Better HDR and the Bilateral Grid

Fredo Durand
MIT EECS 6.815/6.865
Limitations of the medium

- Flatness
- Finite size, frame
- Unique viewpoint
- Static
- Contrast and gamut

Notion pioneered by H. von Helmholtz
Merging HDR

\[ l_i(x, y) = \text{clip}(k_i \ L(x,y) + n) \]

- For each pixel
  - figure out which images are useful
  - scale values appropriately according, ideally, to \( k_i \)
  - Voila!
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

$$out(x, y) = \frac{1}{\sum w_i(x, y)} \sum w_i(x, y) \frac{1}{k_i} I_i(x, y)$$
HDR display
Exception: Sunnybrook HDR display

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

LED Backlight × LCD Screen = Combined Result

Slide from the 2005 Siggraph course on HDR
Production 37” HDR Display
Questions?
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
- 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

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

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

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

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

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

• Read noise
  - Constant variance
  - Dominates for dark pixels

• For a pixel value $x$:
  \[
  \sigma^2[x] \approx ax + \sigma^2_{\text{read}}
  \]
  - where $a$ and $\sigma^2_{\text{read}}$ depend on the camera and ISO
Optimal weights

• Recall irradiance formula

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

• and HDR merging formula

\[ \text{out}(x, y) = \frac{1}{\sum w_i(x, y)} \sum w_i(x, y) \frac{1}{k_i} I_i(x, y) \]
Optimal weights

- Recall irradiance formula
  \[ I_i(x, y) = \text{clip}(k_i L(x,y) + n) \]
- and HDR merging formula
  \[
  \text{out}(x, y) = \frac{1}{\sum \omega_i(x, y)} \sum \omega_i(x, y) \frac{1}{k_i} I_i(x, y)
  \]
- \(1/k\) amplifies signal and noise:
  \[
  \sigma^2 \left[ \frac{1}{k_i} I_i(x, y) \right] = \frac{1}{k_i^2} \left[ aI_i(x, y) + \sigma_{\text{read}}^2 \right]
  \]
- replace I by irradiance
  \[
  \sigma^2 \left[ \frac{1}{k_i} I_i(x, y) \right] = \frac{1}{k_i^2} \left[ ak_i L(x, y) + \sigma_{\text{read}}^2 \right]
  \]
Variance of one scaled image

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

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

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

- 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

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

• Variance per pixel per image:

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

• replace \( w_i \) by \( w'i \)
  - still use \( w_i \) 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

\[
w'_i(x, y) = \frac{w_i(x, y)}{a L(x, y)}
\]

• Which gives us

\[
out(x, y) = \frac{1}{\sum w_i(x, y) \frac{1}{k_i}} \sum w_i(x, y) \frac{I_i(x, y)}{a L(x, y) k_i}
\]

- \(a\) and \(L(x,y)\) are constant per pixel and present both in the main sum and the normalization. Get rid of them
- the two \(k_i\) 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

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

- 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.
- \( k_i \) 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 \( k_i \) and \( 1/k_i \) used to both appear.
Results

Naive weights

Weighted by $1/ki$
Questions?

Naive weights

Weighted by $1/ki$
References

- http://people.csail.mit.edu/hasinoff/hdrnoise/
  - full noise model
  - exploit ISO
  - Also optimizes the set of exposures
Results: “Nancy Church” (simulation)

- **ISO 100**: (1/100, 1/25, 1/6) sec
  - Standard exposure bracketing: 2.8 dB

- **ISO 3200**: (1/3200, 1/125, 1/5) sec
  - Our SNR-optimal sequence: 14.6 dB

- **Ground truth**

Slide by Sam Hasinoff

Thursday, March 8, 12
Response

curve
Response curve

• So far, we have assumed a mostly linear image

\[ l_i(x, y) = \text{clip}(k_i L(x, y) + n) \]

• Makes it easy to translate pixels values into a common reference range:
  scene irradiance \( L(x, y) \)
  - at least up to a global scale

• But often, images have received a non-linear response curve
  - especially if you get JPEG
Response curve

- To make better use of dynamic range
  - compress dark & bright tones, preserve more contrast in the middle
- Prettification

Density

\[ \text{max density} \]

\[ \text{shoulder region} \]

\[ \text{Linear region}\]

\[ \text{slope: gamma} \]

Log exposure
Some response curves

- **Nikon D2X**: Auto Tone
- **Canon EOS 20D**: Parameter 1
- **Canon EOS-1Ds Mark II**: Standard
- **Canon EOS 5D**: Standard Picture Style
Response curve

- Maps scene light intensity (radiance) into pixel color
- We are interested in the inverse response curve
If we know the response curve

• For each pixel
  – for each frame
    • if not black & not saturated, convert to absolute scene value
  – Take average if well-exposed in multiple frames
Questions?
But how do we get the curve?

- Easy when shooting raw (linear)
- Need calibration otherwise
- We’ll study Debevec and Malik’s algorithm
Calibrating the response curve

• Basic idea:
  measure different scene luminance levels and observe pixel value

• How do we vary the luminance reaching the sensor?
Calibrating the response curve

• Two solutions
  – Vary scene luminance
    • Have a calibrated light source with an intensity knob
  – Vary exposure (shutter speed) and observe pixel value for one fixed scene luminance
    • Exposure (e.g. shutter speed) linearly scales luminance
    • But cannot vary exposure more finely than by 1/3 stop
Calibrating the response curve

• Best of both:
  – Vary exposure
  – Exploit the large number of pixels corresponding to different scene luminance

• Challenge:
  – we don’t know the various scene luminances
Calibration Algorithm

Image series

- Given image series (pixel values Z)
  - know (relative) exposure level (shutter speed $\Delta t$)
- Unknown:
  - scene luminance at pixels (Radiance)
  - inverse response curve $f^{-1}$

Slide adapted from Alyosha Efros who borrowed it from Paul Debevec
$\Delta t$ don't really correspond to pictures. Oh well.
Calibration Algorithm

Image series

Pixel Value $Z = f(\text{Exposure})$

exposure: essentially # photons

Exposure = Radiance $\times \Delta t$

$log \text{ Exposure} = log \text{ Radiance} + log \Delta t$

Slide adapted from Alyosha Efros who borrowed it from Paul Debevec

Δt don't really correspond to pictures. Oh well.
Response curve

• Radiance is unknown

Assuming unit radiance for each pixel

\[
\log \text{Exposure} = \log \text{Radiance} + \log \Delta t
\]

Slide stolen from Alyosha Efros who stole it from Paul Debevec
Response curve

- Radiance is unknown, fit to find a smooth curve

Assuming unit radiance for each pixel

After adjusting radiances to obtain a smooth response curve

Slide stolen from Alyosha Efros who stole it from Paul Debevec

Thursday, March 8, 12
Inverse response curve $g$

- Discretize pixel values
  - Store corresponding exposure value
The math

- unknowns: response curve $f$ and radiance of pixels
- for each pixel $i$ and image $j$
  - Pixel Value $Z_{ij} = f(\text{Exposure}_{i,j})$
  - $\log \text{Exposure} = \log \text{Radiance}_i + \log \Delta t_j$
- Easier to deal with inverse function (in log) $g = \log (f^{-1})$

$$\log \text{Radiance}_i + \log \Delta t_j = g(Z_{ij})$$
The math

• Unknowns:
  – inverse response curve $g$
    • represented by discrete values indexed by pixel value
  – log radiance of pixels

\[ \log \text{Radiance}_i + \log \Delta t_j = g(Z_{ij}) \]

• We have $\# \text{pixels} + \# \text{discrete curve values}$
  & $\# \text{pixels} \ast \# \text{images}$ equations

\[ \text{known} \Rightarrow \text{index to a given discrete value of } g \]
Improvement

• We know that response curves are continuous
  – Force \( g \) to be smooth
• very general optimization technique, called “regularization”
• In our case, force the second derivative to be small
  – minimize \( [g(z)-0.5(g(z-1)+g(z+1))]^2 \)

Pixel value \( z \)

exposure value (log)
The Math

- For each pixel site $i$ in each image $j$, want:
  \[
  \log \text{Radiance}_i + \log \Delta t_j = g(Z_{ij})
  \]
  + small $g''$

- Solve the overdetermined linear system:
  \[
  \sum_{i=1}^{N} \sum_{j=1}^{P} \left[ \log \text{Radiance}_i + \log \Delta t_j - g(Z_{ij}) \right] + \lambda \sum_{z=Z_{\text{min}}}^{Z_{\text{max}}} g''(z)^2
  \]

Slide stolen from Alyosha Efros who stole it from Paul Debevec

Thursday, March 8, 12
Matlab code

- As usual with Matlab, goal is to setup a big linear equation $Ax=b$
  - Where $x$ is our response curve concatenated with radiance values
  - Then solve (under least square) using \n
- You can do the same with numpy
Matlab code

• As usual with Matlab, goal is to setup a big linear equation \( Ax=b \)
  – Where \( x \) is our response curve concatenated with radiance values
  – Then solve (under least square) using \( \backslash \)

• \( A \) has width \#pixels+\#curve samples (unknowns)
• \( A \) has height \#pixels*\#frames for the data-fitting equation (each pixel in each frame)
  + \( n+1 \) for the smoothness terms
• Pixels can be weighted if the user wants
Matlab code

% gsolve.m - Solve for imaging system response function
%
% Given a set of pixel values observed for several pixels in several
% images with different exposure times, this function returns the
% imaging system’s response function g as well as the log film irradiance
% values for the observed pixels.
%
% Assumes:
%
%  Zmin = 0
%  Zmax = 255
%
% Arguments:
%
%  Z(i,j) is the pixel values of pixel location number i in image j
%  B(j)  is the log delta t, or log shutter speed, for image j
%  l     is lambda, the constant that determines the amount of smoothness
%  w(z)  is the weighting function value for pixel value z
%
% Returns:
%
%  g(z)   is the log exposure corresponding to pixel value z
%  lE(i)  is the log film irradiance at pixel location i
%
from Debevec and Malik 1997
function [g,lE]=gsolve(Z,B,l,w)

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

%% Include the data-fitting equations

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

%% Fix the curve by setting its middle value to 0
A(k,129) = 1;
k=k+1;

%% Include the smoothness equations

for i=1:n-2
    A(k,i)=l*w(i+1); A(k,i+1)=-2*l*w(i+1); A(k,i+2)=l*w(i+1);
    k=k+1;
end

%% Solve the system using SVD

x = A\b;
g = x(1:n);
lE = x(n+1:size(x,1));
Result: digital camera

Kodak DCS460
1/30 to 30 sec

Recovered response curve

Pixel value

log Exposure

Slide stolen from Alyosha Efros who stole it from Paul Debevec
Reconstructed radiance map

Slide stolen from Alyosha Efros who stole it from Paul Debevec
Result: color film

- Kodak Gold ASA 100, PhotoCD

Slide stolen from Alyosha Efros who stole it from Paul Debevec
Recovered response curves

Slide stolen from Alyosha Efros who stole it from Paul Debevec
Recap

• Curve calibration
  – Take many images of static scene (1/3 stop)
  – Solve optimization problem

• HDR multiple-exposure merging
  – Take multiple exposures (e.g. every 2 stops)
  – (optional) align images
  – for each pixel, use picture(s) where properly exposed
    • use inverse response curve and exposure time
  – Output: one image where each pixel has full dynamic range, stored e.g. in float
  aka radiance map
The Radiance map

Slide stolen from Alyosha Efros who stole it from Paul Debevec
Bilateral Grid
Back to bilateral tone mapping

**CHALLENGE:** SPEED

Input HDR image

Intensity

Bilateral Filter

Large scale

Detail

Reduce contrast

Preserve!

Output

Large scale

Detail

Color

Color

Input HDR image

Intensity

Bilateral Filter

Large scale

Detail

Reduce contrast

Preserve!

Output

Large scale

Detail

Color

Color

Thursday, March 8, 12
Showing off

tone mapping took 4.61149096489 seconds
Bilateral Filter: Weighted Average of Pixels

- Depends on spatial distance and intensity difference
  - Pixels across edges have almost no influence

\[
BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(I_p - I_q) I_q
\]
Review: Gaussian (Bell curve)

\[ G_\sigma(x) = e^{-\frac{x^2}{2\sigma^2}} \]

- \( \sigma \) (standard deviation) determines width
- Can be normalized
  - here, to be 1 at 0
  - or to make area under curve 1 (multiply by \( \frac{1}{\sigma\sqrt{2\pi}} \))
- Nice smooth way to have high influence around center and then decrease rapidly beyond
Brute-force Implementation of Bila.

\[ BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(\| I_p - I_q \|) I_q \]

For each pixel \( p \)
For each pixel \( q \)
Compute

\[ G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(\| I_p - I_q \|) I_q \]

**VERY SLOW!**

More than 10 minute per image
Questions?
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

Downsampling & Simple blur
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
Recall other view

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

Image scanline

Intensity plot

Bilateral Grid

Intensity plot

space
Bilateral Filter on the Bilateral Grid

Image scanline

Intensity plot

Blurred bilateral grid (3D blur)

Query grid with input image

Filtered scanline
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 $\sigma_s$ and $\sigma_r$
    ($\sigma$ 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 \( \sigma_s \) and \( \sigma_r \))

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

where \([\cdot]\) 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:

\[
\begin{align*}
  b(x, y, i) &= \sum_{x'} G_{\sigma_s}(x - x')v(x, y, i) \\
  &\text{If we have chosen sf = 1 cell width, then the Gaussian is simply [1 4 6 4 1]/16}
\end{align*}
\]
Why are Gaussians separable?

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

\[ e^{-\frac{y^2}{2\sigma}} \ast e^{-\frac{x^2}{2\sigma}} = e^{-\frac{x^2+y^2}{2\sigma}} \]

- gives you a radial Gaussian
Blurring

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

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

\[ k(x) = \sum_{\xi} f(x, \xi) \ g(I(\xi) - I(x)) \]
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:

$\left(1-x'+x\right)v(x, y)+\left(x'-x\right)v(x+1, y)$
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
Bilateral Filter on the Bilateral Grid
Bilateral Filter on the Bilateral Grid

Image scanline

Intensity plot

I

Intensity

x

Bilateral Grid

intensity

space

Blurred bilateral grid: both values & weights

intensity

space

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 $x/\sigma$, $y/\sigma$, $I(x,y)/\sigma$
    add 1 to weight of grid cell $x/\sigma$, $y/\sigma$, $I(x,y)/\sigma$

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($x/\sigma$, $y/\sigma$, $I(x,y)/\sigma$) / weight($x/\sigma$, $y/\sigma$, $I(x,y)/\sigma$)
Questions?
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
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
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)+0.000000001
    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
Motion
Smarter HDR capture


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

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

Before

After
Variance-based Detection
Region Masking
Best Exposure in Each Region
Lens Flare Removal

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).
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.
Let’s look at Lightroom
HDR code

- HDRShop [http://gl.ict.usc.edu/HDRShop/](http://gl.ict.usc.edu/HDRShop/) (v1 is free)
- Photoshop CS2
- MPI PFScalibration (includes source code) [http://www.mpii.mpg.de/resources/hdr/calibration/pfs.html](http://www.mpii.mpg.de/resources/hdr/calibration/pfs.html)
- EXR tools [http://scanline.ca/exrtools/](http://scanline.ca/exrtools/)
- HDR Image Editor [http://www.acm.uiuc.edu/siggraph/HDRIE/](http://www.acm.uiuc.edu/siggraph/HDRIE/)
- Artizen HDR [http://www.supportingcomputers.net/Applications/Artizen/Artizen.htm](http://www.supportingcomputers.net/Applications/Artizen/Artizen.htm)
- Automated High Dynamic Range Imaging Software & Images [http://www2.cs.uh.edu/~somalley/hdri_images.html](http://www2.cs.uh.edu/~somalley/hdri_images.html)
HDR images

- [http://www.mpi-sb.mpg.de/resources/hdr/gallery.html](http://www.mpi-sb.mpg.de/resources/hdr/gallery.html)
- [http://www.openexr.com/samples.html](http://www.openexr.com/samples.html)
- [http://www.flickr.com/groups/hdr/](http://www.flickr.com/groups/hdr/)
- [http://www2.cs.uh.edu/~somalley/hdri_images.html#hdr_others](http://www2.cs.uh.edu/~somalley/hdri_images.html#hdr_others)
- [http://www.cs.utah.edu/~7Ereinhard/cdrom/hdr.html](http://www.cs.utah.edu/~7Ereinhard/cdrom/hdr.html)
- [http://www.sachform.de/download_EN.html](http://www.sachform.de/download_EN.html)
- [http://lcavwww.epfl.ch/%7EElmeylan/HdrImages/February06/February06.html](http://lcavwww.epfl.ch/%7EElmeylan/HdrImages/February06/February06.html)
- [http://lcavwww.epfl.ch/%7EElmeylan/HdrImages/April04/april04.html](http://lcavwww.epfl.ch/%7EElmeylan/HdrImages/April04/april04.html)
HDR photography

- http://www.cambridgeincolour.com/tutorials/high-dynamic-range.htm