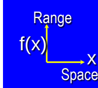
EEE6086 Video Processing and Analysis

Bilateral Filter and Non-Local Means

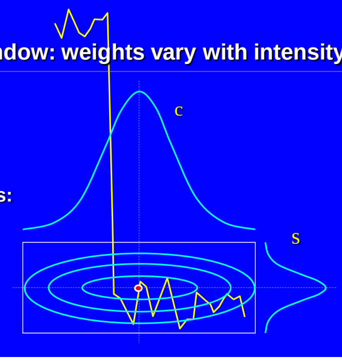
Review: Bilateral Filter

A 2-D filter window: weights vary with intensity



2 Gaussian Weights: product = ellisoidal footprint

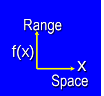
Normalize weights to always sum to 1.0

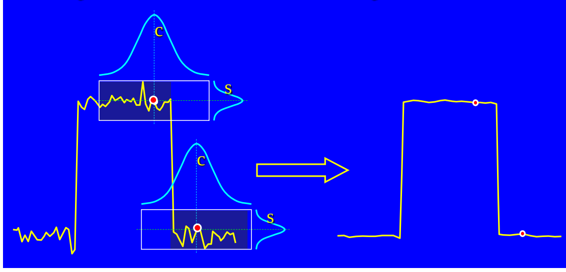


Review: Bilateral Filter

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 :
  - Yaroslavsky (1985): T.D.R.I.M.
  - Smith & Brady (1997): SUSAN

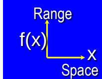
And now, a growing set of named variants:

- 'Trilateral' Filter (Choudhury et al., EGSR 2003)
- Cross-Bilateral (Petschnigg04, Eisemann04)
- NL-Means (Buades 05)

And more coming: application driven…

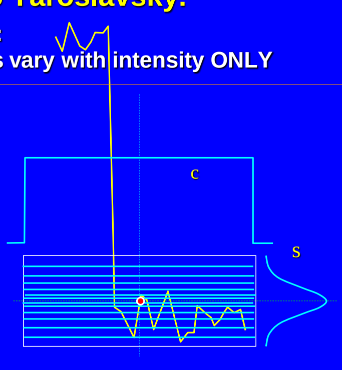
New Idea! 1985 Yaroslavsky:

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



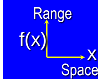
Square neighborhood, Gaussian Weighted 'similarity'

Normalize weights to always sum to 1.0



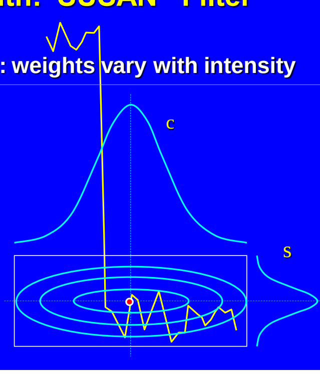
New Idea! 1995 Smith: 'SUSAN' Filter

A 2-D filter window: weights vary with intensity



2 Gaussian Weights: product = ellisoidal footprint

Normalize weights to always sum to 1.0



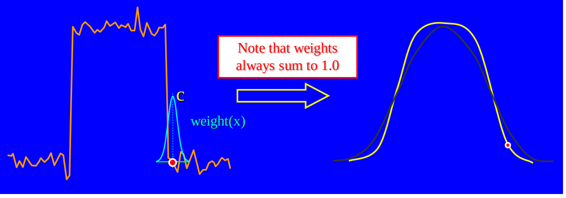
Background: 'Unilateral' Filter

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

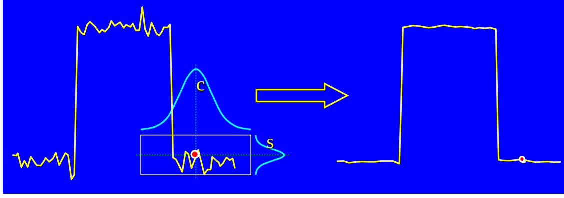


Bilateral Filter: Strengths

Piecewise smooth result

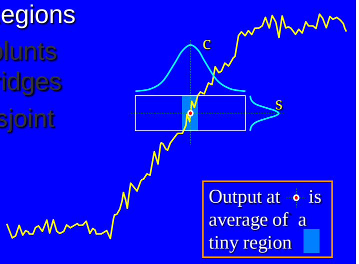
- averages local small details, ignores outliers
- preserves steps, large-scale ramps, and curves,...

- Equivalent to anisotropic diffusion and robust statistics [Black98,Elad02,Durand02]
- Simple & Fast (esp. w/ [Durand02] FFT-based speedup)



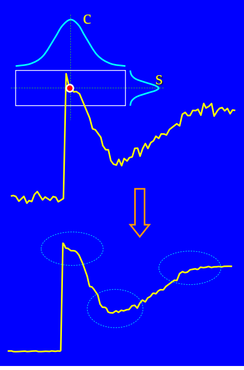
Bilateral Filter: 3 Difficulties

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



Bilateral Filter: 3 Difficulties

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



'Blunted Corners' → Weak Halos

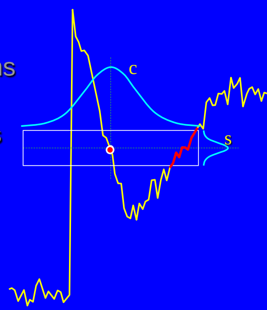
Bilateral :

'Blunted Corners' → Weak Halos

'Trilateral':

Bilateral Filter: 3 Difficulties

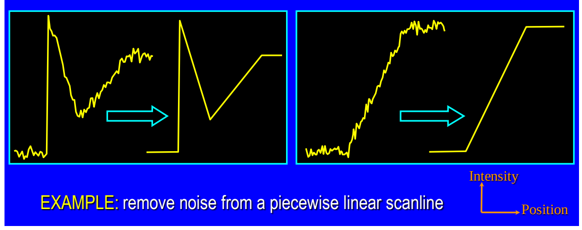
- Poor Smoothing in High Gradient Regions
- Smoothes and blunts cliffs, valleys & ridges
- Disjoint regions can blend together



New Idea! Trilateral Filter (Choudhury 2003)

Goal:
Piecewise linear smoothing, not piecewise constant

Method:
Extensions to the Bilateral Filter

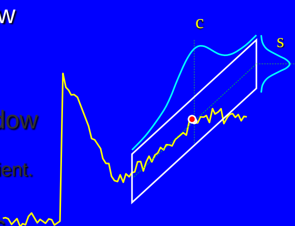


EXAMPLE: remove noise from a piecewise linear scanline

Outline: Bilateral→Trilateral Filter

Three Key Ideas:

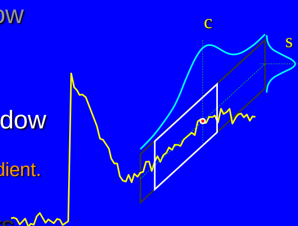
- **Tilt** the filter window according to bilaterally-smoothed gradients
- **Limit** the filter window to connected regions of similar smoothed gradient.
- **Adjust Parameters** from measurements of the windowed signal



Outline: Bilateral→Trilateral Filter

Key Ideas:

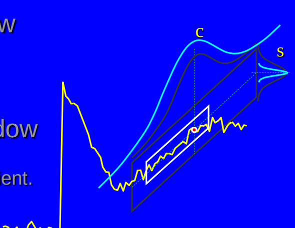
- **Tilt** the filter window according to bilaterally-smoothed gradients
- **Limit** the filter window to connected regions of similar smoothed gradient.
- **Adjust Parameters** from measurements of the windowed signal

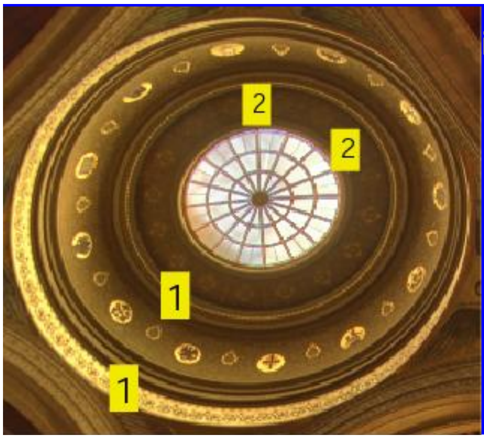


Outline: Bilateral→Trilateral Filter

Key Ideas:

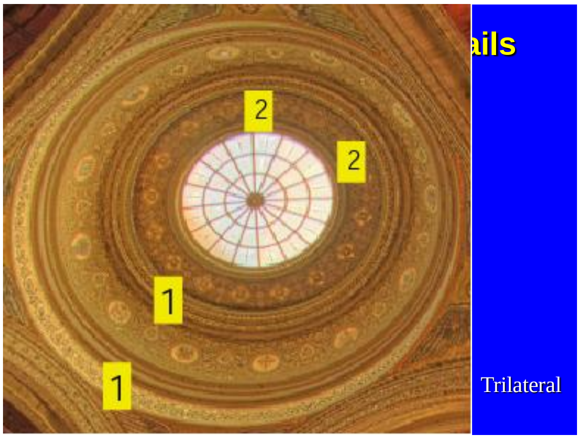
- **Tilt** the filter window according to bilaterally-smoothed gradients
- **Limit** the filter window to connected regions of similar smoothed gradient.
- **Adjust Parameters** from measurements of the windowed signal





ils

Bilateral



Trilateral 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' Petschnigg(2004) and Eisemann(2004)

Bilateral → *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**

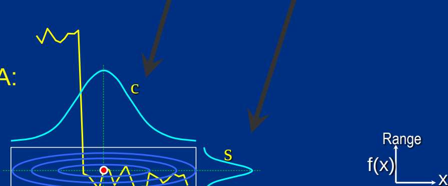
Why do this? To get **'best of both images'**

Ordinary Bilateral Filter

Bilateral → *two kinds* of weights, one image A :

$$BF[A]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|A_{\mathbf{p}}-A_{\mathbf{q}}|)\, A_{\mathbf{q}}$$

Image A:



'Joint' or 'Cross' Bilateral Filter

NEW: *two kinds* of weights, *two* images

$$BF[A]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|B_{\mathbf{p}}-B_{\mathbf{q}}|)\, A_{\mathbf{q}}$$

A: Noisy, dim
(ambient image)

B: Clean,strong
(Flash image)

Image A: Warm, shadows, but too Noisy
(too dim for a good quick photo)

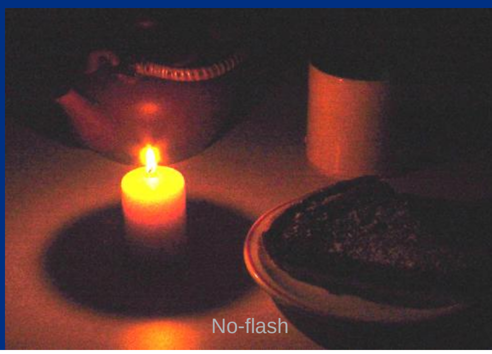
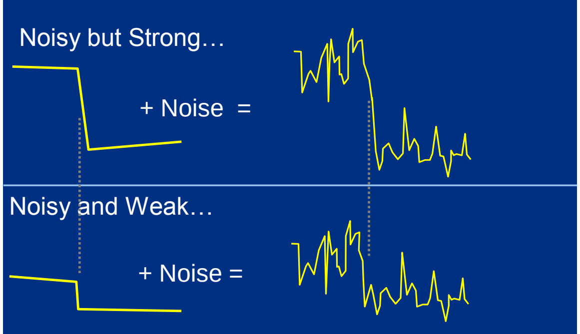


Image B: Cold, Shadow-free, Clean
(flash: simple light, ALMOST no shadows)

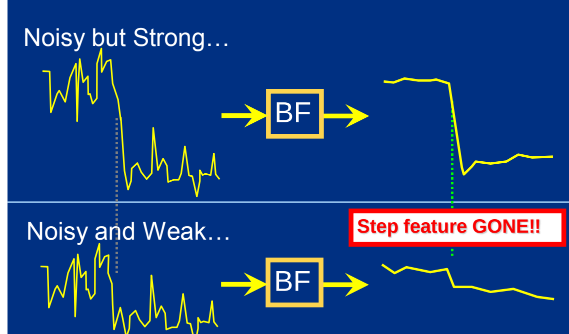
MERGE BEST OF BOTH: apply ‘Cross Bilateral’ or ‘Joint Bilateral’

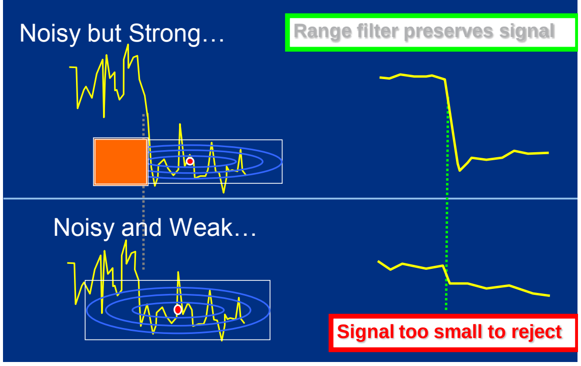
(it really is *much* better!)

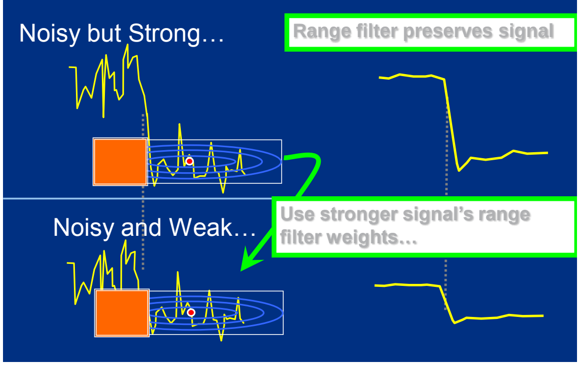
Recovers Weak Signals Hidden by Noise



Ordinary Bilateral Filter?







'Joint' or 'Cross' Bilateral Filter Petschnigg(2004) and Eisemann(2004)

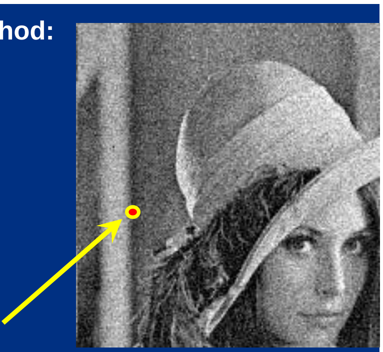
- CBF(A,B): smoothes image A only; (e.g. no flash)
- Limits smoothing to stay within regions where Image B is ~uniform (e.g. flash)

- **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)

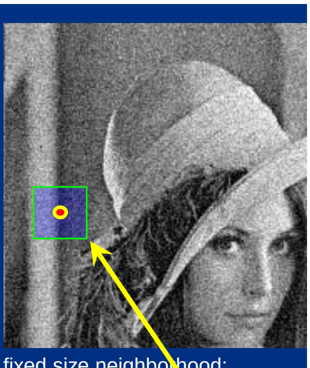
- Same goals: 'Smooth within Similar Regions'
- **KEY INSIGHT**: Generalize, extend 'Similarity'
  - **Bilateral:** Averages neighbors with **similar intensities**;
  - **NL-Means:** Averages neighbors with **similar neighborhoods!**

NL-Means Method: Buades (2005)



- For each and every pixel **p:**

NL-Means Method: Buades (2005)



- For each and every pixel **p:**
  - Define a small, simple fixed size neighborhood;

NL-Means Method: Buades (2005)

$$V_p = \begin{bmatrix} 0.74 \\ 0.32 \\ 0.41 \\ 0.55 \\ \dots \\ \dots \\ \dots \end{bmatrix}$$

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

NL-Means Method: Buades (2005)

'Similar' pixels **p, q**

→ **SMALL** vector distance;

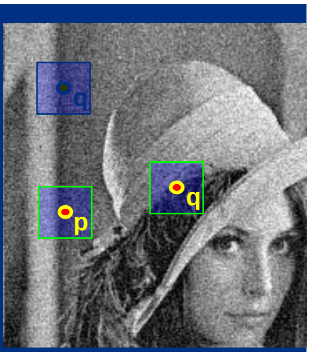
$\| V_p - V_q \|^2$

NL-Means Method: Buades (2005)

'Dissimilar' pixels **p, q**

→ **LARGE** vector distance;

$\| V_p - V_q \|^2$



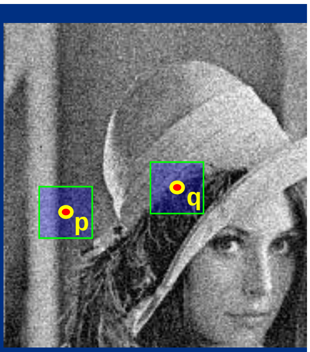
NL-Means Method: Buades (2005)

'Dissimilar' pixels **p, q**

→ **LARGE** vector distance;

$\| V_p - V_q \|^2$

Filter with this!



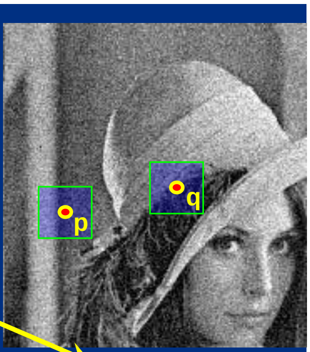
NL-Means Method: Buades (2005)

p, q neighbors define a vector distance;

$\boxed{\| V_p - V_q \|^2}$

Filter with this:

No spatial term!



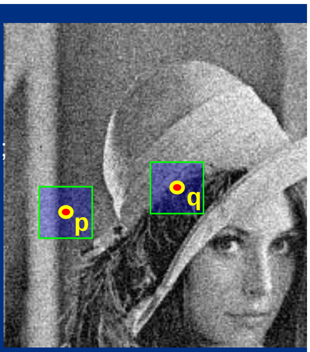
$$NLMF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} \cancel{G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)}\, \boxed{G_{\sigma_r}\left(\|\vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}}\|^2\right)} I_{\mathbf{q}}$$

NL-Means Method: Buades (2005)

pixels **p, q** neighbors Set a vector distance;

$\boxed{\| V_p - V_q \|^2}$

Vector Distance to p sets weight for each pixel q



$$NLMF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} \boxed{G_{\sigma_r}\left(\|\vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}}\|^2\right)} I_{\mathbf{q}}$$

NL-Means Filter (Buades 2005)

- Noisy source image:

NL-Means Filter (Buades 2005)

- Gaussian Filter

Low noise,

Low detail

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.

NL-Means Filter: Properties

- Works perfectly for images with repetitive patterns, e.g. edges, texture.
- Major drawback: efficiency, because searching for similar neighbourhood pixels is time-consuming

  - Efficient search techniques, e.g. sub-sampled search, can be used in practice