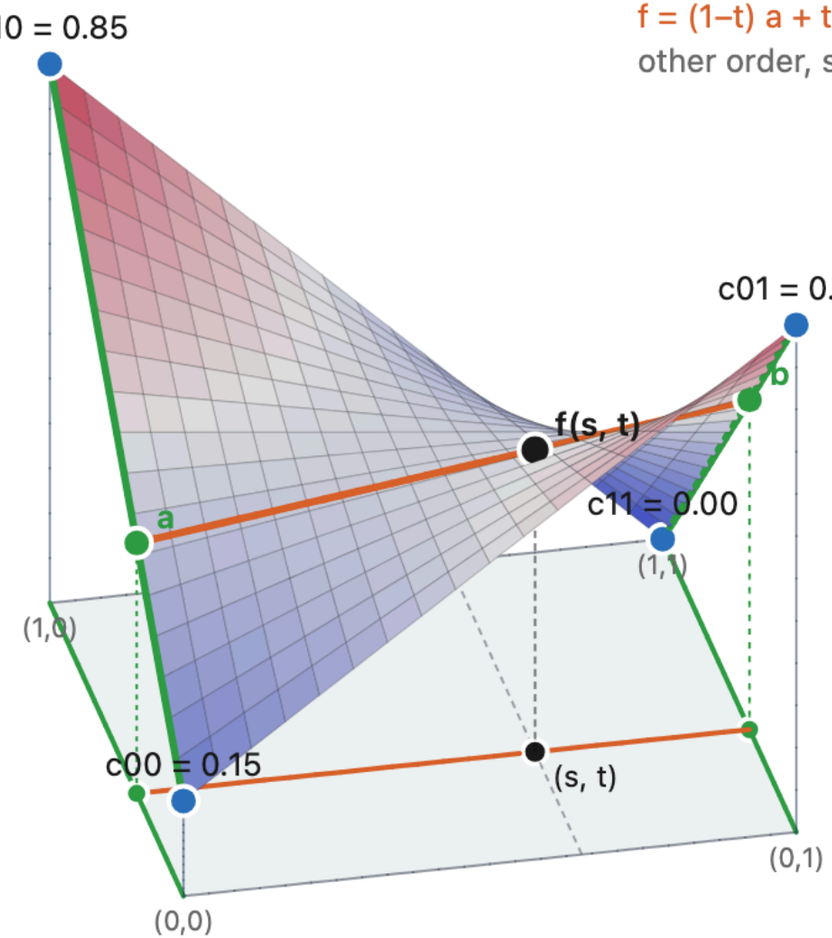
- **Say we only have values v at integer x & y**
- **We want to reconstruct at real-valued x', y'**
- **Bilinear reconstruction:**

$$(1-y'+y)[(1-x'+x)v(x, y)+(x'-x)v(x+1, y)] +$$
$$(y'-y)[(1-x'+x)v(x, y+1)+(x'-x)v(x+1, y+1)]$$



Demo: Bilinear height-field

a = (1−s) c00 + s c10 = 0.395
b = (1−s) c01 + s c11 = 0.520
f = (1−t) a + t b = 0.476
other order, same f = 0.476



non-coplanar → saddle: straight grid lines

Bilinear: order does not matter

- **Linear along x followed by linear along y**

$$(1-y'+y)[(1-x'+x)v(x, y)+(x'-x)v(x+1, y)] +$$
$$(y'-y)[(1-x'+x)v(x, y+1)+(x'-x)v(x+1, y+1)]$$

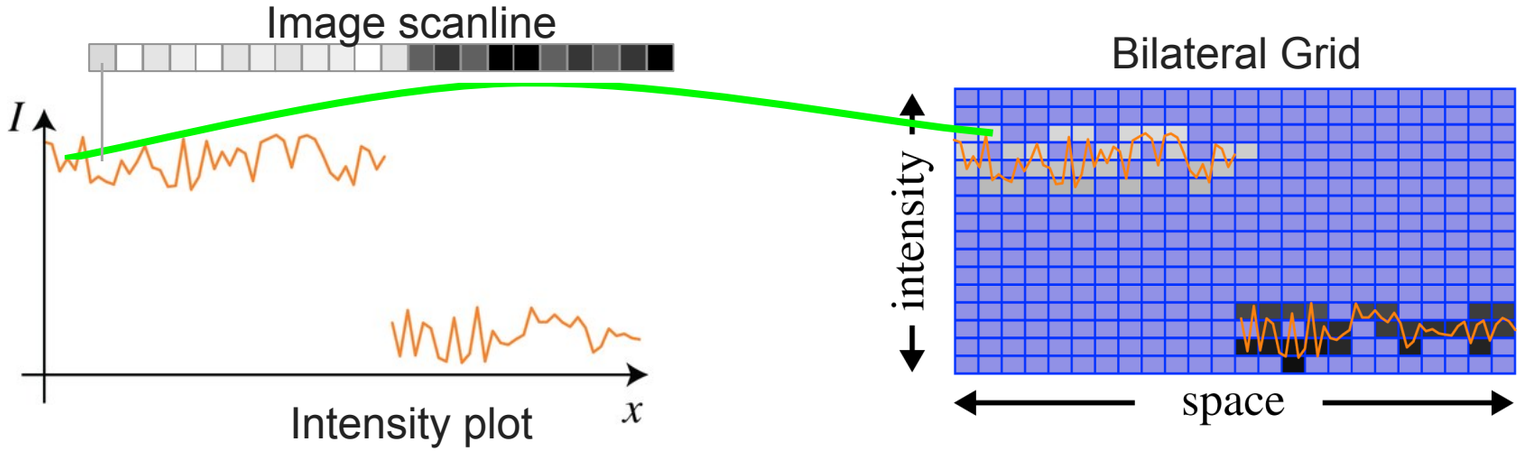
- **Linear along y followed by linear along x**

$$(1-x'+x)[(1-y'+y)v(x, y)+(y'-y)v(x, y+1)] +$$
$$(x'-x)[(1-y'+y)v(x+1, y)+(y'-y)v(x+1, y+1)]$$

- **Reduces to the same terms**

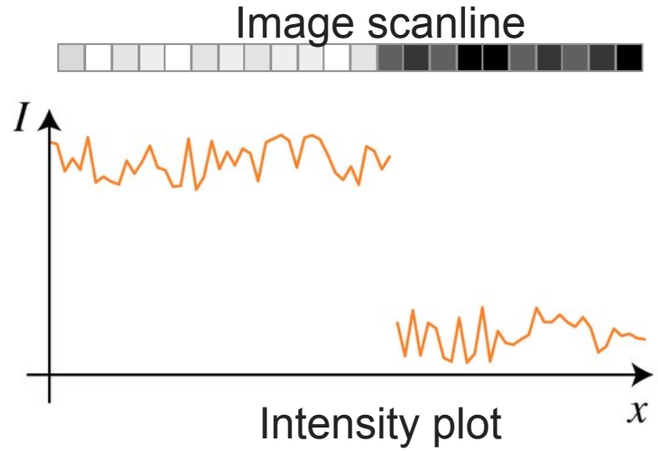
RECAP

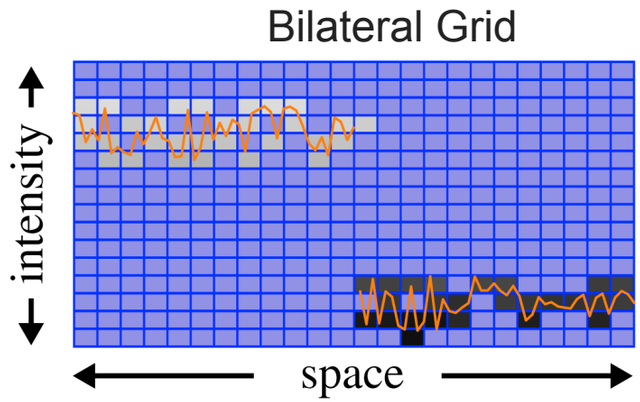
Bilateral Filter on the Bilateral Grid

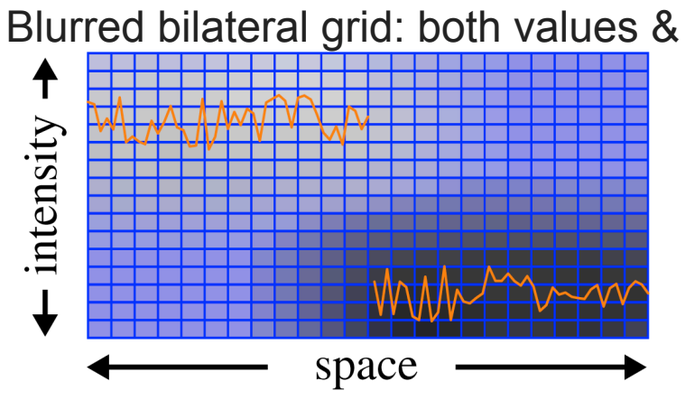


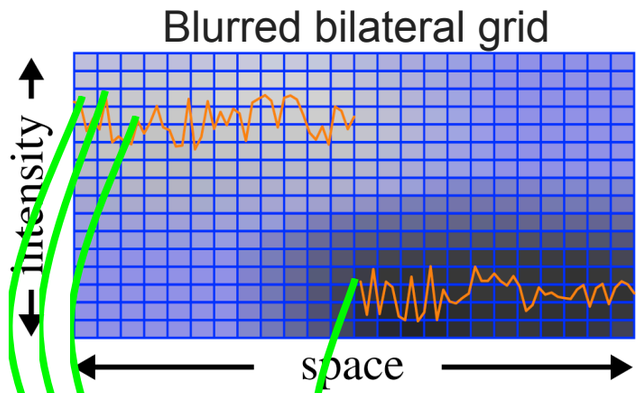
RECAP

Bilateral Filter on the Bilateral Grid









Query grid with
input image: output =
blurred value/blurred weight

Filtered scanline

Video: Real-time Edge-Aware Image Processing with the Bilateral Grid, Chen, Paris &…



Pseudo code

```
For each pixel x, y
      add I(x,y) to grid cell x/σ, y/σ, I(x,y)/σ
      add 1 to weight of grid cell x σ, y σ, I(x,y) σ
Blur values & weights along X axis
Blur values & weights along Y axis
Blur values & weights along I axis
For each pixel x, y
      output = value(x/σ, y/σ, I(x,y)/σ)
               / weight(x/σ,y/σ,I(x,y)/σ)
```

Grid creation

Grid Blur

Slicing

Questions?

Real code

```
class bilaGrid:
      def __init__(self, ref, sigmaS, sigmaR, factor=3.0):
            self.sigmaS=sigmaS
            self.sigmaR=sigmaR
            self.fxy=1.0/sigmaS*factor
            self.fr=1.0/sigmaR*factor
            self.factor=factor
            self.height=ref.shape[0]*self.fxy+2
            self.width=ref.shape[1]*self.fxy+2
            self.miniR=min(ref.flatten())
            self.offsetr=-self.miniR*self.fr+0.5
            self.range=(max(ref.flatten())-self.miniR)*self.fr+2
            self.grid=zeros([self.height, self.width, self.range, 3])
            self.weight=zeros([self.height, self.width, self.range])
            self.ref=ref
```

Real code

```
def creation(self, im):
      for y, x, in imIter(im):
            self.grid[y*self.fxy+0.5, x*self.fxy+0.5, self.ref[y, x]*self.fr+self.offsetr]+=im[y, x]
            self.weight[y*self.fxy+0.5, x*self.fxy+0.5, self.ref[y, x]*self.fr+self.offsetr]+=1.0
def blur(self):
      self.grid=ndimage.filters.gaussian_filter(self.grid, [self.factor, self.factor, self.factor, 0])
      self.weight=ndimage.filters.gaussian_filter(self.weight, [self.factor, self.factor, self.factor])
def slicing(self):
      h=self.ref.shape[0]
      w=self.ref.shape[1]
      newy, newx=self.fxy*mgrid[0:h, 0:w]
      coord=array([newy, newx, self.ref*self.fr+self.offsetr])
      out=empty([h, w, 3])
      weight=ndimage.map_coordinates(self.weight, coord, order=1)+0.0000000001
      for i in xrange(3):
            out[:, :, i]=ndimage.map_coordinates(self.grid[:, :, :, i], coord, order=1)/weight
```

Scipy to the rescue

- `ndimage.filters.gaussian_filter(self.grid, [self.factor, self.factor, self.factor, 0])`
- applies separable Gaussian blur
- with possibly-different stddev for each axis

Scipy to the rescue

- `ndimage.map_coordinates(self.grid[:, :, :, i], coord, order=1)`
- resamples the first argument at the locations given by the second one.
- Can use various orders of interpolation
  - I used 1 for tri-linear

Real code

```
ef creation(self, im):
     for y, x, in imIter(im):
           self.grid[y*self.fxy+0.5, x*self.fxy+0.5, self.ref[y, x]*self.fr+self.offsetr]+=im[y, x]
           self.weight[y*self.fxy+0.5, x*self.fxy+0.5, self.ref[y, x]*self.fr+self.offsetr]+=1.0
ef blur(self):
     self.grid=ndimage.filters.gaussian_filter(self.grid, [self.factor, self.factor, self.factor, 0])
     self.weight=ndimage.filters.gaussian_filter(self.weight, [self.factor, self.factor, self.factor])
ef slicing(self):
     h=self.ref.shape[0]
     w=self.ref.shape[1]
     newy, newx=self.fxy*mgrid[0:h, 0:w]
     coord=array([newy, newx, self.ref*self.fr+self.offsetr])
     out=empty([h, w, 3])
     weight=ndimage.map_coordinates(self.weight, coord, order=1)+0.0000000001
     for i in xrange(3):
           out[:, :, i]=ndimage.map_coordinates(self.grid[:, :, :, i], coord, order=1)/weight
```

Bells and whistles

- http://dspace.mit.edu/handle/1721.1/67030

- only modify pixels according to sum of weights
- avoids problems for pixels who have few neighbors in the same range
- implemented as weighted average of input and standard bilateral filter

Figure 10.1.12: Edge-aware manipulation in real time



From Chen, Paris & Durand (2007).

Running the splat–blur–slice pipeline on the GPU makes local tone mapping, and other edge-respecting adjustments, fast enough to apply live as the user works.

Bilateral upsampling

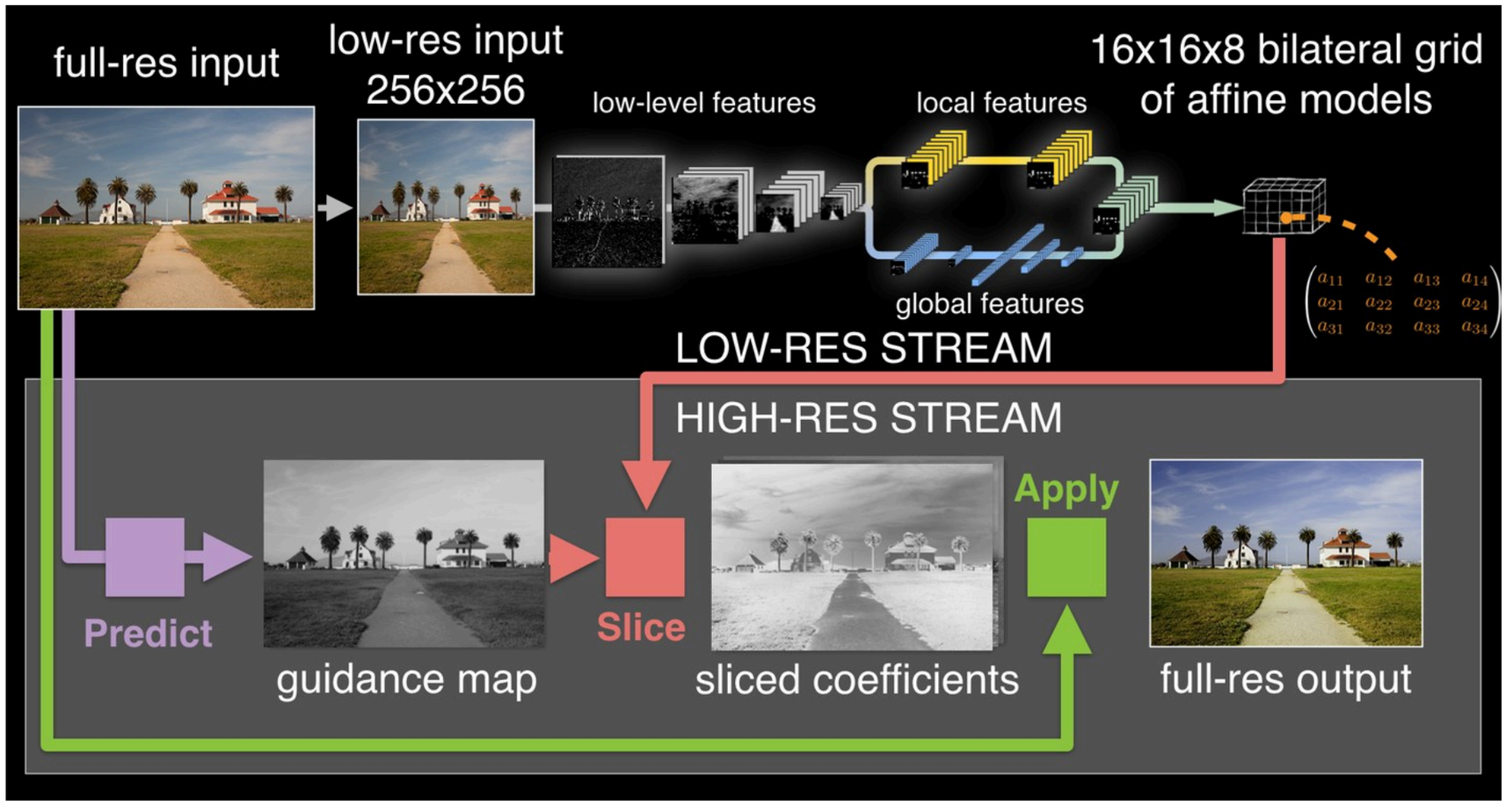
- Compute on a small copy; upsample guided by the full image
  - Joint bilateral upsampling [Kopf et al. 2007]

$$\tilde{S}_p = \frac{1}{W_p} \sum_{q_\downarrow} S_{q_\downarrow}\, g_s\big(\|p_\downarrow - q_\downarrow\|\big)\, g_r\big(|J_p - J_q|\big)$$

- Upsample the operator, not the image [Chen et al. 2016]
  - fit an affine color transform per bilateral-grid cell at low resolution; slice it at full resolution and apply it to the original pixels, which keep their detail
- Deep learning: keep the network at low resolution, let the edge-aware step carry it back
  - HDRnet [Gharbi et al. 2017]: the network predicts the grid of affine transforms; runs live in a phone viewfinder
  - Bilateral solver [Barron and Poole 2016]: differentiable edge-aware refinement of depth, segmentation, colorization
  - Trainable bilateral and guided-filter layers [Jampani et al. 2016; Wu et al. 2018]
  - Neural rendering: a learned grid per view absorbs the phone's processing [Wang et al. 2024]
- The guide transfers the edges it has, not the ones it lacks

Deep learning with Bilateral grid

- Gharbi et al. 2017
  - **https://groups.csail.mit.edu/graphics/hdrnet/**



Video: Deep Bilateral Learning for Real-Time Image Enhancement (HDRnet)

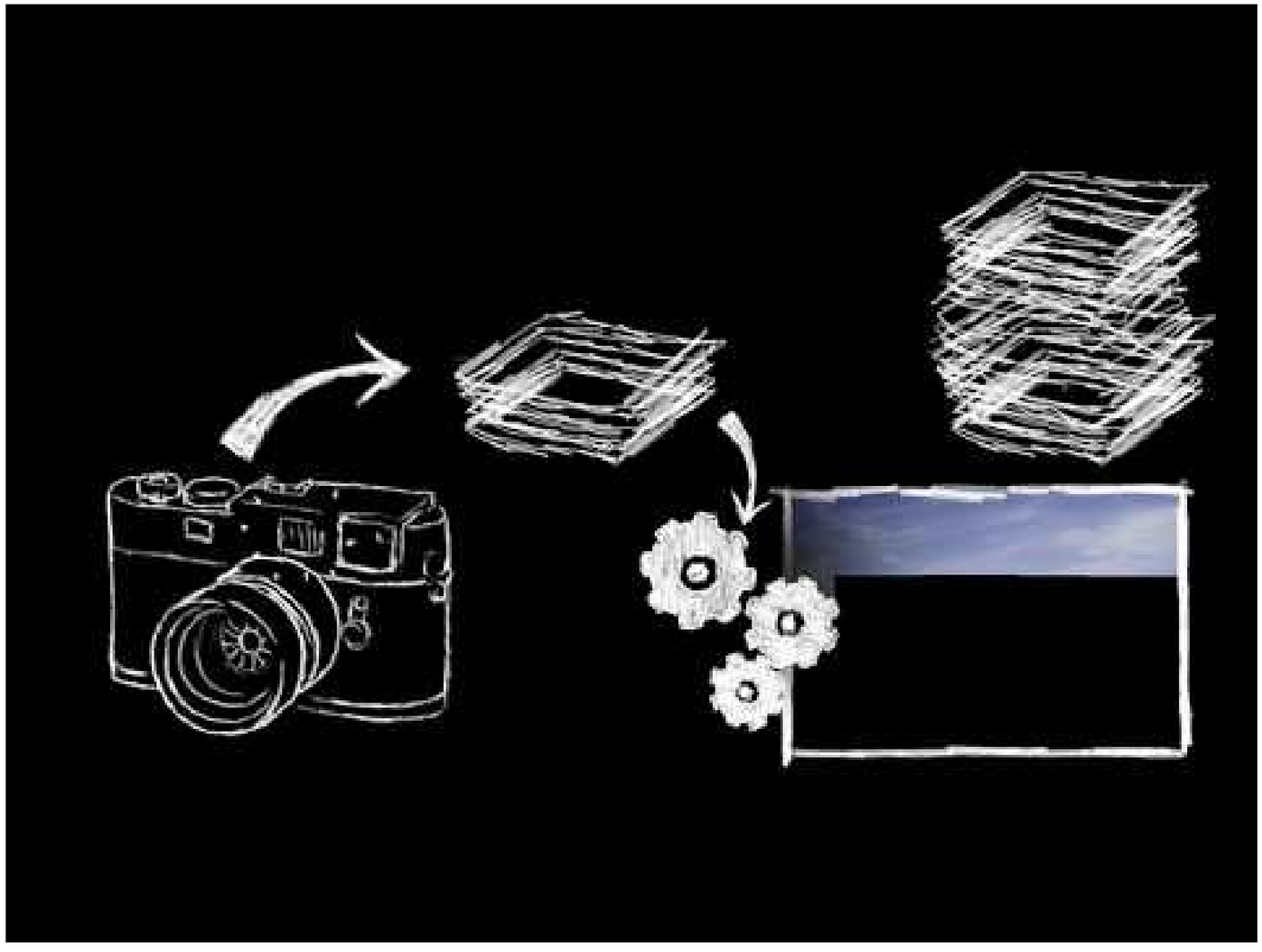


Figure 7.3.4: What learned retouching buys you



From Gharbi, Chen, Barron, Hasinoff & Durand (2017).

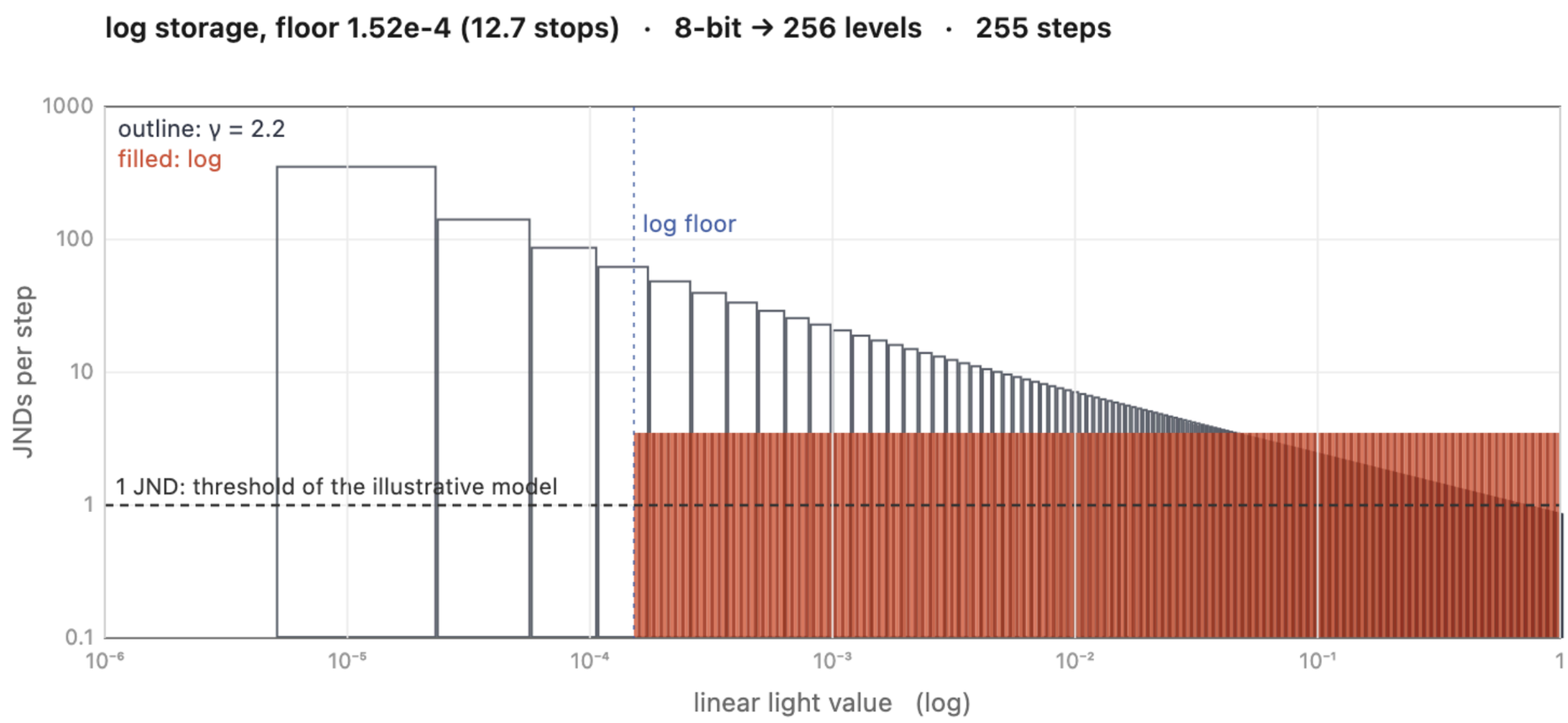
A set of ordinary input photos beside HDRnet's enhanced versions, the adjustment learned from expert before/after pairs rather than dialed in by hand.

HDR encoding

HDR encoding

- **Most formats are lossless**
- **Adobe DNG (digital negative)**
  - Specific for RAW files, avoid proprietary formats
- **RGBE**
  - 24 bits/pixels as usual, plus 8 bit of common exponent
  - Introduced by Greg Ward for Radiance (light simulation)
  - Enormous dynamic range
- **OpenEXR**
  - By Industrial Light + Magic, also standard in graphics hardware
  - 16bit per channel (48 bits per pixel) 10 mantissa, sign, 5 exponent
  - Fine quantization (because 10 bit mantissa), only 9.6 orders of magnitude
- **JPEG 2000**
  - Has a 16 bit mode, lossy

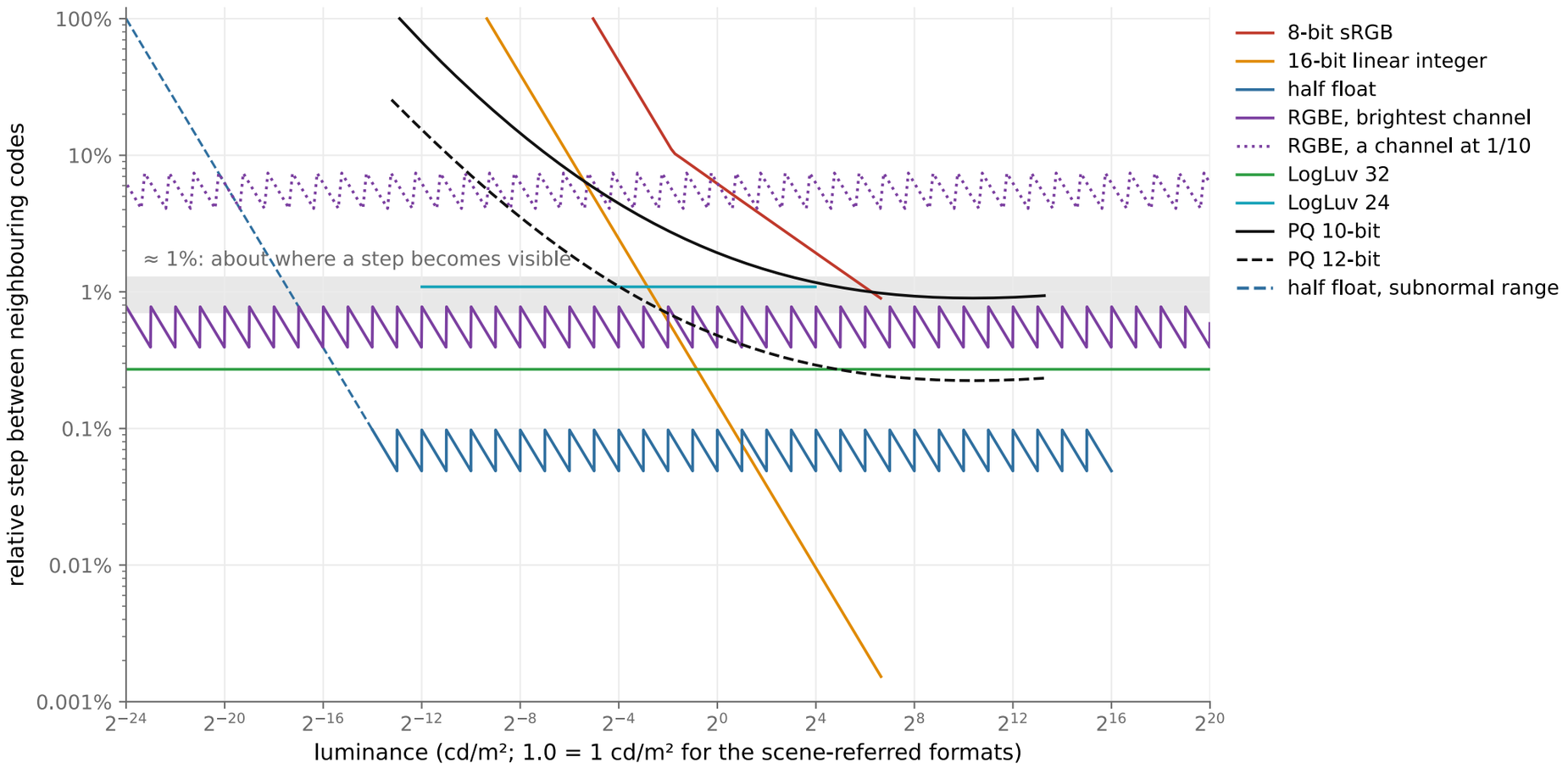
Demo: How many just-noticeable differences is one log code step worth?…



log: 255/255 steps over 1 JND, worst ≈ 3.5, median ≈ 3.5

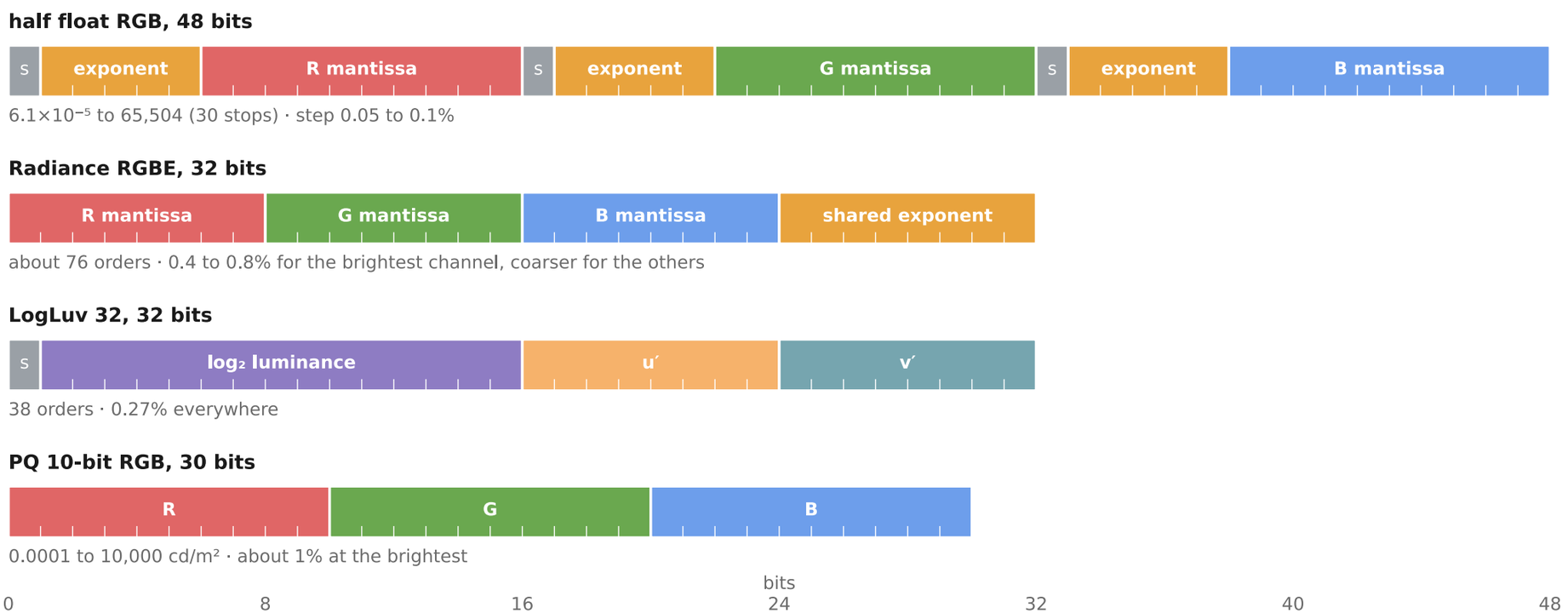
γ = 2.2: 220/254 steps over 1 JND, worst ≈ 359, median ≈ 1.7 (1 left out: from 0 or above white)

Figure 17.7.1: The relative step between neighboring codes, against luminance over…



The relative step between neighboring codes, against luminance over about 43 stops, for each encoding in this chapter, every curve computed code by code from the…

Figure 17.7.2: One pixel's bits in four formats, to scale



Half float RGB spends 48 bits, a sign, a 5-bit exponent and a 10-bit mantissa per channel, for about 30 stops at a step under 0.1%. Radiance RGBE spends 32, three 8-bit…

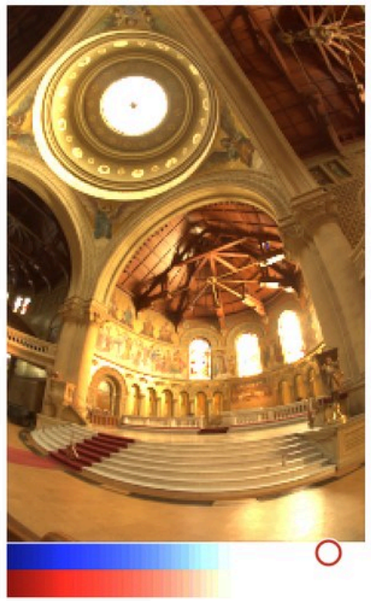
HDR formats

- Summary of all HDR encoding formats (Greg Ward): **http://www.anyhere.com/gward/hdrenc/hdr_encodings.html**
- Greg's notes: **http://www.anyhere.com/gward/pickup/CIC13course.pdf**
- **http://www.openexr.com/**
- High Dynamic Range Video Encoding

(MPI) **http://www.mpi-sb.mpg.de/resources/hdrvideo/**

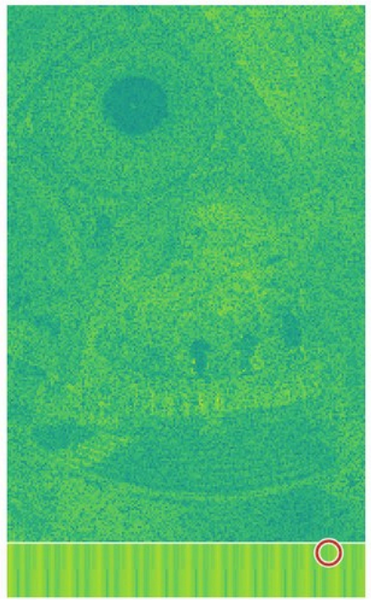
Demo: HDR formats, round trip

decoded from RGBE



probe at pixel (162, 276), on the blue ramp

relative error, worst channel



0.0001% 0.001% 0.01% 0.1% 1% 10% 100%

		decoded at the probe (mid-gray = 0.18)				over the photograph		
format	**bits / px**	**R**	**G**	**B**	**error there**	**median error**	**99th pct.**	**> 1%**
original (float32)		9.894	19.79	494.7				
sRGB, 8 bit	24	1.440	1.440	1.440	100%	1.3%	98%	64%
linear, 16 bit	48	9.902	19.79	494.7	0.085%	11%	≥ 100%	93%
half float	48	9.891	19.78	494.8	0.030%	0.027%	0.045%	0
RGBE	**32**	**9.000**	**19.00**	**495.0**	**9.0%**	**1.2%**	**6.1%**	**57%**
LogLuv24	24	8.847	20.30	491.1	11%	4.7%	≥ 100%	94%
LogLuv32	32	9.119	20.09	494.7	7.8%	3.1%	10%	84%
PQ, 10 bit	30	39.00	22.97	48.70	≥ 100%	1.0%	25%	52%
gain map	25.53	9.808	19.63	491.4	0.87%	1.5%	≥ 100%	71%

Optimal Weights

Problem setup

- We may have multiple valid observations for a given pixel
- They have different noise characteristics
- How can we combine them optimally?

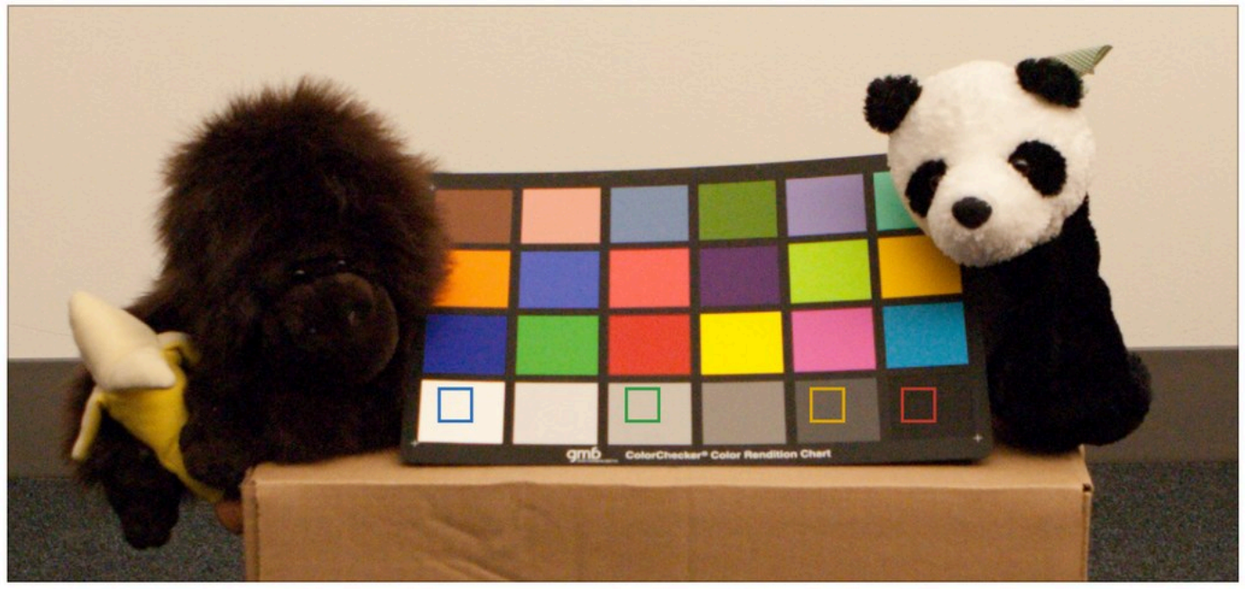


Simple cases

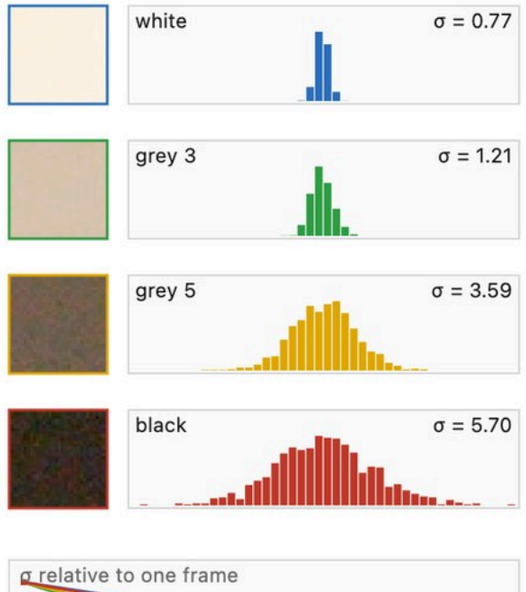
- **They have the same noise**
  - Just take the average: see pset 3
  - Noise reduced by sqrt(N)
- **If one is a lot more noisy?**
  - Probably focus on the other one.
  - But if we only use the less noisy one, we don’t get any noise reduction
  - There has to be a way to use the second one at least a little bit.

Demo: The same claim on the course's own frames

the average of 8 frames

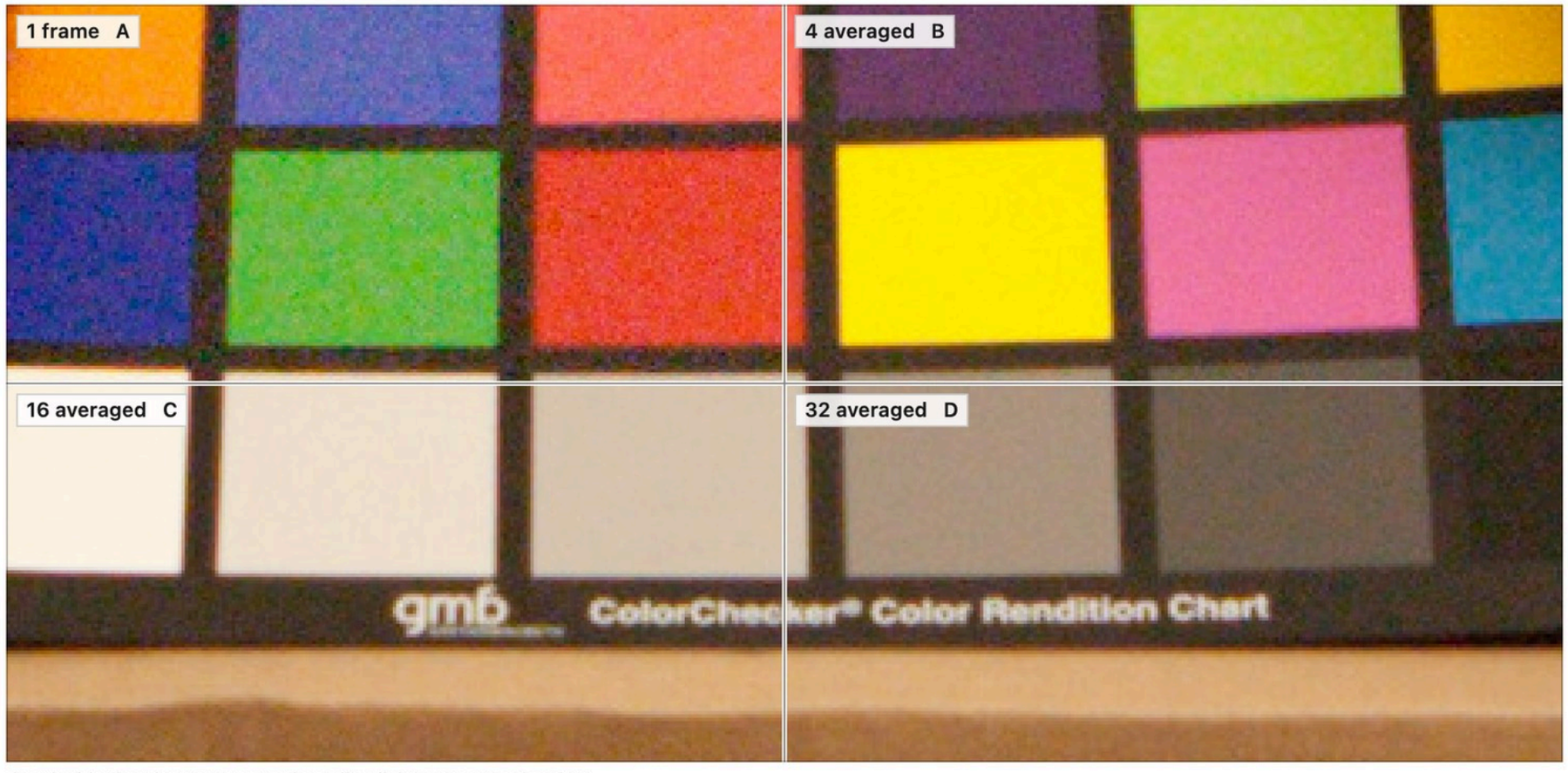


ISO 3200, aligned in advance; the same scene the problem set uses · hover to magnify



Demo: Whether you can see it

4 panes of the same view



drag inside the picture to move the split; click to recentre the view

Simple case

- Two observations x & y of the same quantity
  - with given variances

$$\sigma^2[x] \quad \sigma^2[y]$$

- Compute estimate as ax+(1-a)y
- What is the optimal a?

Minimize variance

- Variance of the combination:

$$\sigma^2[ax + (1-a)y] = a^2\sigma^2[x] + (1-a)^2\sigma^2[y]$$

- To minimize: derive, set to zero

$$2a\sigma^2[x] - 2(1-a)\sigma^2[y] = 0$$

$$a\left(\sigma^2[x] + \sigma^2[y]]\right) = \sigma^2[y]$$

$$a = \frac{\sigma^2[y]}{\sigma^2[x] + \sigma^2[y]]}$$

Optimal combination

$$\frac{\sigma^2[y]}{\sigma^2[x]+\sigma^2[y]]}x + \frac{\sigma^2[x]}{\sigma^2[x]+\sigma^2[y]]}y$$

- Put the x terms together

$$\frac{\sigma^2[x]\sigma^2[y]}{\sigma^2[x]+\sigma^2[y]}\left(\frac{1}{\sigma^2[x]}x + \frac{1}{\sigma^2 y}y\right)$$

normalization term

- The optimal combination weights estimators according to the inverse of their variance

Verify for same variance

General formula

- Weight each estimator by the inverse variance

$$\frac{1}{\sum \frac{1}{\sigma^2[x_i]}} \sum \frac{x_i}{\sigma^2[x_i]}$$

Optimal HDR

Recall: Assembling HDR

- **Figure out scale factor between images**
  - From exposure data
  - Or by looking at ratios $I_i(x,y)/I_i(x,y)$
    - but only when both are good
- **Compute weight map w_i for each image**
  - binary so far
- **reconstruct full image using weighted combination**

$$out(x,y) = \frac{1}{\sum w_i(x,y)} \sum w_i(x,y) \frac{1}{k_i} I_i(x,y)$$

Pixel noise and variance

- Recall: noise is characterized by its variance
- i.e. each pixel value comes from a true value plus some noise added
- We can calibrate this noise by taking multiple exposures
- or we can derive variance equations using pen and paper

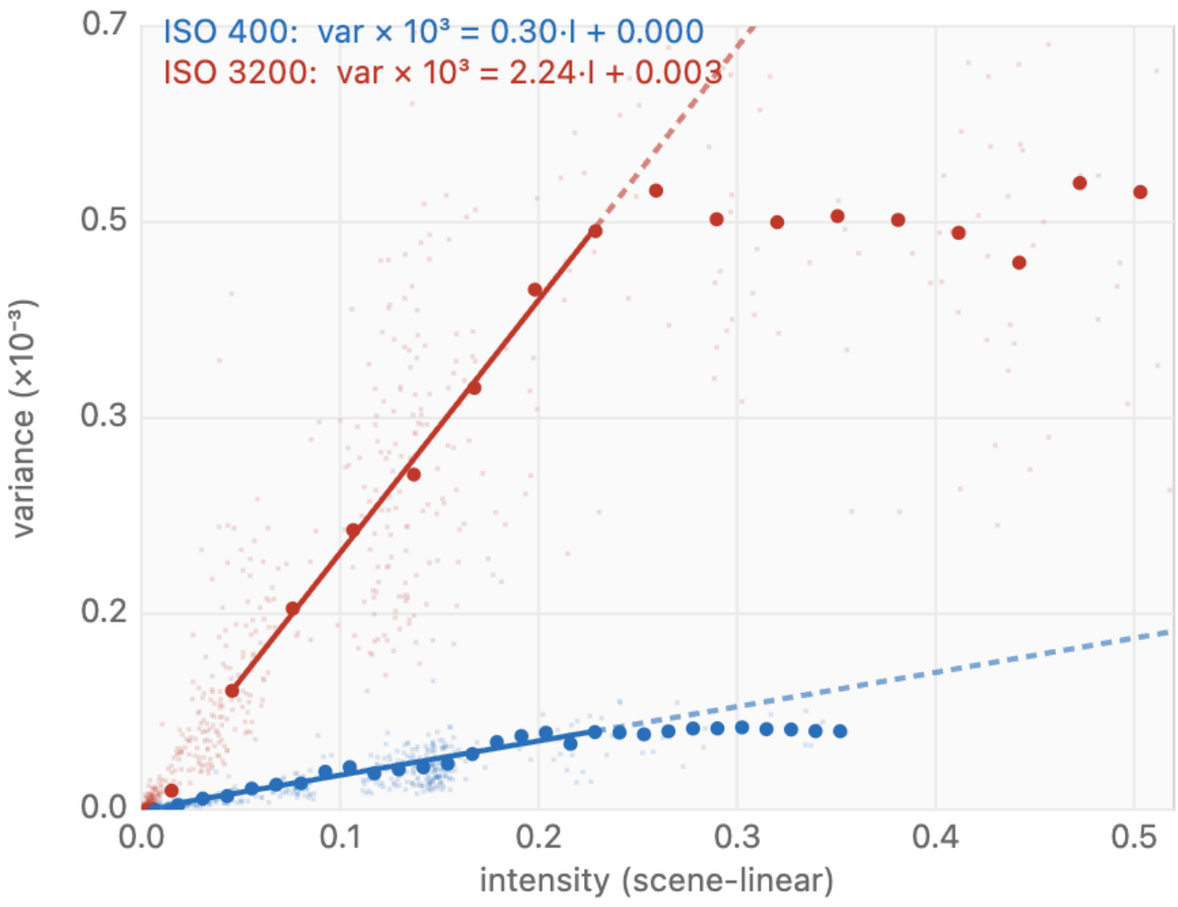
Sources of noise

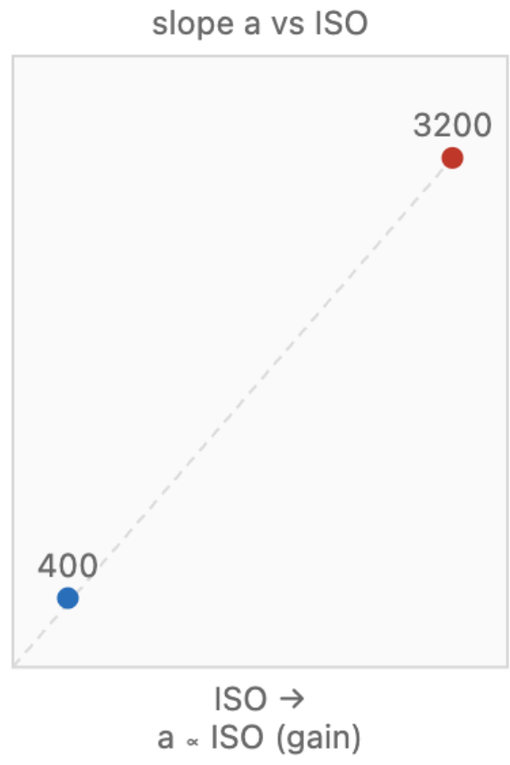
- **Photon noise**
  - Variance proportional to signal
  - dominates for bright pixels
- **Read noise**
  - Constant variance
  - Dominates for dark pixels
- **For a pixel value x:**

  - where *a* and $\sigma^2[x] \approx ax + \sigma^2_{read}$era and ISO

$$\sigma^2_{read}$$

Demo: Noise is affine





Demo: Noise calculator

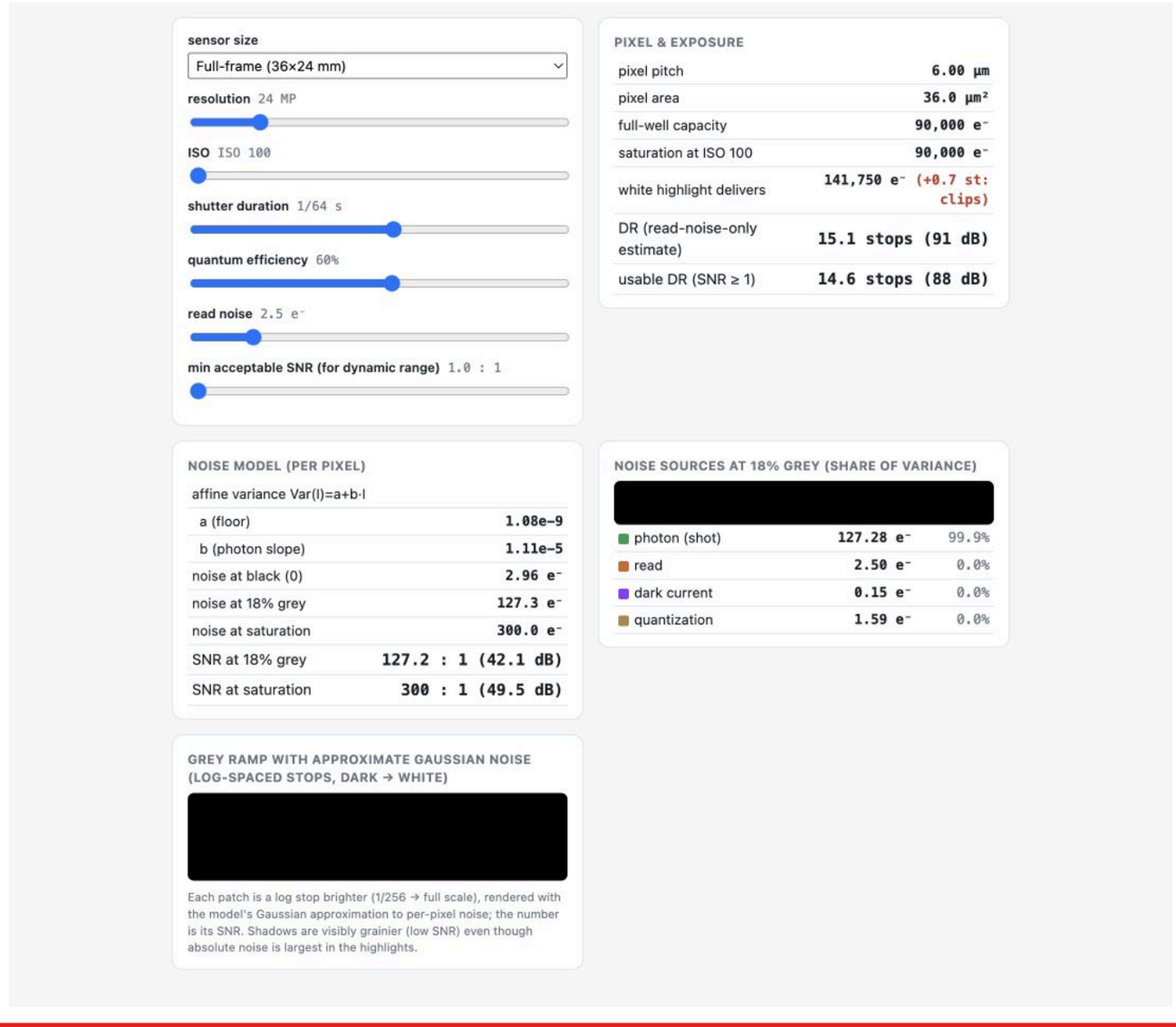
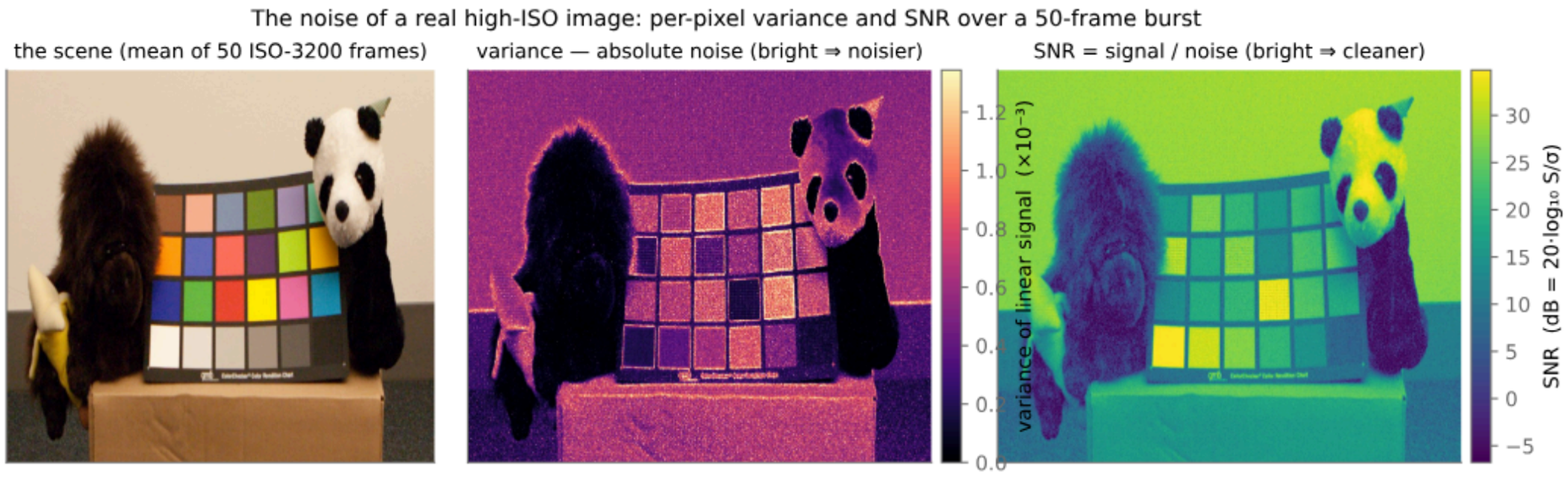


Figure 2.8.4: The noise of a real high-ISO image, and its SNR



From a 50-frame ISO-3200 burst of one static scene: the scene itself (left, mean of the burst), the per-pixel variance across the burst (middle), and the per-pixel SNR =…

Optimal weights

- Recall irradiance formula

$$I_i(x, y)=clip(k_i\ L(x,y)+n)$$

- and HDR merging formula

$$out(x,y) = \frac{1}{\sum w_i(x,y)} \sum w_i(x,y) \frac{1}{k_i} I_i(x,y)$$

Optimal weights

- Recall irradiance formula

$$I_i(x, y)=\text{clip}(k_i\, L(x,y)+n)$$

- and HDR merging formula

$$out(x,y) = \frac{1}{\sum w_i(x,y)} \sum w_i(x,y) \frac{1}{k_i} I_i(x,y)$$

- 1/k amplifies signal and noise:

$$\sigma^2[\frac{1}{k_i} I_i(x,y)] = \frac{1}{k_i^2} \left[a I_i(x,y) + \sigma_{read}^2 \right]$$

- repla

$$\sigma^2[\frac{1}{k_i} I_i(x,y)] = \frac{1}{k_i^2} \left[a k_i L(x,y) + \sigma_{read}^2 \right]$$

Variance of one scaled image

$$\sigma^2[\frac{1}{k_i}I_i(x,y)] = \frac{1}{k_i^2}\left[aI_i(x,y) + \sigma_{read}^2\right]$$

$$\sigma^2[\frac{1}{k_i}I_i(x,y)] = \frac{1}{k_i^2}\left[ak_iL(x,y) + \sigma_{read}^2\right]$$

$$\sigma_{read}^2$$

- If we only look at photon noise, mostly proportional to scale factor
- Ideally, should all be calibrated
- Note that we ignored ISO variations

Improved weight maps

- Old formula

$$out(x,y) = \frac{1}{\sum w_i(x,y)} \sum w_i(x,y) \frac{1}{k_i} I_i(x,y)$$

- Variance per pixel per image:

$$\sigma^2[\frac{1}{k_i} I_i(x,y)] \approx \frac{a}{k_i} L(x,y)$$

- replace wi by w'i
  - still use wi to reject dark and bright pixels
  - but also weight by inverse variance

$$w_i'(x,y) = w_i(x,y) / \frac{a}{k_i} L(x,y)$$

Improved weight maps

- New formula

$$out(x,y) = \frac{1}{\sum w_i'(x,y)} \sum w_i'(x,y) \frac{1}{k_i} I_i(x,y)$$

  with

- Which gives us

$$w_i'(x,y) = w_i(x,y) / \frac{a}{k_i} L(x,y)$$

  - *a* and *L(x,y)* are constant per pixel and present both in the main sum and t
  - the two ki in the ma

$$out(x,y) = \frac{1}{\sum w_i(x,y) \frac{k_i}{}} \sum w_i(x,y) \frac{1}{} I_i(x,y)$$



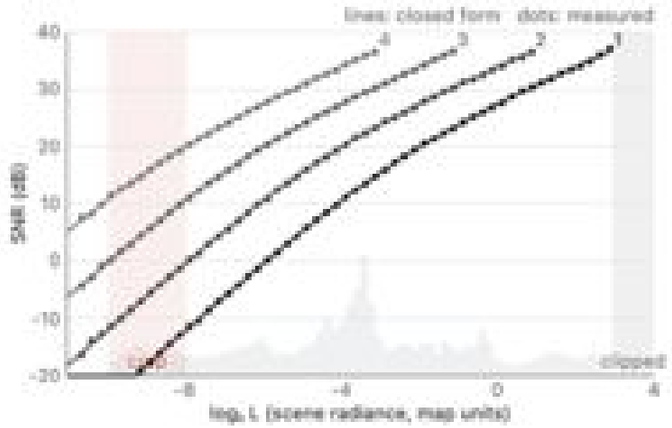
Improved weight maps

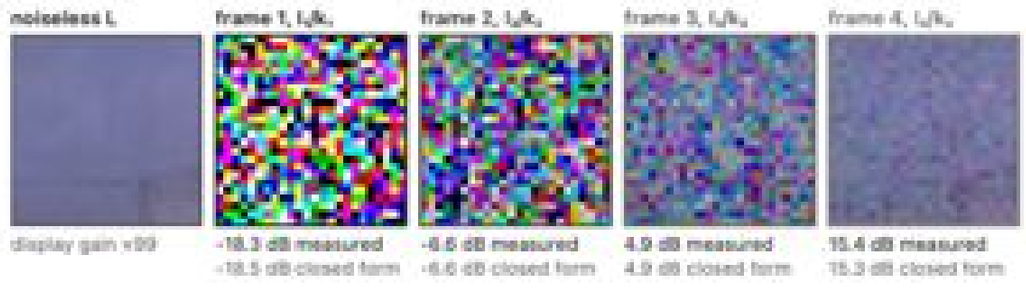
- **New formula**

  - The radiant [redacted] pixels of a given exposure are weighted the same.
  - This is because the *relative photon* noise changes similarly for all pixels between a pair of exposures
    - Would be different with read noise
- ki has disappeared from the main sum. The images are not really rescaled to scene radiant power
  - But they indirectly are because of the normalization
  - Recall that ki and 1/ki used to both appear

Demo: Optimal weights for the HDR merge: noise model: each frame's SNR







1 noise model: each frame's SNR | 2 merge: binary validity | 3 merge: hat weights | 4 merge: weights ~ k_i | 5 merge: inverse variance

radiance map: office window | frames N: 4

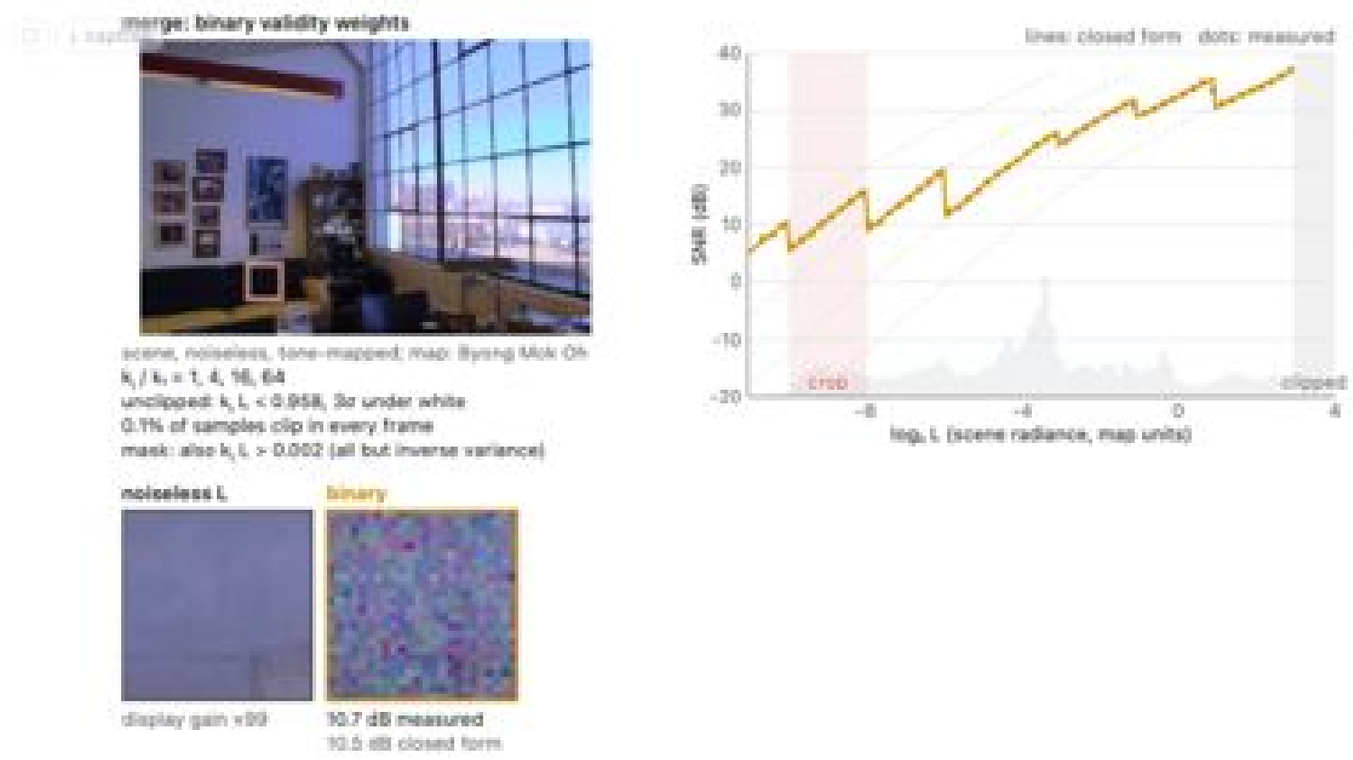
EV spacing: 2 stops | shortest exposure, vs the highlights: 0 stops

photon gain a: $a = 2.0 \times 10^{-4}$ (full well $1/a = 5{,}012$ e$^-$)

read noise σ_{read}: $\sigma_{read} = 2.0 \times 10^{-4}$ (10 e$^-$) | reset

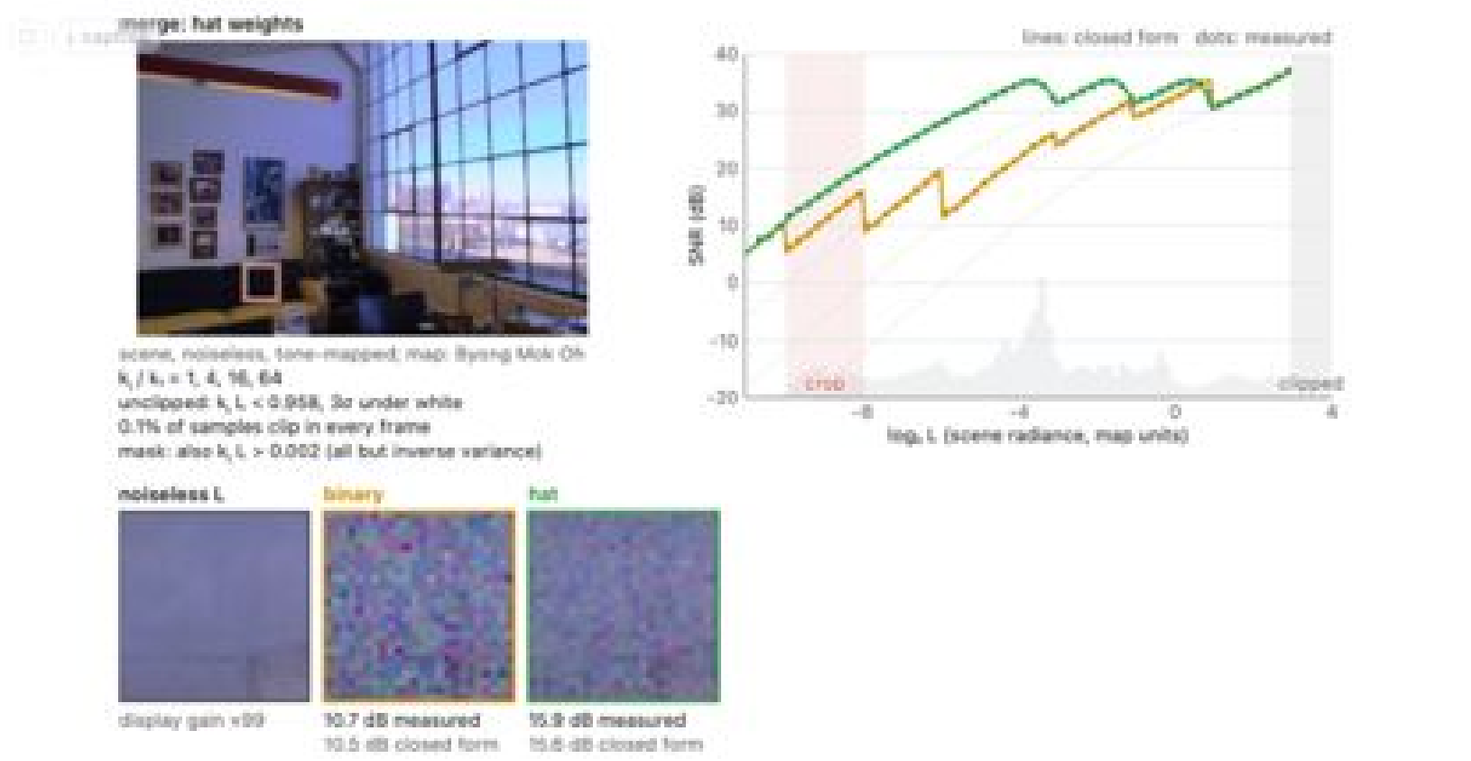
Demo courtesy of Frédo Durand, MIT.

Demo: Optimal weights for the HDR merge: binary validity



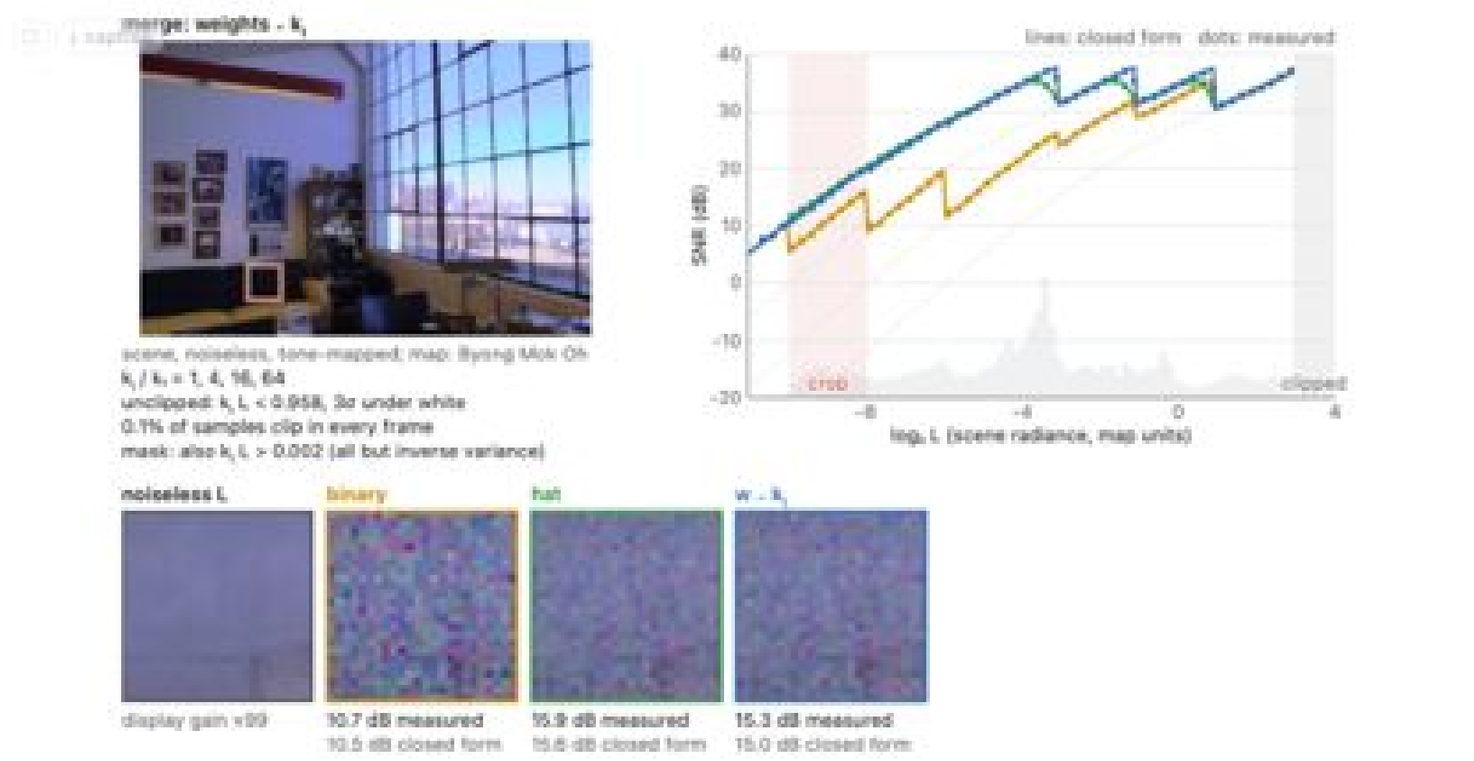
Demo courtesy of Frédo Durand, MIT.

Demo: Optimal weights for the HDR merge: hat weights



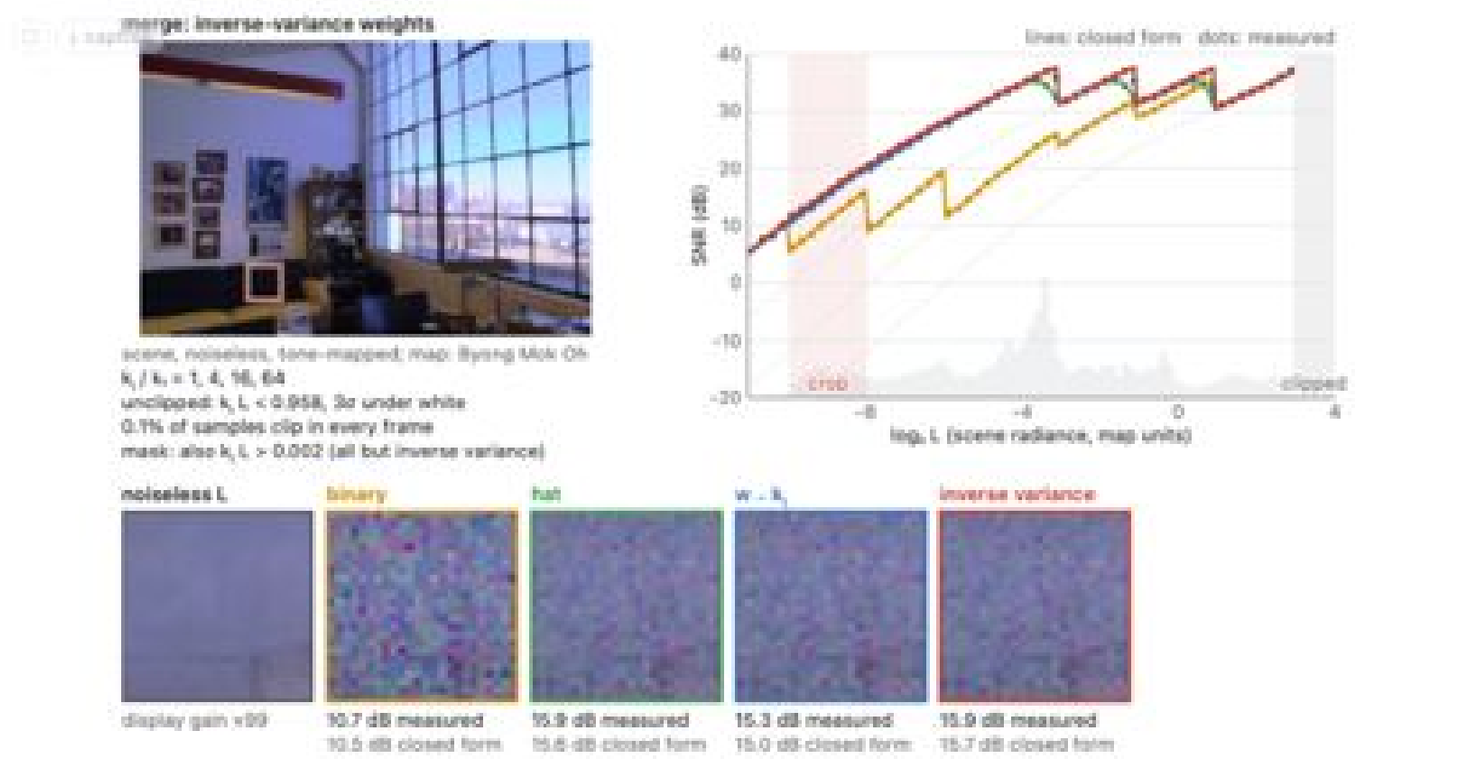
Demo courtesy of Frédo Durand, MIT.

Demo: Optimal weights for the HDR merge: weights $\propto k_i$



Demo courtesy of Frédo Durand, MIT.

Demo: Optimal weights for the HDR merge: inverse variance



Demo courtesy of Frédo Durand, MIT.

Results





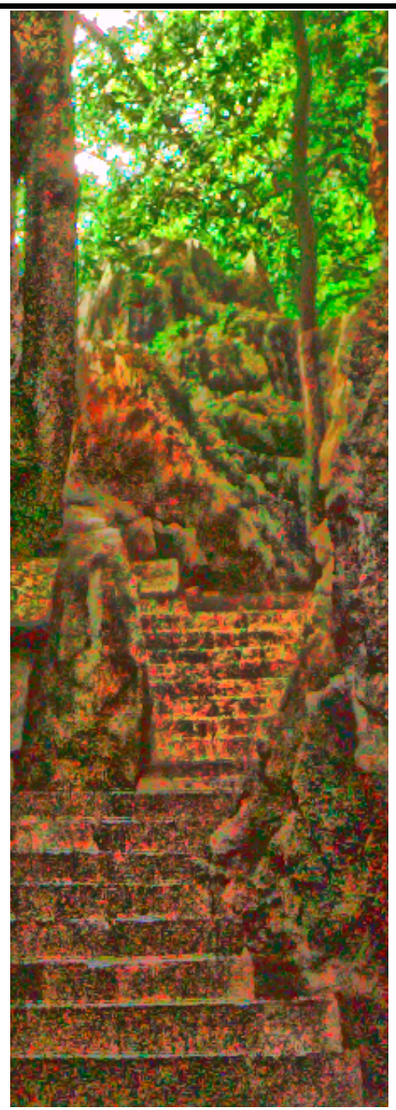
Naive weights





Weighted by k_i

Questions?





Naive weights

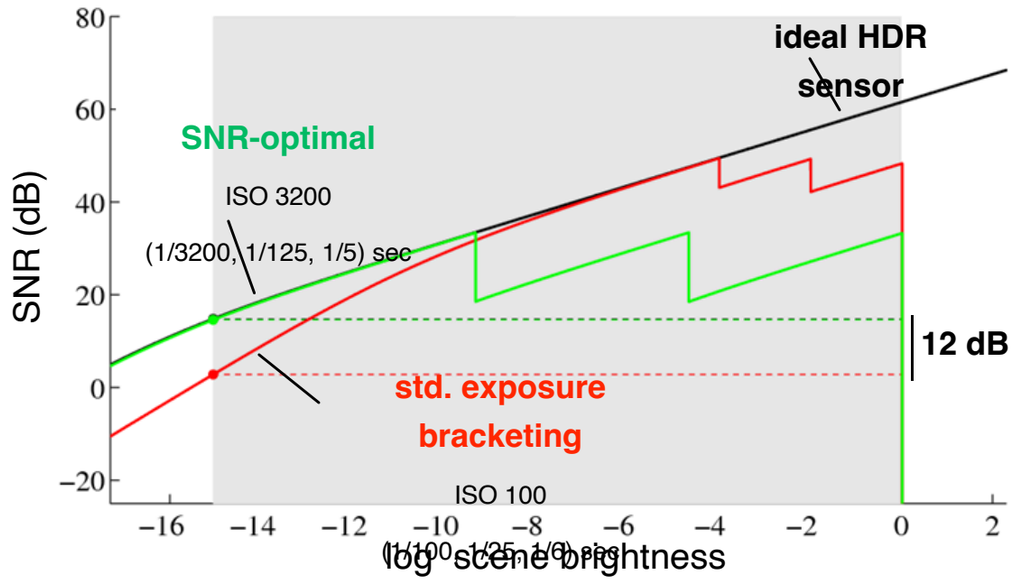




Weighted by k_i

References

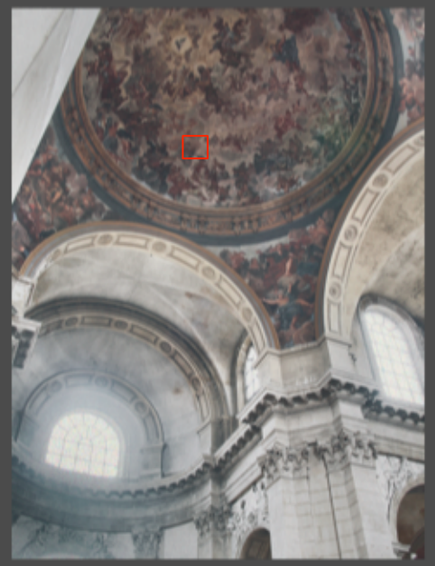
- **http://www.macs.hw.ac.uk/bmvc2006/papers/372.pdf**
- **http://people.csail.mit.edu/hasinoff/hdrnoise/**
  - full noise model
  - exploit ISO
  - Also optimizes the set of exposures

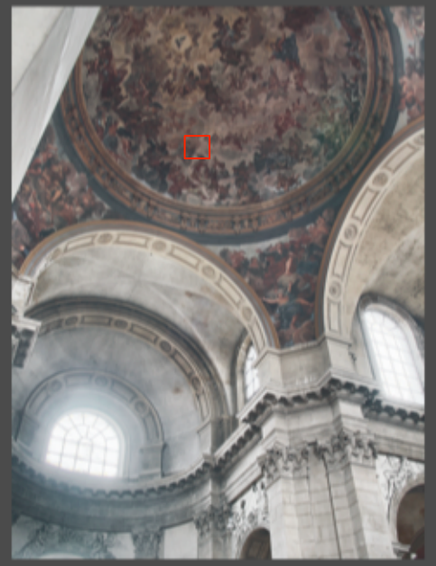


Results: “Nancy Church” (simulation)

tone-mapped merged HDR



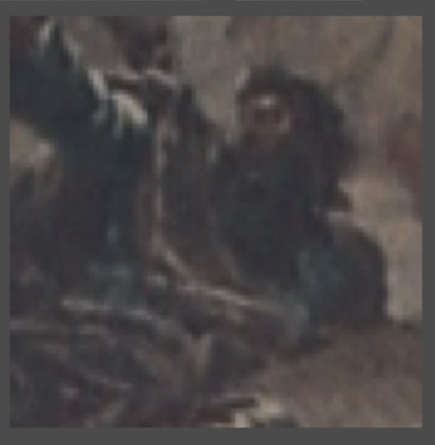




dark region







ISO 100: (1/100, 1/25, 1/6) sec

standard exposure bracketing

ISO 3200: (1/3200, 1/125, 1/5) sec

our SNR-optimal sequence

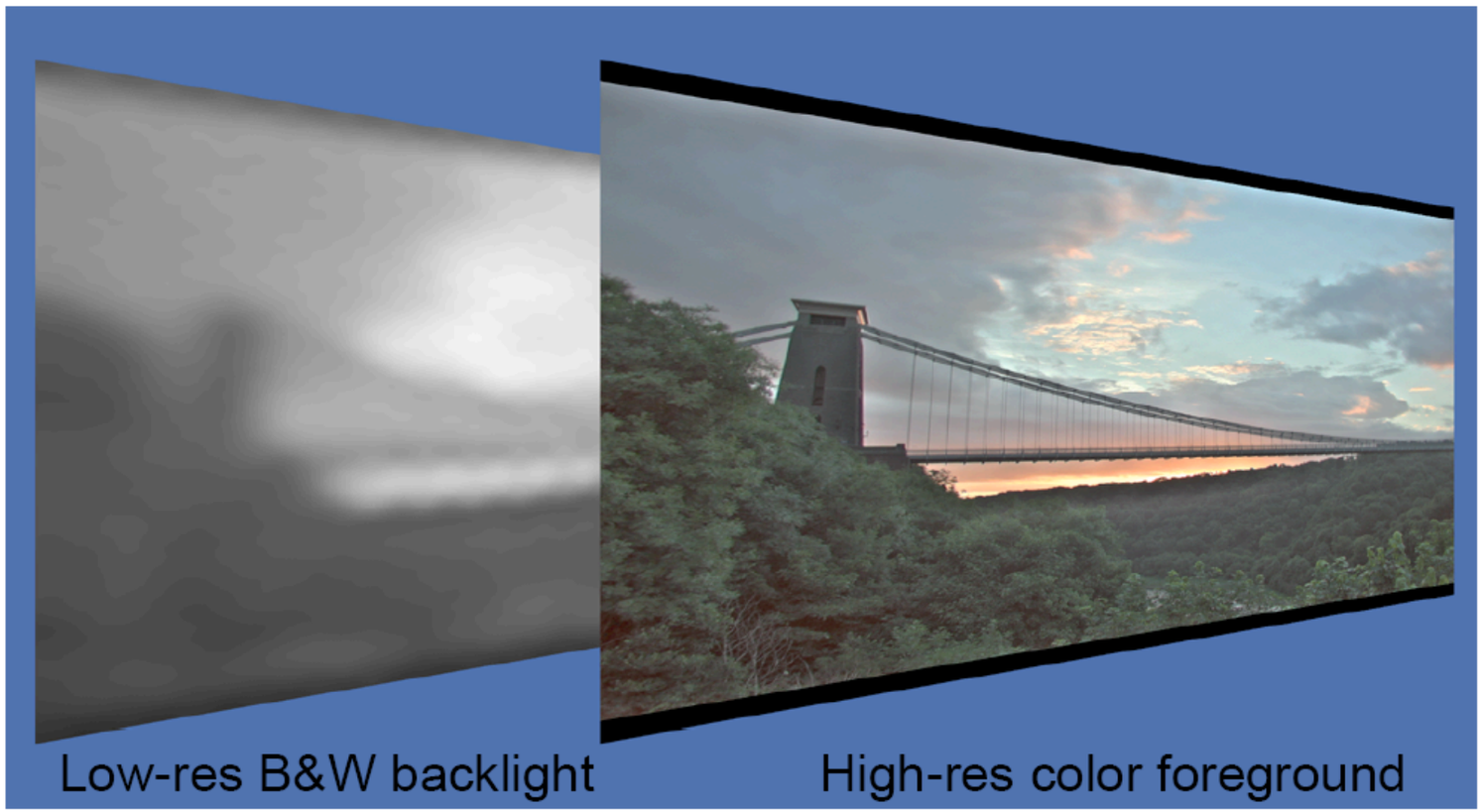
ground truth

Slide by Sam Hasinoff

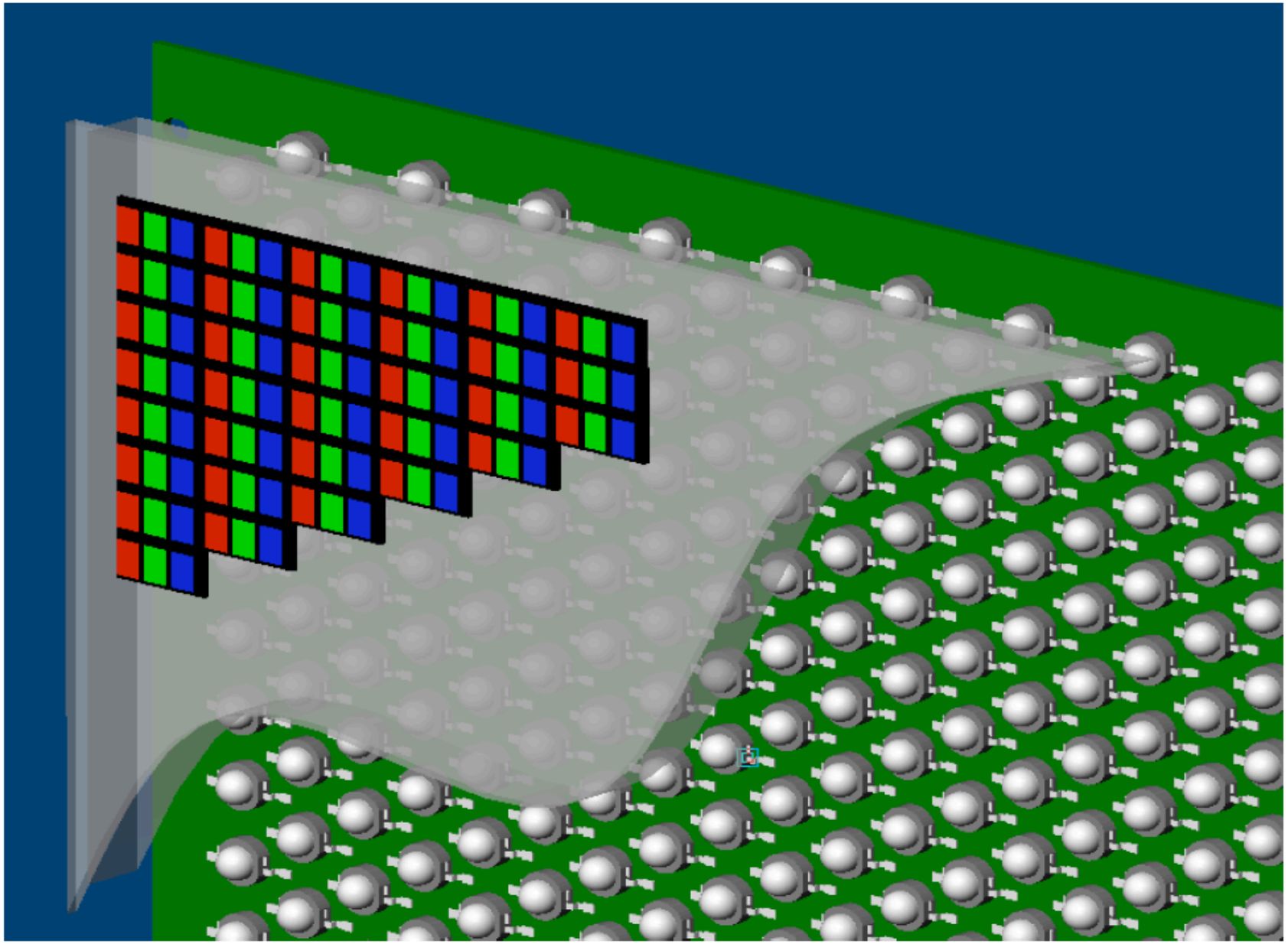
HDR display

Exception: Sunnybrook HDR display

- Use Bright Source + Two 8-bit Modulators
  - Transmission multiplies together
  - Over 10,000:1 dynamic range possible

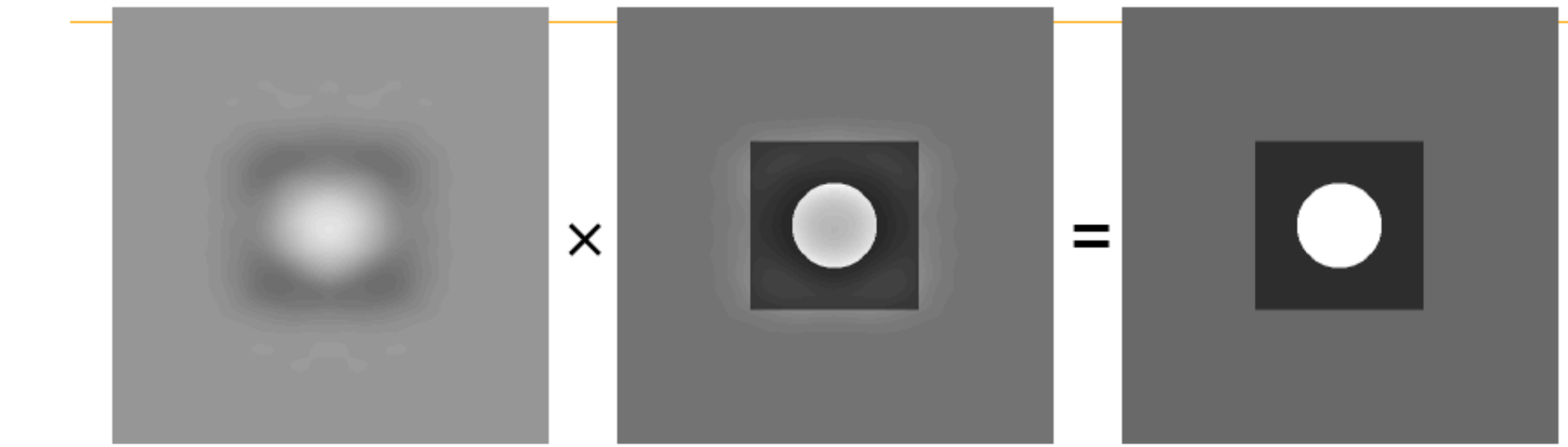


Slide from the 2005 Siggraph course on HDR

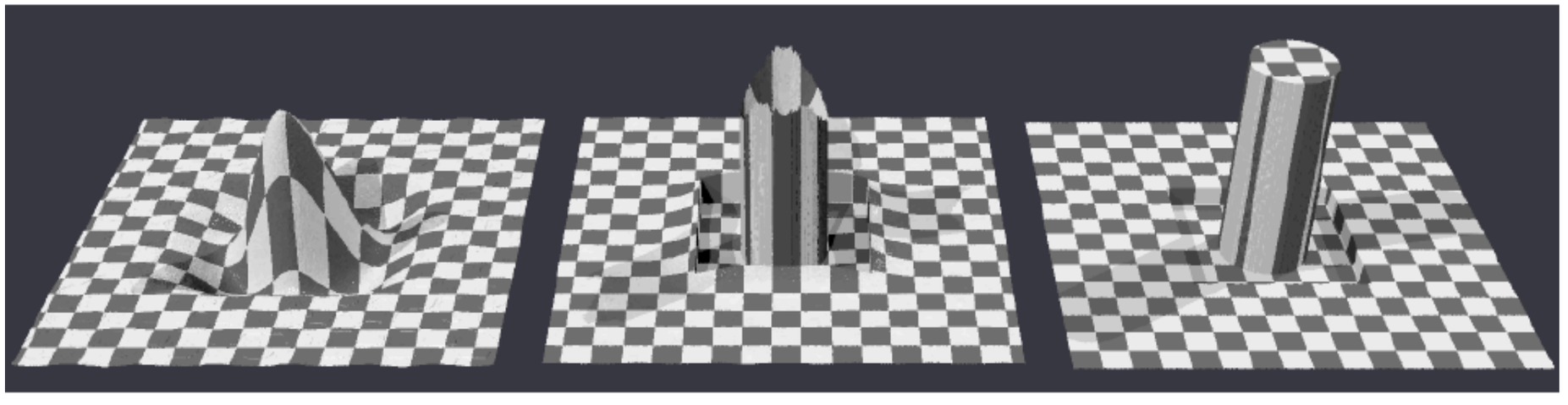


Slide from the 2005 Siggraph course on HDR

How It Works



LED Backlight LCD Screen Combined Result



Slide from the 2005 Siggraph course on HDR

Production 37” HDR Display



Slide from the 2005 Siggraph course on HDR

Questions?

Motion

Smarter HDR capture

Ward, Journal of Graphics Tools, 2003

http://www.anyhere.com/gward/papers/jgtpap2.pdf

Implemented in Photosphere http://www.anyhere.com/

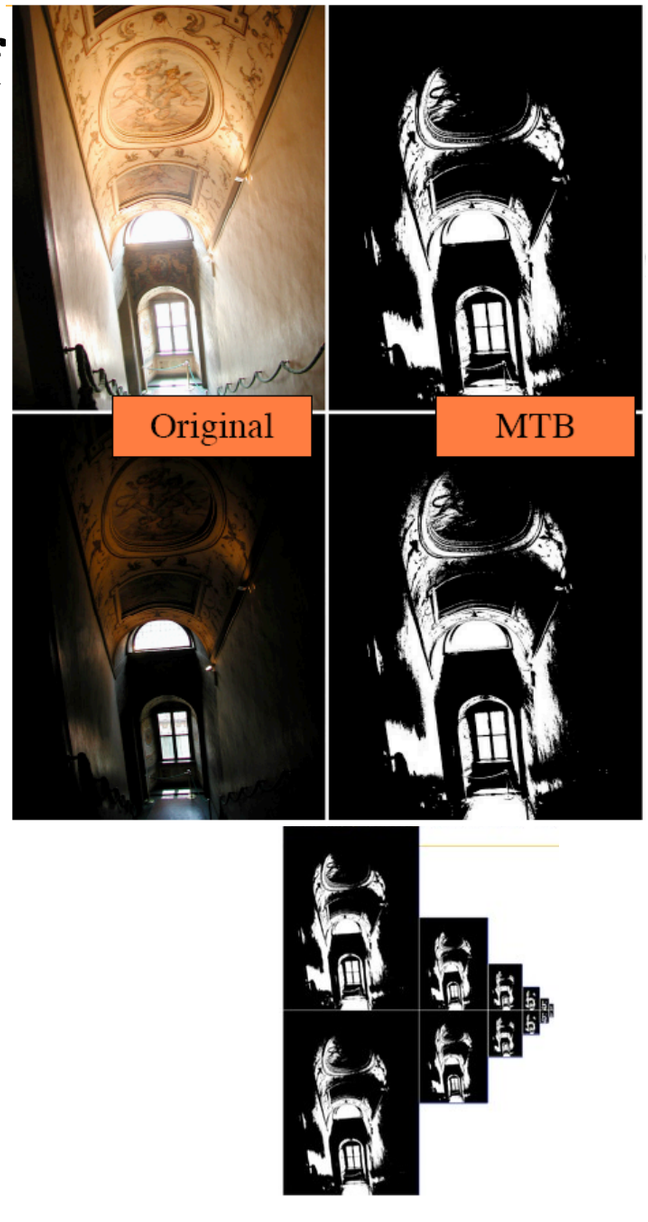
- **Image registration (no need for tripod)**
- **Lens flare removal**
- **Ghost removal**



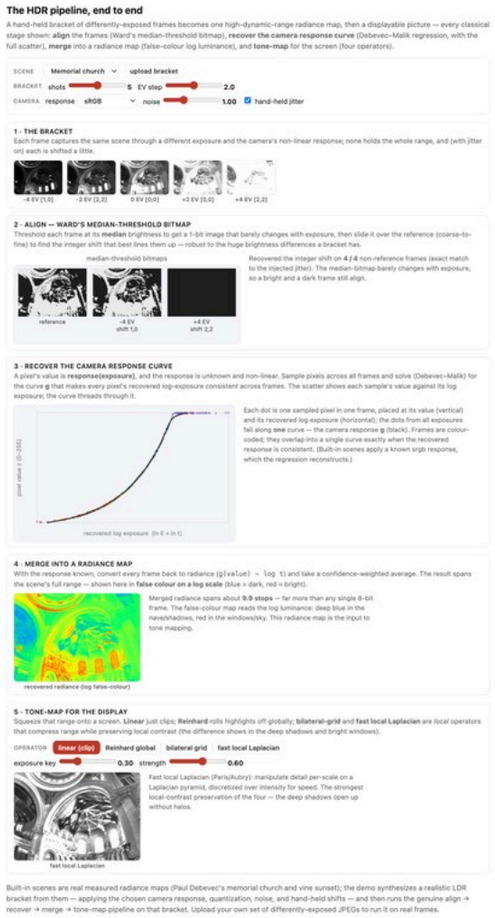
Images Greg Ward

Image registration

- **How to robustly compare images of different exposure?**
- **Use a black and white version of the image thresholded at the median**
  - Median-Threshold Bitmap (MTB)
- **Find the translation that minimizes difference**
- **Accelerate using pyramid**



Demo: Full HDR pipeline



Alignment Results

SIGGRAPH2005



5 unaligned exposures | Close-up detail | MTB alignment

Time: About .2 second/exposure for 3 MPixel image

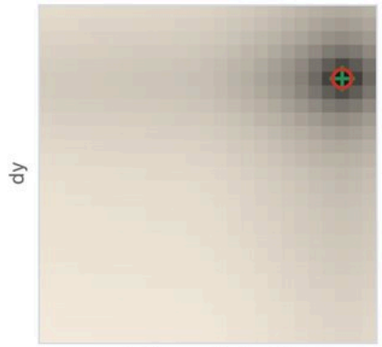
Slide from Siggraph 2005 course on HDR

Demo: Brute-force alignment of a hand-held burst

frame 6, as it came



SSD against frame 1, every shift

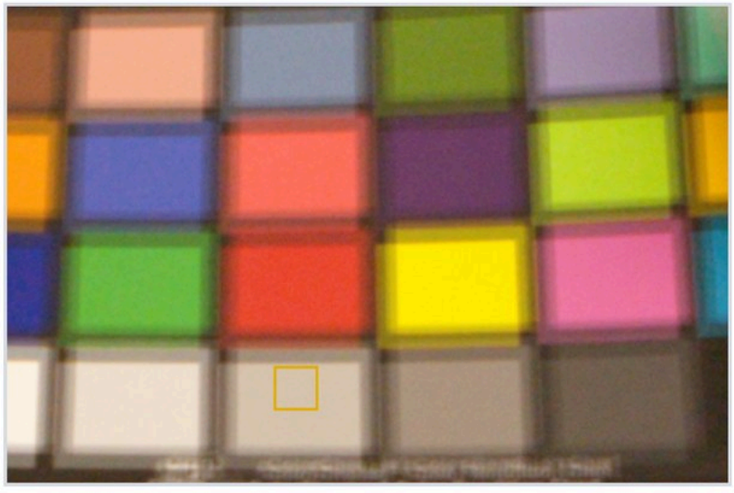


+ true shift
○ smallest SSD

frame	true shift	found
2	(0, 6)	(0, 6) ✓
3	(6, 4)	(6, 4) ✓
4	(4, 1)	(4, 1) ✓
5	(7, 1)	(7, 1) ✓
6	(10, -7)	(10, -7) ✓
7	(12, -8)	(12, -8) ✓
8	(12, -8)	(12, -8) ✓

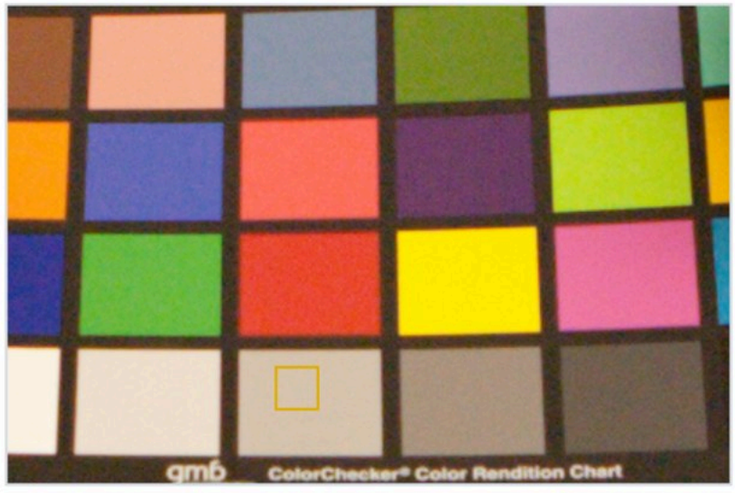
7 of 7 found

8 frames, averaged as they came



noise in the grey patch 1.48 · error against the truth 33.19

8 frames, aligned, then averaged



noise in the grey patch 0.95 · error against the truth 1.85

one frame: noise 2.81, so 8 aligned frames should reach 1.06 (1/√N, plus the truth's own noise). Codes out of 255; the truth is the mean of all 50 tripod frames.

Automatic “Ghost” Removal

SIGGRAPH2005



Before

After

Demo: **A handheld burst of a scene with something moving in it, merged three…**



2.3% of in-bounds samples rejected · gate 0.25

Gated minus ungated PSNR: 1.83 dB. Texture and noise can also trigger rejection.

Variance-based Detection

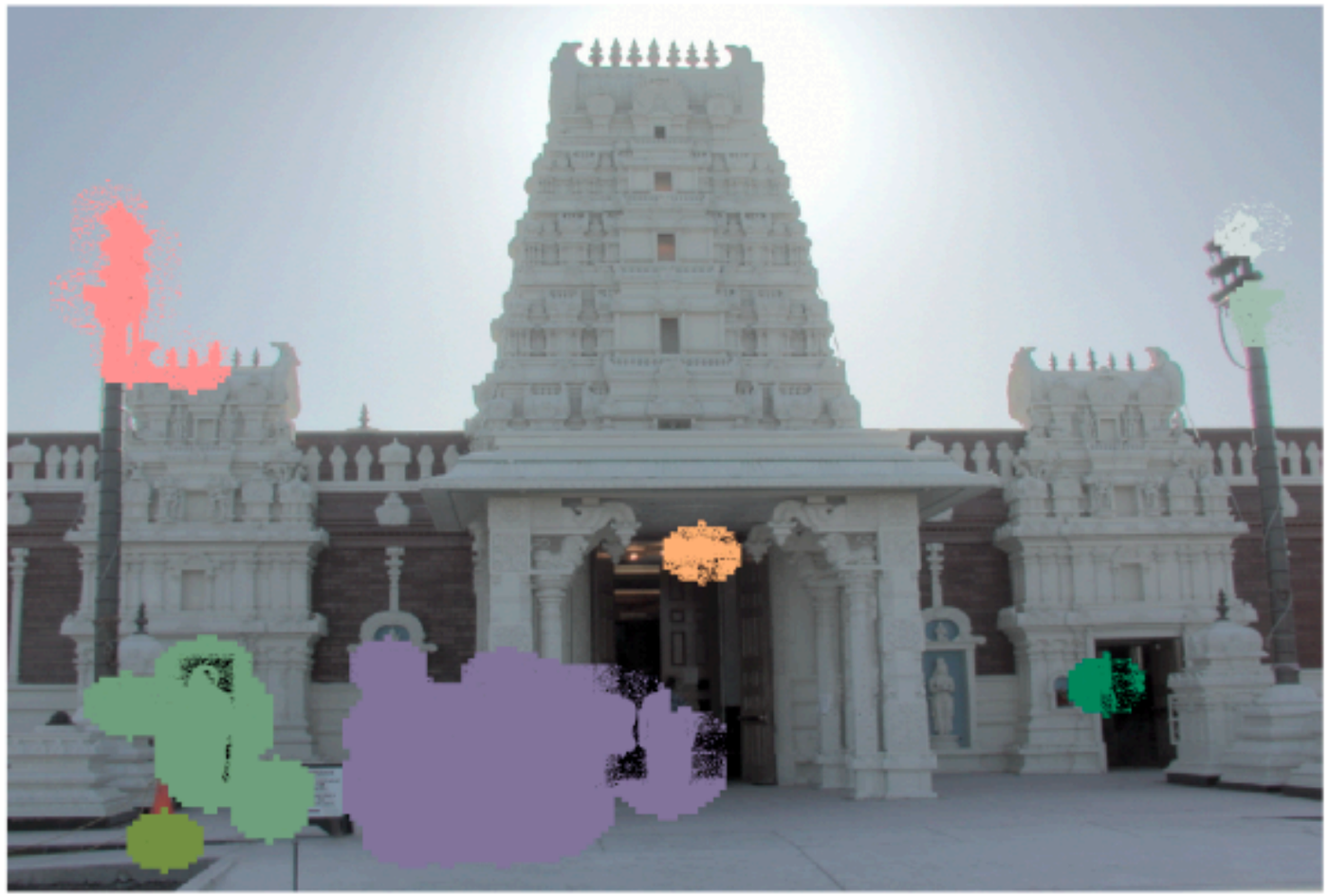
SIGGRAPH2005



Slide from Siggraph 2005 course on HDR

Region Masking

SIGGRAPH2005



Slide from Siggraph 2005 course on HDR

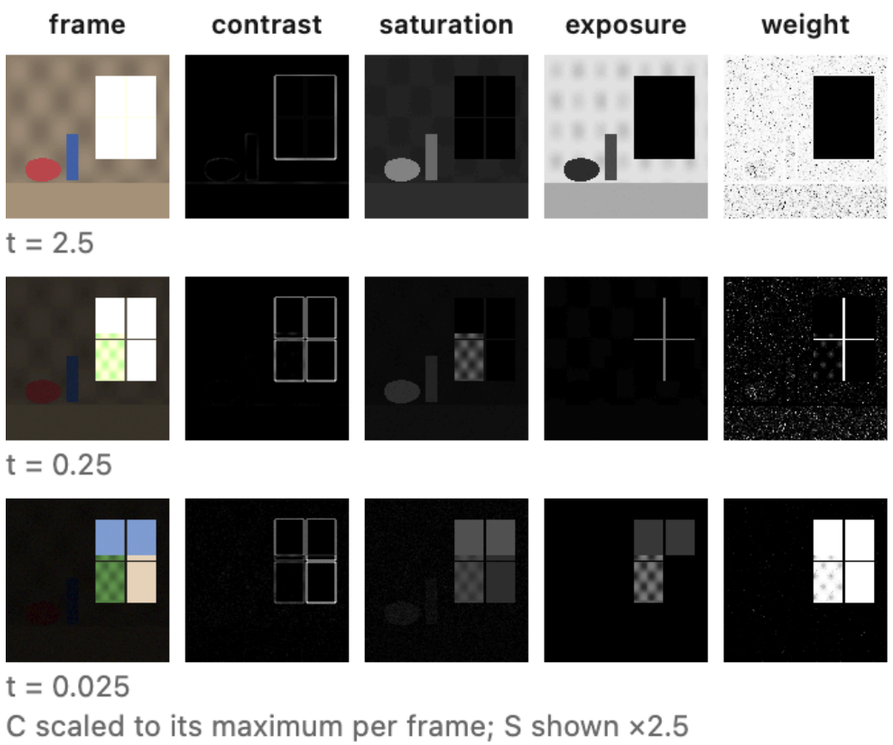
Best Exposure in Each Region

SIGGRAPH2005



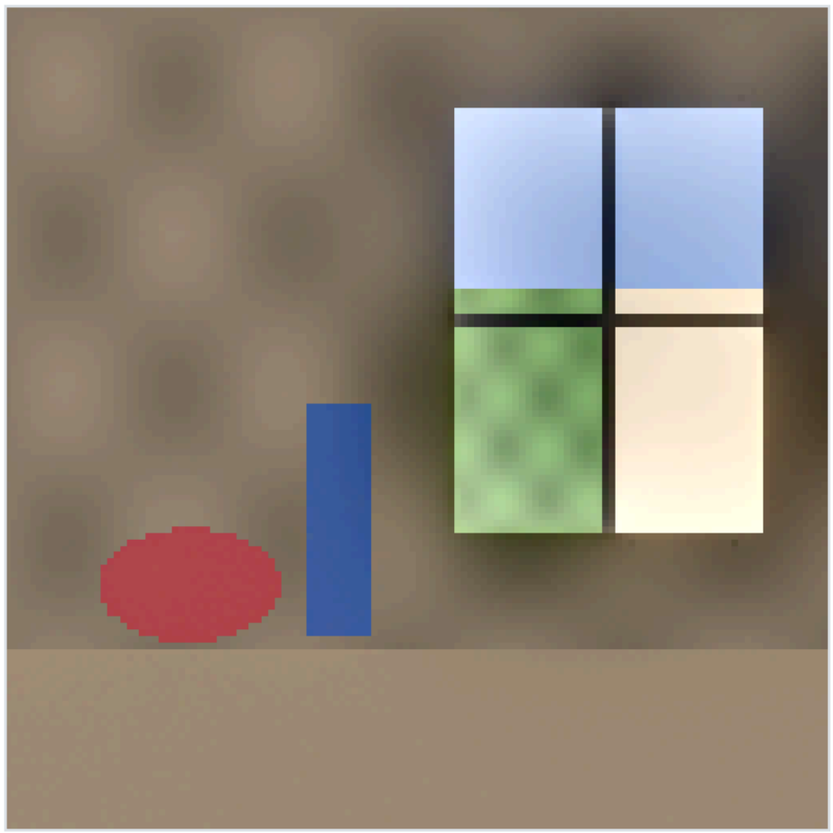
Slide from Siggraph 2005 course on HDR

Demo: One bracket, with each frame's three quality measures shown as images…



C scaled to its maximum per frame; S shown ×2.5

pyramid blend, 5 levels



seams 0.0039 · drift 0.089 · out of [0,1] 1.1%

seams: structure added where the scene is smooth
drift: distance from the frame that wins a region

Synthetic scene and clean sensor model; weights computed from the encoded frames alone.

Lens Flare Removal

SIGGRAPH2005



Before

After

Slide from Siggraph 2005 course on HDR

Extension: HDR video

- **Kang et al. Siggraph 2003**
  http://portal.acm.org/citation.cfm?id=882262.882270



Figure 1: High dynamic range video of a driving scene. *Top row: Input video with alternating short and long exposures. Bottom row: High dynamic range video (tonemapped).*

Extension: HDR video

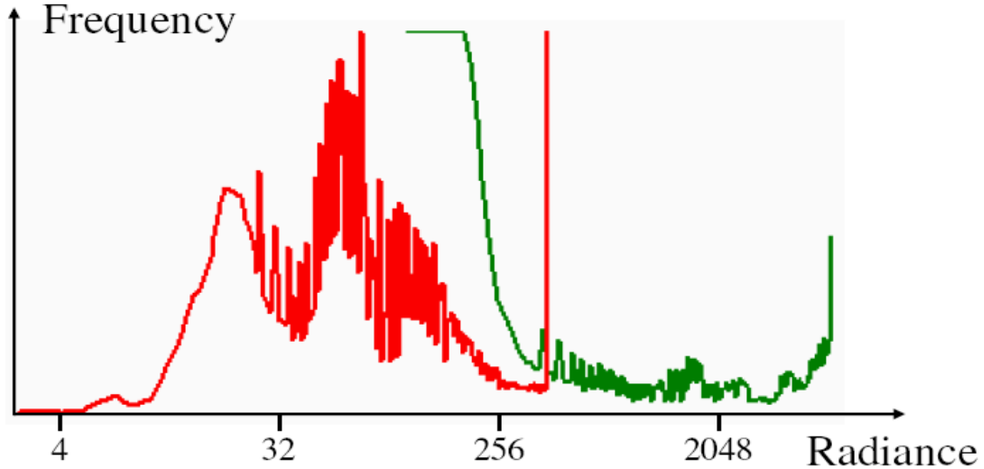




Figure 3: Two input exposures from the driving video. *The radiance histogram is shown on top. The red graph goes with the long exposure frame (bottom left), while the green graph goes with the short exposure frame (bottom right). Notice that the combination of these graphs spans a radiance range greater than a single exposure can capture.*

HDR code

- **HDRShop http://gl.ict.usc.edu/HDRShop/ (v1 is free)**
- **Columbia's camera calibration and HDR combination with source code Mitsunaga, Nayar , Grossberg http://www1.cs.columbia.edu/CAVE/projects/rad_cal/rad_cal.php**
- **Greg Ward Phososphere HDR browser and image combination with regsitration (Macintosh, command-line version under Linux) with source code http://www.anyhere.com/**
- **Photoshop CS2**
- **Idruna http://www.idruna.com/photogenicshdr.html**
- **MPI PFScalibration (includes source code) http://www.mpii.mpg.de/resources/hdr/calibration/pfs.html**
- **EXR tools http://scanline.ca/exrtools/**
- **HDR Image Editor http://www.acm.uiuc.edu/siggraph/HDRIE/**
- **CinePaint http://www.cinepaint.org/**
- **Photomatix http://www.hdrsoft.com/**
- **EasyHDR http://www.astro.leszno.net/easyHDR.php**
- **Artizen HDR http://www.supportingcomputers.net/Applications/Artizen/Artizen.htm**
- ***Automated High Dynamic Range Imaging Software & Images*** **http://www2.cs.uh.edu/~somalley/hdri_images.html**
- **Optipix http://www.imaging-resource.com/SOFT/OPT/OPT.HTM**

HDR images

- http://www.debevec.org/Research/HDR/
- http://www.mpi-sb.mpg.de/resources/hdr/gallery.html
- http://people.csail.mit.edu/fredo/PUBLI/Siggraph2002/
- http://www.openexr.com/samples.html
- http://www.flickr.com/groups/hdr/
- http://www2.cs.uh.edu/~somalley/hdri_images.html#hdr_others
- http://www.anyhere.com/gward/hdrenc/pages/originals.html
- http://www.cis.rit.edu/mcsl/icam/hdr/rit_hdr/
- http://www.cs.utah.edu/%7Ereinhard/cdrom/hdr.html
- http://www.sachform.de/download_EN.html
- http://lcavwww.epfl.ch/%7Elmeylan/HdrImages/February06/February06.html
- http://lcavwww.epfl.ch/%7Elmeylan/HdrImages/April04/april04.html
- http://books.elsevier.com/companions/0125852630/hdri/html/images.html

HDR photography

Speaker's note:

- http://luminous-landscape.com/essays/hdr-plea.shtml
- http://en.wikipedia.org/wiki/High_dynamic_range_imaging
- http://www.cambridgeincolour.com/tutorials/high-dynamic-range.htm
- http://www.luminous-landscape.com/tutorials/hdr.shtml