Fast Bilateral Filtering & Applications

Frédo Durand
MIT - EECS

Many slides by
Sylvain Paris & Jiawen Chen
Demosaicking pset

• Don’t forget to convert to double
• See how to avoid loops on web page
Recap: HDR imaging

• Multiple-exposure HDR capture
  – calibrate response curve
  – combine multiple exposures

• Bilateral tone mapping
  – decompose luminance into large scale & detail
    • use bilateral filter
  – reduce contrast of large scale only
    • preserve detail
    • preserve colors
Alternative: exposure fusion

- One single step for both multiple-exposure merging & tone mapping
  - http://research.edm.uhasselt.be/~tmertens/exposure_fusion/

Figure 2. Exposure fusion is guided by weight maps for each input image. A high weight means that a pixel should appear in the final image. These weights reflect desired image qualities, such as high contrast and saturation. Image courtesy of Jacques Joffre.
Back to bilateral tone mapping

Input HDR image

Intensity

Bilateral Filter

Large scale

Detail

Reduce contrast

Preserve!

Large scale

Detail

Color

Input HDR image

Output

Large scale

Detail

Color
Bilateral Filter: Weighted Average of Pixels

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

Mathematical expression:

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

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

8 megapixel photo: 64,000,000,000,000 iterations!

VERY SLOW!

More than 10 minute per image
Better Brute-force Implementation

Idea: Far away pixels are negligible, truncate the Gaussian

– usually truncate at $3\sigma$ or $4\sigma$

For each pixel $p$

– For each pixel $q$ such that $\| p - q \| < constant \times \sigma_s$
Discussion

- Complexity: $\tilde{I} \left( |S| \times \sigma_s^2 \right)$

- Fast for small kernels: $\sigma_s \sim 1$ or $2$ pixels

- BUT: slow for larger kernels
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
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
Bilateral Filter on the Bilateral Grid

Image scanline

Intensity plot

Bilateral Grid

Blurred bilateral grid

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 [ ] 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 \([1 \ 4 \ 6 \ 4 \ 1]/16\)
• JUSTIFY
SEPARABLE
Blurring

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

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

\[
k(x) = \sum_{\xi} f(x, \xi) g(I(\xi) - I(x))
\]
[Tomasi and Manduchi 1998]

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

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

Image scanline

Bilateral Grid

Intensity plot
Bilateral Filter on the Bilateral Grid

Image scanline

Bilateral Grid

Intensity plot

Blurred bilateral grid: both values & weights

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)
Questions?
brute-force implementation
bilateral grid
visually similar
Performance

Image size: 2 MPixels

- Brute force: 10 minutes
- CPU Bilateral grid: 1 second
- GPU bilateral grid
  - 2004 card (NV40): 28 ms (36 Hz)
  - 2006 card (G80): 9 ms (111 Hz)
Figure 4: Bilateral filter running times as a function of the image size (using $\sigma_s = 16$ and $\sigma_r = 0.1$). The memory requirements increase linearly from 625 kB at 1 megapixel to 6.25 MB at 10 megapixels.
Discussion

• Complexity: \[ I \left( |S| + \frac{|S||R|}{\sigma_s^2 \sigma_r^2} \right) \]

• Fast for medium and large kernels
  – Can be ported on Graphics Processing Units (graphics cards) [Chen 07]: always very fast

• Can be extended to color images but slower

• Visually similar to brute-force computation

Tuesday, October 27, 2009
Surprising behavior

• Faster when spatial footprint gets bigger
  • Because we can downsample more aggressively
Real-Time Bilateral Filtering using the Bilateral Grid
More bilateral grid operations
Edge-aware brush

• Classical paint brush
  • Ignores edges

• Our edge-aware brush
  • Respects edges
Bilateral Grid Painting

• When mouse is held down, paint only at intensity level of initial mouse click
Scribble-based Selection

- User scribbles to specify selection [Lischinski 06]
- Piecewise-smooth interpolation to get full selection
  - Respects intensity discontinuities

Input image with scribbles

Input image

Our interpolated selection

Tuesday, October 27, 2009
Scribble-based Selection

Image scanline

Hard constraints in grid

Bilateral Grid

Intensity

Space
Scribble-based Selection

Hard constraints in grid

Smooth interpolation

Slice: query grid with input image

Image scanline

More about the use of scribbles & optimization next week

Tuesday, October 27, 2009
Many Operations and Applications

- Local histogram equalization
- Interactive tone mapping
- Video abstraction [Winnemoller 06, DeCarlo 02]
- Photographic style transfer [Bae 06]
Questions?
Discussion

• Respects luminance edges
• Color bilateral grid would be 5D
  - high memory cost
  - Might not fit on current graphics hardware
  - Luminance edges are often sufficient

• Grid resolution depends on the operator
  - E.g., for edge-aware brush:
    space sampling rate ~ brush radius
    intensity sampling rate ~ edge-awareness
Summary: the Bilateral Grid

• 3D representation for 2D data

• Intelligent downsampling

• Many edge-aware operations
  • Painting, scribble interpolation, bilateral filter, local histogram equalization

• Real-time for HD video
Refs

• A Gentle Introduction to Bilateral Filtering and its Applications
  – http://people.csail.mit.edu/sparis/bf_course/


  – http://groups.csail.mit.edu/graphics/bilagrid/

• Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. Frédo Durand and Julie Dorsey. SIGGRAPH 2002
Questions ?
Alternative

- Merge exposures & tone map in one single step
- Burt
• Another acceleration technique
Box Kernel [Weiss 06]

- Bilateral filter with a square box window [Yaroslavsky 85]

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

- The bilateral filter can be computed only from the list of pixels in a square neighborhood.
Box Kernel [Weiss 06]

- Idea: fast histograms of square windows

- **input:** full histogram is known
- **update:** add one line, remove one line
Box Kernel [Weiss 06]

- Idea: fast histograms of square windows

**input:** full histograms are known

**update:** add one line, remove one line, add two pixels, remove two pixels
Discussion

- Complexity: \( \bar{I} (|S| \times \log \sigma_s) \)
  - always fast

- Only single-channel images

- Exploit vector instructions of CPU

- Visually satisfying results (no artifacts)
  - 3 passes to remove artifacts due to box windows (Mach bands)
brute-force implementation
box kernel
visually different,
yet no artifacts