Fast Bilateral Filtering & Applications

Frédo Durand
MIT - EECS

Many slides by Sylvain Paris & Jiawen Chen
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


---

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!

Output

Large scale

Detail

Color

Color

Tuesday, February 24, 2009
Weighted Average of Pixels

- Depends on spatial distance and intensity difference
  - Pixels across edges have almost influence

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

Tuesday, February 24, 2009
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 $\| p - q \| < constant \times \sigma_s$

looking at all pixels

looking at neighbors only
Discussion

- Complexity: \( \bar{I} \left( |S| \times \sigma_s^2 \right) \)
- Fast for small kernels: \( \sigma_s \approx 1 \) 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
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 v2.0: The product of spatial and intensity Gaussian defines a 3D Gaussian in x, y, I

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

Bilateral Grid

Intensity plot

Tuesday, February 24, 2009
Bilateral Filter on the Bilateral Grid

Image scanline

Bilateral Grid

Intensity plot

Query grid with input image

Blurred bilateral grid

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 number of pixels
  - empty cells shown in blue here
  - store a weight to keep track of number of pixels
  - 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 roughly a cell width)
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

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

where [] denotes integer truncation
Blurring the grid

- Same as in 2D
- Each cell replaced by Gaussian-weighted average of neighbors

\[ b(x, y, i) = \sum_{x', y', i'} G_{\sigma_f}(x - x') G_{\sigma_f}(y - y') G_{\sigma_g}(i - i') v(x, y, i) \]
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]/6
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 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 at integer \( x \) & \( y \)
• We want to reconstruct at real-valued \( x', y' \)
• Bilinear reconstruction:

\[
(1-y' + y)[(1-x' + x)v(x, y) + (x' - x)v(x+1, y)] +
(y' - y)[(1-x' + x)v(x, y+1) + (x' - x)v(x+1, y+1)]
\]
Bilateral Filter on the Bilateral Grid

Image scanline

Intenstity plot

Bilateral Grid
Bilateral Filter on the Bilateral Grid

- Image scanline
- Bilateral Grid
- Blurred bilateral grid
- Query grid with input image
- Filtered scanline

Tuesday, February 24, 2009
Pseudo code

For each pixel $x, y$

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

For each grid cell $x, y, i$

value = 0; weight = 0;
for each grid cell $x', y', i'$

value += value($x', y', i'$)
weight += weight($x', y', i'$)

For each pixel $x, y$

output = value($x, y, I(x,y)$)/weight($x, y, I(x,y)$)

---

Grid creation

Grid Blur

Slicing
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: $\hat{I} \left( |S| + \frac{|S| |R|}{\sigma_s^2 \sigma_r} \right)$

- 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
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
A Simple Illustration

- Classical paint brush
  - Ignores edges

- Our edge-aware brush
  - Respects edges

Stroke with classical brush

Stroke with bilateral brush
Bilateral Grid Painting

Image scanline

Bilateral Grid

intensity

space

Tuesday, February 24, 2009
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
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

Interpolated selection
Many Operations and Applications

• Local histogram equalization
• Interactive tone mapping
  ![Local histogram equalization](image1.png)

• Video abstraction
  [Winnemoller 06, DeCarlo 02]
  ![Video abstraction](image2.png)

• Photographic style transfer
  [Bae 06]
  ![Photographic style transfer](image3.png)
Discussion

• Respects luminance edges
• Color bilateral grid would be 5D
  • Does not fit on current hardware
  • Luminance edges are often sufficient

• Crosses thin lines
  • Diffusion vs. bilateral filter
  • Useful in many cases

• Grid resolution depends on the operator
  • E.g., for edge-aware brush:
    space sampling rate \sim brush radius
    intensity sampling rate \sim 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
• 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
EXTRA MATERIAL

• Another acceleration technique
Box Kernel [Weiss 06]

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

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

Restrict the sum

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

- 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

**Tracking one window**

**input:**
full histogram is known

**update:**
add one line, remove one line
Box Kernel [Weiss 06]

- Idea: fast histograms of square windows

Tracking two windows at the same time

**Input:**
full histograms are known

**Update:**
add one line, remove one line,
add two pixels, remove two pixels

Tuesday, February 24, 2009
Discussion

- Complexity: $\tilde{I} (|S| \times \log \sigma_s)$
  - 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