Fast Bilateral Filtering
Image Decomposition

• Splitting an image into two layers
  – Large-scale and small-scale components
  – Global and local contrasts, illumination and texture

• Caveats
  – Speed: This decomposition happens often, we cannot afford to wait.
  – Accuracy: This decomposition is the starting point of many applications, it has to be accurate.
Gaussian Convolution

input

smoothed (large scale)

residual (texture, small scale)

edge-preserving: Bilateral Filter

Obtained by subtraction
Halos are a Major Issue in Computational Photography

Bilateral filtering solves this problem.

Sample manipulation:
increasing local contrast three times.
Edge-preserving Smoothing

- Separate **large-scale component** from **texture**
- Prevent halos and blurry contours

**Bilateral Filter**

Obtained by subtraction
Traditional Denoising versus Computational Photography

Edge-preserving filtering introduced for denoising.

• Denoising: decompose into signal + noise
  – Throw away noise
  – Small kernels

• Computational photography: decompose into base + detail
  – Detail is valuable
  – Large kernels

ROWSER filter [Aurich 95, Smith 97, Tomasi 98]
Bilateral Filter: Ubiquitous in Computational Photography

- **Style transfer** [Bae 06]
  - Input (low-light video)
  - Output

- **Contrast management** [Durand 02]
  - Input
  - Output

- **Exposure correction** [Bennett 05]
  - Input
  - Output

- **Abstraction** [Winnemoeller 06]
  - Input
  - Output
Illustration a 1D Image

- 1D image = line of pixels

![Illustration of a 1D image with pixel intensities and positions](image)

- Better visualized as a plot
Definition

Gaussian blur

\[ I^b_p = \sum_{q \in S} G_{\sigma_s}(\|p - q\|) I_q \]

- only spatial distance, intensity ignored

Bilateral filter

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

- spatial and range distances
- weights sum to 1
Example on a Real Image

- Kernels can have complex, spatially varying shapes.
Bilateral Filter is Expensive

• Brute-force computation is slow (several minutes)
  – Two nested for loops:
    *for each pixel, look at all pixels*
  – Non-linear, depends on image content
    ⇒ no FFT, no pre-computation…

• Fast approximations exist [Durand 02, Weiss 06]
  – Significant *loss of accuracy*
  – *No formal understanding* of accuracy versus speed
Today

• We will reformulate the bilateral filter
  – Link with linear filtering
  – Fast and accurate algorithm
Questions ?
Outline

• Reformulation of the BF

• Fast algorithm to compute the BF

• Practical implementation

• Application and extension
  – Photographic style transfer
  – Bilateral grid
Bilateral Filter on 1D Signal

![Graph showing Bilateral Filter on 1D Signal](image)
Our Strategy

Reformulate the bilateral filter

- More complex space:
  - Homogeneous intensity
  - Higher-dimensional space

- Simpler expression: mainly a convolution
  - Leads to a fast algorithm
Link with Linear Filtering

1. Handling the Division

Handling the division with a projective space.

\[ I_{p}^{bf} = \frac{1}{W_{p}^{bf}} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_x}(\|I_p - I_q\|) I_q \]
Formalization: Handling the Division

\[ I_{p}^{bf} = \frac{1}{W_{p}^{bf}} \sum_{q \in S} G_{\sigma_{s}}(\|p - q\|) G_{\sigma_{r}}(|I_{p} - I_{q}|) I_{q} \]

\[ W_{p}^{bf} = \sum_{q \in S} G_{\sigma_{s}}(\|p - q\|) G_{\sigma_{r}}(|I_{p} - I_{q}|) \]

- Normalizing factor as homogeneous coordinate
- Multiply both sides by \( W_{p}^{bf} \)

\[
\begin{pmatrix}
W_{p}^{bf} & I_{p}^{bf} \\
W_{p}^{bf} & W_{p}^{bf}
\end{pmatrix}
= 
\sum_{q \in S} G_{\sigma_{s}}(\|p - q\|) G_{\sigma_{r}}(|I_{p} - I_{q}|)
\begin{pmatrix}
I_{q} \\
1
\end{pmatrix}
\]
Formalization: Handling the Division

\[
\begin{pmatrix}
W_p^{bf} & I_p^{bf} \\
W_p^{bf} & W_p^{bf}
\end{pmatrix} = \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_t}(\|I_p - I_q\|) \begin{pmatrix}
W_q & I_q \\
W_q & W_q
\end{pmatrix} \text{ with } W_q = 1
\]

- Similar to homogeneous coordinates in projective space
- Division delayed until the end
Questions ?
Link with Linear Filtering

2. Introducing a Convolution

\[
\begin{pmatrix}
W_p^{bf} & I_p^{bf} \\
W_p^{bf} & W_p^{bf}
\end{pmatrix}
= \sum_{q \in S} \begin{pmatrix}
G_{\sigma_s}(\|p - q\|) & G_{\sigma_t}(\|I_p - I_q\|)
\end{pmatrix} \begin{pmatrix}
W_q & I_q \\
W_q & W_q
\end{pmatrix}
\]

2D Gaussian
Link with Linear Filtering
2. Introducing a Convolution

\[
\begin{pmatrix}
W_{bf}^{p} & I_{bf}^{p} \\
W_{bf}^{p} & W_{bf}^{p}
\end{pmatrix}
= \sum_{q \in S} G_{\sigma_{\text{space}}} (\|p - q\|) \ G_{\sigma_{\text{range}}} (\|I_{p} - I_{q}\|)
\begin{pmatrix}
W_{q} & I_{q} \\
W_{q} & W_{q}
\end{pmatrix}
\]

Corresponds to a 3D Gaussian on a 2D image. Result appeared previously in [Barash 02].
Link with Linear Filtering

2. Introducing a Convolution

\[
\begin{pmatrix}
W_p^{bf} & I_p^{bf} \\
W_p^{bf} & W_p^{bf}
\end{pmatrix}
= \sum_{(q, \zeta) \in S \times R} \sum \text{space-range Gaussian}
\]

sum all values multiplied by kernel \(\Rightarrow\) convolution
Link with Linear Filtering

2. Introducing a Convolution

\[
\begin{pmatrix}
W_p^{bf} & I_p^{bf} \\
W_p^{bf} & W_p^{bf}
\end{pmatrix}
= \sum \sum_{(q, \zeta) \in S \times R} \begin{pmatrix}
W_q & I_q \\
W_q & W_q
\end{pmatrix}
\]

result of the convolution
Link with Linear Filtering

2. Introducing a Convolution

\[
\begin{pmatrix}
W_p^{bf} & I_p^{bf} \\
W_p^{bf} & W_q
\end{pmatrix}
= \sum_{(q, \zeta) \in S \times R} \sum (W_q I_q)
\]

result of the convolution

space-range Gaussian
higher dimensional functions

Gaussian convolution

division

slicing
Reformulation: Summary

1. Convolution in higher dimension
   • expensive but well understood (linear, FFT, etc)

2. Division and slicing
   • nonlinear but simple and pixel-wise

Exact reformulation

linear: \( (w^{bf} \cdot bf, w^{bf}) = g_{s, r} \otimes (w_i, w) \)

nonlinear: \( I_p^{bf} = \frac{w^{bf}(p, I_p) \cdot bf(p, I_p)}{w^{bf}(p, I_p)} \)
Questions ?
Outline

• Reformulation of the BF

• Fast algorithm to compute the BF

• Practical implementation

• Application and extension
  – Photographic style transfer
  – Bilateral grid
Recap:
- simple operations
- complex space
Strategy: downsampling convolution

**Higher dimensional functions**

**Gaussian convolution**

**Downsample**

**Upsample**

**Division**

**Slicing**
Sampling Theorem

• Sampling a signal at a least twice its smallest wavelength is enough.

Not enough
Sampling Theorem

- Sampling a signal at a least twice its smallest wavelength is enough.
Sampling Theorem

- Sampling a signal at a least twice its smallest wavelength is enough.

Enough
Signal processing analysis

higher dimensional functions

Low-pass filter
Gaussian convolution

Almost only low freq.
High freq. negligible

division

slicing
higher dimensional functions

“Safe” downsampling

D O W N S A M P L E

Gaussian convolution

Almost no information loss

U P S A M P L E

division

slicing

Almost no information loss
Fast Convolution by Downsampling

- Downsampling cuts frequencies above Nyquist limit (half the sampling rate)
  - Less data to process
  - But introduces error

- Evaluation of the approximation

- Efficient implementation
Accuracy versus Running Time

À 1 second instead of several minutes

• Finer sampling increases accuracy.
• More precise than previous work.

Accuracy as function of Running Time

Brute-force bilateral filter takes over 10 minutes.
Visual Results

• Comparison with previous work [Durand 02]
  – running time = 1s for both techniques

<table>
<thead>
<tr>
<th>input</th>
<th>exact BF</th>
<th>our result</th>
<th>prev. work</th>
</tr>
</thead>
</table>

difference with exact computation (intensities in [0:1])

1200 × 1600
More on Accuracy and Running Times
Kernel Size

- Larger kernels are faster because we downsample more.
  ➕ Useful for photography.

**Running Time as a function of Kernel Size**

Brute-force bilateral filter takes over 10 minutes.
Questions ?
Outline

• Reformulation of the BF

• Fast algorithm to compute the BF

• Practical implementation

• Application and extension
  – Photographic style transfer
  – Bilateral grid
Efficient Implementation

- Never build the full resolution 3D space
  - Bin pixels on the fly
  - Interpolate on the fly

- Separable Gaussian kernel

- 5-tap approximation
Sampling Rate

• 1 sample every sigma
  – Kernel parameters are all equal to 1 sample
  – 5-tap approximation is sufficient

 1 – 4 – 6 – 4 – 1
Fast Bilateral Filter

- input: image $I$
- Gaussian parameters $\sigma_s$ and $\sigma_r$
- sampling rates $s_s$ and $s_r$

- output: filtered image $I^b$

1. Initialize all $w_i$ and $w_i'$ values to 0. **Initialize the 3D grid to 0.**
Fast Bilateral Filter

input: image $I$
Gaussian parameters $\sigma_s$ and $\sigma_r$
sampling rates $s_s$ and $s_r$

output: filtered image $I^b$

1. Initialize all $w_i$ and $w_\downarrow$ values to 0. Initialize the 3D grid to 0.

2. Compute the minimum intensity value: Useful later to save space. $I_{\text{min}} \leftarrow \min_{(X,Y) \in S} I(X,Y)$
3. For each pixel \((X, Y) \in S\) with an intensity \(I(X, Y) \in \mathcal{R}\). **Look at each pixel.**

   (a) Compute the homogeneous vector \((wi, w)\):

   **Create the data.** \((wi, w) \leftarrow (I(X, Y), 1)\)
3. For each pixel \((X, Y) \in S\) with an intensity \(I(X, Y) \in \mathcal{R}\). Look at each pixel.

(a) Compute the homogeneous vector \((wi, w)\):

**Create the data.** \((wi, w) \leftarrow (I(X, Y), 1)\)

(b) Compute the downsampling coordinates (with \([ \cdot ]\) the rounding operator)

**Compute the grid location.** \((x, y, \zeta) \leftarrow \left(\left[\frac{X}{s_{s}}\right], \left[\frac{Y}{s_{s}}\right], \left[\frac{I(X, Y) - I_{\text{min}}}{s_{r}}\right]\right)\)
3. For each pixel \((X, Y) \in \mathcal{S}\) with an intensity \(I(X, Y) \in \mathcal{R}\). Look at each pixel.

(a) Compute the homogeneous vector \((w_i, w)\):

Create the data. \[ (w_i, w) \leftarrow (I(X, Y), 1) \]

(b) Compute the downsampled coordinates (with \([ \cdot ]\) the rounding operator)

Compute the grid location. \[ (x, y, \zeta) \leftarrow \left( \left[ \frac{X}{s_s} \right], \left[ \frac{Y}{s_s} \right], \left[ \frac{I(X, Y) - I_{\min}}{s_r} \right] \right) \]

(c) Update the downsampled \(\mathcal{S} \times \mathcal{R}\) space

Update the grid. \[
\begin{pmatrix}
  w \downarrow i \downarrow (x, y, \zeta) \\
  w \downarrow (x, y, \zeta)
\end{pmatrix}
\leftarrow
\begin{pmatrix}
  w \downarrow i \downarrow (x, y, \zeta) \\
  w \downarrow (x, y, \zeta)
\end{pmatrix}
+ \begin{pmatrix}
  w_i \\
  w
\end{pmatrix}
\]
4. Convolv \( (w_{\downarrow} i_{\downarrow}, w_{\downarrow}) \) with a 3D Gaussian \( g \) whose parameters are \( \sigma_s/s_s \) and \( \sigma_r/s_r \)

\[
(w^b_{\downarrow} i^b_{\downarrow}, w^b_{\downarrow}) \leftarrow (w_{\downarrow} i_{\downarrow}, w_{\downarrow}) \otimes g
\]

**Convolve with 1 – 4 – 6 – 4 – 1 along each axis.**

**3 for loops needed, one for each axis.**

**In 3D, 15 samples (3 times 5) considered instead of 125 (5^3) for a full convolution.**

**Same result!**
5. For each pixel \((X, Y) \in S\) with an intensity \(I(X, Y) \in \mathcal{R}\) **Look at each pixel.**

(a) Tri-linearly interpolate the functions \(w^b_{1} i^b_{1}\) and \(w^b_{1}\) to obtain \(W^b I^b\) and \(W^b\):

\[
W^b I^b(X, Y) \leftarrow \text{interpolate} \left( w^b_{1} i^b_{1}, \frac{X}{s_s}, \frac{Y}{s_s}, \frac{I(X, Y)}{s_r} \right)
\]

\[
W^b(X, Y) \leftarrow \text{interpolate} \left( w^b_{1}, \frac{X}{s_s}, \frac{Y}{s_s}, \frac{I(X, Y)}{s_r} \right)
\]
5. For each pixel \((X, Y) \in S\) with an intensity \(I(X, Y) \in R\) \textbf{Look at each pixel.}

(a) Tri-linearly interpolate the functions \(w^b\) and \(i^b\) to obtain \(W^b I^b\) and \(W^b:\)

\[
W^b I^b(X, Y) \leftarrow \text{interpolate} \left( w^b, \frac{X}{s_s}, \frac{Y}{s_s}, \frac{I(X, Y)}{s_r} \right)
\]

\[
W^b(X, Y) \leftarrow \text{interpolate} \left( w^b, \frac{X}{s_s}, \frac{Y}{s_s}, \frac{I(X, Y)}{s_r} \right)
\]

(b) Normalize the result

\[
I^b(X, Y) \leftarrow \frac{W^b I^b(X, Y)}{W^b(X, Y)}
\]
Comments

• Every sample is processed even if empty
  – The grid is coarse and fairly dense in 3D.
    • e.g. parameters (16,0.1): 256 pixels for 10 bins
      convolution spans 5 bins
      at least 50% occupancy
  – Simple data structure, simple sweep ⇒ fast

  – Maybe: can be improved
Complexity

- There is no nested loops over the whole set of samples
  - At most: “for each sample, for 5 samples”

\[ O \left( |S| + \frac{|S|}{s_s^2} \frac{|R|}{s_r} \right) \]

- Creation + slicing “for each pixel”
- Convolution “for each sample”
Color Images

• Range domain = RGB, Lab… $\Rightarrow$ 3D space

• Space (2D) + range (3D) $\Rightarrow$ 5D space

• Same algorithm works but:
  – Larger space $\Rightarrow$ more memory required
  – Larger, higher-dimensional space $\Rightarrow$ slower
  – Still faster than brute force, especially for large kernels
Running Times (0.5 megapixels)
Fast Bilateral Filter

- No visible difference with brute force
- Based on signal processing argument
- Works well with medium to large kernels
- Easy to code
Outline

• Reformulation of the BF

• Fast algorithm to compute the BF

• Practical implementation

• Application and extension
  – Photographic style transfer
  – Bilateral grid
Control of Photographic Style using an Example

Transfer the visual style of to

Focus on black and white photographs.

Today’s presentation:
Quick overview with emphasis on BF.
Overview

Input Image

Global contrast

Careful combination

Post-process

Result

Split

Local contrast

Overview
Overview

Input Image

Split

Global contrast

Local contrast

Careful combination

Post-process

Result
Bilateral Filter

• Better than blurring: preserve edges (no halo).
• Fast computation.

**base** (result of the bilateral filter) **detail** (residual)
Global contrast

Careful combination

Post-process

Result

Input Image

Bilateral Filter

Local contrast
Local contrast

Global contrast

Input Image

Bilateral Filter

Careful combination

Post-process

Result
Global Contrast

- Intensity remapping of base layer

brighter  input  more contrasted

curve applied
Transfer: Histogram Matching

Applied only on the base
♀ Alter global contrast **independently** of local contrast
Global contrast

Input Image

Bilateral Filter

Intensity matching

Careful combination

Post-process

Local contrast

Result
Local contrast

Global contrast

Input Image

Bilateral Filter

Intensity matching

Careful combination

Post-process

Result
Texture Control with Detail Layer

- Halo-free manipulation
Measuring the Texture Level

• We introduce the notion of “textureness”
  – “amplitude of the local variations”
  – at each pixel, according to surrounding region

smooth region $\Rightarrow$ low value

textured region $\Rightarrow$ high value
“Textureness”: 1D Example

Previous work:
- low pass
  [Li 05, Su 05]

- edge-preserving filter

input signal

range weight set on input values

edges
Textureness Transfer

Step 1:
Histogram transfer

Step 2:
Per pixel scaling of detail layer to match desired textureness

input detail x 0.5 x 2.7 x 4.3 
output detail
Local contrast

Global contrast

Input Image

Intensity matching

Careful combination

Post-process

Result

Bilateral Filter

Textureness matching
**Global contrast**

- Input Image
- Intensity matching
- Careful combination
- Post-process

**Local contrast**

- Image
- Textureness matching
- Result

**Careful combination**

- Bilateral Filter
Preserving Details

direct combination
(detail + base)

corrected result
Local contrast

Global contrast

Input Image

Intensity matching

Bilateral Filter

Constrained Poisson

Texturedness matching

Post-process

Result
Local contrast

Global contrast

Input Image

Bilateral Filter

Intensity matching

Constrained Poisson

Textureness matching

Post-process

Result
Additional Effects

- **Soft focus** (high frequency manipulation)
- **Film grain** (texture synthesis [Heeger 95])
- **Color toning** (chrominance = \( f(\text{luminance}) \))
Intensity matching

Bilateral Filter

Intensity matching

Constrained Poisson

Textureness matching

Soft focus Toning Grain

Global contrast

Local contrast

Input Image

Result
Recap

**Global contrast**

- Intensity matching
- Bilateral Filter
- Textureness matching
- Constrained Poisson
- Soft focus Toning Grain

**Local contrast**

- Input Image
- Result
Results

User provides input and model photographs.

- Our system *automatically* produces the result.
Comparison with Naïve Histogram Matching

Model

Snapshot, Alfred Stieglitz

Input

Naïve Histogram Matching

Local contrast, sharpness unfaithful

Our result
Comparison with Naïve Histogram Matching

Model
*Clearing Winter Storm*, Ansel Adams

Input

Histogram Matching

Local contrast too low

Our Result
Color Images

- Lab color space: modify only luminance
Fast Algorithm

• Fast BF and multi-grid Poisson solver

• A few seconds for 1-megapixel picture
  – Interactive feedback: important for usability, easy to test several options…
Bilateral Grid

• Extend the use of the volumetric representation of images

• Maps on graphics hardware
  – Real-time HD video processing
References

• Webpage with code, bibliography, PDFs
  http://people.csail.mit.edu/sparis/bf/

• Tech report (~course note), available on above webpage:
  MIT-CSAIL-TR-2006-073

• Two-scale Tone Management for Photographic Look
  Soonmin Bae, Sylvain Paris, and Frédo Durand
  SIGGRAPH 2006

• Real-time Edge-Aware Image Processing with the Bilateral Grid
  Jiawen Chen, Sylvain Paris, Frédo Durand

• Other acceleration technique, always fast but less flexible
  Fast Median and Bilateral Filtering.
  Ben Weiss.
  SIGGRAPH 2006
  http://www.shellandslate.com/fastmedian.html