HDR and the Bilateral Grid

Fredo Durand
MIT EECS 6.8370/6.8371

Today

- Multiple-exposure High-Dynamic-Range imaging



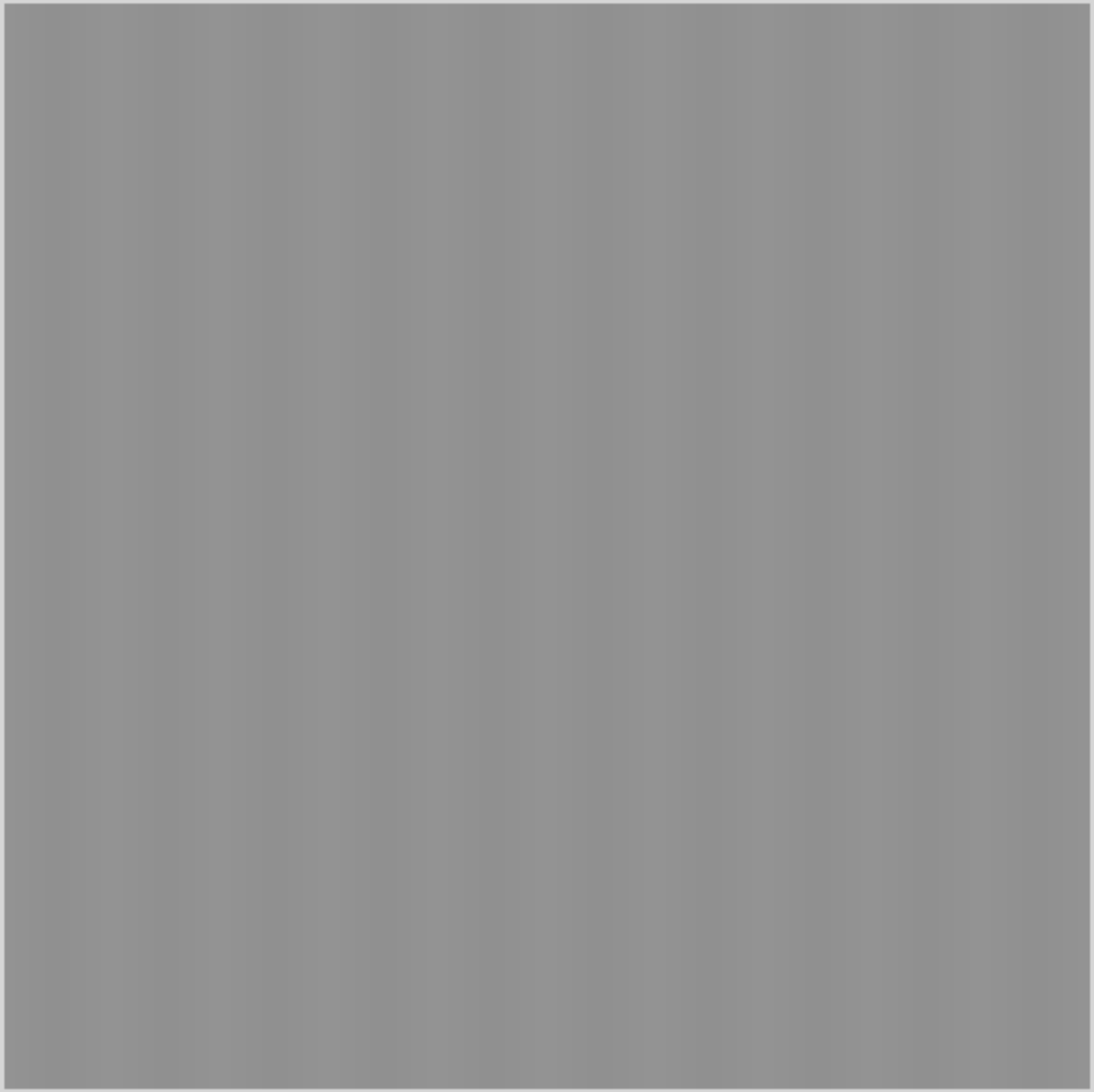
- **Tone mapping using the bilateral filter**

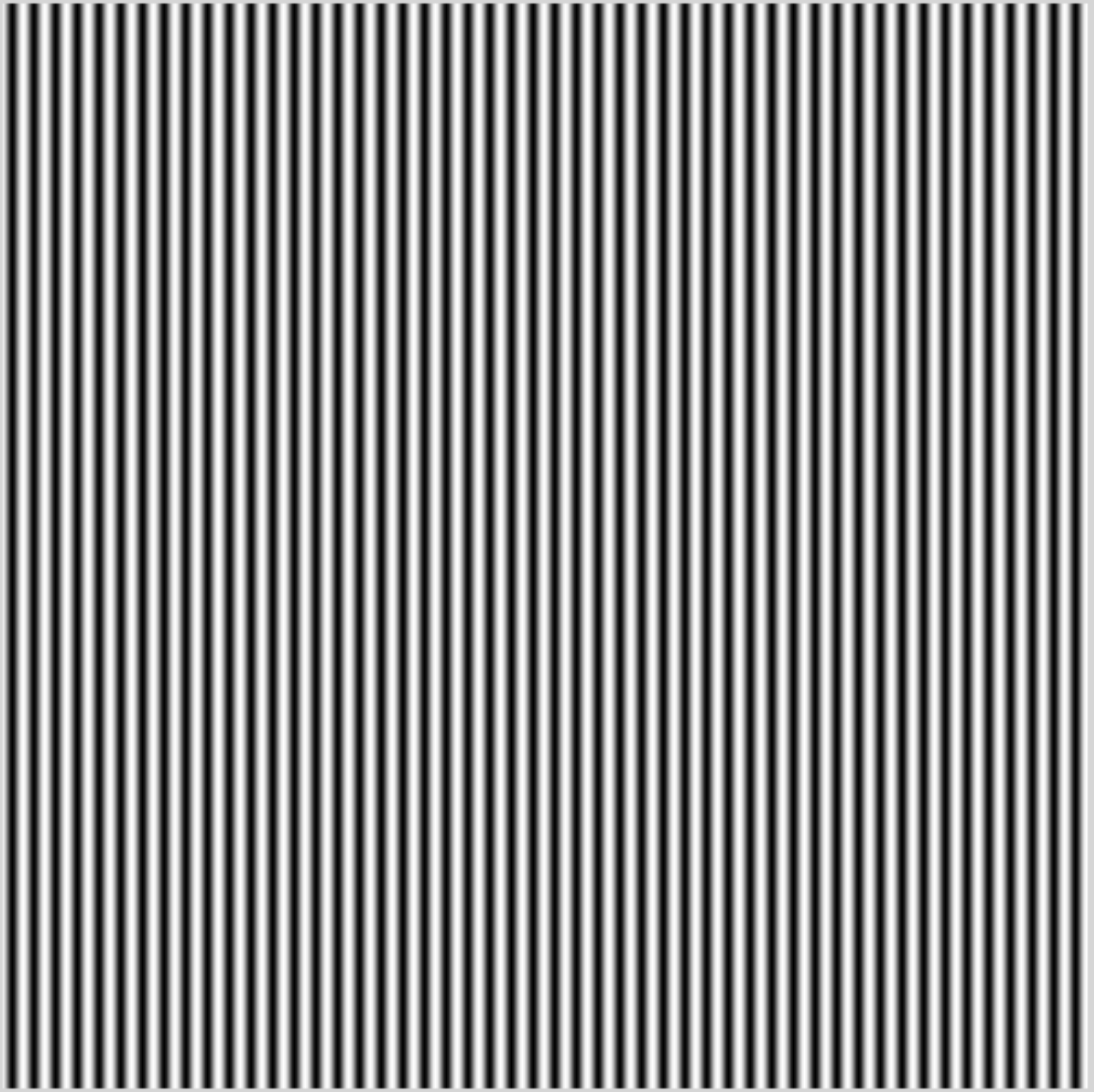


Spatial color vision

Contrast Sensitivity

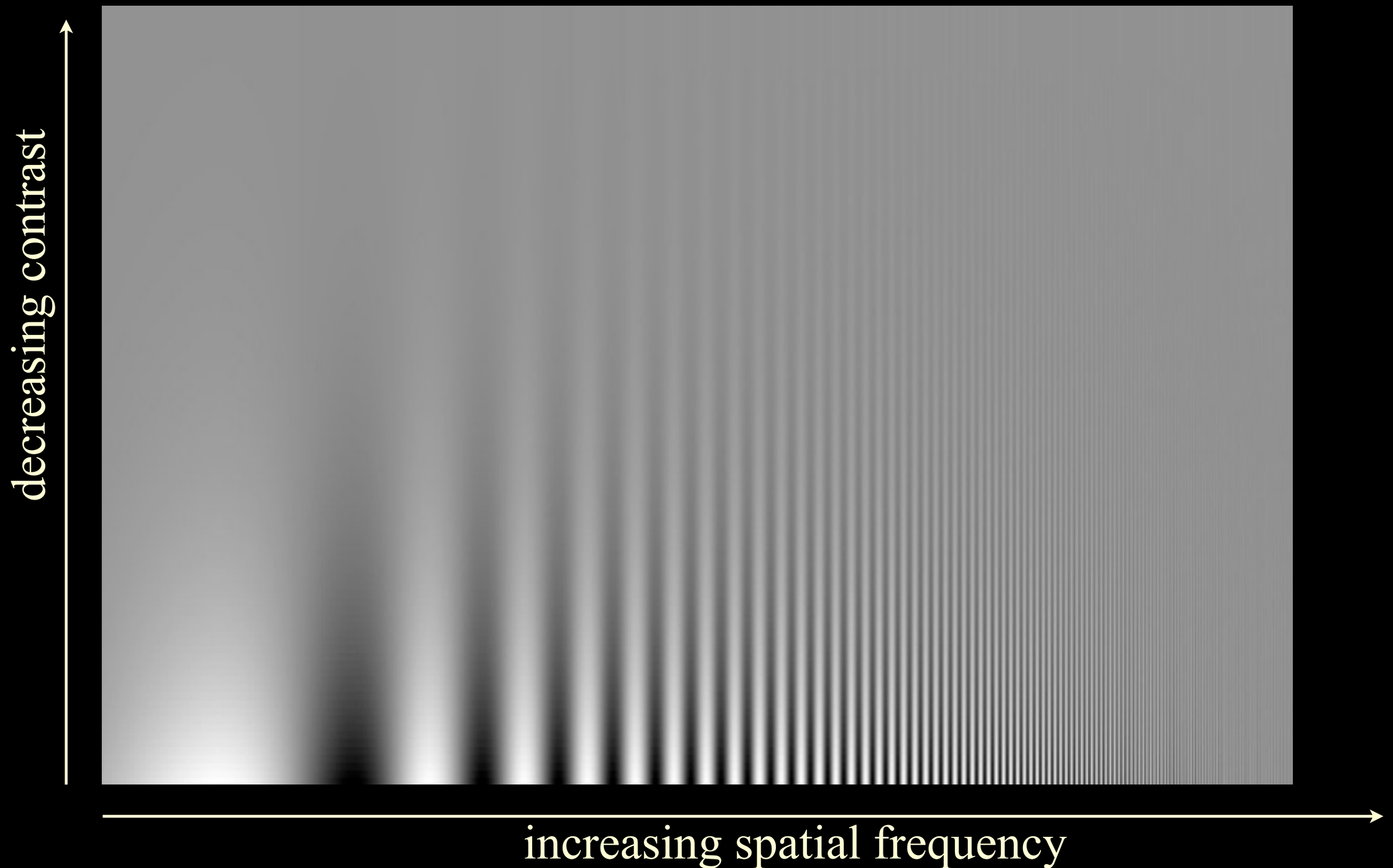
- Sine Wave grating
- What contrast is necessary to make the grating visible?
- How does it change with spatial frequency





Contrast Sensitivity Function

Campbell-Robson Chart



Contrast Sensitivity Function (CSF)

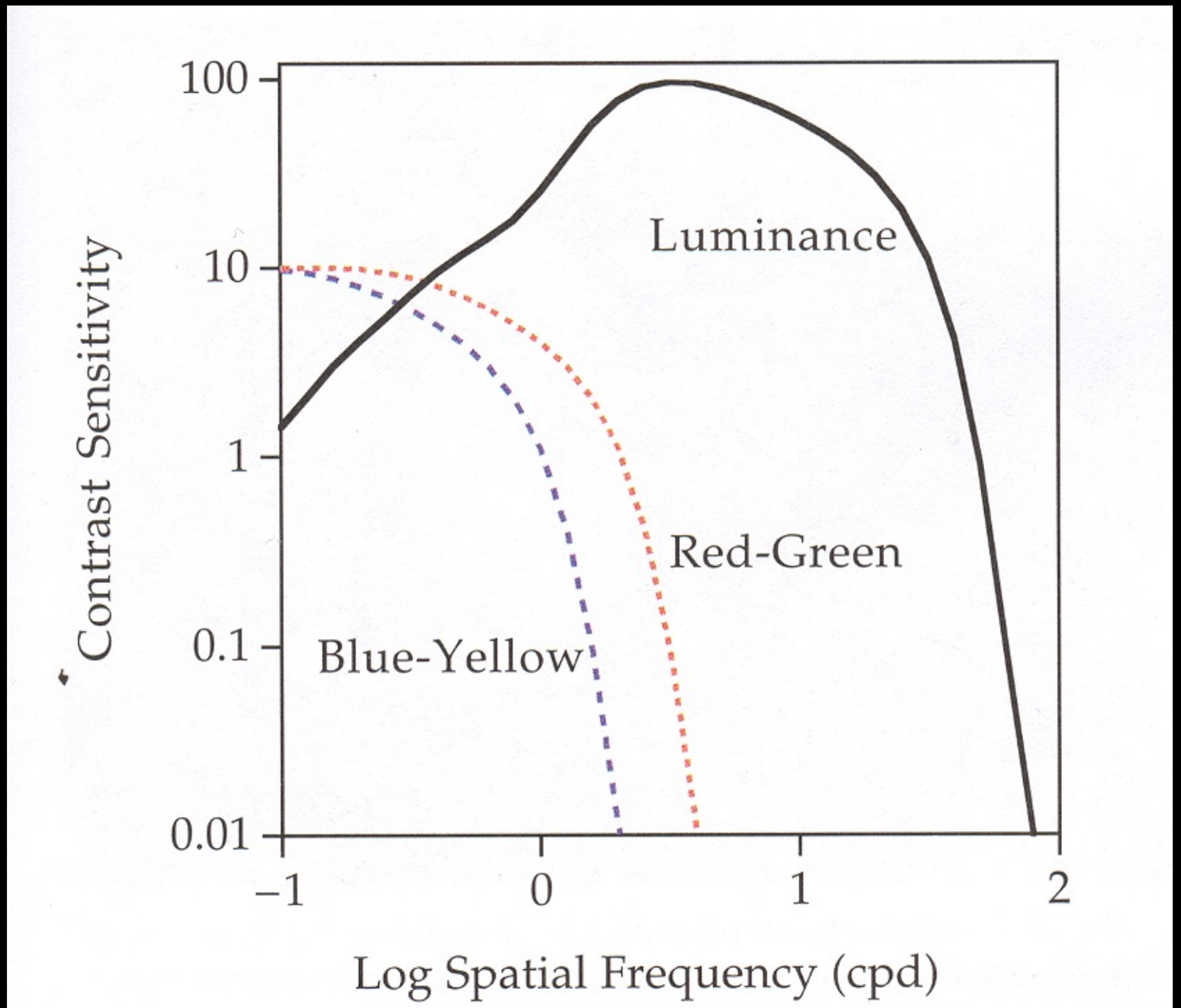


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

Photoshop demo



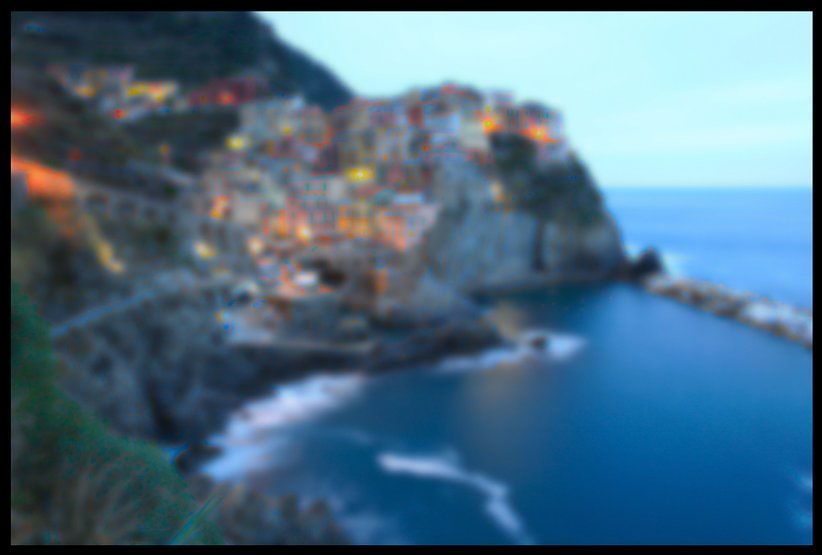
- Image > Mode > Lab color
- Go to channel panel, select Lightness
- Filter > Blur > Gaussian Blur , e.g. 4 pixel radius
  - –very noticeable



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



Original

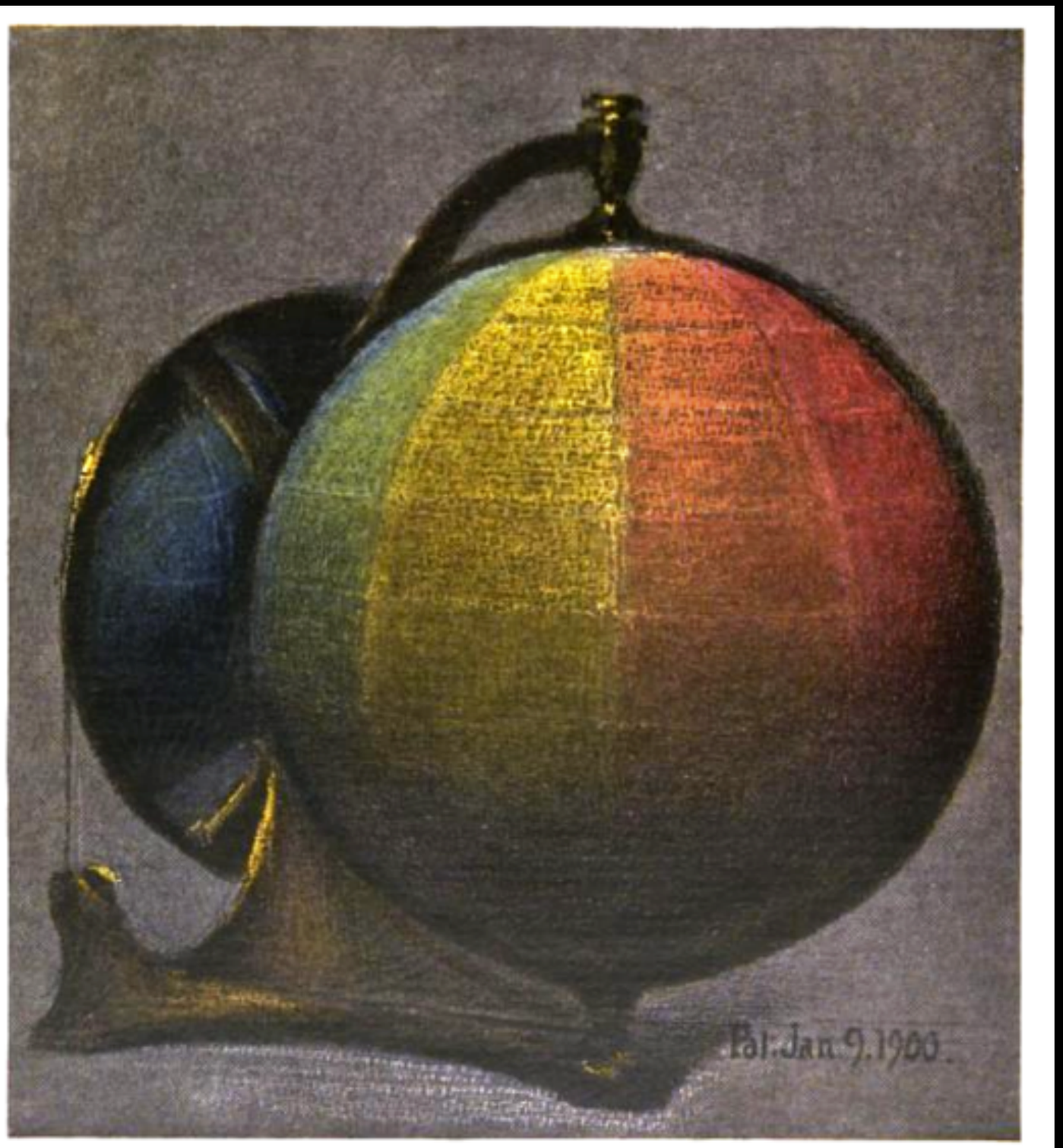


Blur Lightness



Blur a & b

Questions?

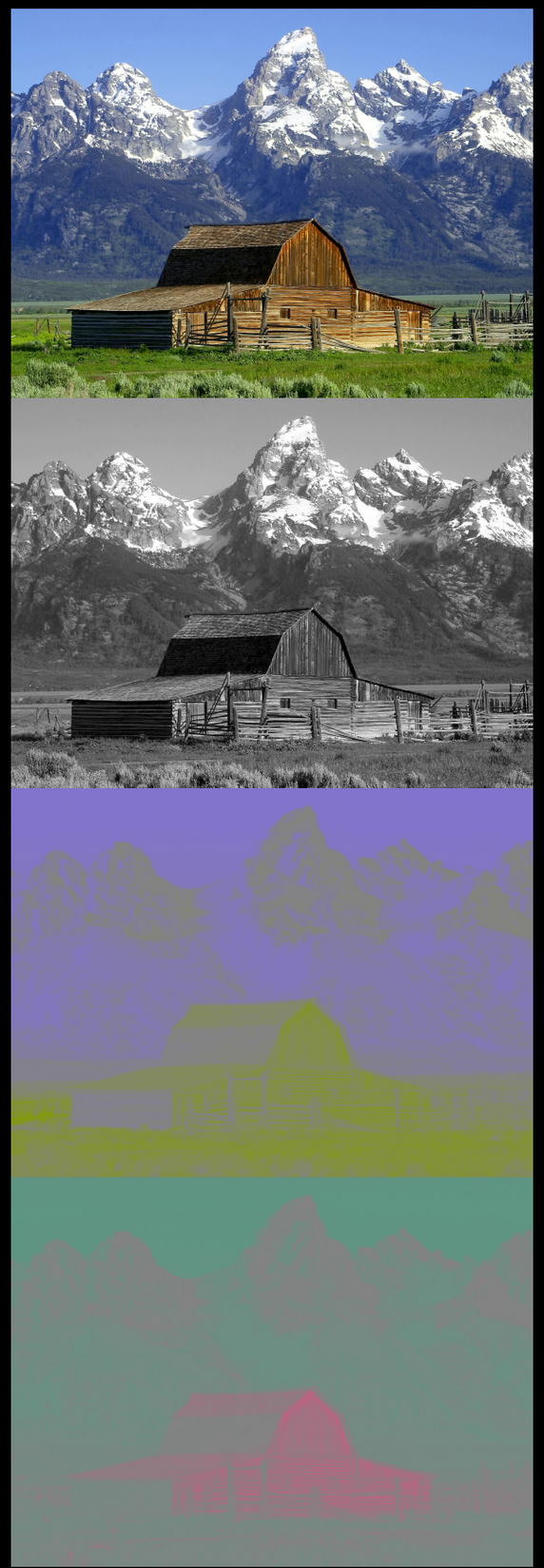


A BALANCED COLOR SPHERE

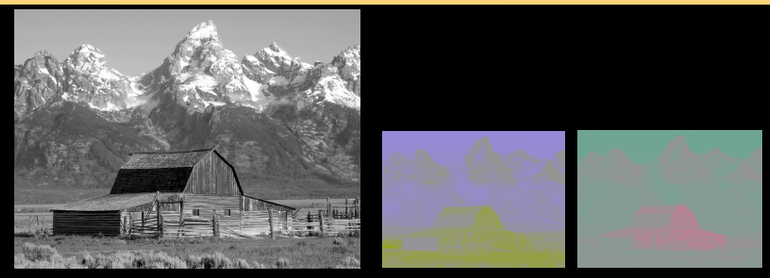
JPEG

Opponents and image compression

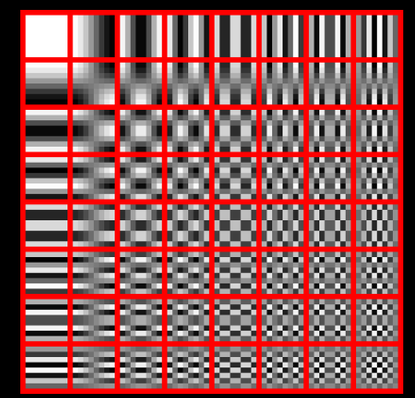
- JPG, MPG, television
- Color opponents instead of RGB
  - YCrCB, similar to YUV
- Compress color more than luminance
  - downsample by factor of two for jpeg
  - less bandwidth for TV
- Exploit contrast sensitivity function
  - Compress high frequencies more



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)

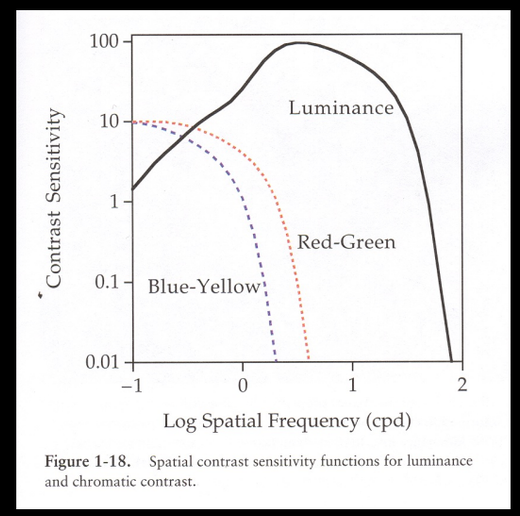
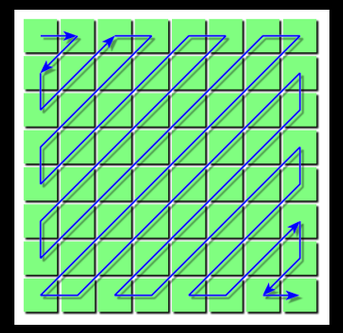


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

- Other usual coding tricks
  - –entropy coding, smart order of blocks



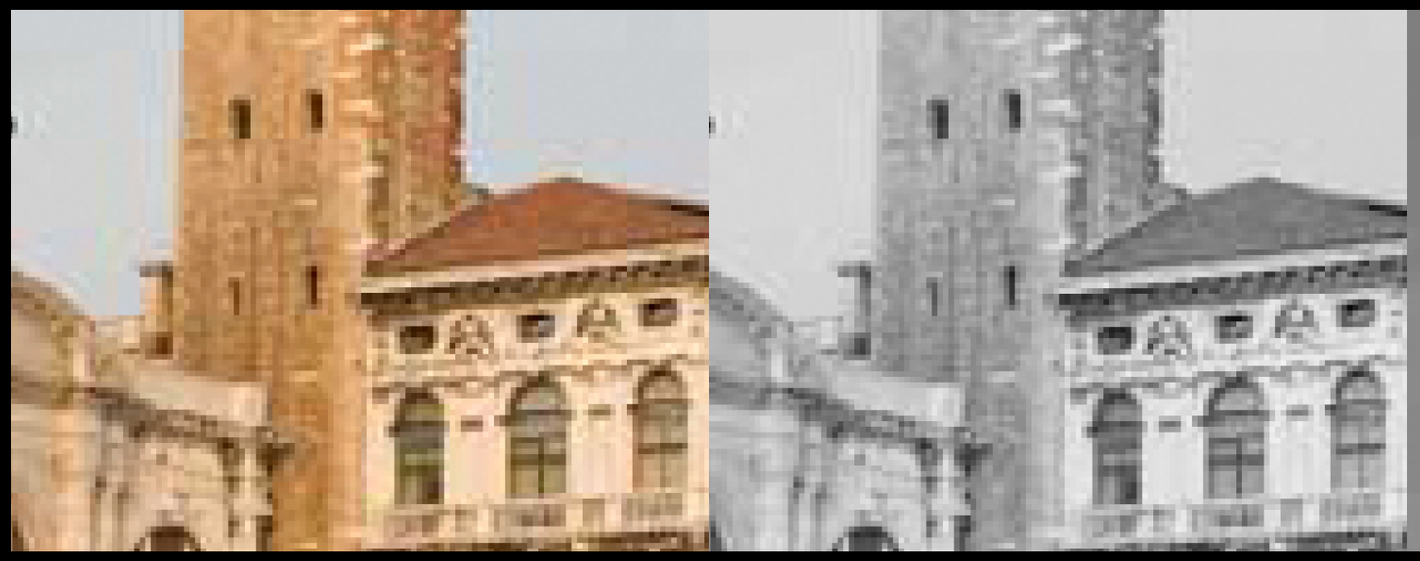
Example

- 800 x 533 image

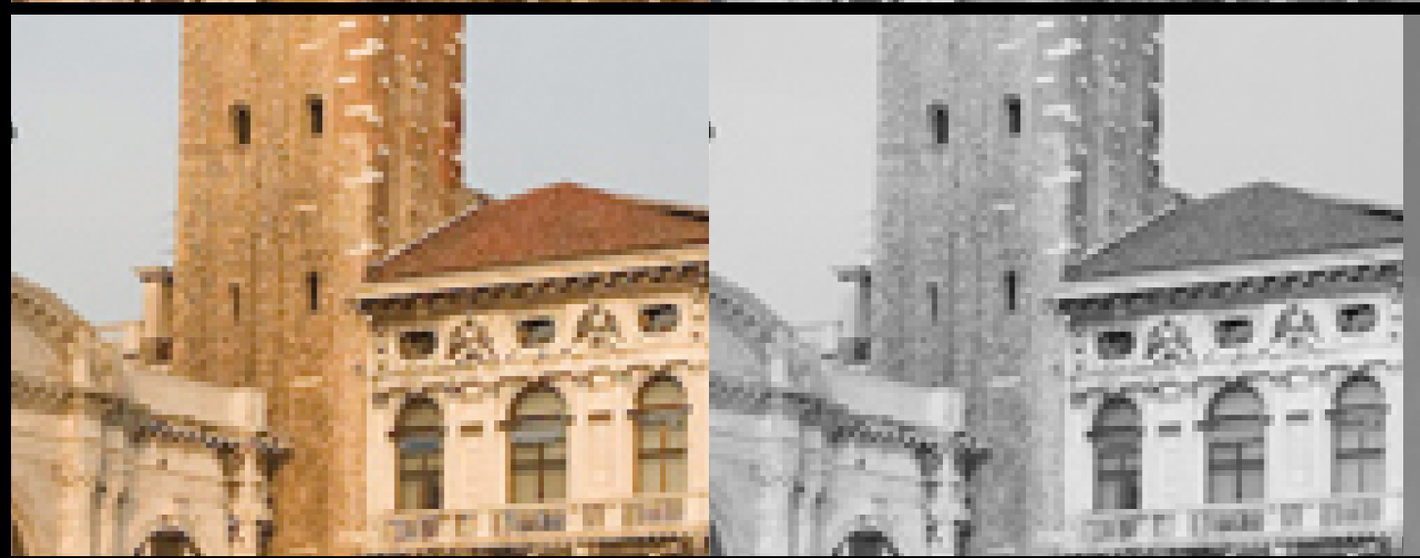


RGB Lightness a

Low quality JPEG (0 in Photoshop, 172 KB)



High quality JPEG (12 in Photoshop, 460 KB)



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?**

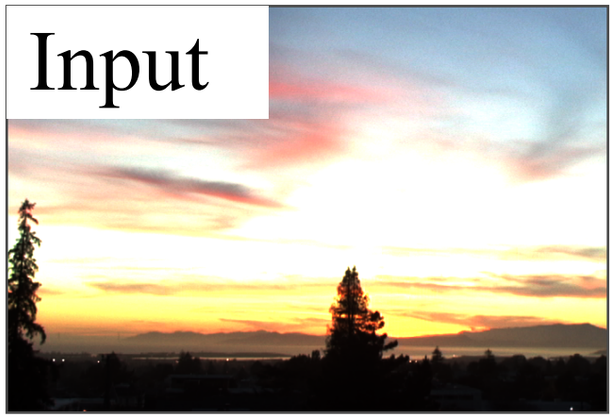


Naïve: Gamma compression

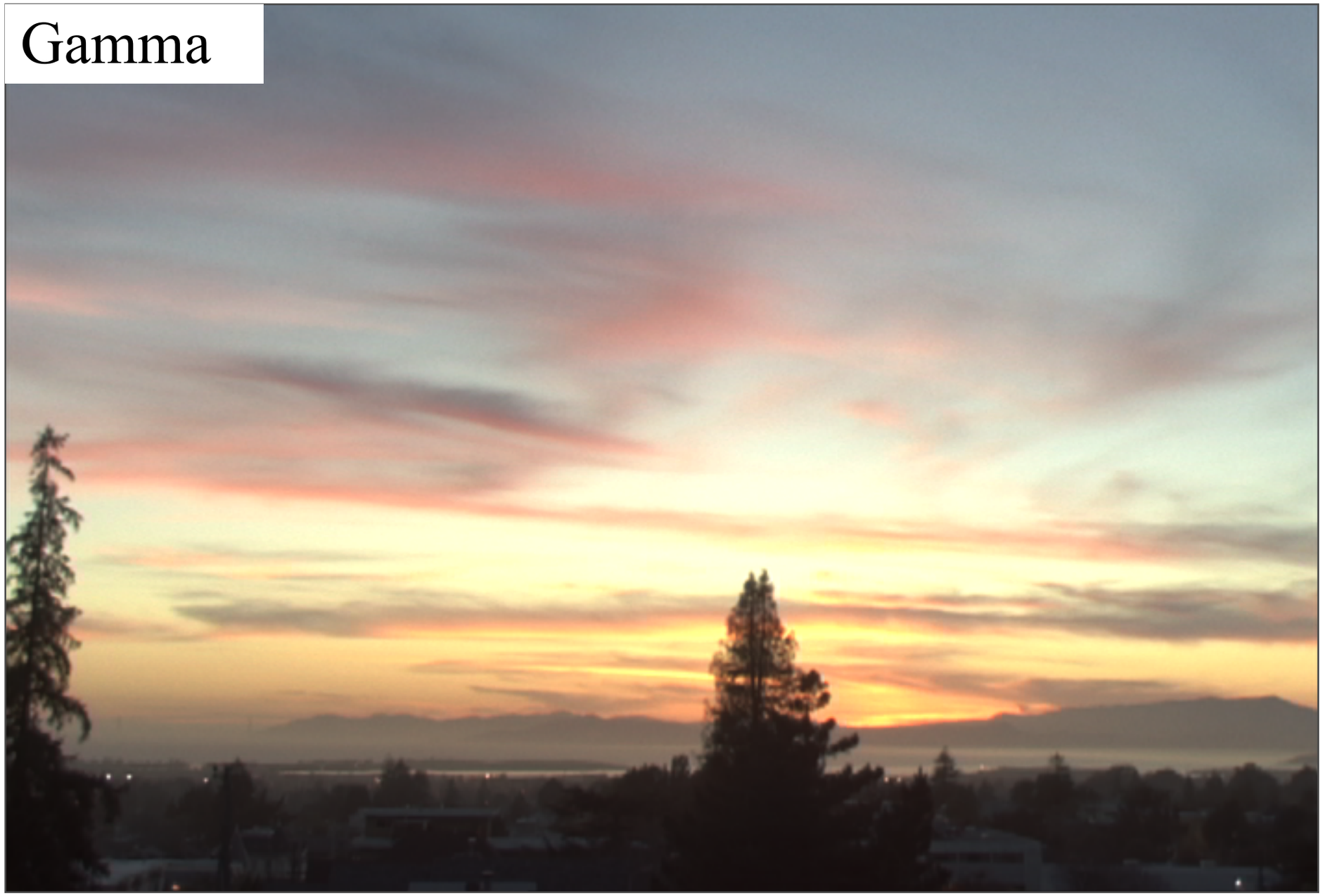
- **X –> X^{γ} (where γ=0.5 in our case)**
- **But… colors are washed-out. Why?**

applied independently on R, G & B

Input



Gamma



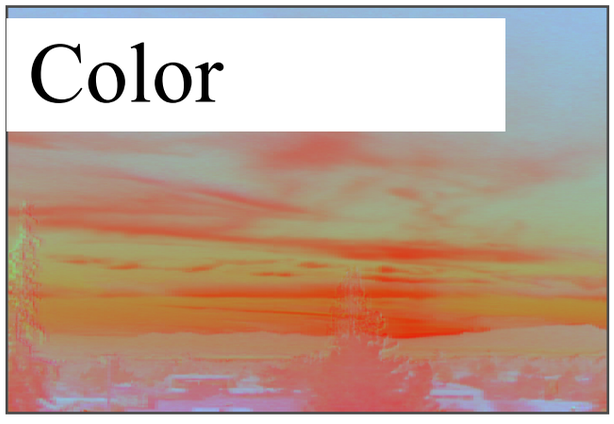
Gamma compression on intensity

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

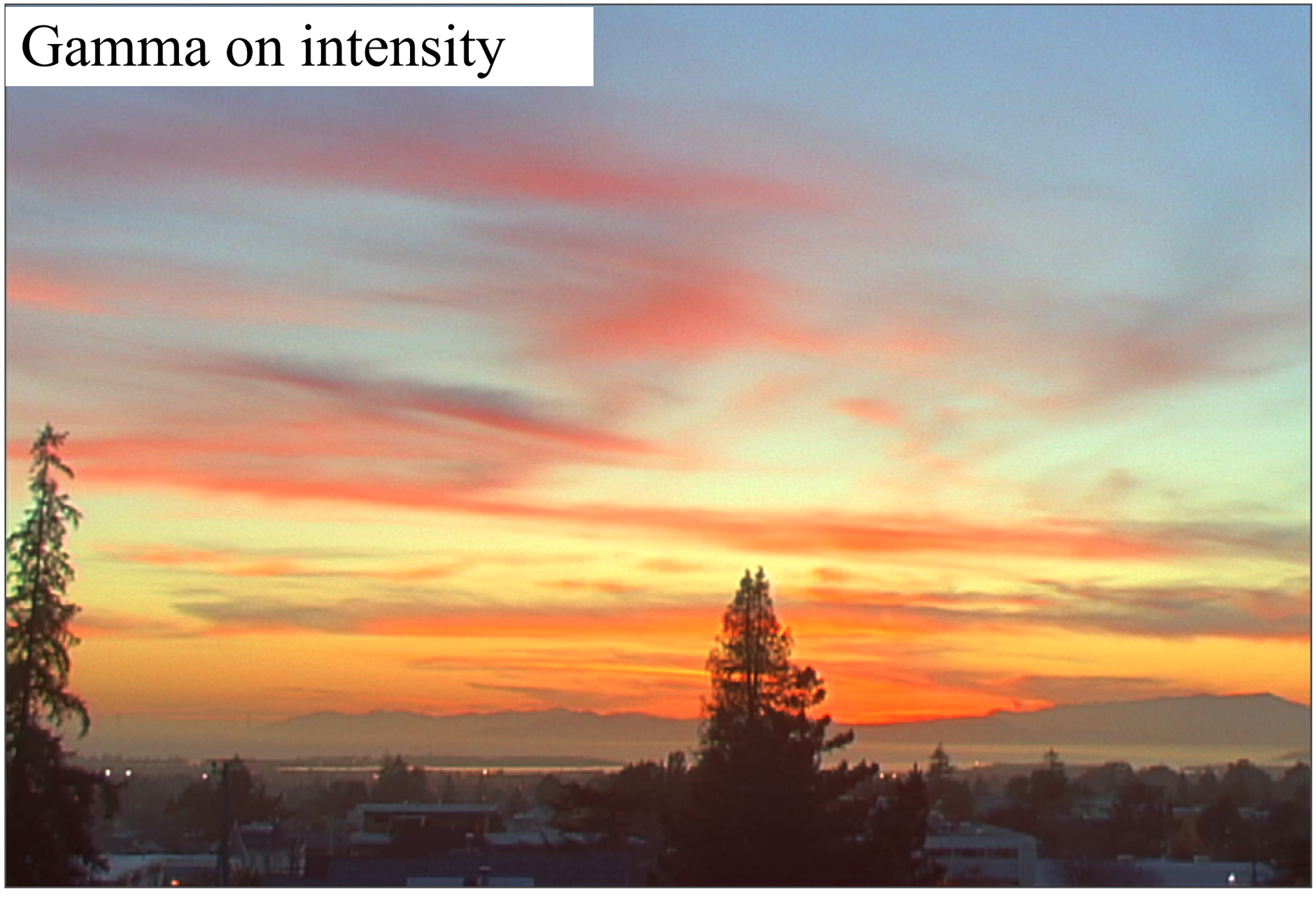
Intensity



Color



Gamma on intensity



Contrast Sensitivity Function (CSF)

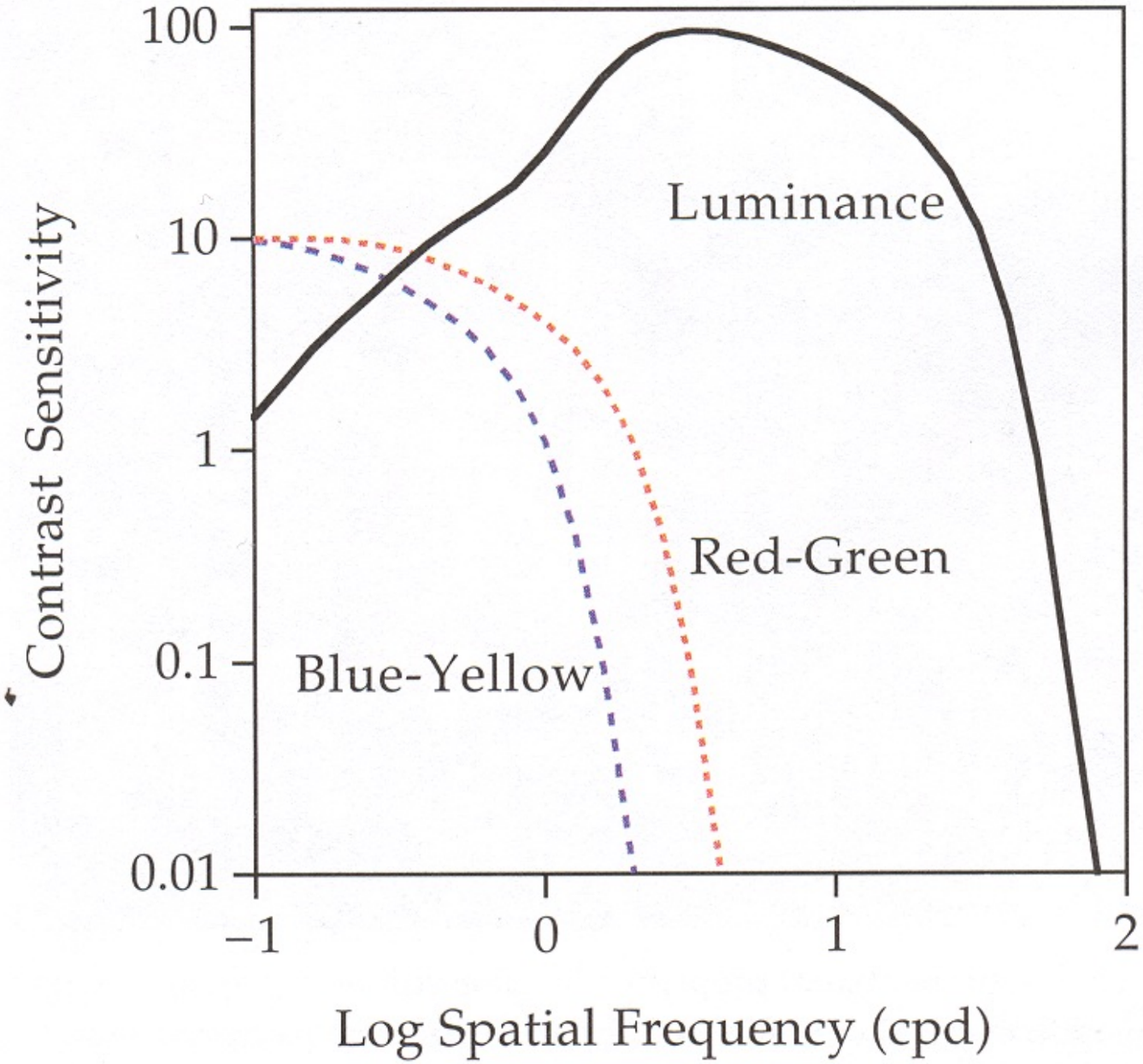


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

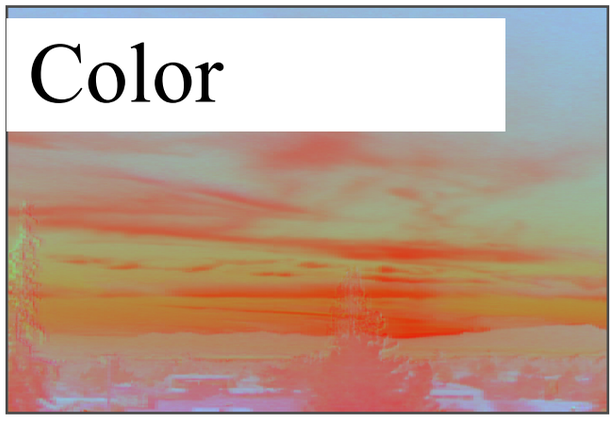
Gamma compression on intensity

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

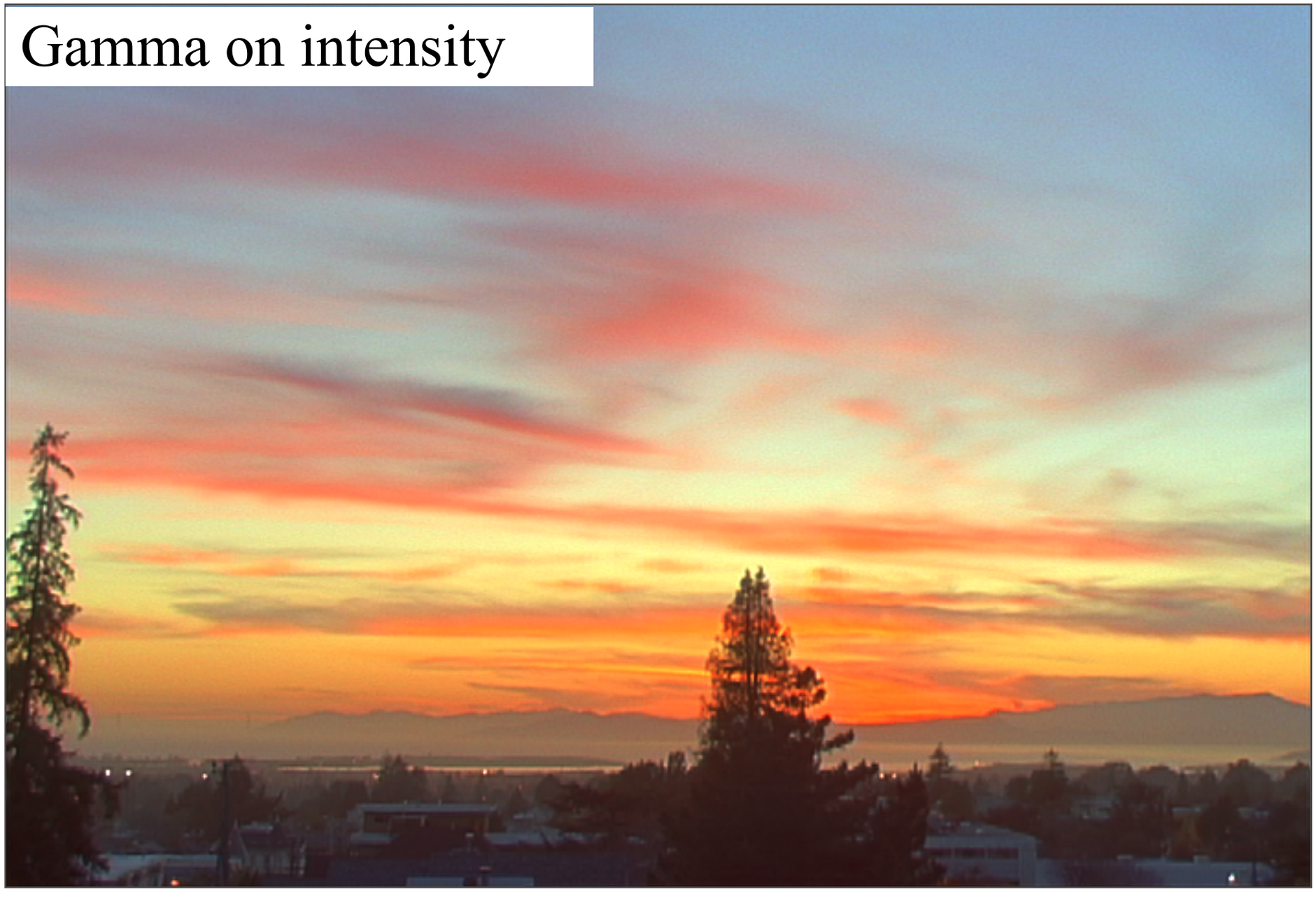
Intensity



Color



Gamma on intensity

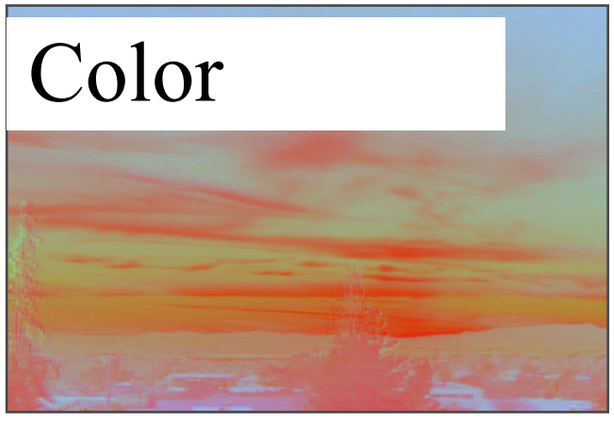


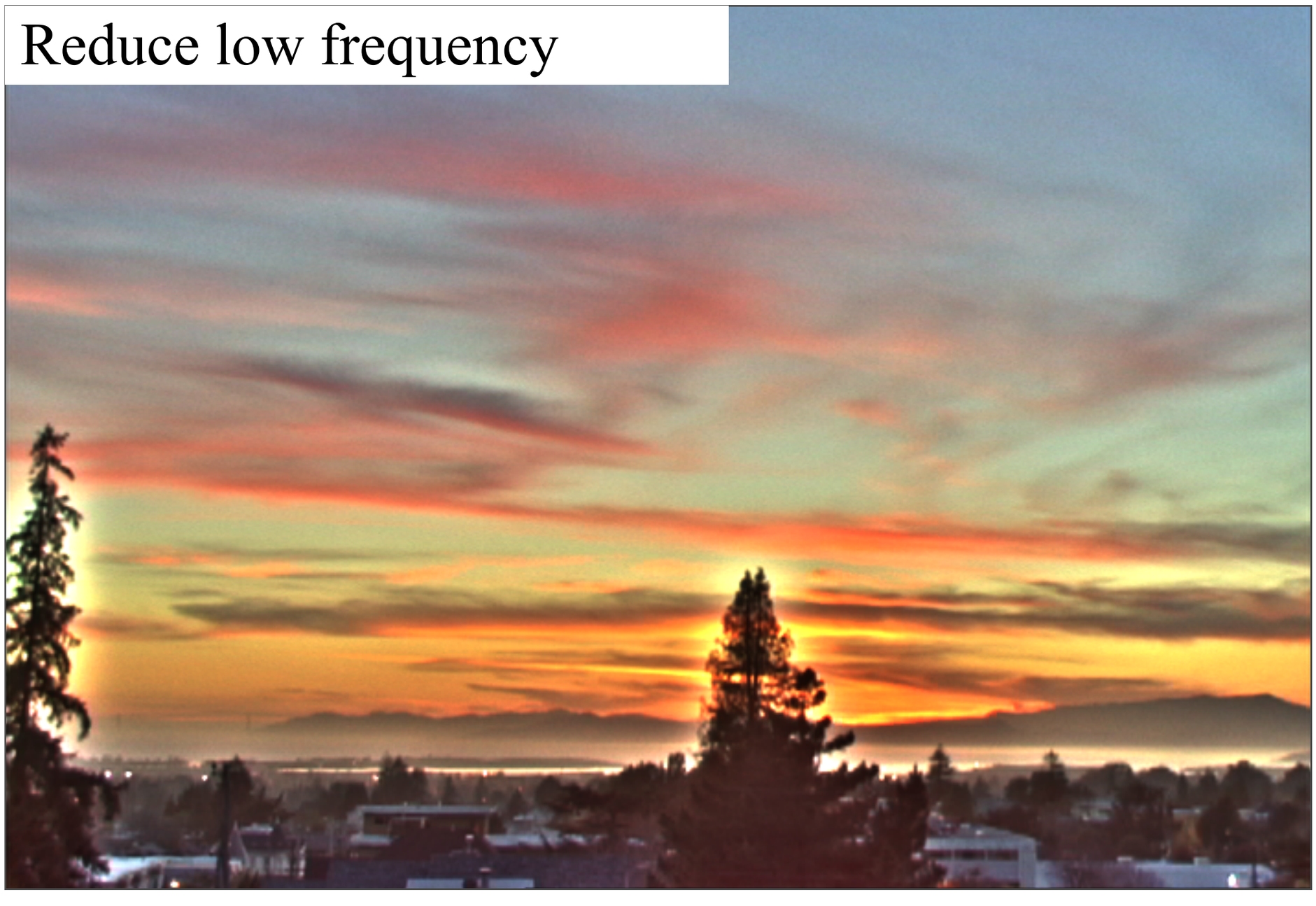
Oppenheim 1968, Chiu et al. 1993

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









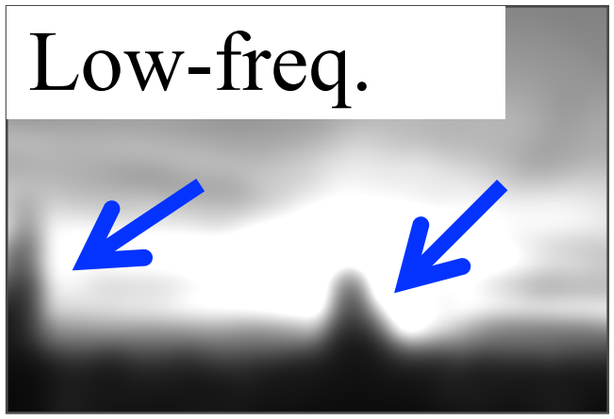
Homomorphic filtering

- **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**

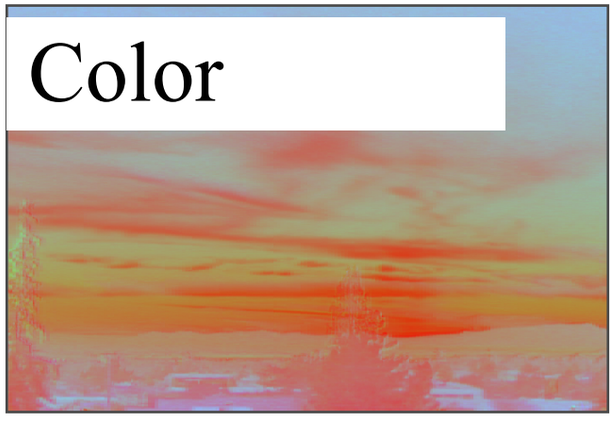
- **http://www.cs.sfu.ca/~stella/papers/blairthesis/main/node33.html**
- **See also Koenderink "Image processing done right" http://www.springerlink.com/(l1bpumaapconcbjngteojwqv)/app/home/contribution.asp?referrer=parent&backto=issue,11,53;journal,1538,3333;linkingpublicationresults,1:105633,1**

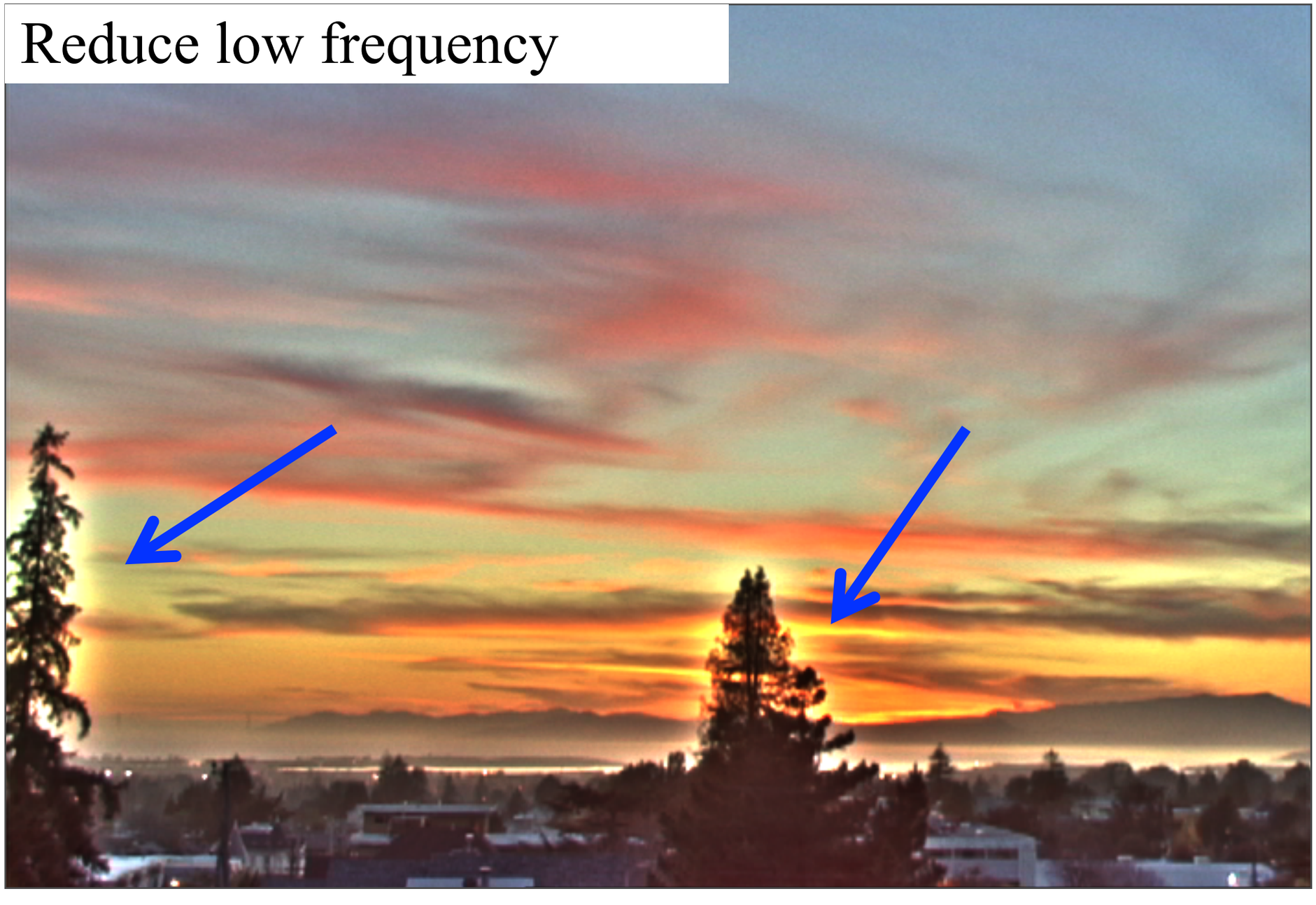
The halo nightmare

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





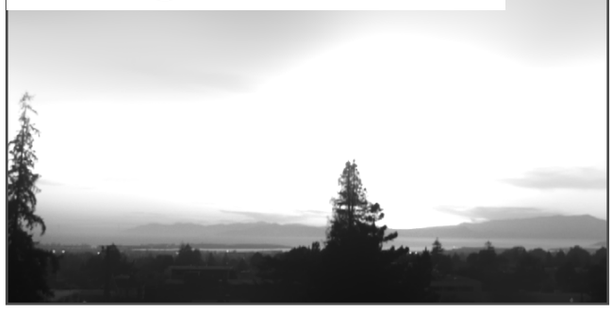




Our approach

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

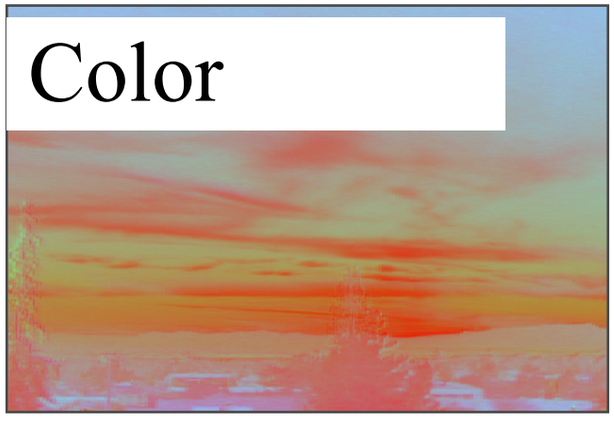
Large-scale



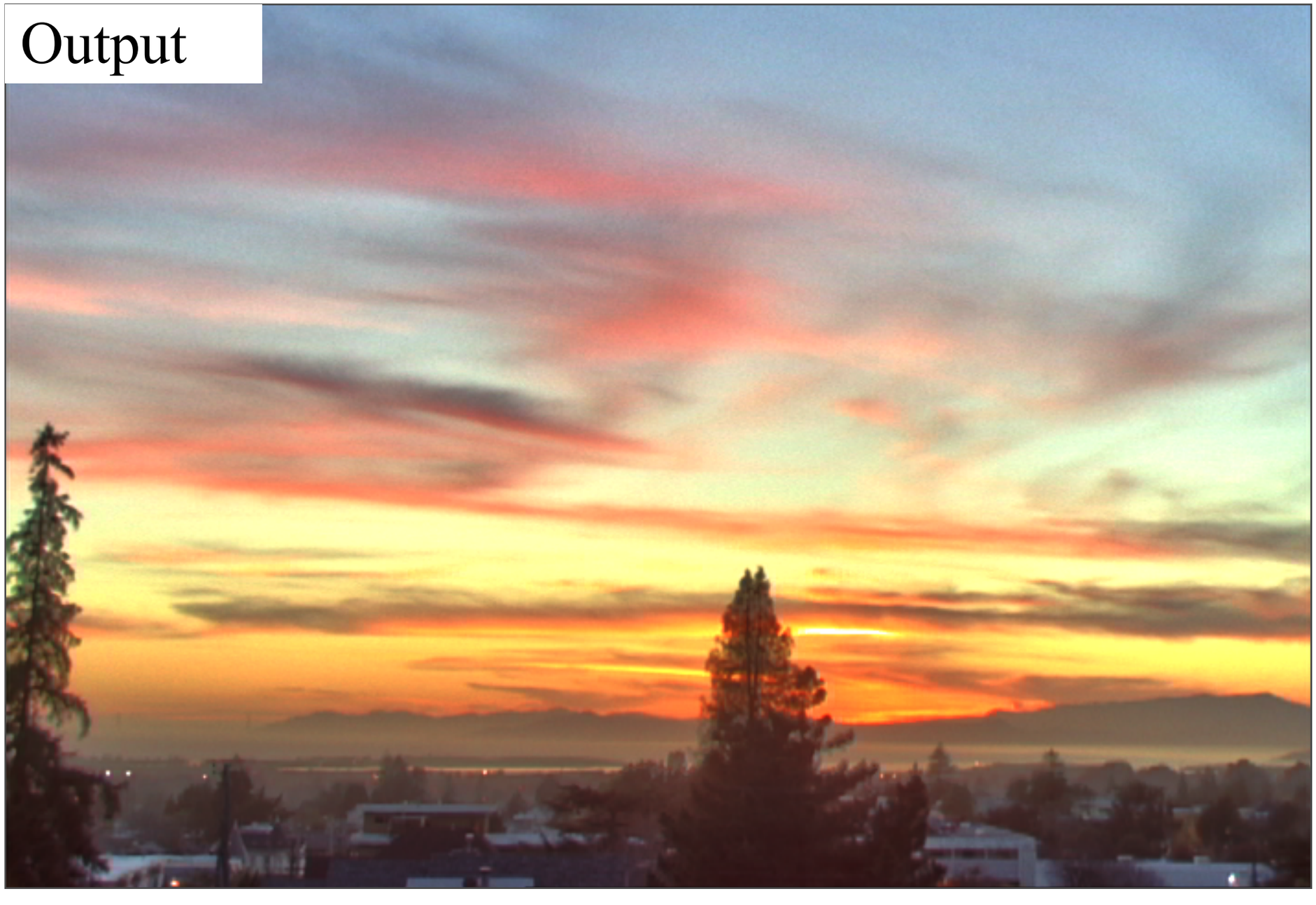
Detail



Color



Output



Bilateral filter

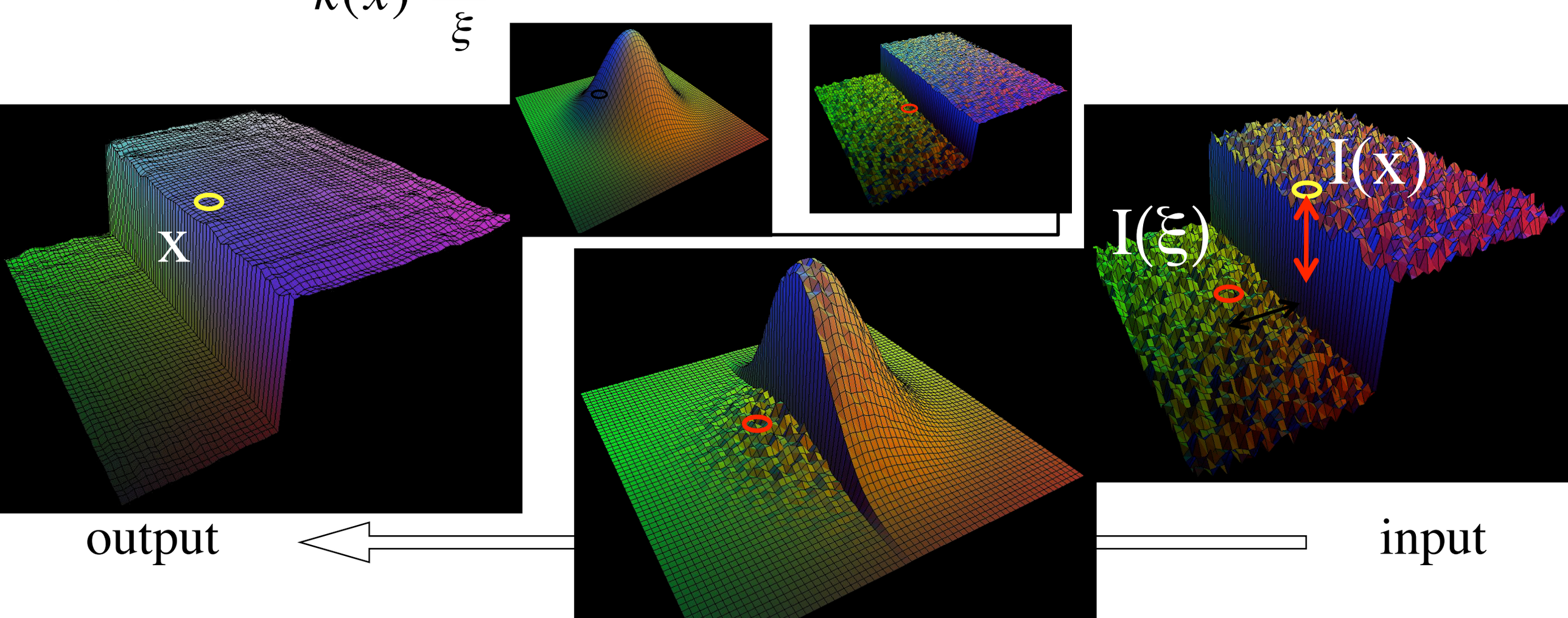
- **Tomasi and Manduci 1998]**
  - –http://www.cse.ucsc.edu/~manduchi/Papers/ICCV98.pdf
- **Developed for denoising**
- **Related to**
  - –SUSAN filter [Smith and Brady 95] http://citeseer.ist.psu.edu/smith95susan.html
  - –Digital-TV [Chan, Osher and Chen 2001] http://citeseer.ist.psu.edu/chan01digital.html
  - –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**

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

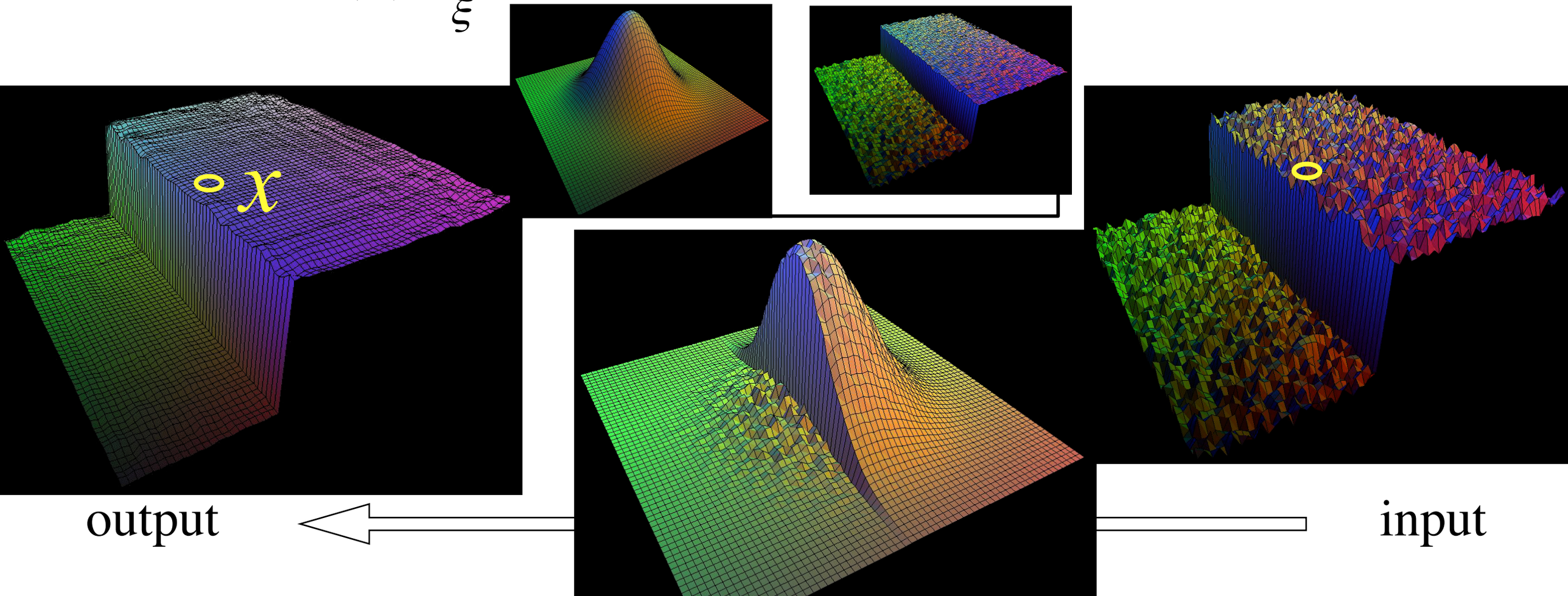


Normalization factor

[Tomasi and Manduchi 1998]

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

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

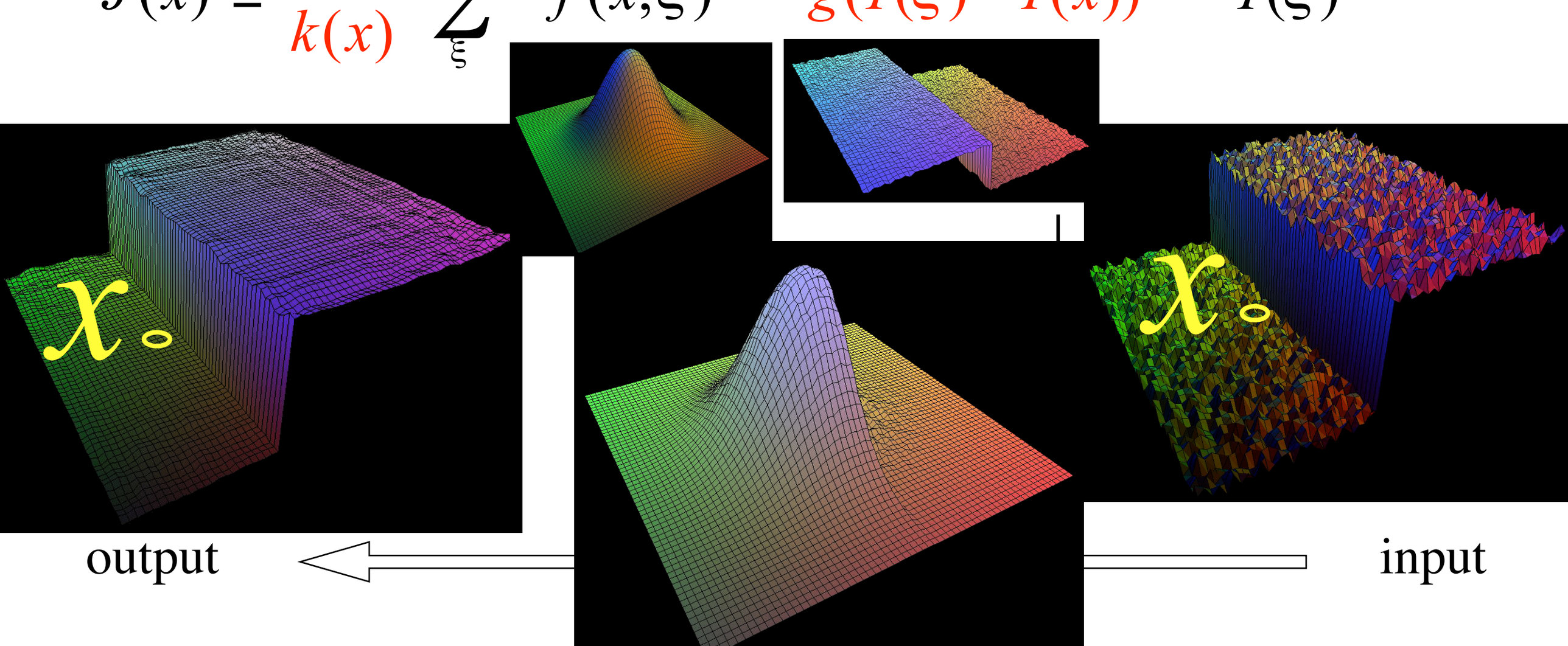


Bilateral filtering is non-linear

[Tomasi and Manduchi 1998]

- **The weights are different for each output pixel**

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



Recall bilateral denoising



Noisy input



After bilateral filter

Contrast reduction

Input HDR image



Contrast
too high!

Contrast reduction





intensity
= 0.4R+0.7G+0.01B



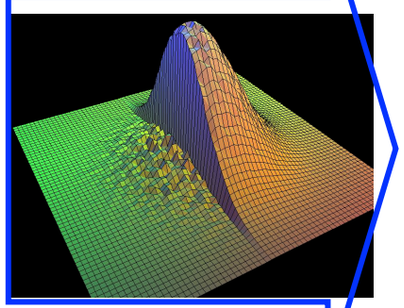
R’=R/intensity
G’=G/intensity
B’=B/intensity

important to use ratios (makes it luminance invariant)

Contrast reduction



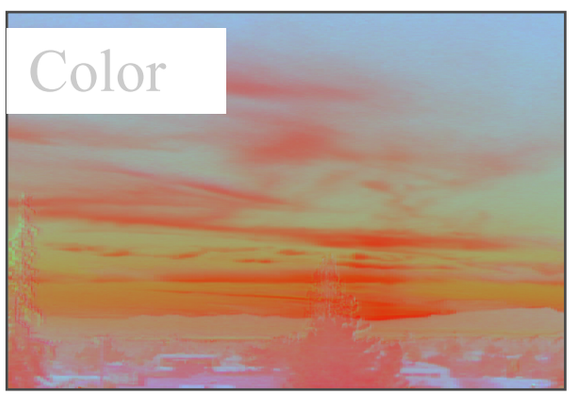




Bilateral Filter

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

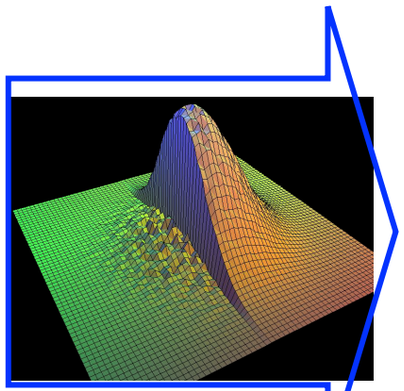
Contrast reduction

Input HDR image



Intensity





Bilateral Filter

in log

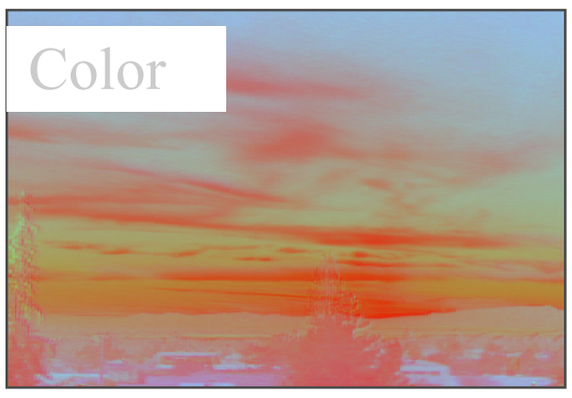
Large scale



Detail



Color



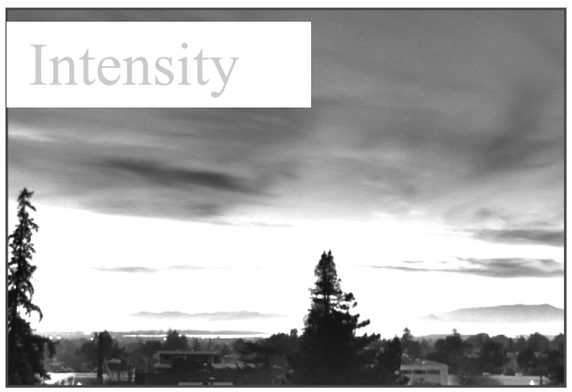
Detail = log intensity - large scale (residual)

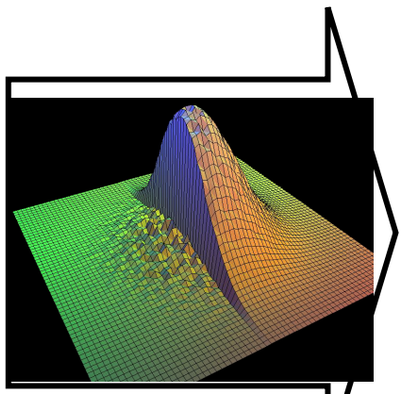
Contrast reduction

Input HDR image



Intensity

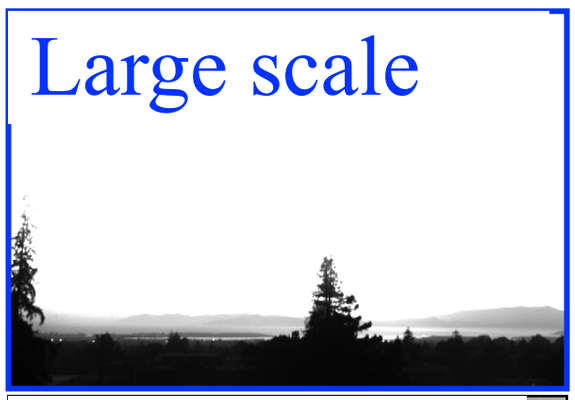




Bilateral Filter

in log

Large scale



Detail

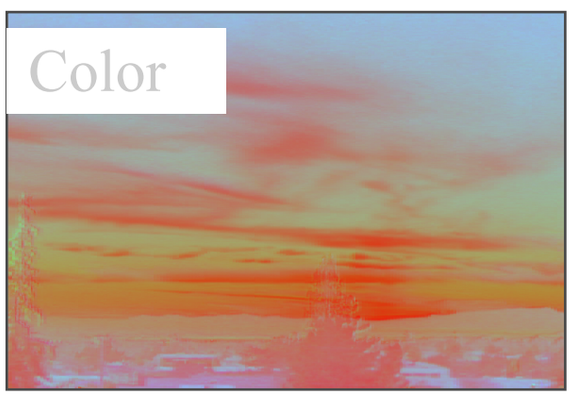


Reduce contrast

Large scale

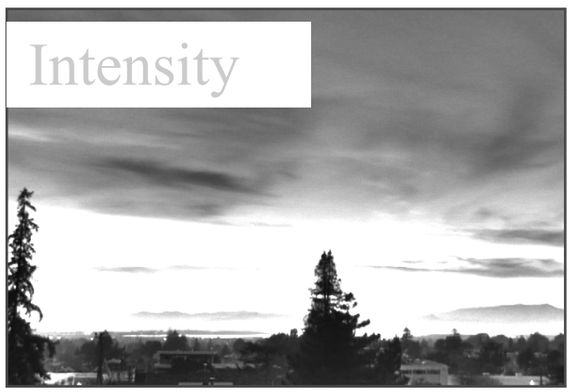


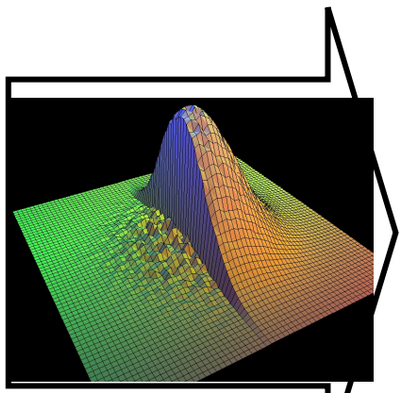
Color



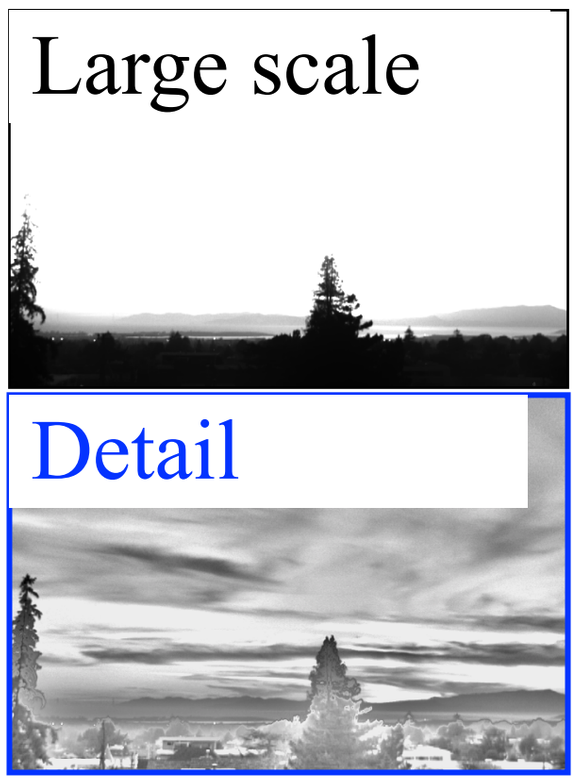
Contrast reduction





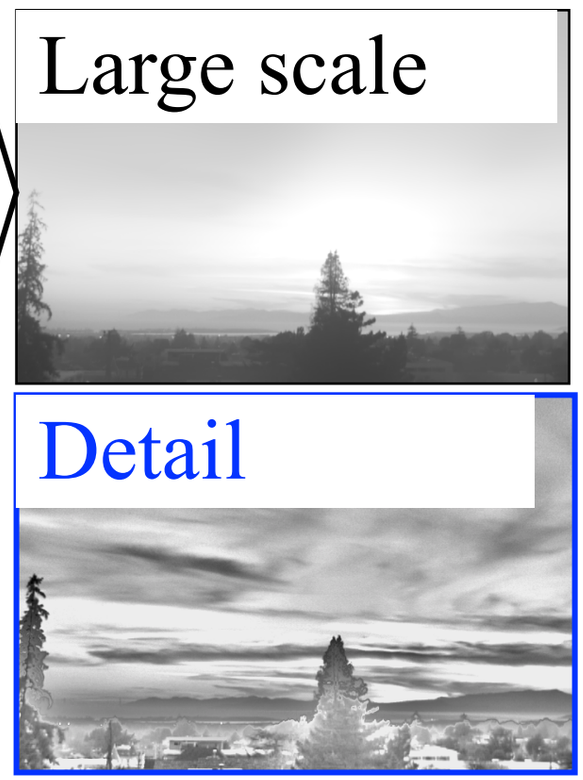


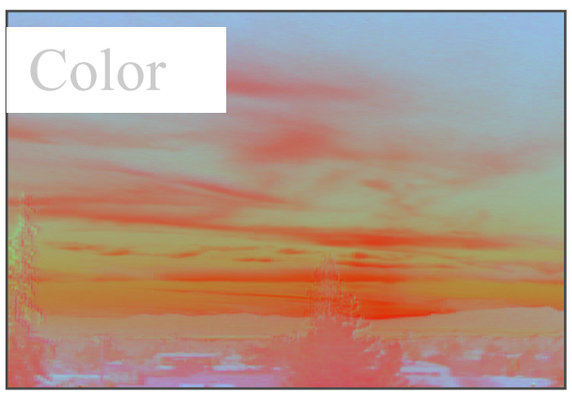
Fast
Bilateral
Filter



Reduce
contrast

Preserve!

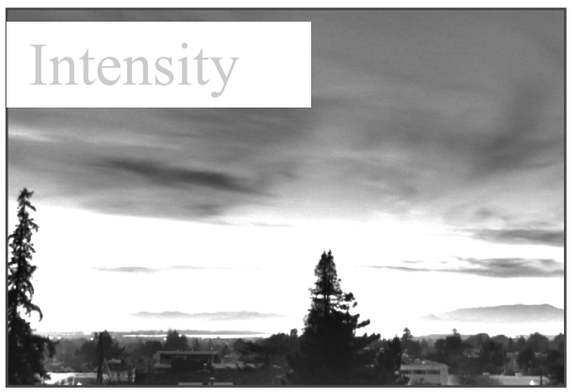


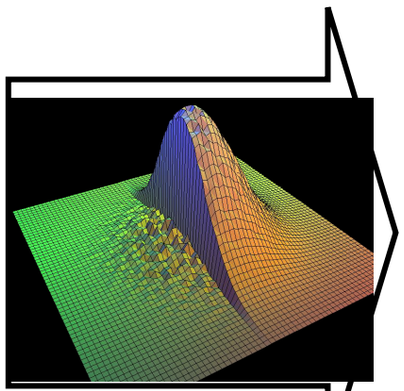


Contrast reduction









Fast
Bilateral
Filter



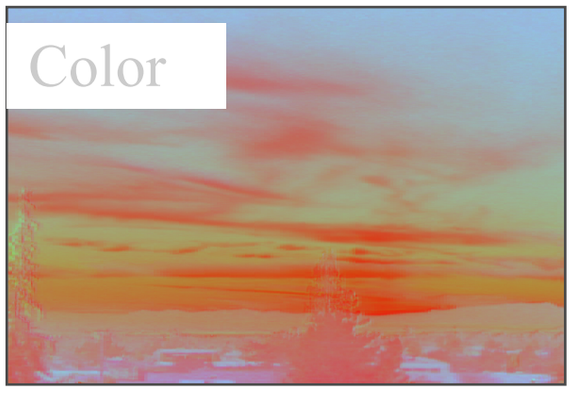


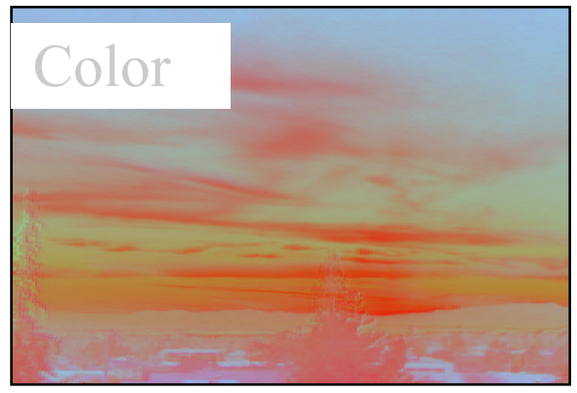
Reduce
contrast

Preserve!









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**

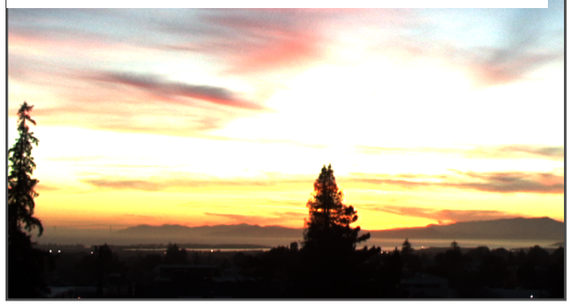
outLog= detailAmp*inLogDetail +
k(inLogLarge – max(inLogLarge))

Final output

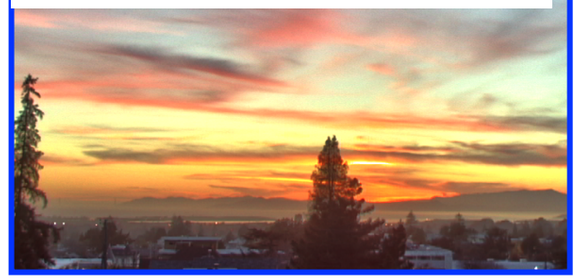
- From last slide:
  outLog= detailAmp*inLogDetail +
  k(inLogLarge – max(inLogLarge))
- outIntensity=10^{outLog}
- Recall that R’,G’,B’ is the intensity-normalized RGB color
- outR=outIntensity * R’
- outG=outIntensity * G’
- outB=outIntensity * B’

Recap

Input HDR image

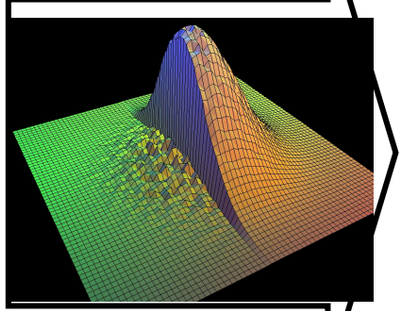


Output



Intensity





Fast Bilateral Filter **IN LOG**

Large scale



Detail



detail=
input log - large scale

Reduce contrast

Preserve!

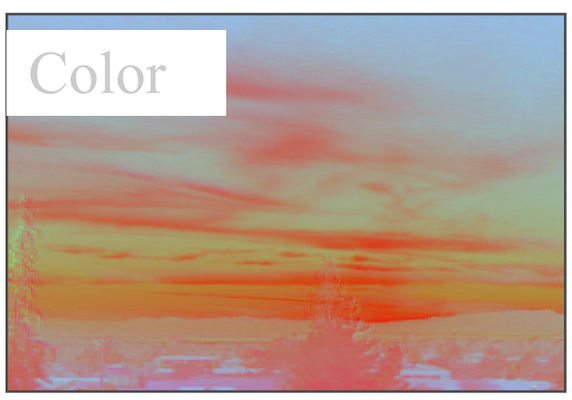
Large scale



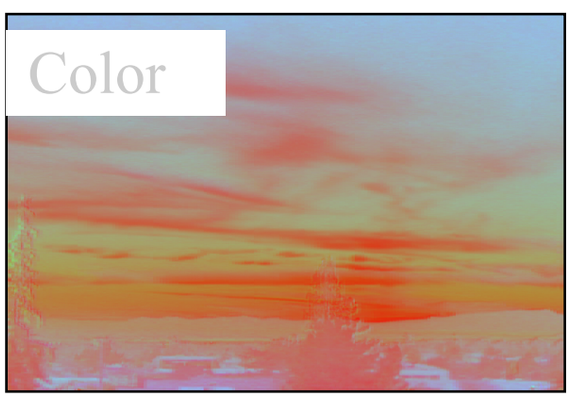
Detail



Color



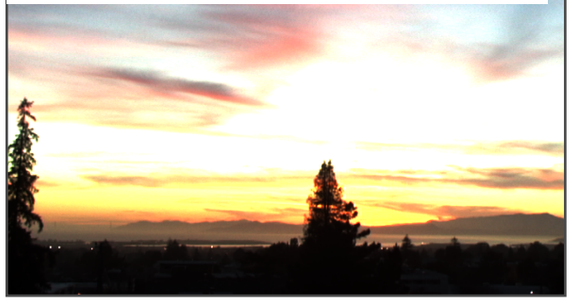
Color



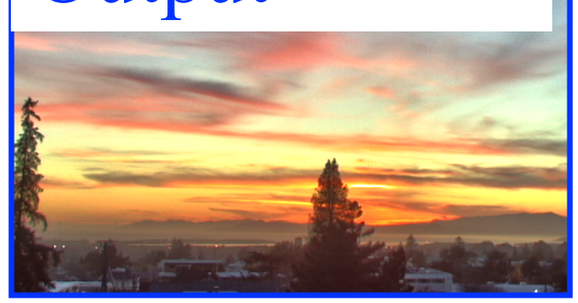
Bells and whistles: increase detail

CSAIL

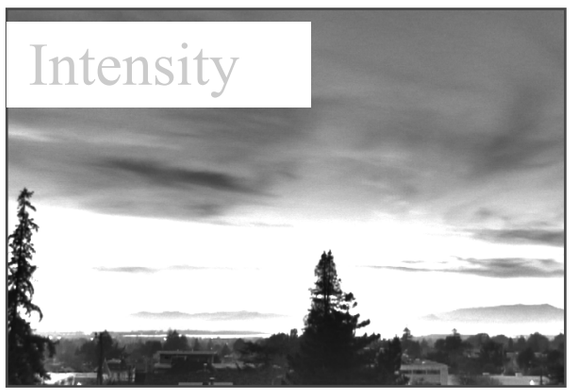
Input HDR image

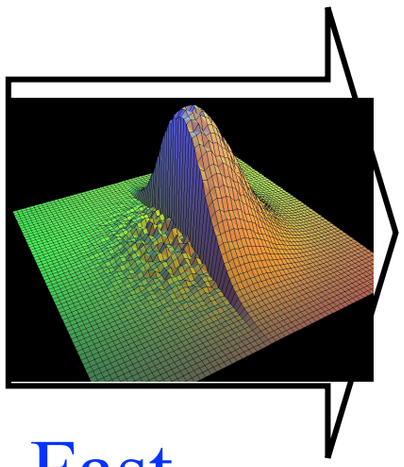


Output



Intensity





Fast Bilateral Filter **IN LOG**

Large scale



Detail



detail=
input log - large scale

Reduce contrast

Amplify

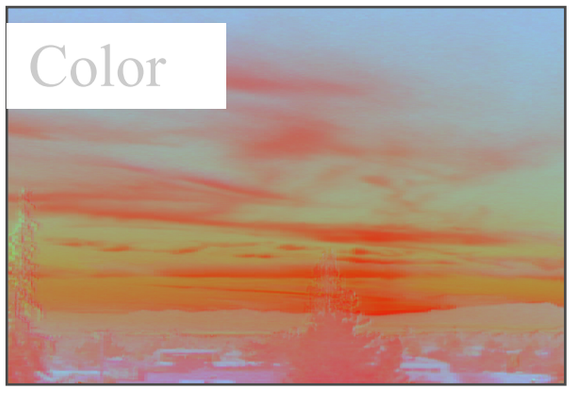
Large scale



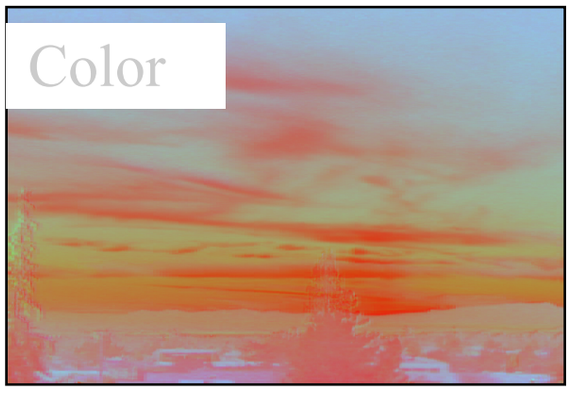
Detail

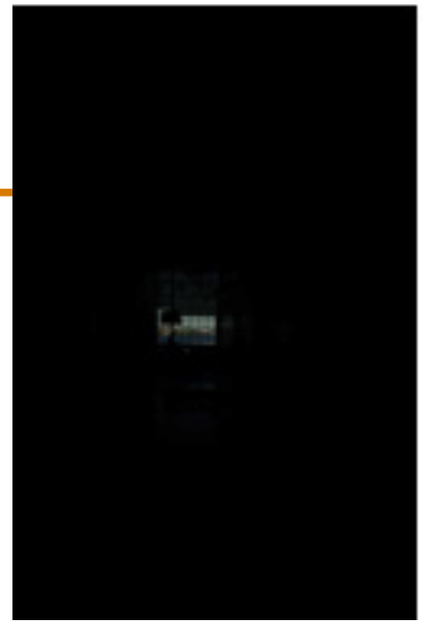


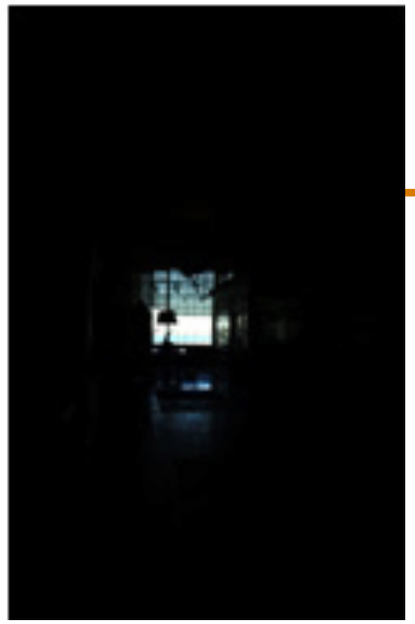
Color



Color













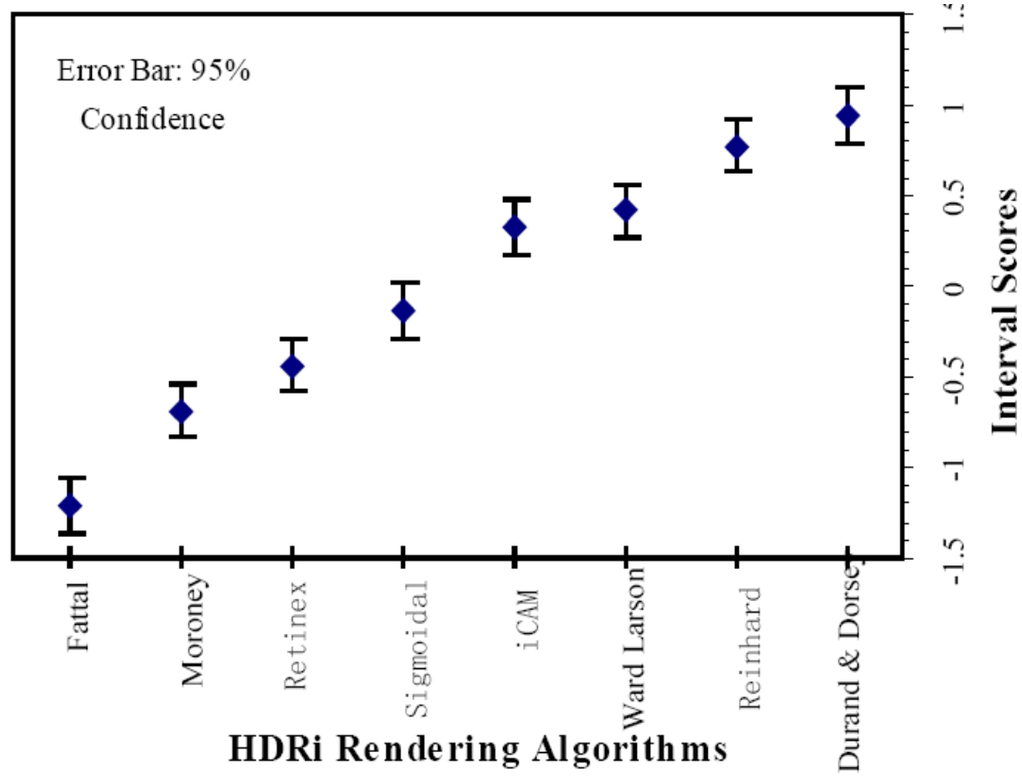


What matters

- **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**



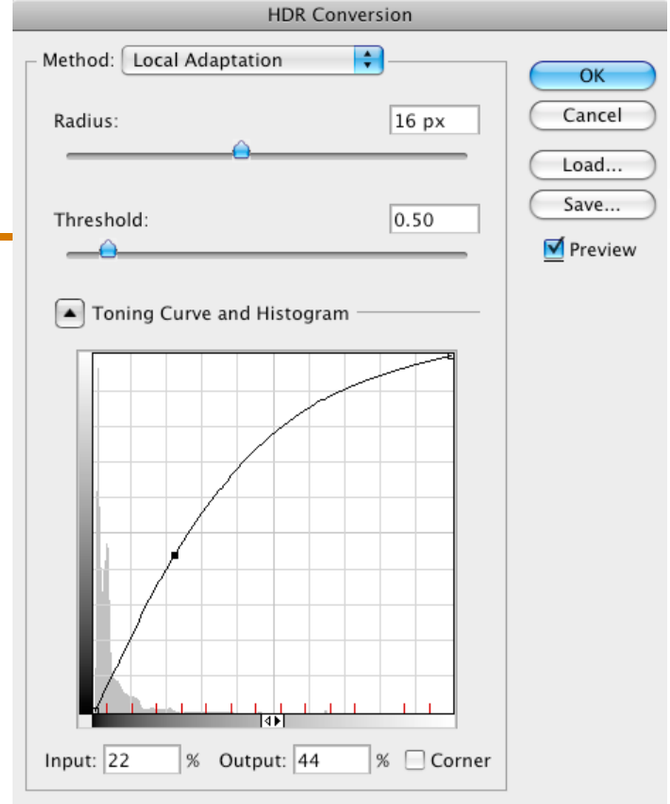
From Kuang et al.

	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

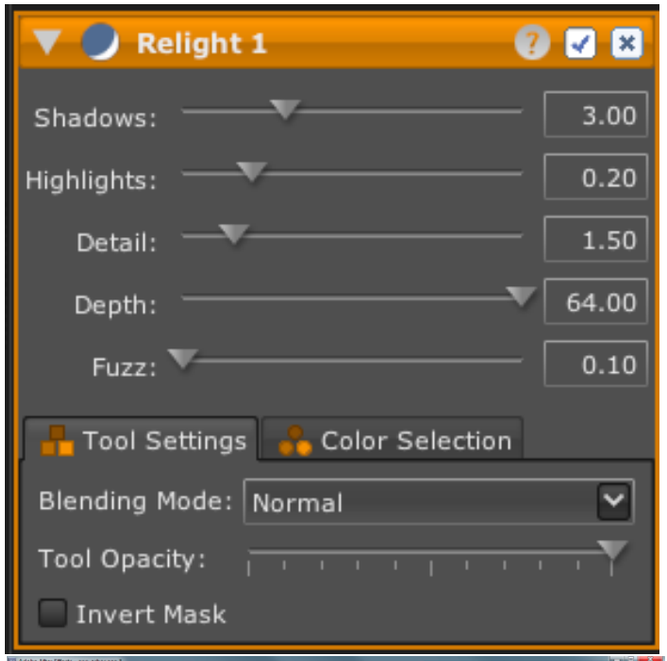
- **Photoshop “Local adaptation”**



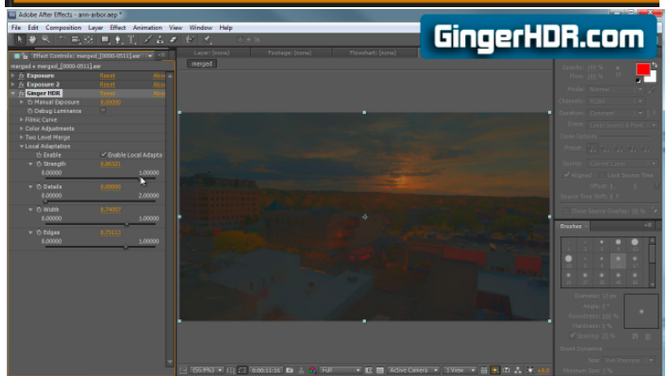
- **Lightroom Fill Light**
  - –or shadow



- **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**

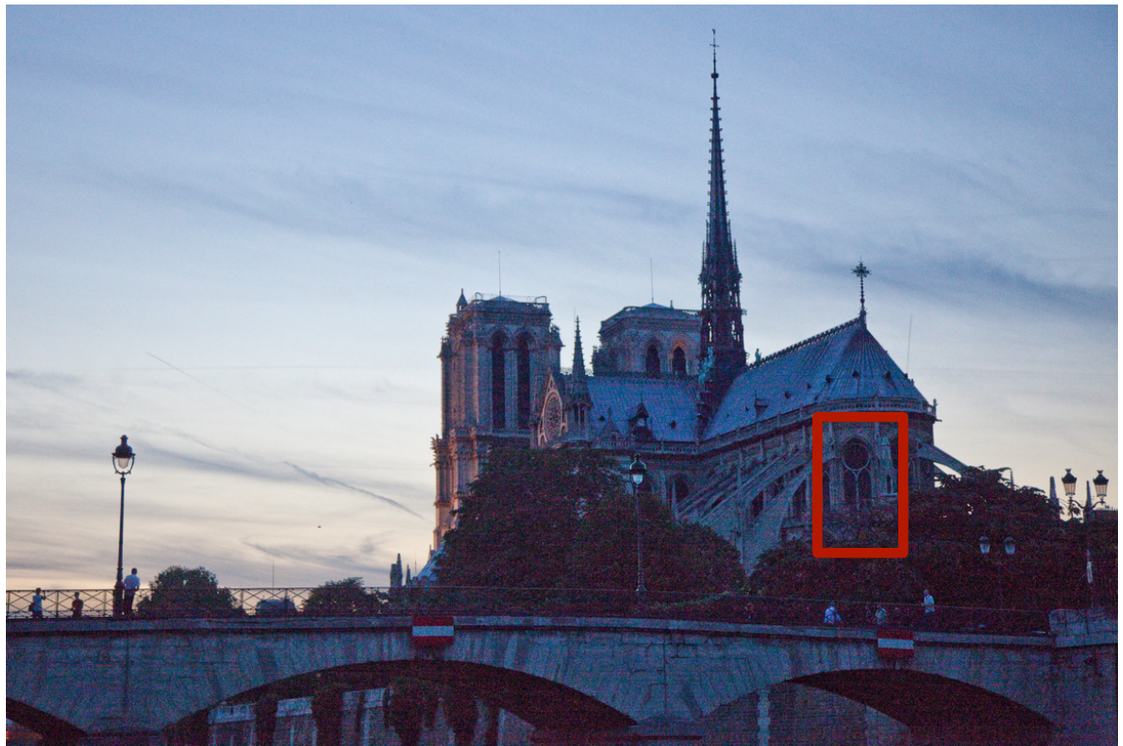




Up to a point

- **Noise gets amplified in the shadows**





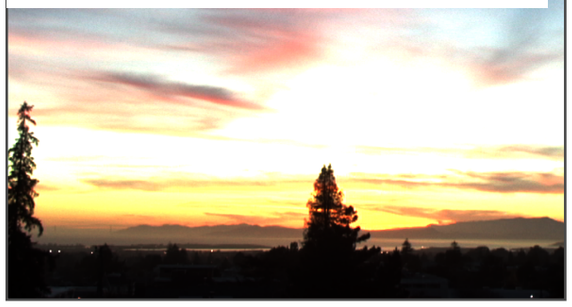


Questions?

Bilateral Grid

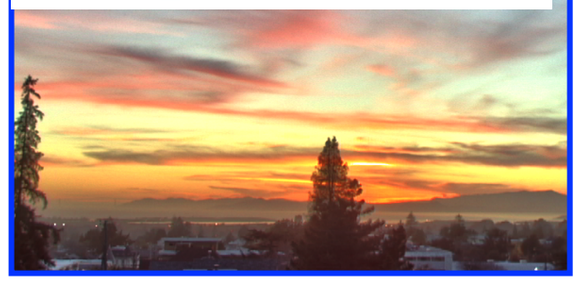
Back to bilateral tone mapping

Input HDR image



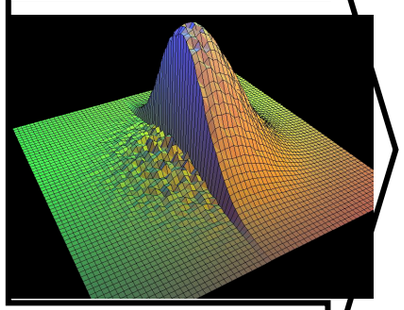
CHALLENGE: SPEED

Output



Intensity





Bilateral Filter

Large scale



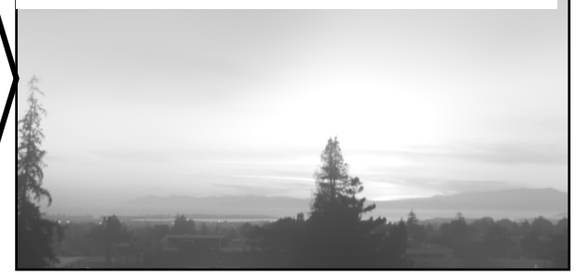
Detail



Reduce contrast

Preserve!

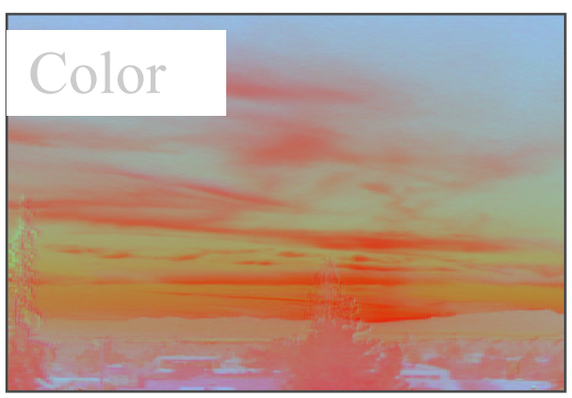
Large scale



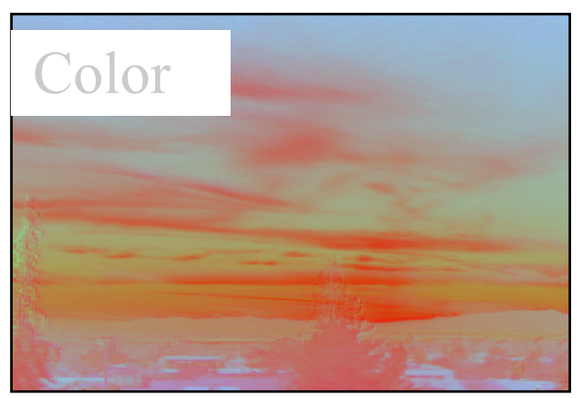
Detail



Color



Color



Speed

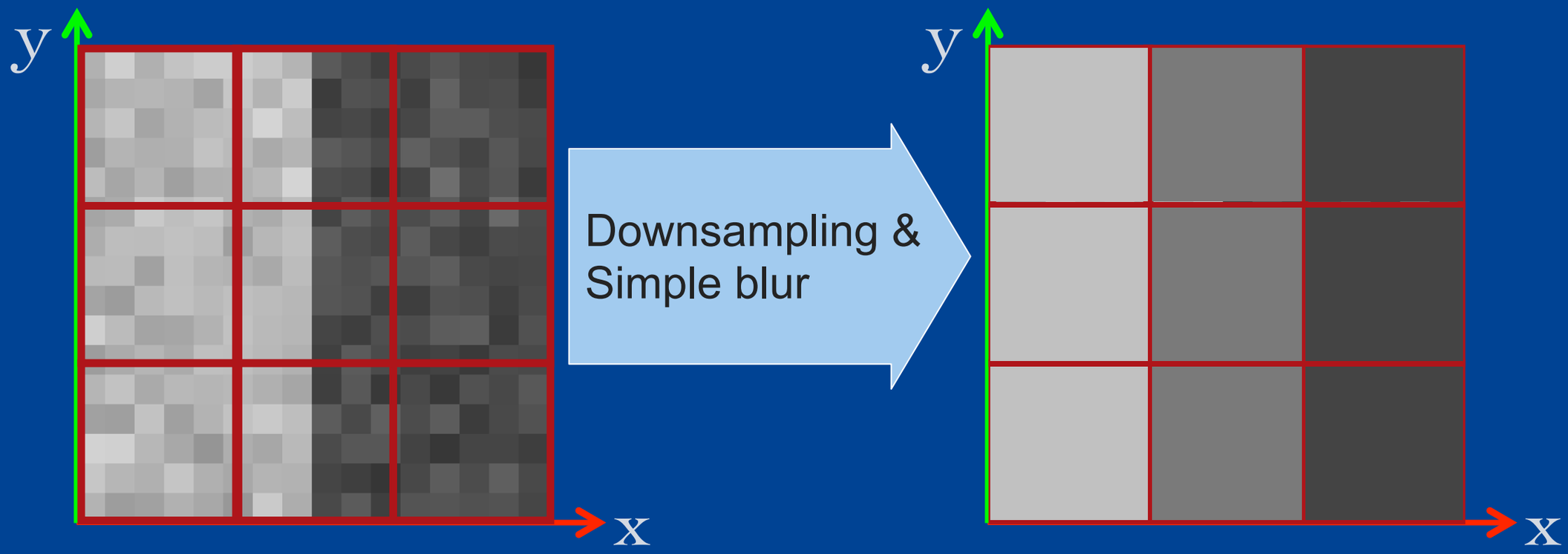
- **Direct bilateral filtering is slow (minutes)**

- **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]

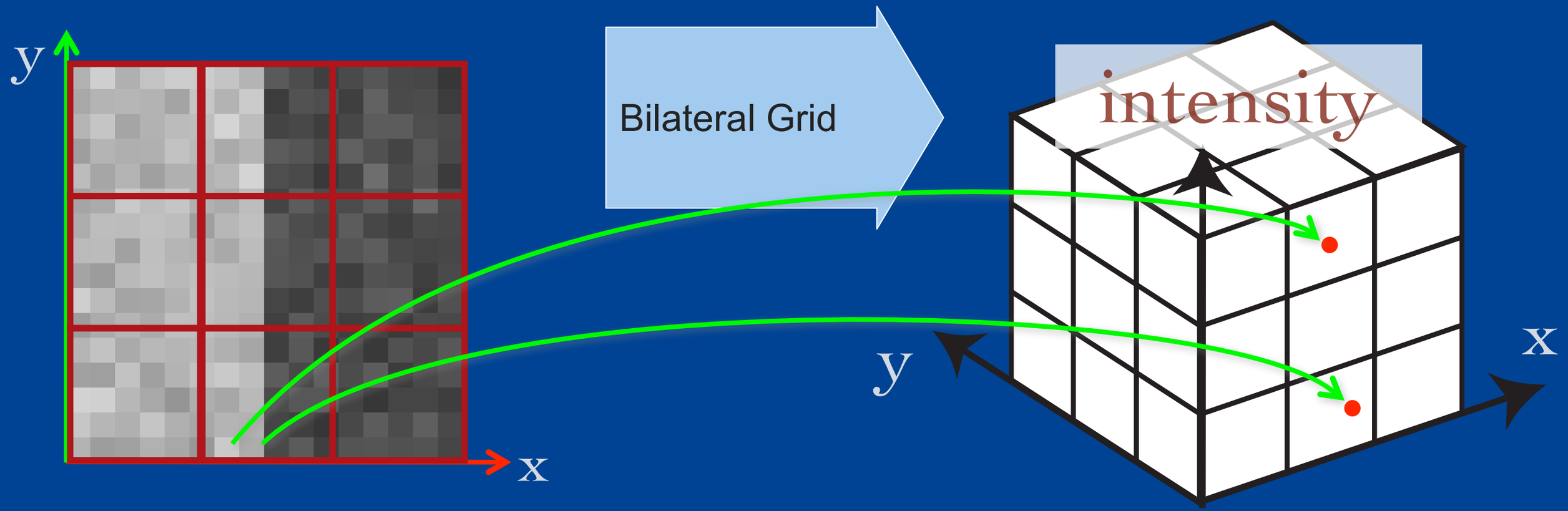
- When we smooth, we reduce complexity of image => we should be able to do it at a lower resolution



Bilateral Grid: basic motivation

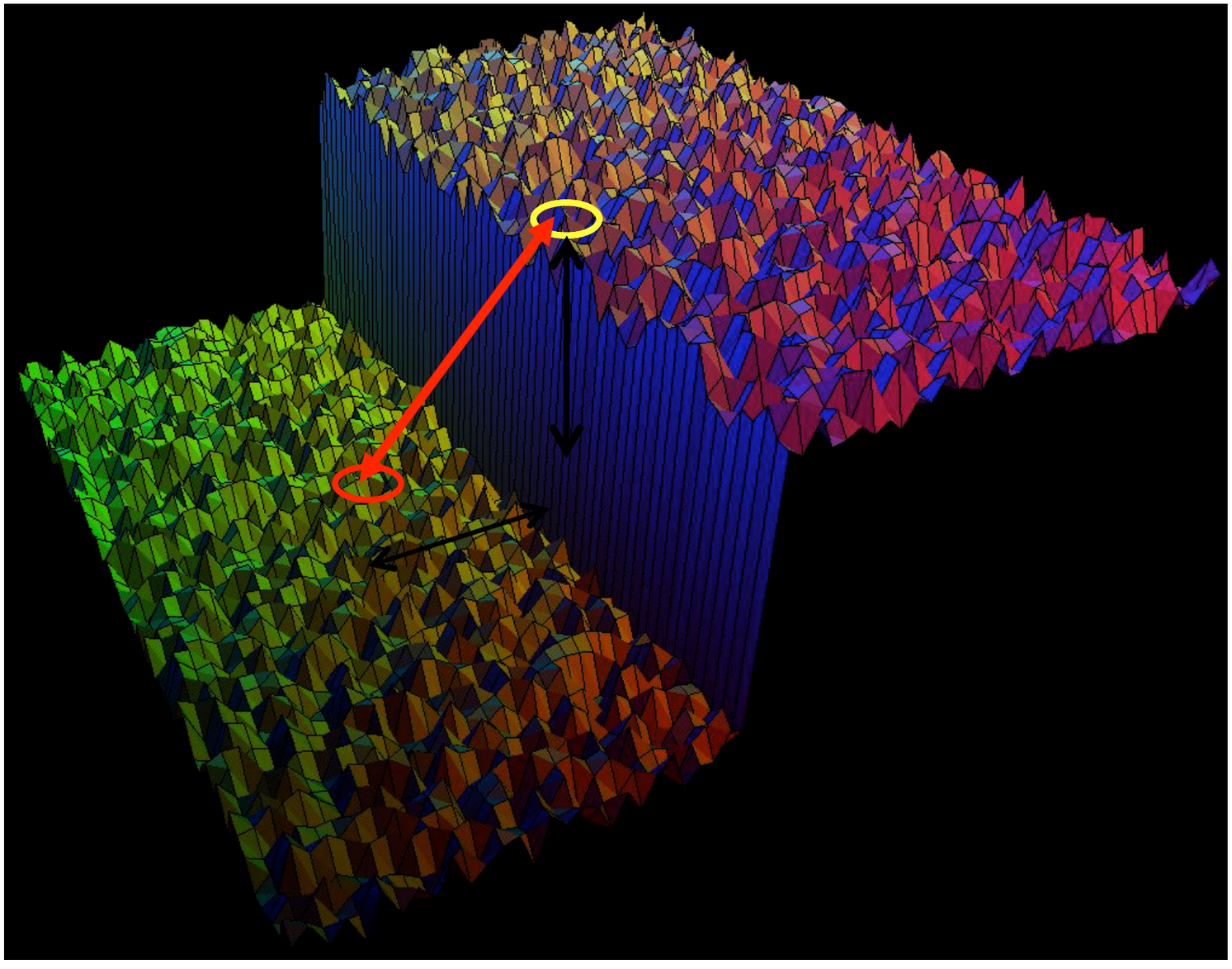
[Paris and Durand 06, Chen et al. 07]

- 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



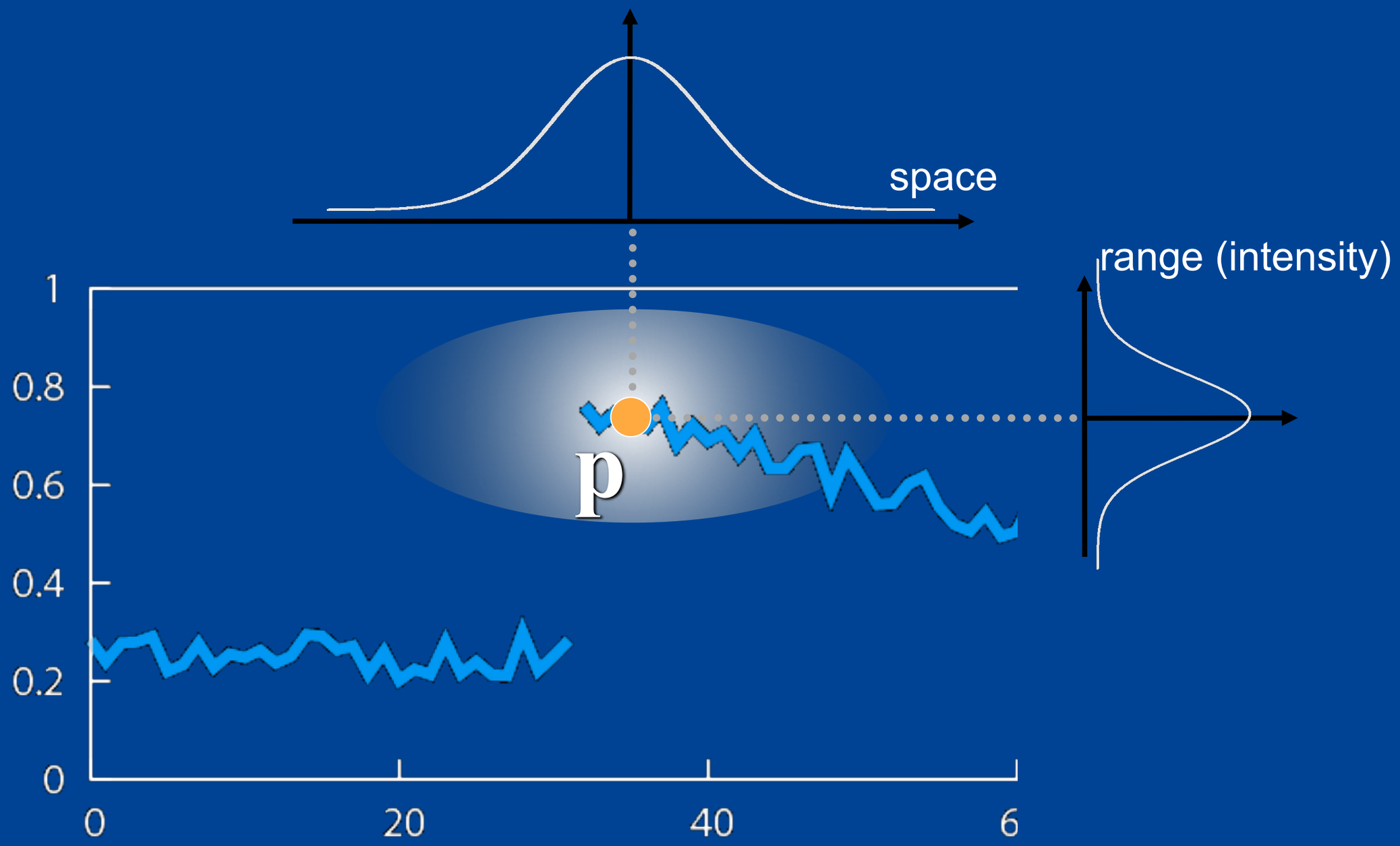
3D gaussian

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



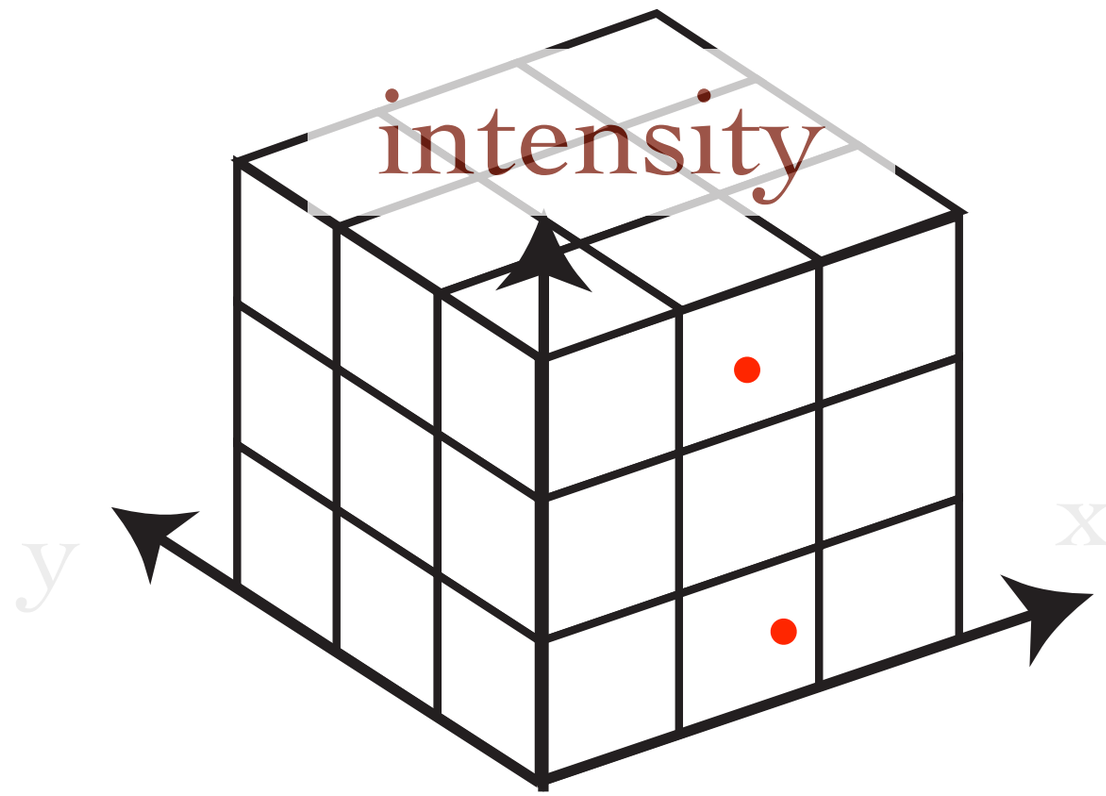
Bilateral Grid [Paris and Durand 06, Chen et al. 07]

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

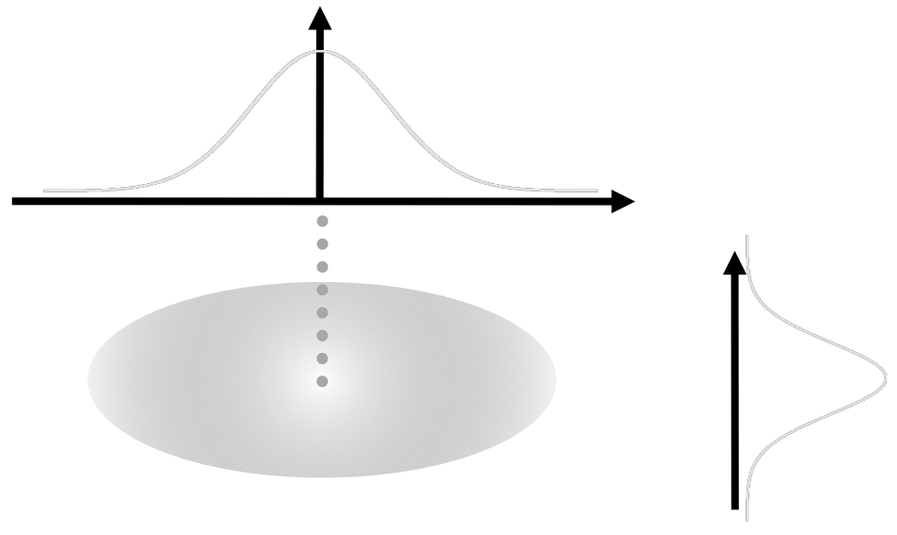


Fast bilateral filter idea

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



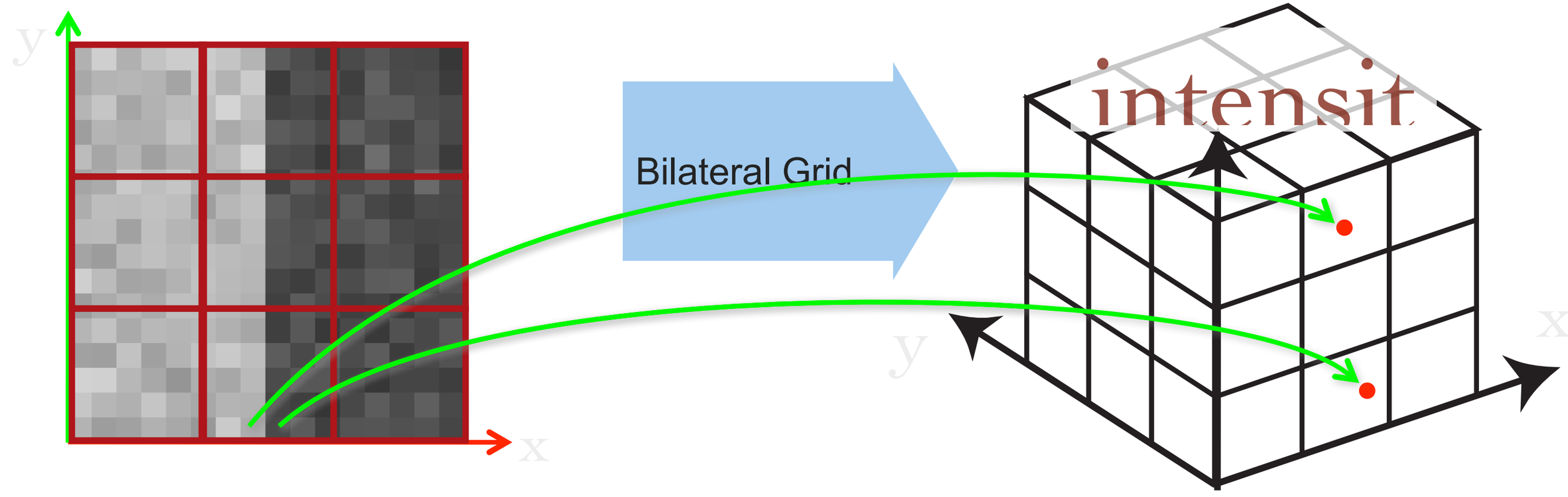
- **3D blur combines space and intensity terms**



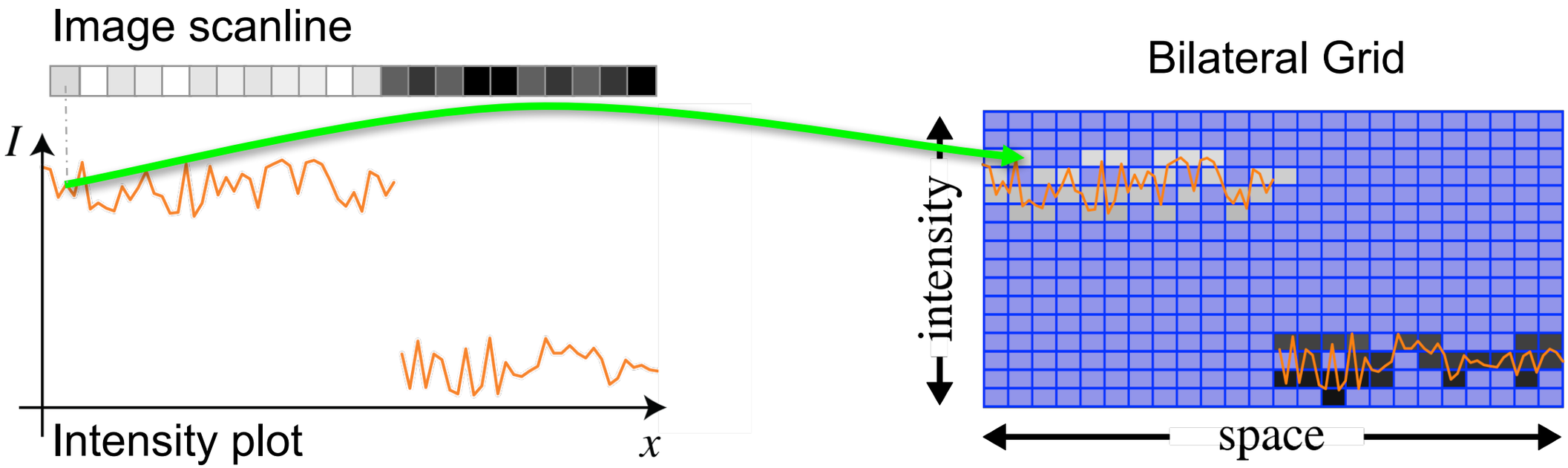
Questions?

Overview

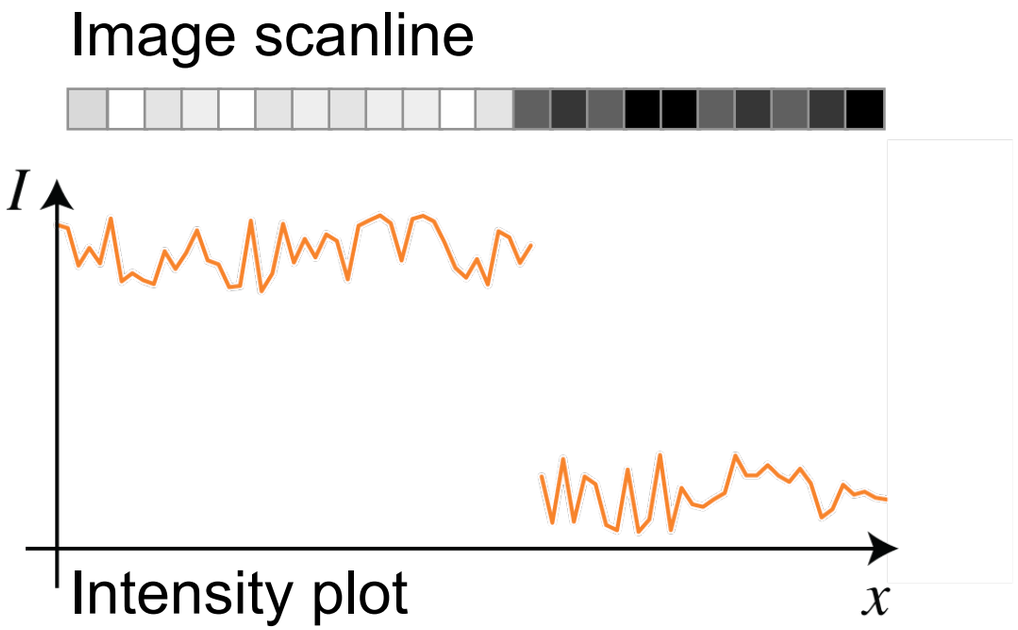
- **Convert image to bilateral grid**
  - (x, y) pixel goes to x, y, I(x,y)
- **Blur the grid**
  - 3D Gaussian combines 2D x,y term f and I term g
- **Convert back to 2D image space**

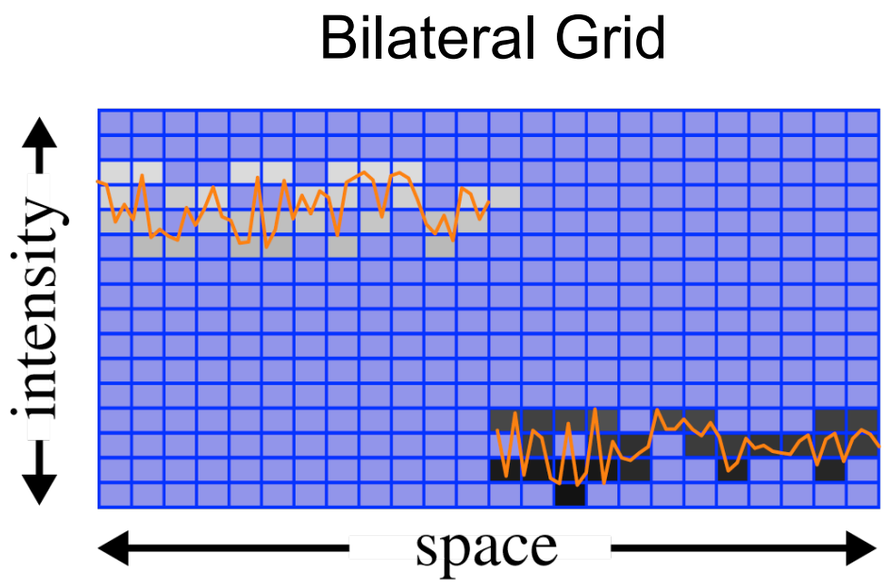


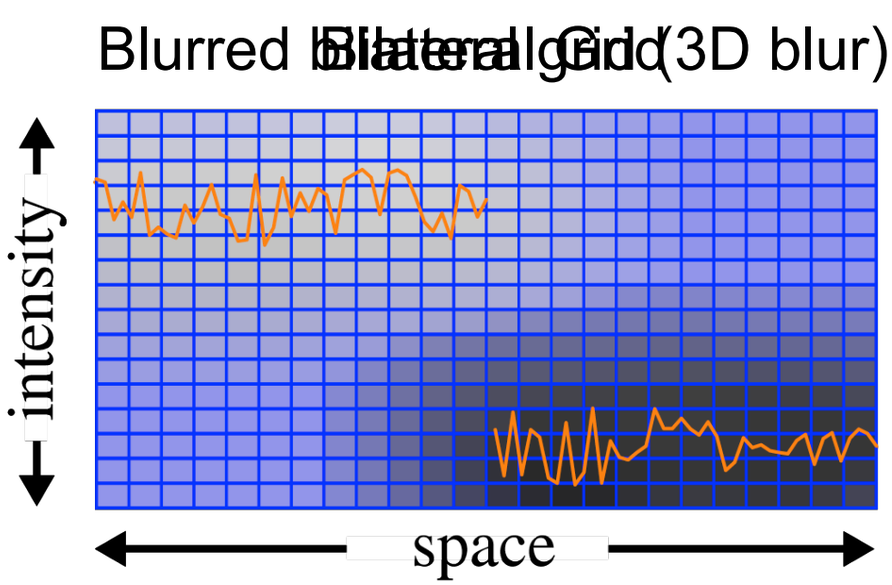
Bilateral Filter on the Bilateral Grid



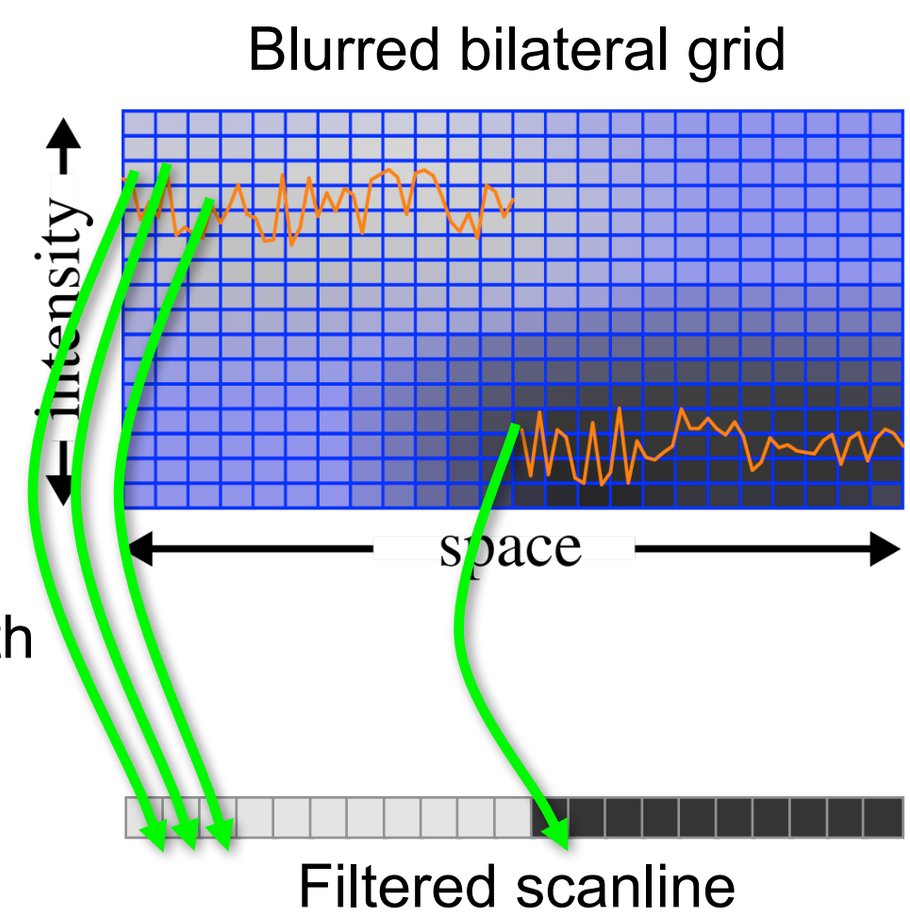
Bilateral Filter on the Bilateral Grid





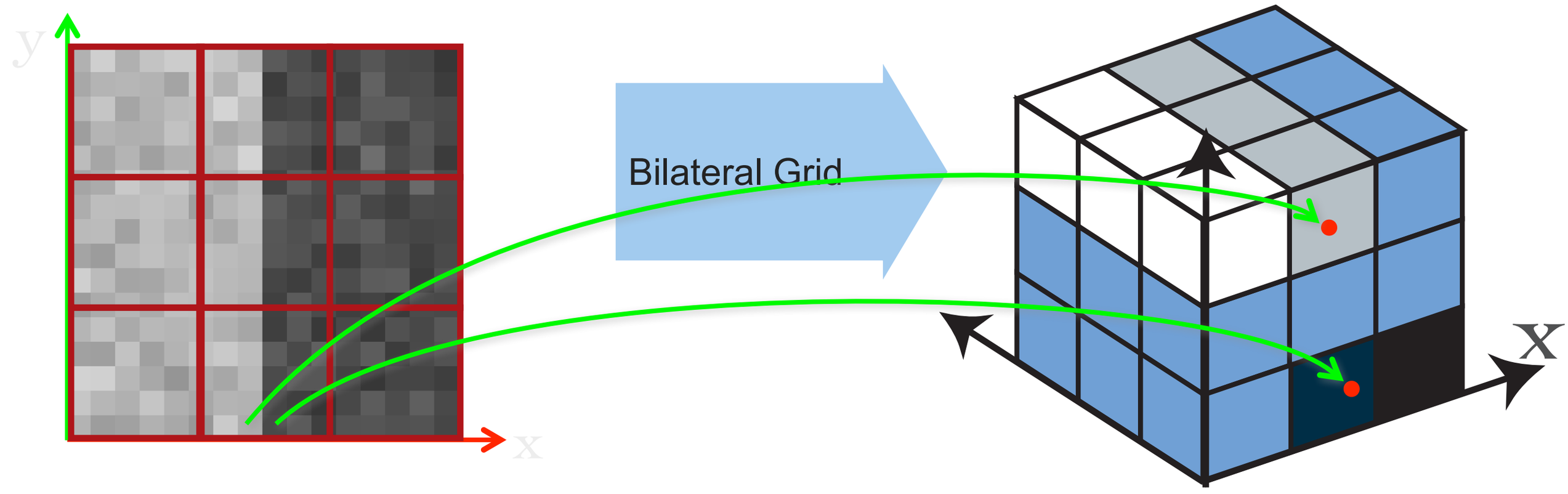


Query grid with input image



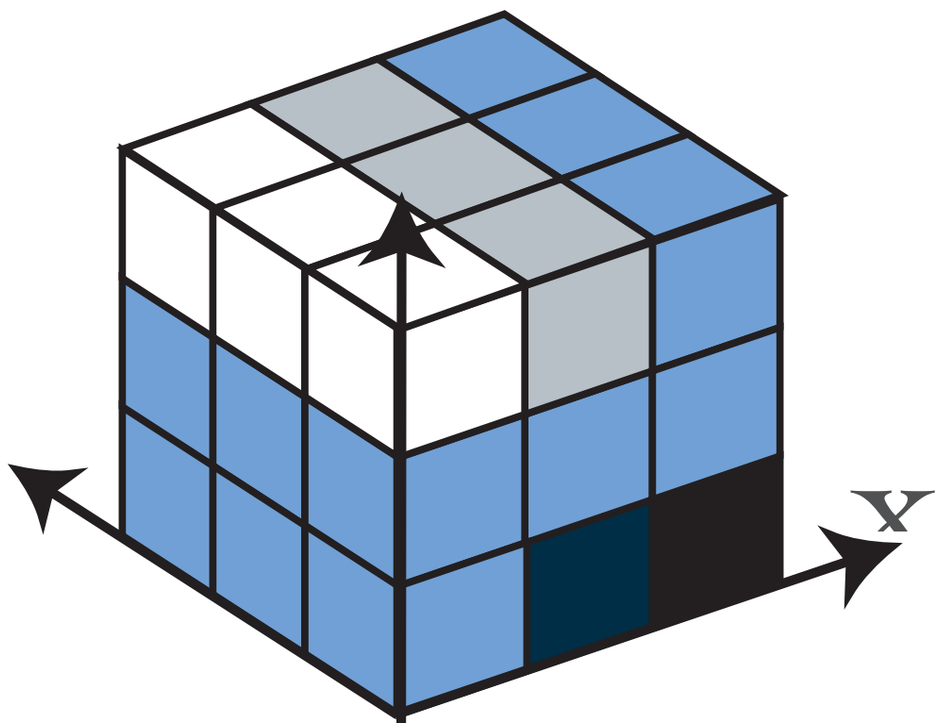
Grid creation

- **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)

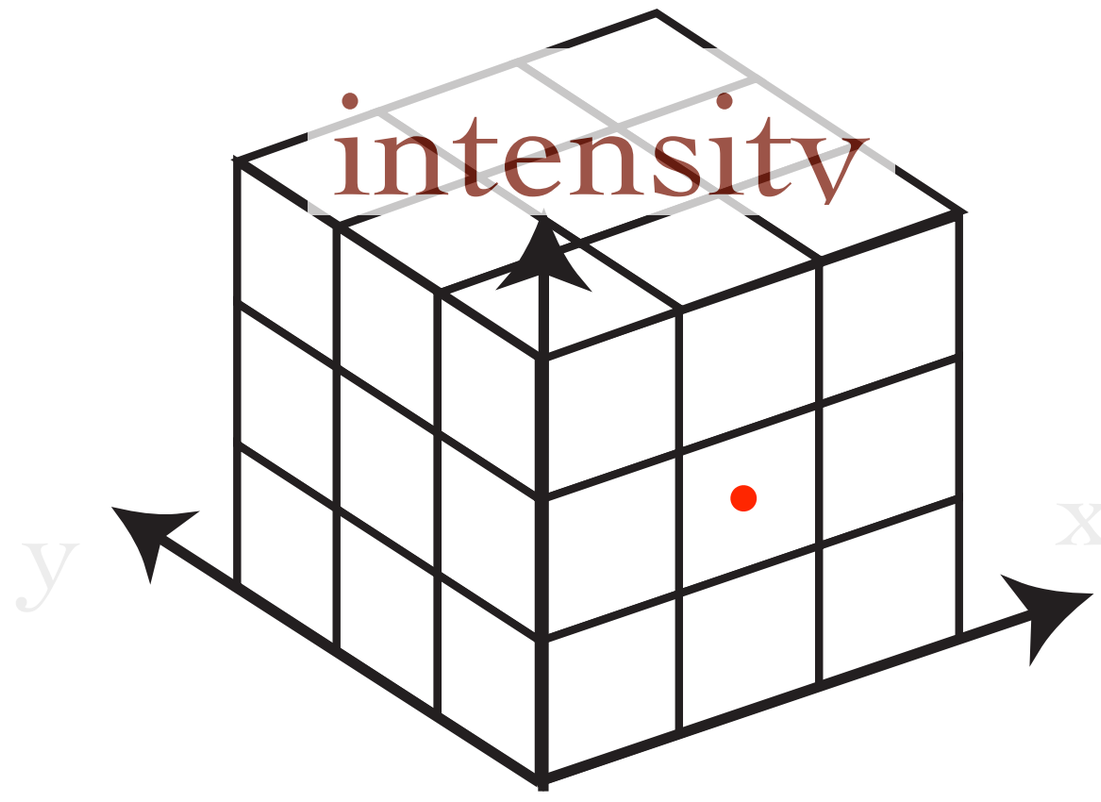
$$x, y, I \rightarrow \left[\frac{x}{\sigma_s}\right], \left[\frac{y}{\sigma_s}\right], \left[\frac{I}{\sigma_r}\right]$$

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:

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

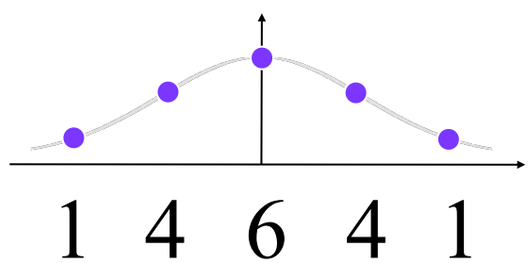


Even smarter: separable

- **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**

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

- **gives you a radial Gaussian**

Blurring

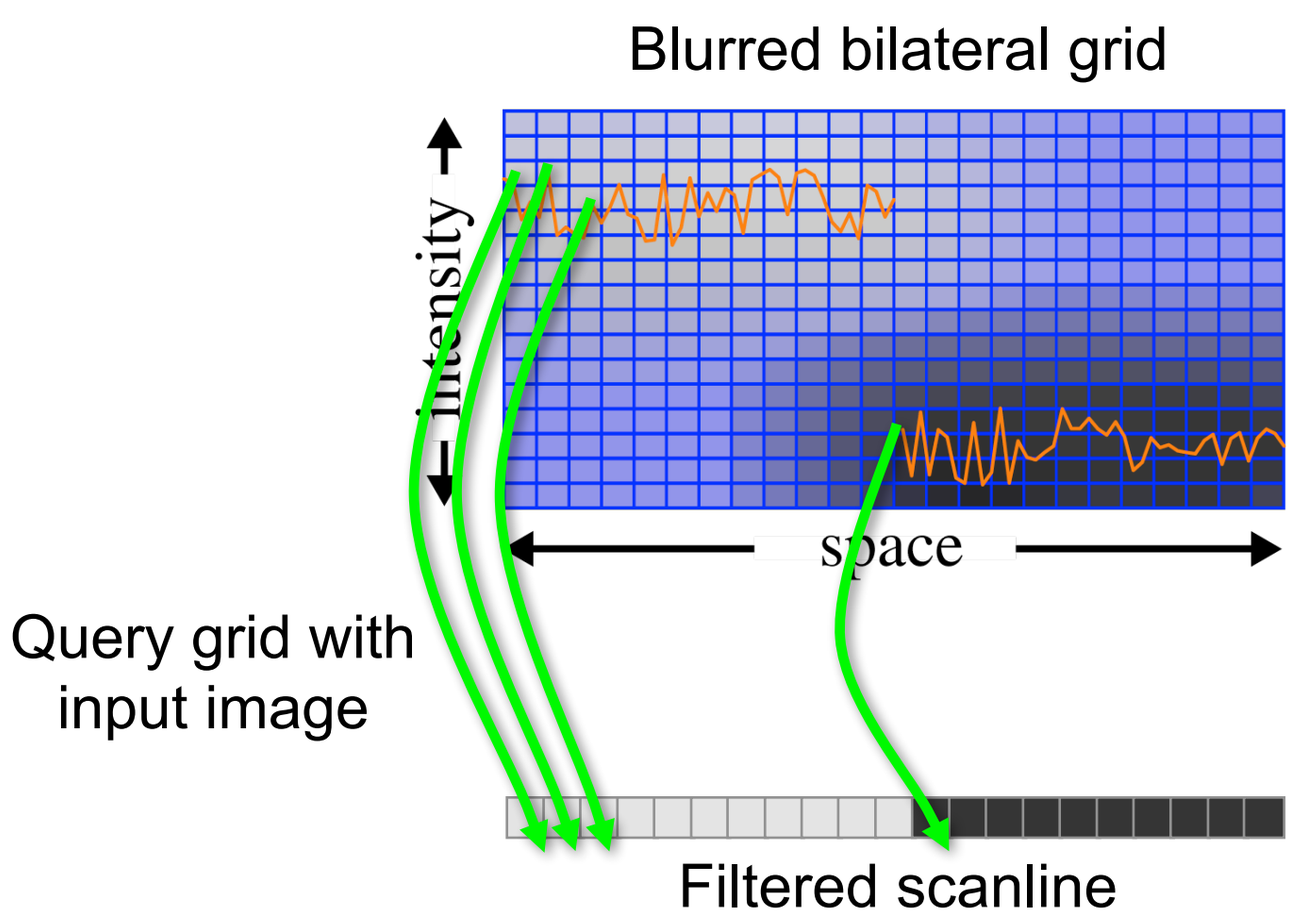
- **Blur BOTH the values and the weights**
- **Recall original bilateral filter formulas**

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

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

Slicing: critical step

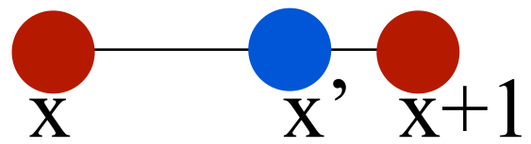
- **Read the grid at locations specified by input image**
  - Output at pixel x, y, is read from grid cell x, y, I(x, y)
- **Trilinear reconstruction**
  - because the grid is downsampled



Linear reconstruction

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

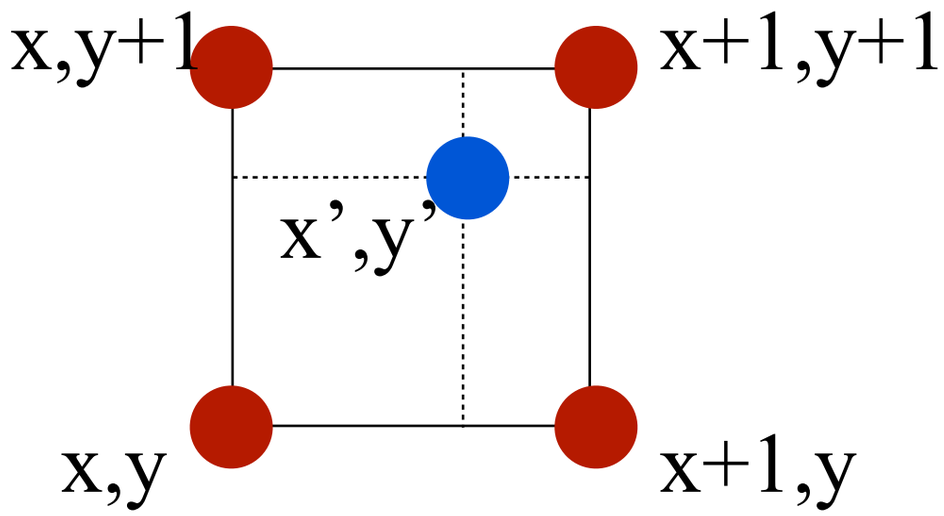
$$(1-x'+x)v(x, y)+(x'-x)v(x+1, y)]$$



BiLinear reconstruction

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

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



Bilinear: order does not matter

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

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

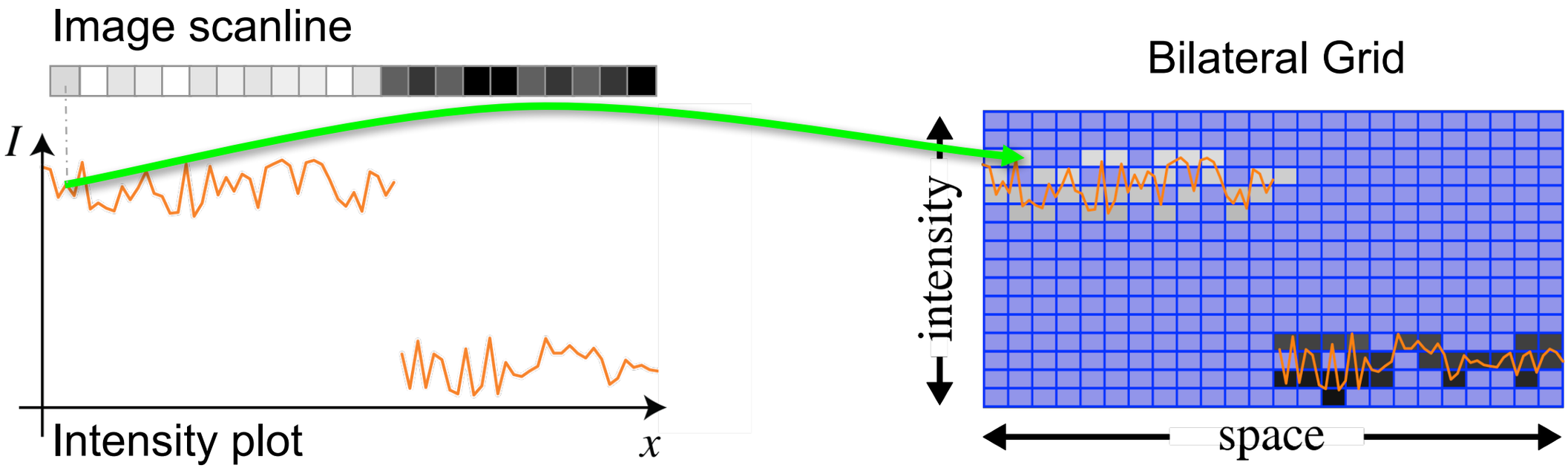
- **Linear along y followed by linear along x**

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

- **Reduces to the same terms**

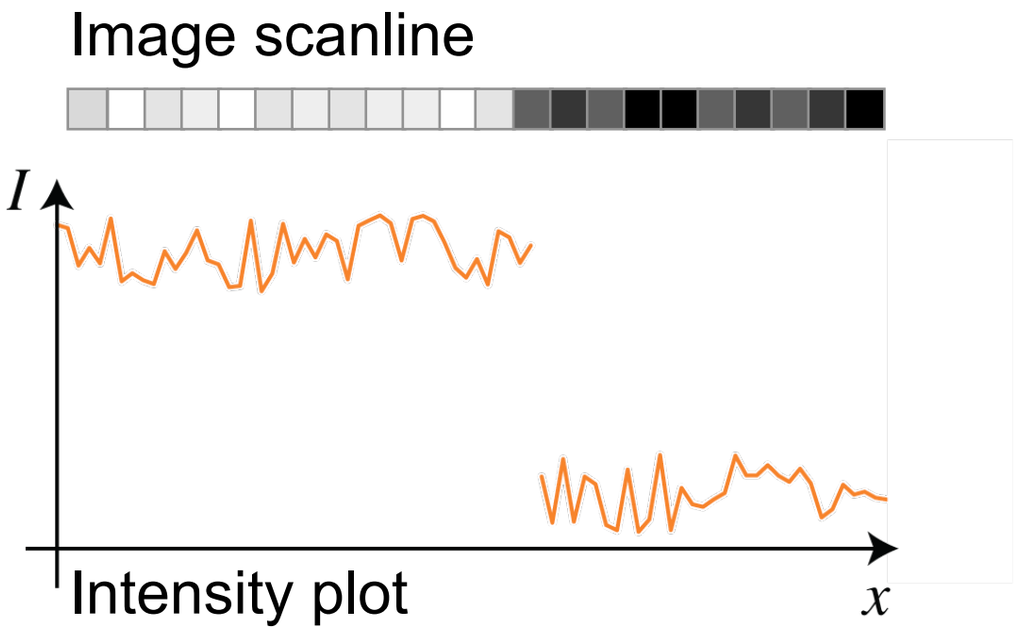
RECAP

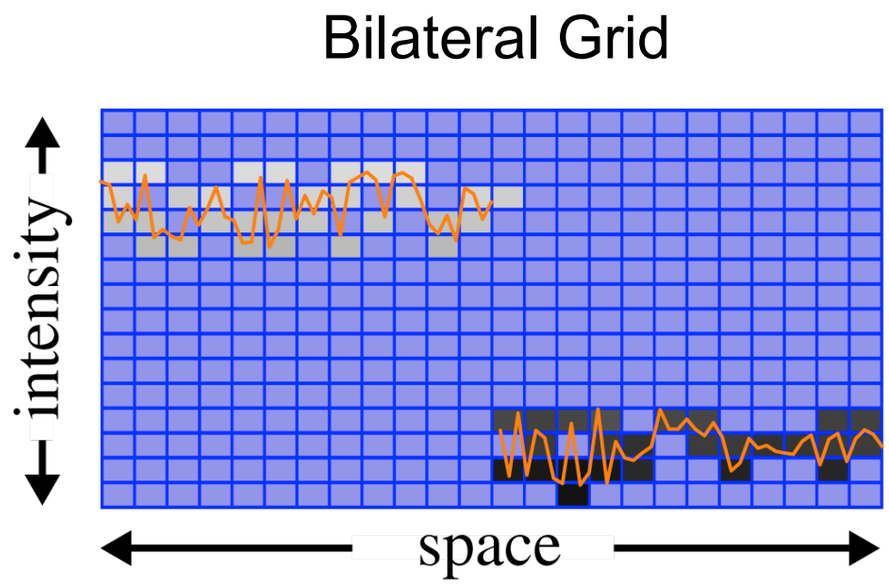
Bilateral Filter on the Bilateral Grid



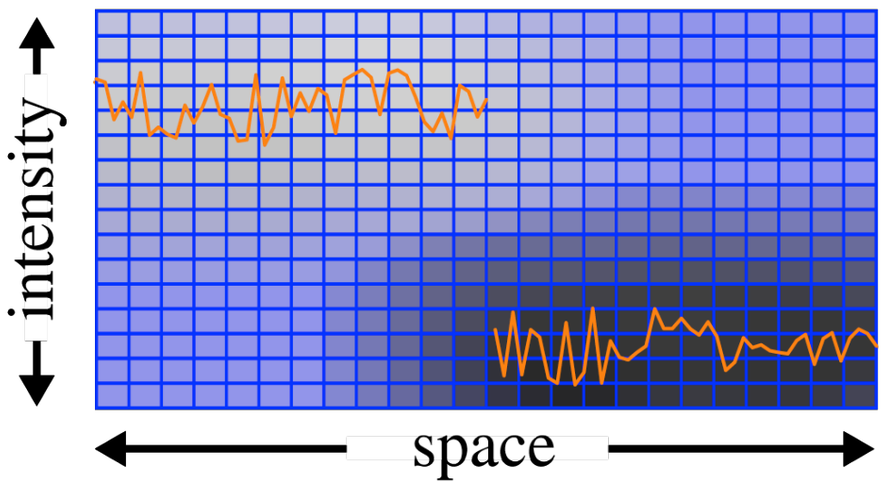
RECAP

Bilateral Filter on the Bilateral Grid

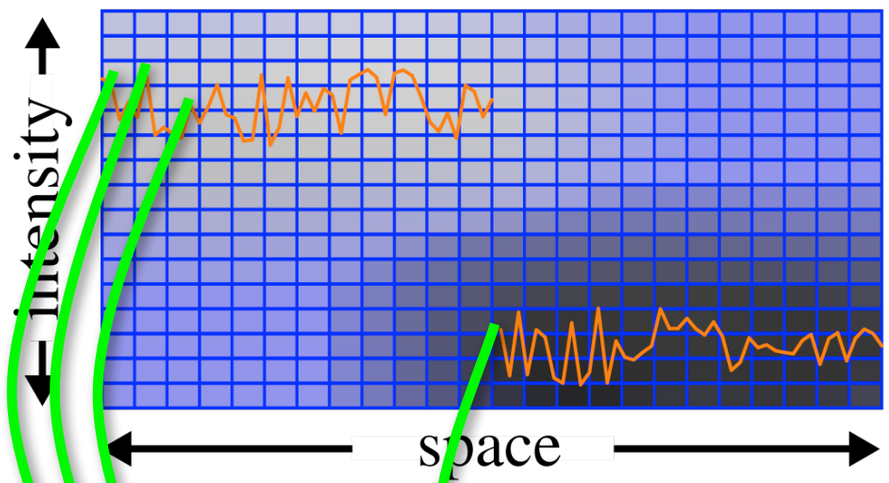




Blurred bilateral grid: both values & weights



Blurred bilateral grid



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



Filtered scanline

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
      return out
```

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

```
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
      return out
```

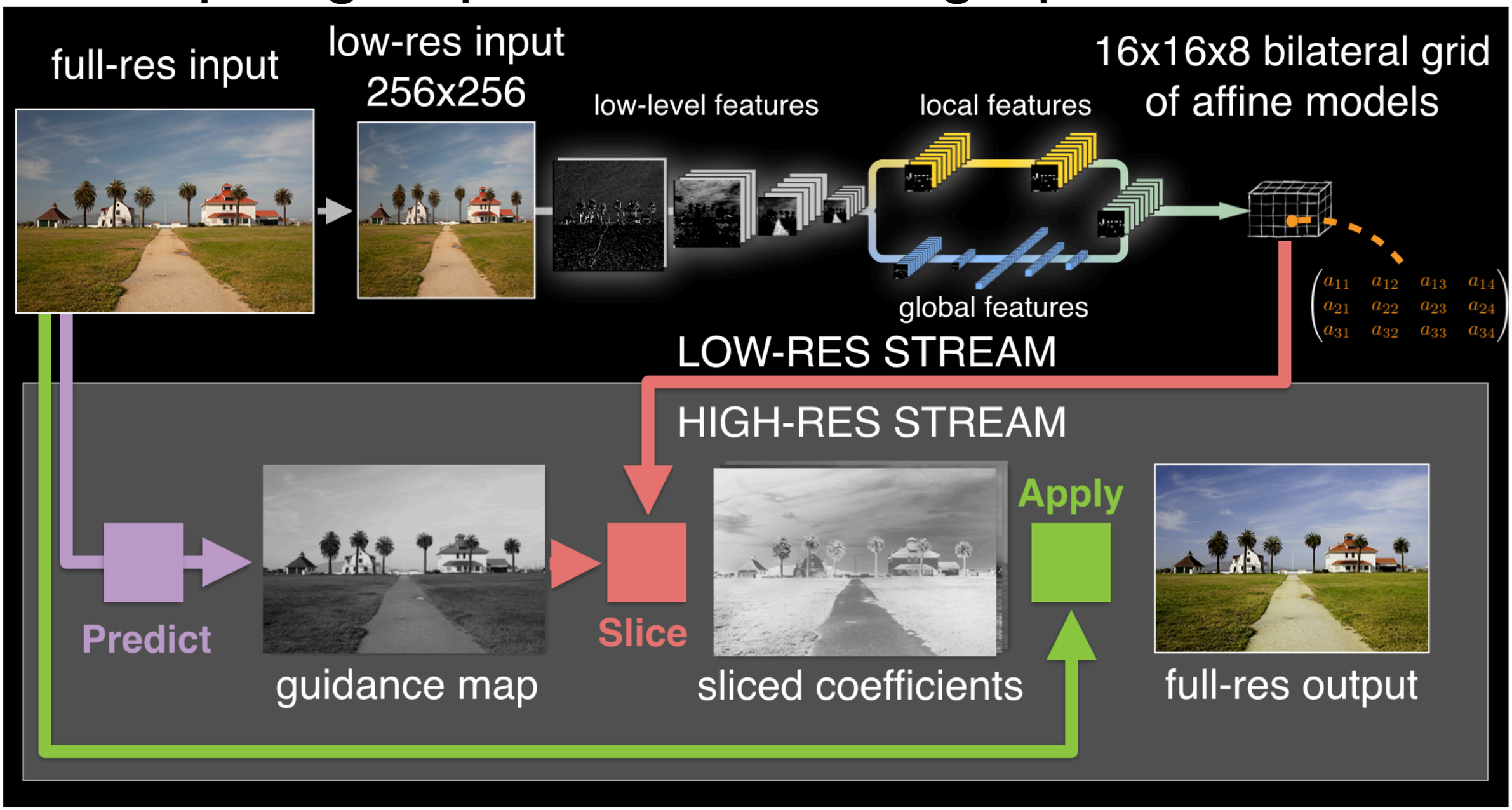
Bells and whistles

- 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

Deep learning with Bilateral grid

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



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

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/

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 4
  - 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.

Simple case

- Two observations x & y of the same quantity
  - with given variances $\sigma^2[x]$ $\sigma^2[y]$
- 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

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

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 dark pixels
- **Read noise**
  - Constant variance
  - Dominates for dark pixels
- **For a pixel value x:**

$$\sigma^2[x] \approx ax + \sigma^2_{read}$$

  - where *a* and σ^2_{read} depend on the camera and ISO

Optimal weights

- Recall irradiance formula

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

- and HDR merging formula

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

Optimal weights

- Recall irradiance formula

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

- and HDR merging formula

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

- 1/k amplifies signal and noise:

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

- replace I by irradiance

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

Variance of one scaled image

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

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

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

- 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

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

- Which gives us

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

- *a* and *L(x,y)* are constant per pixel and present both in the main sum and the normalization. Get rid of them
- the two ki in the main sum cancel each other

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

Improved weight maps

- New formula

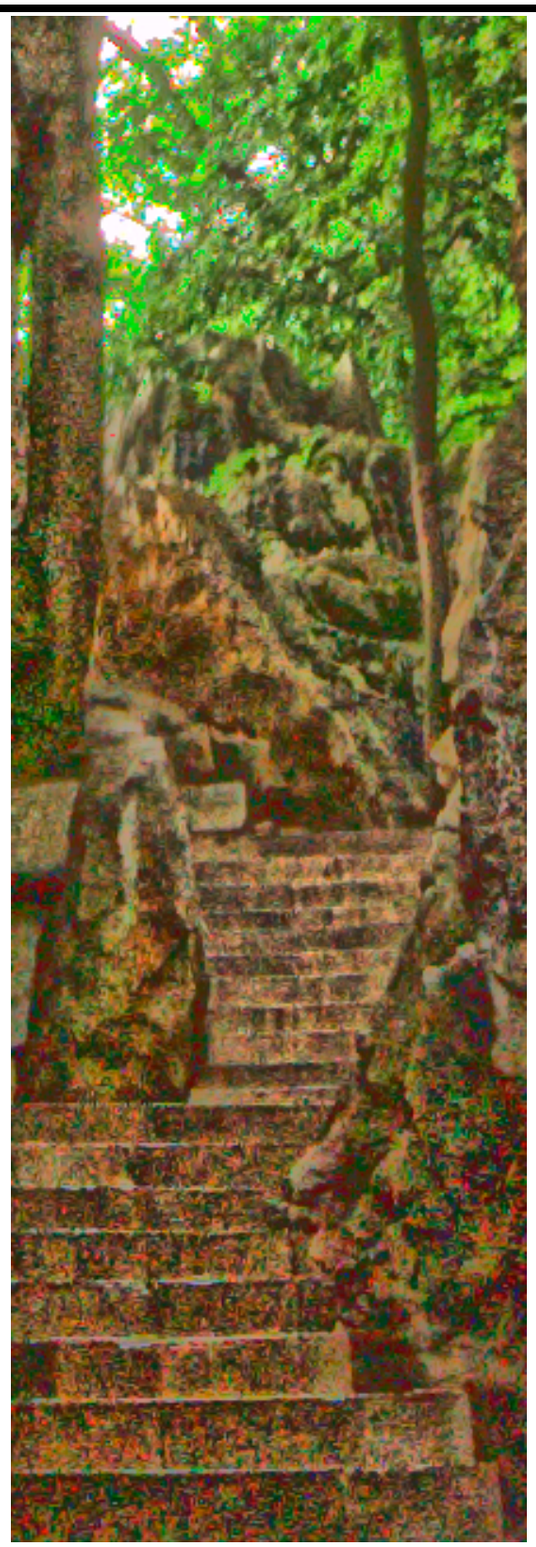
$$out(x,y) = \frac{1}{\sum w_i(x,y)k_i} \sum w_i(x,y) I_i(x,y)$$

  - The radiant power reaching the pixel has disappeared. All 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

Results



Naive weights



Weighted by 1/ki

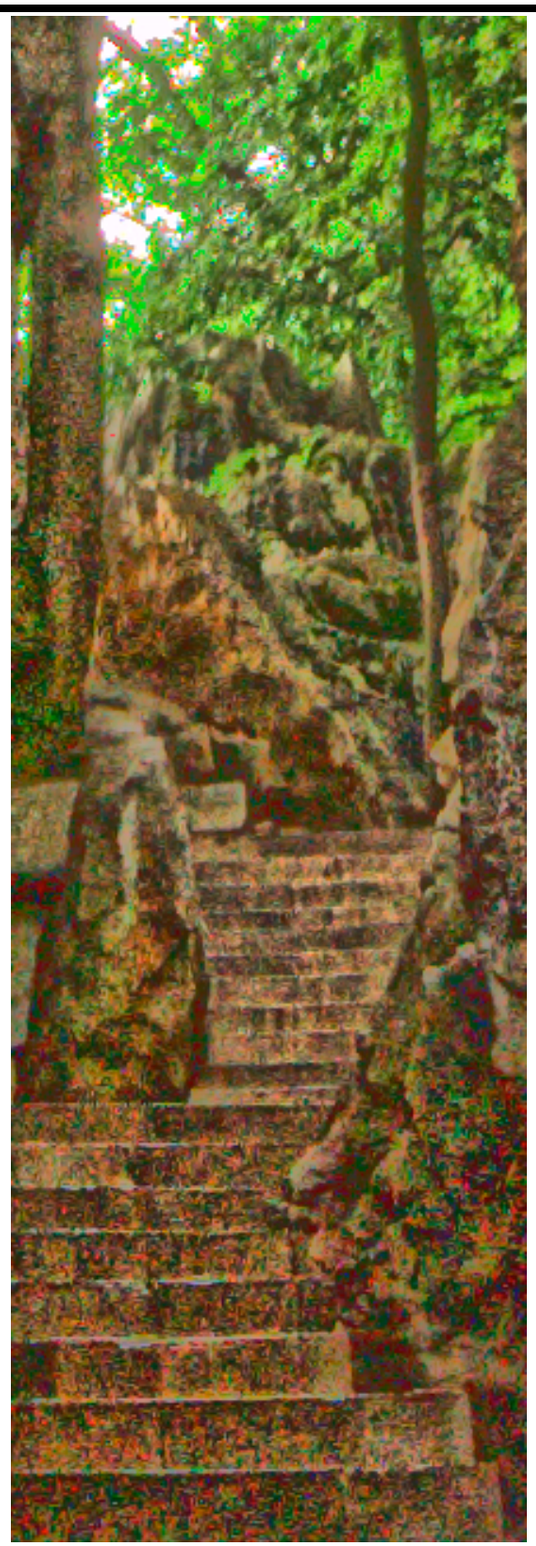




Questions?



Naive weights



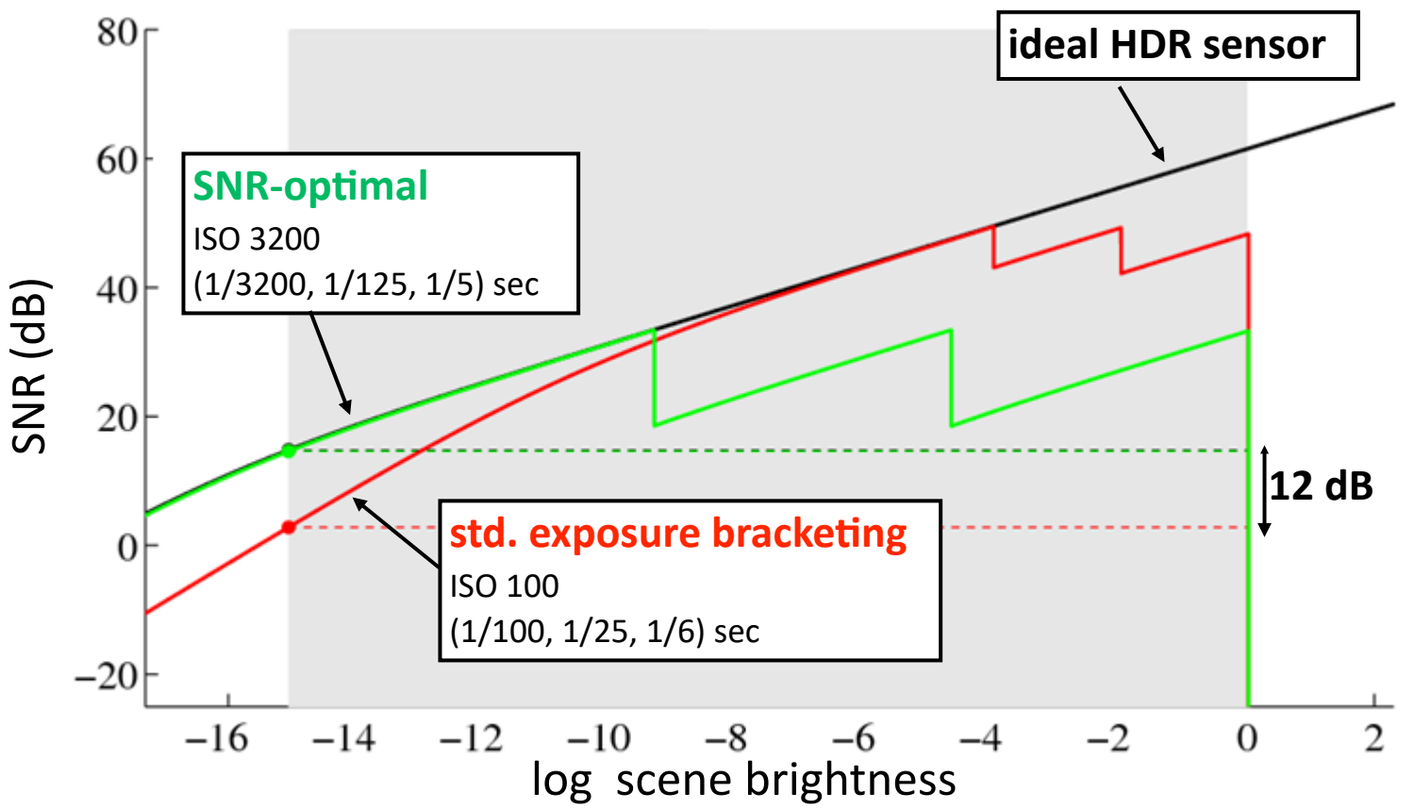
Weighted by 1/ki





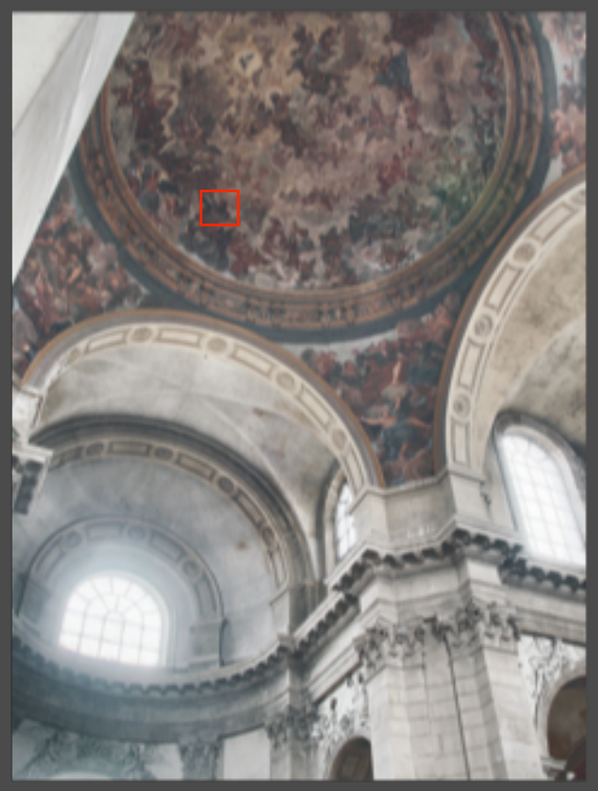
References

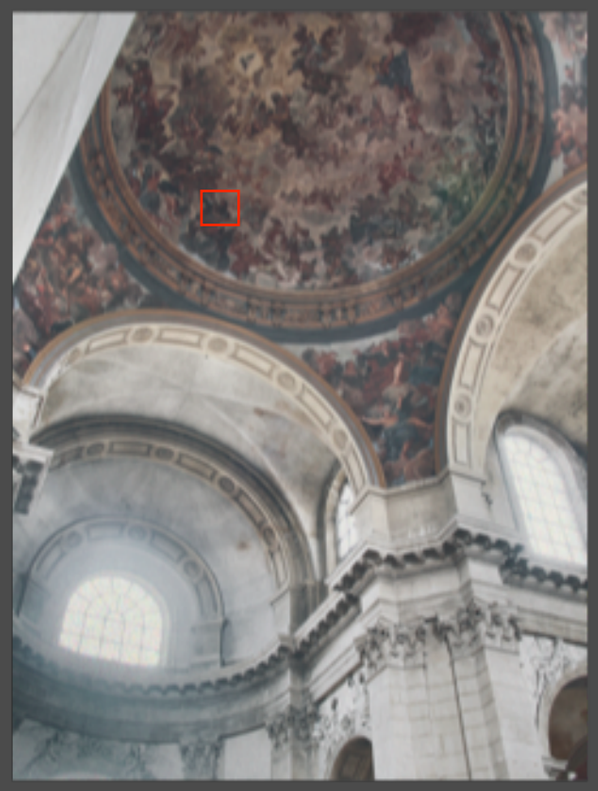
- http://www.macs.hw.ac.uk/bmvc2006/papers/372.pdf
- http://people.csail.mit.edu/hasinoff/hdrnoise/
  - full noise model
  - exploit ISO
  - Also optimizes the set of exposures

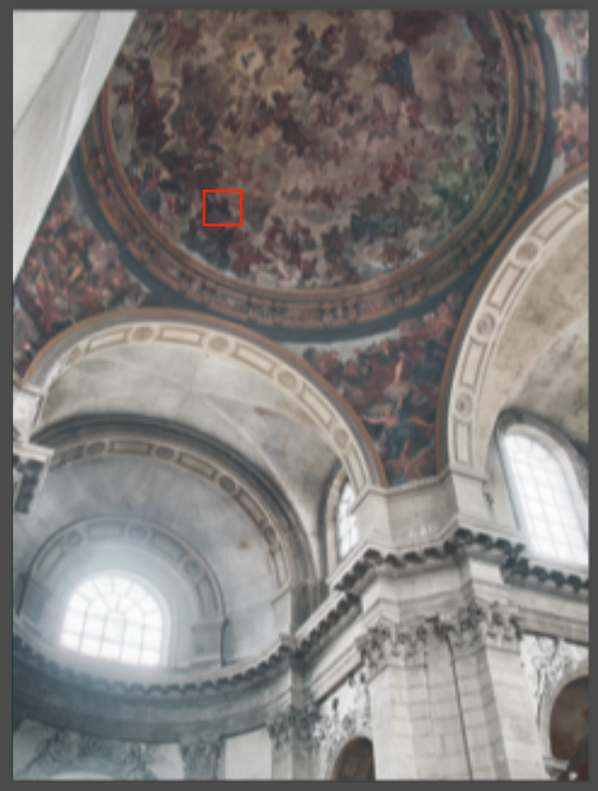


Results: “Nancy Church” (simulation)

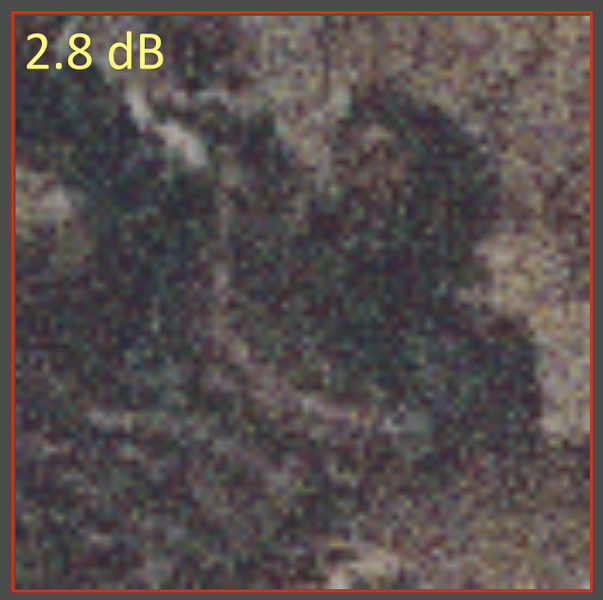
tone-mapped merged HDR







dark region







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

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

standard exposure bracketing

our SNR-optimal sequence

ground truth

Slide by Sam Hasinoff

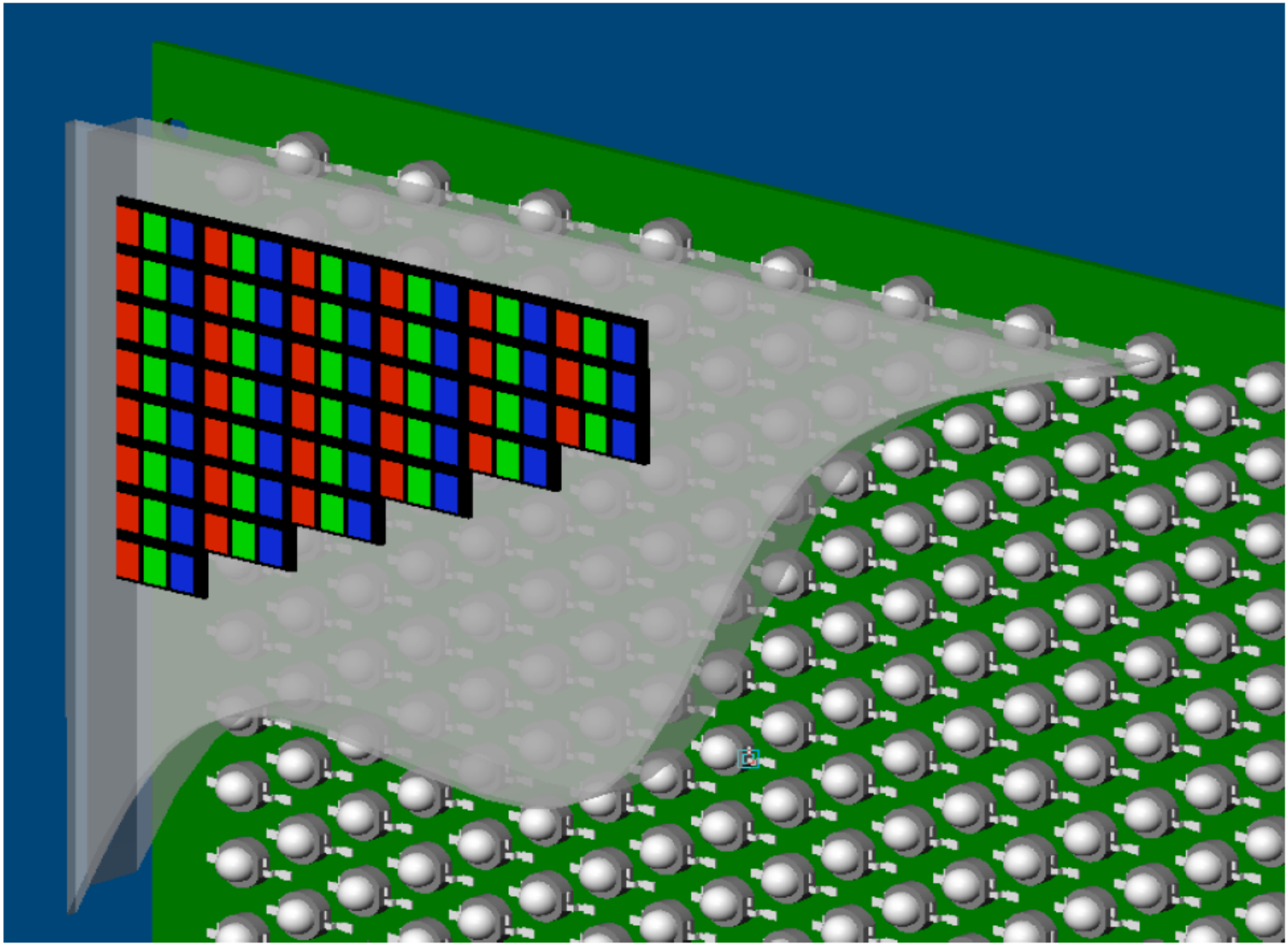
HDR display

Exception: Sunnybrook HDR display

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



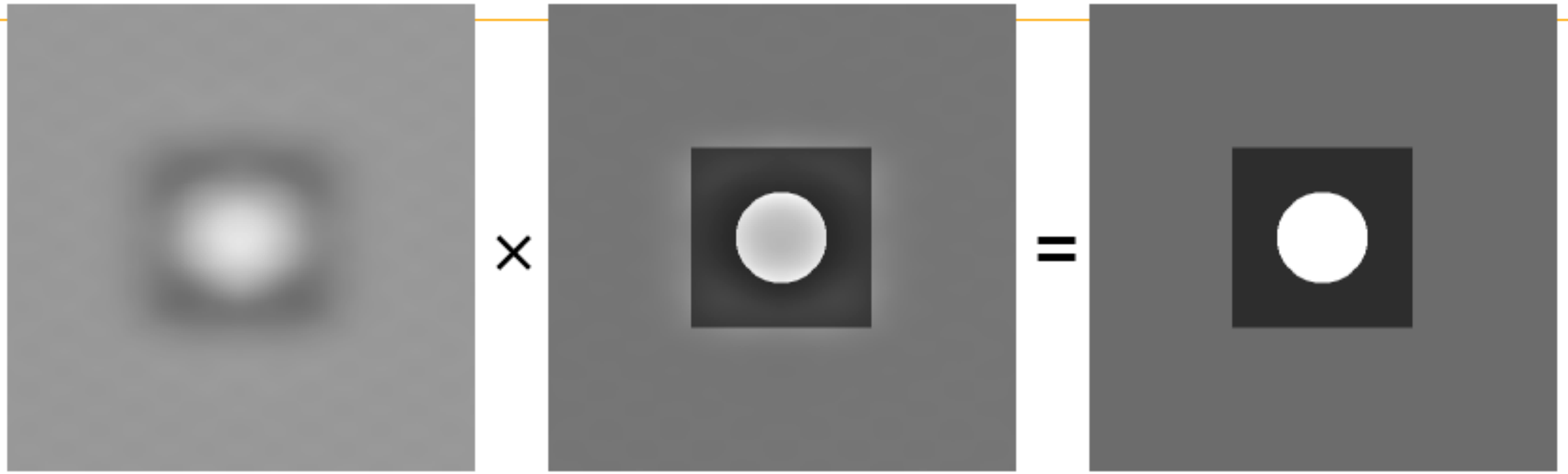
Slide from the 2005 Siggraph course on HDR



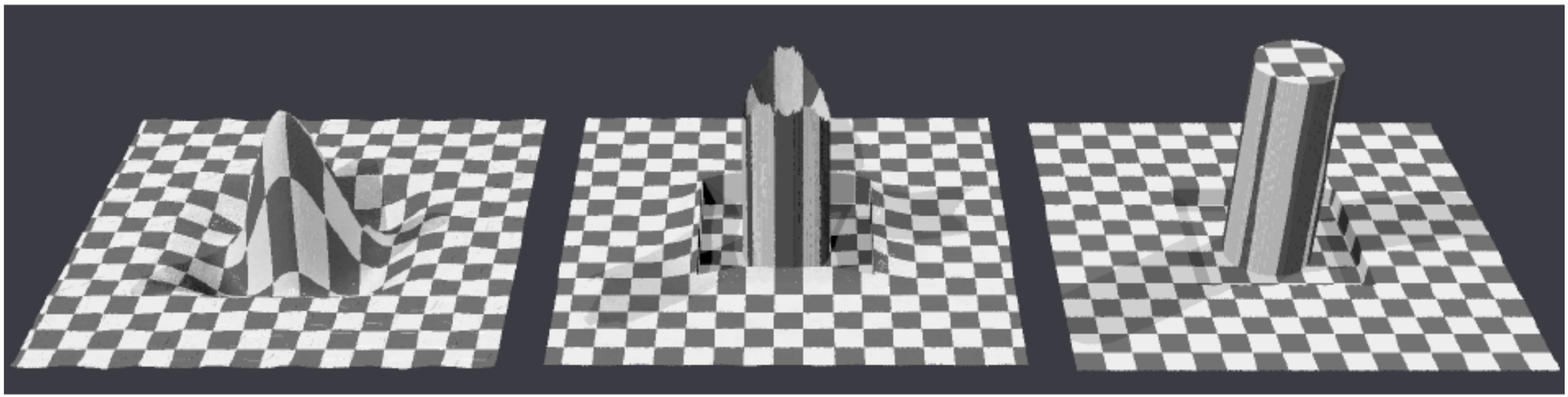
Slide from the 2005 Siggraph course on HDR

How It Works

SIGGRAPH2005



LED Backlight × LCD Screen = Combined Result



Slide from the 2005 Siggraph course on HDR

Production 37” HDR Display

SIGGRAPH2005



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/

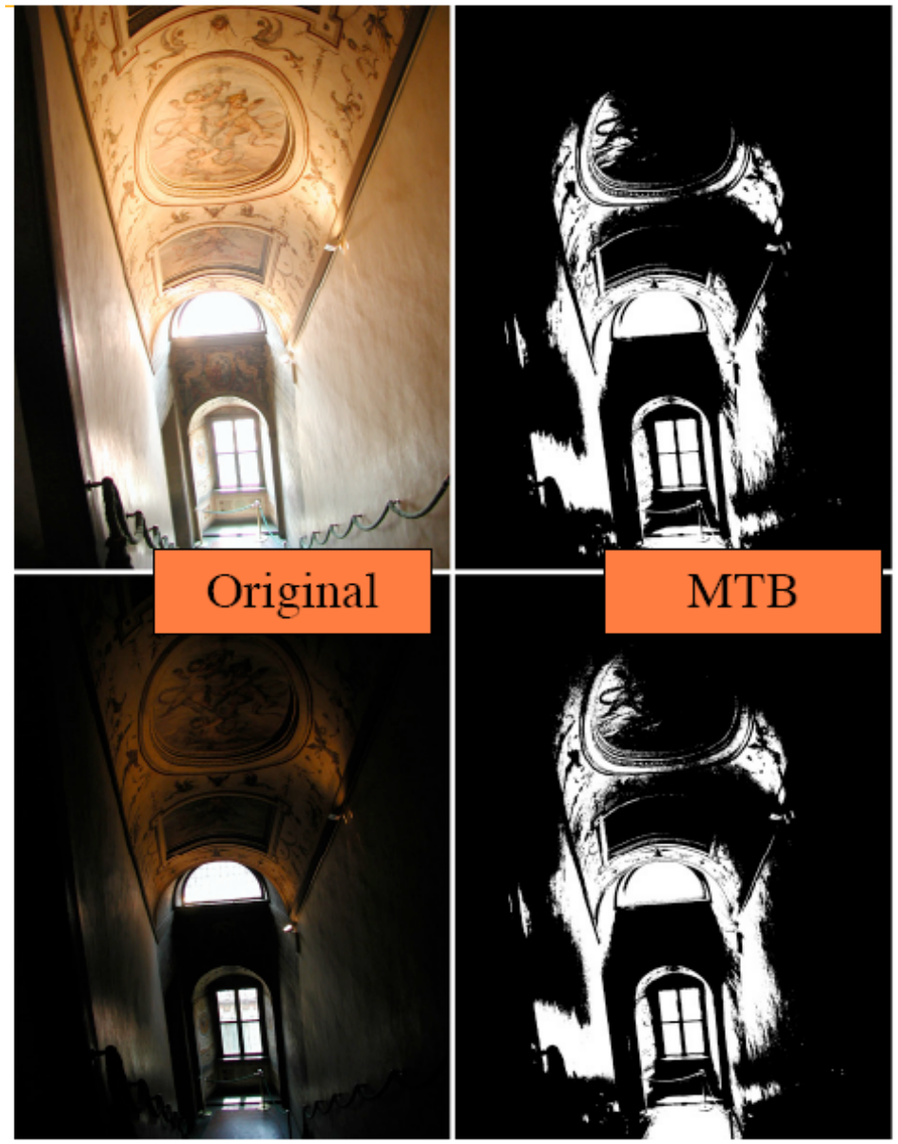
- **Image registration (no need for tripod)**
- **Lens flare removal**
- **Ghost removal**



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**





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

Automatic “Ghost” Removal

SIGGRAPH2005



Before

After

Slide from Siggraph 2005 course on HDR

Variance-based Detection

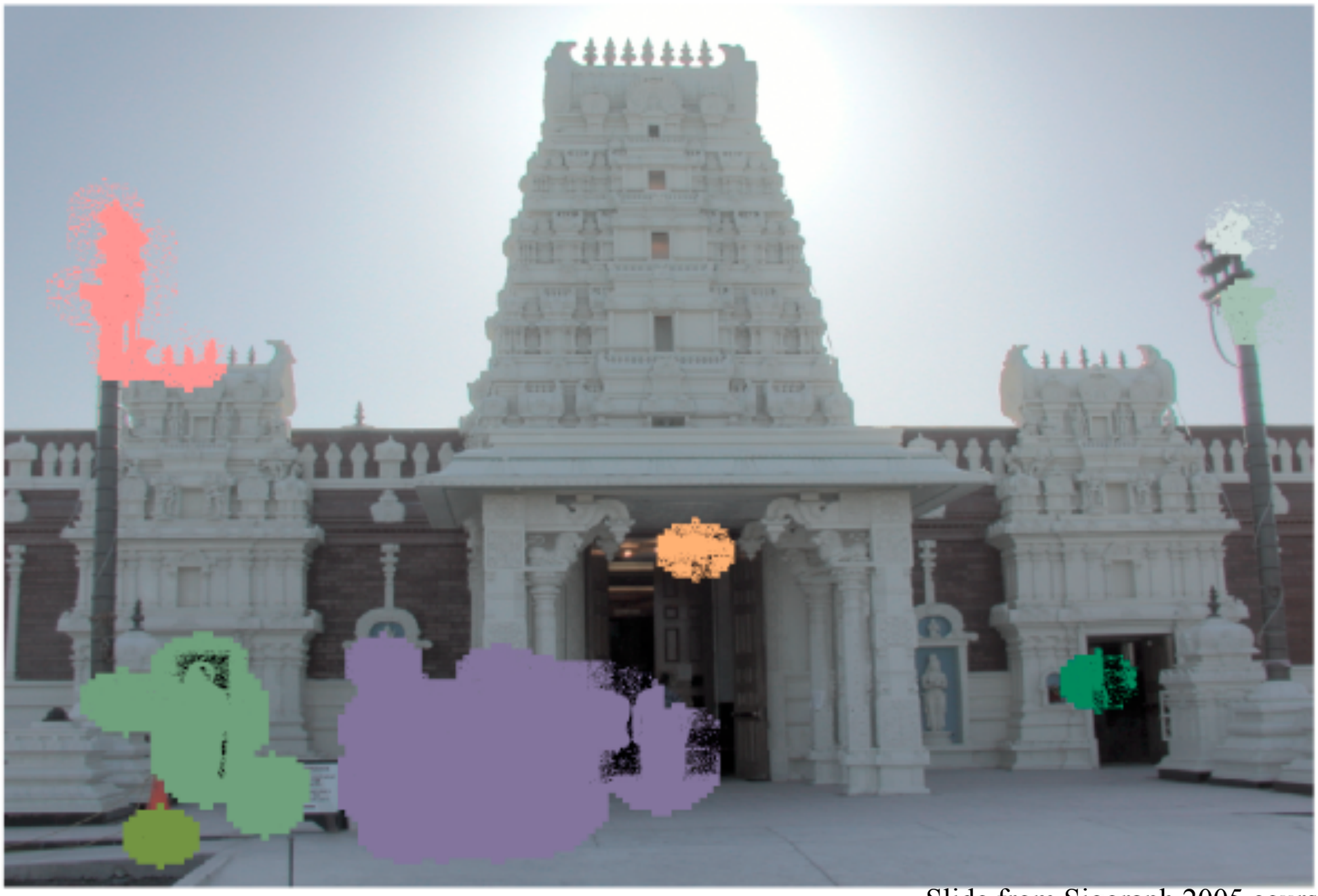
SIGGRAPH2005



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

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

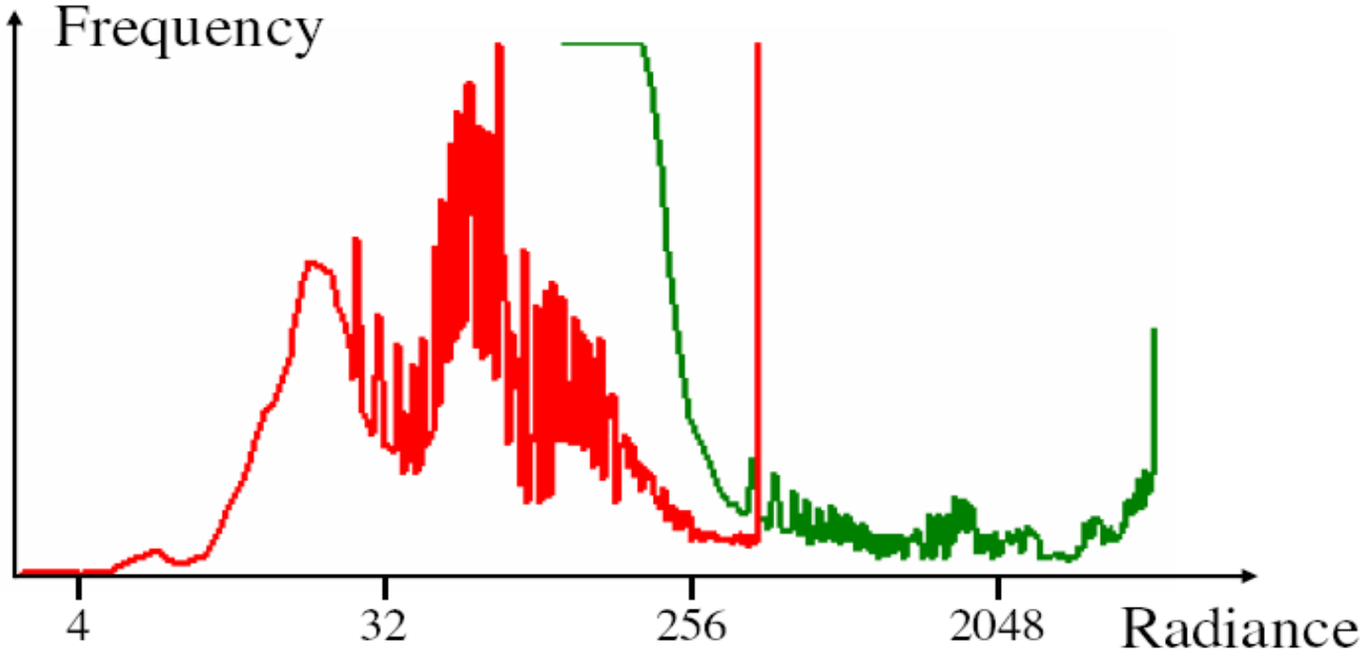




Figure 3: Two input exposures from the driving video. *The radiance histogram is shown on top. The red graph goes with the long exposure frame (bottom left), while the green graph goes with the short exposure frame (bottom right). Notice that the combination of these graphs spans a radiance range greater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

- **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**
-