A Gentle Introduction to Bilateral Filtering and its Applications

07/10: Novel Variants of the Bilateral Filter

Jack Tumblin – EECS, Northwestern University
Review: Bilateral Filter

A 2-D filter window: weights vary with intensity

2 Gaussian Weights: product = ellipsoidal footprint

Normalize weights to always sum to 1.0
Review: Bilateral Filter

Why it works: graceful segmentation

- Smoothing for ‘similar’ parts ONLY
- Range Gaussian $s$ acts as a ‘filtered region’ finder
Bilateral Filter Variants

• before the ‘Bilateral’ name:
  – Smith & Brady (1997): SUSAN

And now, a growing set of named variants:
• ‘Trilateral’ Filter (Choudhury et al., EGSR 2003)
• Cross-Bilateral (Petschnigg04, Eisemann04)
• NL-Means (Buades 05)

And more coming: application driven…
Who was first? Many Pioneers

- Elegant, Simple, Broad Idea
  →  →
  ‘Invented’ several times

- Different Approaches, Increasing Clarity
  - Smith & Brady (1995): ‘SUSAN’
    “Smallest Univalue Segment Assimilating Nucleus”
  - Yaroslavsky (1985)
    ‘Transform Domain Image Restoration Methods’
1985 Yaroslavsky:

A 2-D filter window: weights vary with intensity ONLY

Square neighborhood, Gaussian Weighted ‘similarity’

Normalize weights to always sum to 1.0
New Idea!

1995 Smith: ‘SUSAN’ Filter

A 2-D filter window: weights vary with intensity

2 Gaussian Weights: product = ellisoidal footprint

Normalize weights to always sum to 1.0
**Background:** ‘Unilateral’ Filter

e.g. traditional, linear, FIR filters

**Key Idea: Convolution**
- Output(x) = local weighted avg. of inputs.
- Weights vary within a ‘window’ of nearby x

- Smoothes away details, **BUT** blurs result

Note that weights always sum to 1.0
Bilateral Filter: **Strengths**

Piecewise smooth result
- averages local small details, ignores outliers
- preserves steps, large-scale ramps, and curves,...

- **Equivalent to anisotropic diffusion and robust statistics**
  [Black98, Elad02, Durand02]

- **Simple & Fast**
  (esp. w/ [Durand02] FFT-based speedup)
Bilateral Filter: 3 Difficulties

- Poor Smoothing in High Gradient Regions
- Smoothes and blunts cliffs, valleys & ridges
- Can combine disjoint signal regions

Output at \( \bullet \) is average of a tiny region
Bilateral Filter: 3 Difficulties

- Poor Smoothing in High Gradient Regions
- Smoothes and blunts cliffs, valleys & ridges
- Can combine disjoint signal regions
‘Bluntend Corners’ → Weak Halos

Bilateral:
‘Blunted Corners’ → Weak Halos

‘Trilateral’: 
Bilateral Filter: 3 Difficulties

- Poor Smoothing in High Gradient Regions
- Smoothes and blunts cliffs, valleys & ridges
- Disjoint regions can blend together
New Idea!

Trilaterial Filter (Choudhury 2003)

Goal:
Piecewise linear smoothing, not piecewise constant

Method:
Extensions to the Bilateral Filter

EXAMPLE: remove noise from a piecewise linear scanline
Outline: Bilateral → Trilateral Filter

Three Key Ideas:

• **Tilt** the filter window according to bilaterally-smoothed gradients

• **Limit** the filter window to connected regions of similar smoothed gradient.

• **Adjust** Parameters from measurements of the windowed signal
Outline: Bilateral → Trilaterial Filter

Key Ideas:

- **Tilt** the filter window according to bilaterally-smoothed gradients

- **Limit** the filter window to connected regions of similar smoothed gradient.

- **Adjust** Parameters from measurements of the windowed signal
Outline: Bilateral \( \rightarrow \) Triliteral Filter

**Key Ideas:**

- **Tilt** the filter window according to bilaterally-smoothed gradients.

- **Limit** the filter window to connected regions of similar smoothed gradient.

- **Adjust** Parameters from measurements of the windowed signal.
Trilaterial Filter (Choudhury 2003)

- **Strengths**
  - Sharpens *corners*
  - Smoothes similar *gradients*
  - Automatic *parameter* setting
  - 3-D *mesh de-noising, too!*

- **Weaknesses**
  - *S-L-O-W*; very costly connected-region finder
  - Shares Bilateral’s ‘Single-pixel region’ artifacts
  - *Noise Tolerance* limits; disrupts ‘tilt’ estimates
NEW IDEA: ‘Joint’ or ‘Cross’ Bilateral’

Bilateral $\rightarrow$ **two kinds** of weights

**NEW:** get them from **two kinds** of images.

- Smooth image A pixels locally, but
- Limit to ‘similar regions’ of image B

**Why do this?** To get ‘best of both images’
Ordinary Bilateral Filter

Bilateral $\rightarrow$ **two kinds** of weights, one image $A$:

$$BF\ [A]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s} (||p - q||) G_{\sigma_r} (|A_p - A_q|) A_q$$
NEW: **two kinds** of weights, **two** images

\[
BF [A]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s} (\| p - q \|) G_{\sigma_r} (| B_p - B_q |) A_q
\]

A: Noisy, dim (ambient image)

B: Clean, strong (Flash image)
Image A: Warm, shadows, but too Noisy
(too dim for a good quick photo)
Image B: Cold, Shadow-free, Clean
(flash: simple light, ALMOST no shadows)
MERGE BEST OF BOTH: apply
‘Cross Bilateral’ or ‘Joint Bilateral’
(it really is *much* better!)
Recovers Weak Signals Hidden by Noise

Noisy but Strong…

+ Noise =

Noisy and Weak…

+ Noise =
Ordinary Bilateral Filter?

Noisy but Strong…

Noisy and Weak…

Step feature GONE!!
Ordinary Bilateral

Noisy but Strong…

Range filter preserves signal

Noisy and Weak…

Signal too small to reject
‘Cross’ or ‘Joint’ Bilateral Idea:

Noisy but Strong…

Range filter preserves signal

Noisy and Weak…

Use stronger signal’s range filter weights…
‘Joint’ or ‘Cross’ Bilateral Filter

- **CBF(A,B):** smoothes image A only; (e.g. no flash)
- Limits smoothing to stay within regions where Image B is ~uniform (e.g. flash)

**Useful Residues.** To transfer details,
- CBF(A,B) to remove A’s noisy details
- CBF(B,A) to remove B’s clean details;
- add to CBF(A,B) – clean, detailed image!
**New Idea:**

**NL-Means Filter (Buades 2005)**

- Same goals: ‘Smooth within Similar Regions’

- **KEY INSIGHT:** Generalize, extend ‘Similarity’
  - **Bilateral:**
    Averages neighbors with **similar intensities**;
  - **NL-Means:**
    Averages neighbors with **similar neighborhoods**!

- For each and every pixel $p$: 

- For each and every pixel $p$:
  - Define a small, simple fixed size neighborhood;
• For each and every pixel $p$:
  - Define a small, simple fixed size neighborhood;
  - Define vector $V_p$: a list of neighboring pixel values.

\[
V_p = \begin{bmatrix}
0.74 \\
0.32 \\
0.41 \\
0.55 \\
\vdots \\
\vdots \\
\vdots \\
\vdots
\end{bmatrix}
\]

‘Similar’ pixels $p, q$

$\rightarrow$ SMALL

vector distance;

$|| V_p - V_q ||^2$

‘Dissimilar’ pixels \( p, q \)

\( \rightarrow \) LARGE

vector distance;

\[ \| V_p - V_q \| ^2 \]

‗Dissimilar‘ pixels \( p, q \)

\( \rightarrow \) LARGE

vector distance;

\[ \| V_p - V_q \|^2 \]

Filter with this!

\[ p, q \text{ neighbors define a vector distance;} \]

\[ \| V_p - V_q \|^2 \]

Filter with this:

No spatial term!

\[
NLMF [I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) \cdot G_{\sigma_r}(\| V_p - V_q \|^2) I_q
\]

pixels \( p, q \) neighbors

Set a vector distance;

\[ \| V_p - V_q \|^2 \]

Vector Distance to \( p \) sets weight for each pixel \( q \)

\[
NLMF \left[ I \right]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_r} \left( \| V_p - V_q \|^2 \right) I_q
\]
NL-Means Filter (Buades 2005)

- Noisy source image:
<table>
<thead>
<tr>
<th>NL-Means Filter (Buades 2005)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Gaussian Filter</td>
</tr>
<tr>
<td>Low noise,</td>
</tr>
<tr>
<td>Low detail</td>
</tr>
</tbody>
</table>
NL-Means Filter (Buades 2005)

- Anisotropic Diffusion

(Note ‘stairsteps’: ~ piecewise constant)
**NL-Means Filter (Buades 2005)**

- **Bilateral Filter**

(better, but similar ‘stairsteps’):
NL-Means Filter (Buades 2005)

- NL-Means:
  - Sharp,
  - Low noise,
  - Few artifacts.
Many More Possibilities: EXPERIMENT!

- Bilateral goals are *subjective*;
  - ‘Local smoothing within similar regions’
  - ‘Edge-preserving smoothing’
  - ‘Separate large structure & fine detail’
  - ‘Eliminate outliers’
  - ‘Filter within edges, not across them’

- It’s simplicity *invites new inventive answers.*