# BBM 413 Fundamentals of Image Processing

Erkut Erdem
Dept. of Computer Engineering
Hacettepe University

## Image Smoothing

**Acknowledgement:** The slides are mostly adapted from the course "A Gentle Introduction to Bilateral Filtering and its Applications" given by Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand (http://people.csail.mit.edu/sparis/bf_course/)

## Review - Smoothing and Edge Detection

- While eliminating noise via smoothing, we also lose some of the (important) image details.
  - Fine details
  - Image edges
  - etc.
- What can we do to preserve such details?
  - Use edge information during denoising!
  - This requires a definition for image edges.

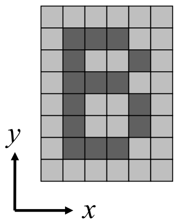
**Chicken-and-egg dilemma!**

- Edge preserving image smoothing

## Today

- Bilateral filter (Tomasi et al., 1998)
- NL-means filter (Buades et al., 2005)
- Structure-texture decomposition via region covariances (Karacan et al. 2013)

## Notation and Definitions

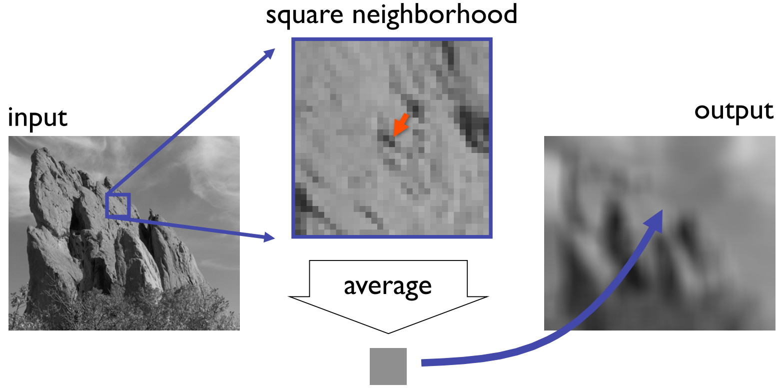



- Image = 2D array of pixels
- Pixel = intensity (scalar) or color (3D vector)
- $I_{\mathbf{p}}$ = value of image $I$ at position: $\mathbf{p} = (p_x, p_y)$
- $F[I]$ = output of filter $F$ applied to image $I$

# Strategy for Smoothing Images

- Images are not smooth because adjacent pixels are different.
- Smoothing = making adjacent pixels look more similar.
- Smoothing strategy
  pixel as average of its neighbors

# Box Average




# Equation of Box Average

$$BA[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} B_\sigma(\mathbf{p}-\mathbf{q})\, I_{\mathbf{q}}$$

result at pixel **p**

sum over all pixels **q**

normalized box function

intensity at pixel **q**

# Square Box Generates Defects

- Axis-aligned streaks
- Blocky results

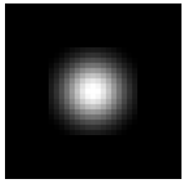
input

output

## Strategy to Solve these Problems

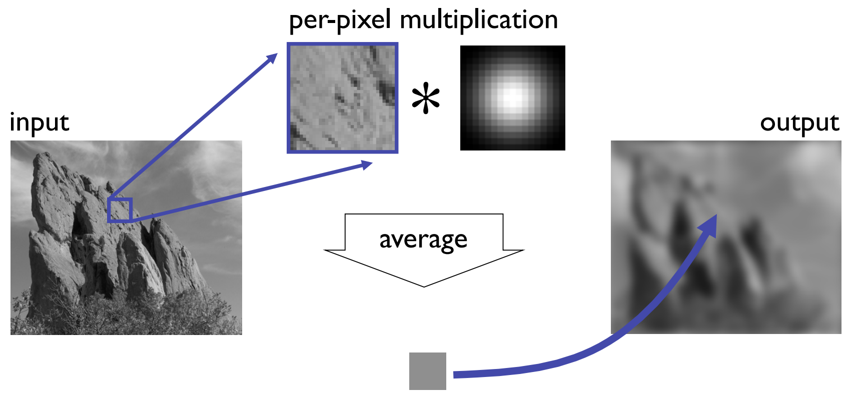
- Use an isotropic (*i.e.* circular) window.
- Use a window with a smooth falloff.

box window



Gaussian window

## Gaussian Blur



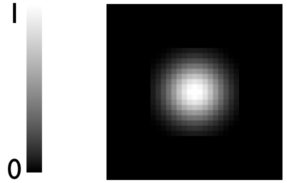



## Equation of Gaussian Blur

Same idea: **weighted average of pixels**.

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} \boxed{G_\sigma(\|\mathbf{p}-\mathbf{q}\|)}\, I_{\mathbf{q}}$$

normalized Gaussian function

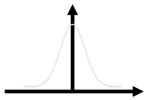



## Spatial Parameter

input

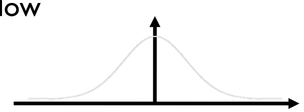
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\boxed{\sigma}}(\|\mathbf{p}-\mathbf{q}\|)\, I_{\mathbf{q}}$$

size of the window

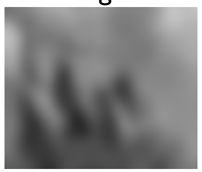


small *s*

limited smoothing



large *s*



strong smoothing

## How to set *s*

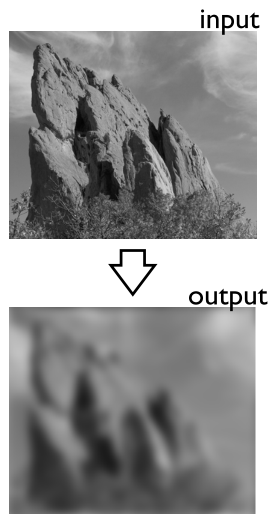
- Depends on the application.
- Common strategy: proportional to image size
  - e.g. 2% of the image diagonal
  - property: independent of image resolution

## Properties of Gaussian Blur

- Weights independent of spatial location
  - linear convolution
  - well-known operation
  - efficient computation (recursive algorithm, FFT…)

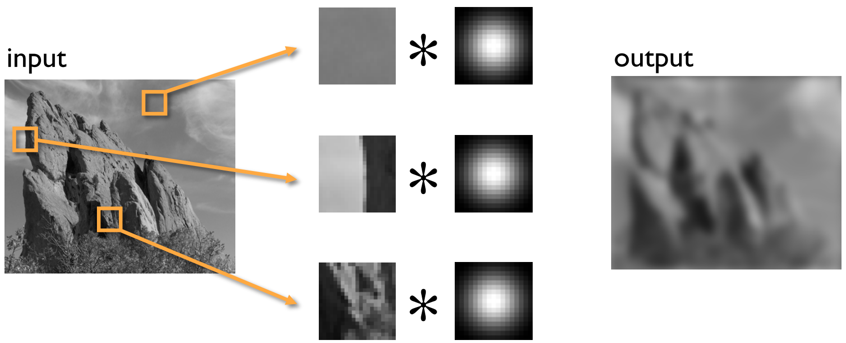
## Properties of Gaussian Blur

- Does smooth images
- But smoothes too much: **edges are blurred**.
  - Only spatial distance matters
  - No edge term




$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} \underbrace{G_\sigma(\|\mathbf{p}-\mathbf{q}\|)}_{\text{space}} I_{\mathbf{q}}$$

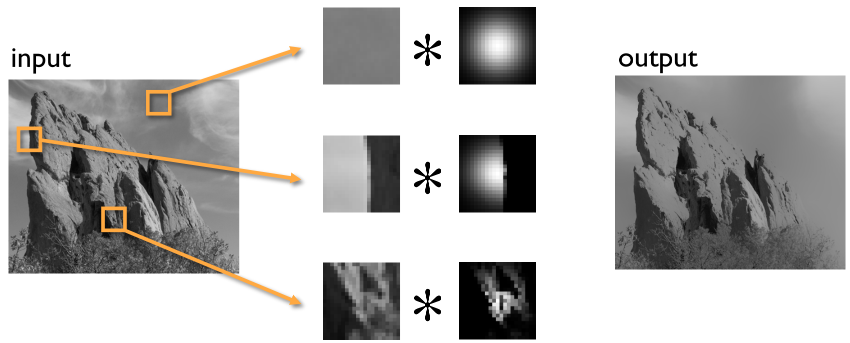
## Blur Comes from Averaging across Edges




Same Gaussian kernel everywhere.

[Aurich 95, Smith 97, Tomasi 98]

## Bilateral Filter No Averaging across Edges




The kernel shape depends on the image content.

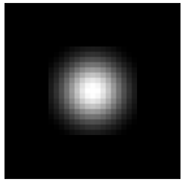
## Bilateral Filter Definition: an Additional Edge Term

Same idea: **weighted average of pixels**.

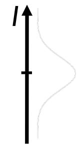
$$BF[I]_{\mathbf{p}} = \underbrace{\frac{1}{W_{\mathbf{p}}}}_{\text{new}} \sum_{\mathbf{q} \in S} \underbrace{G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|)}_{\text{not new}} \underbrace{G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |)}_{\text{new}} I_{\mathbf{q}}$$

normalization factor

***space*** weight



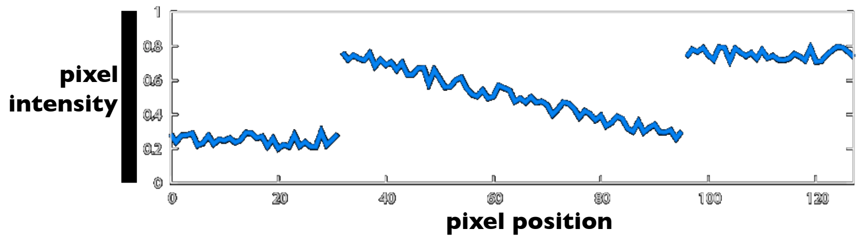
***range*** weight




## Illustration a 1D Image

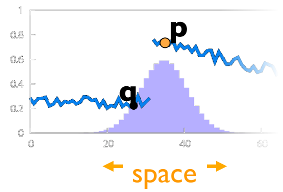
- 1D image = line of pixels

- Better visualized as a plot



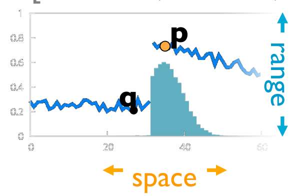

## Gaussian Blur and Bilateral Filter

Gaussian blur




$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} \underbrace{G_{\sigma}(\| \mathbf{p} - \mathbf{q} \|)}_{\text{space}} I_{\mathbf{q}}$$

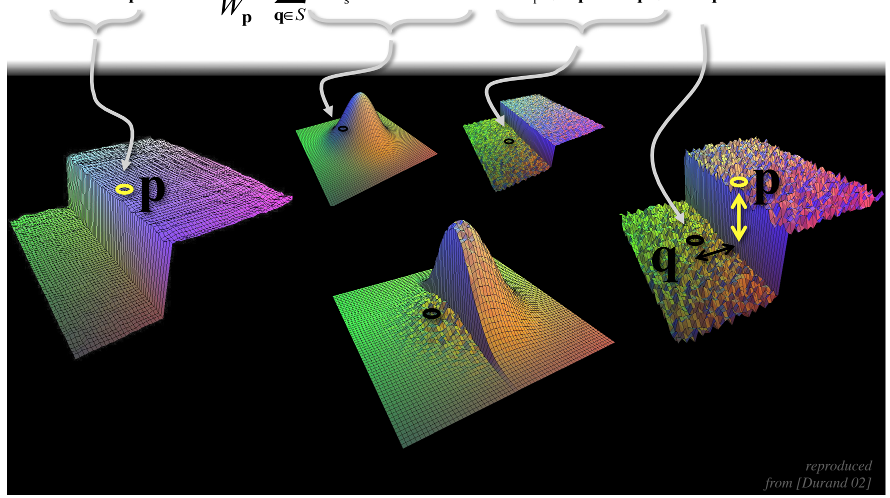
Bilateral filter
[Aurich 95, Smith 97, Tomasi 98]




$$BF[I]_{\mathbf{p}} = \underbrace{\frac{1}{W_{\mathbf{p}}}}_{\text{normalization}} \sum_{\mathbf{q} \in S} \underbrace{G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|)}_{\text{space}} \underbrace{G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |)}_{\text{range}} I_{\mathbf{q}}$$

## Bilateral Filter on a Height Field

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



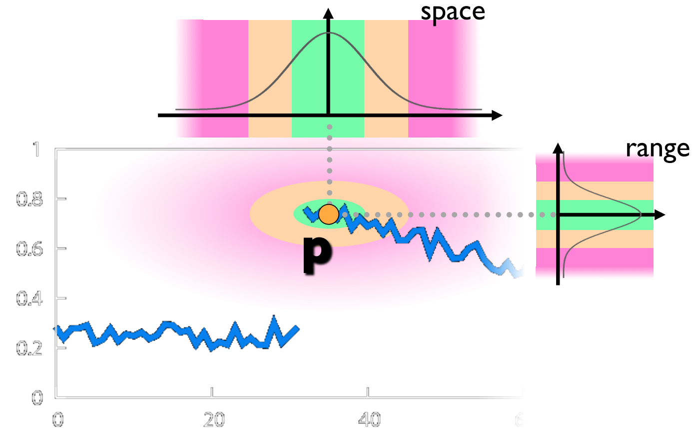
## Space and Range Parameters

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

- space $\sigma_s$ : spatial extent of the kernel, size of the considered neighborhood.
- range $\sigma_r$ : "minimum" amplitude of an edge

## Influence of Pixels

Only pixels close in space and in range are considered.




## Exploring the Parameter Space

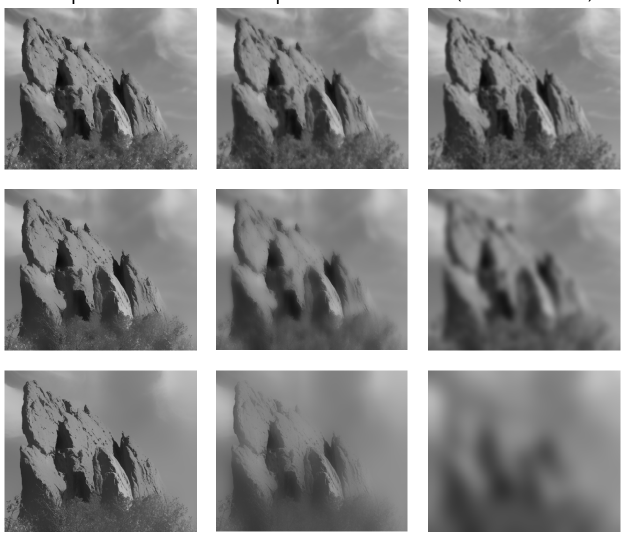
input

$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (Gaussian blur)

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$



## Varying the Range Parameter

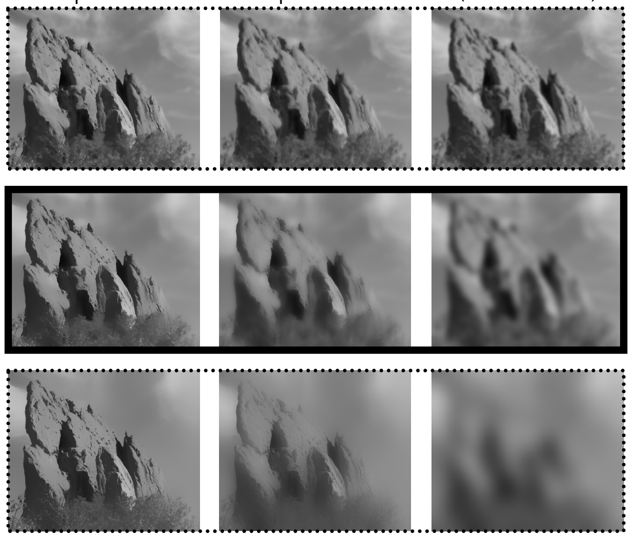
input

$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (Gaussian blur)

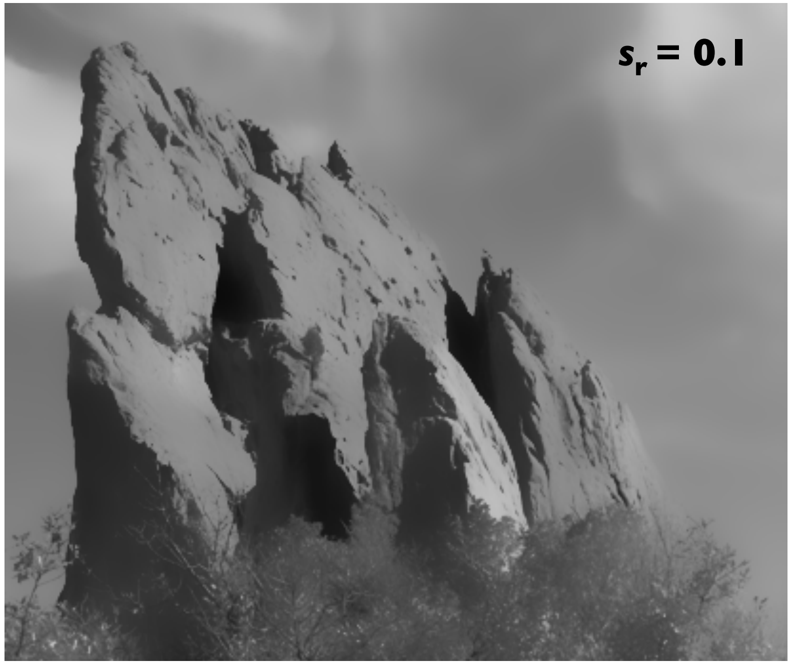
$\sigma_s = 2$

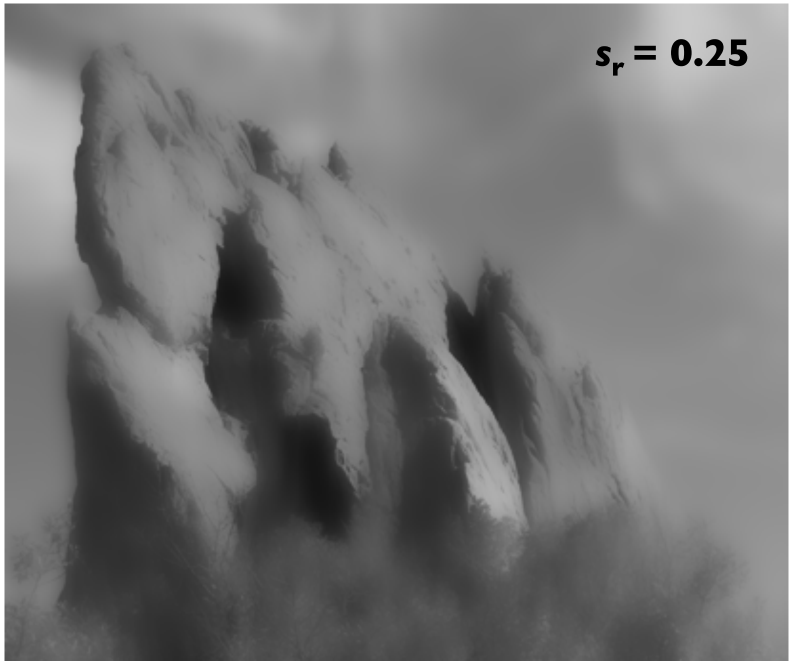
$\sigma_s = 6$

$\sigma_s = 18$



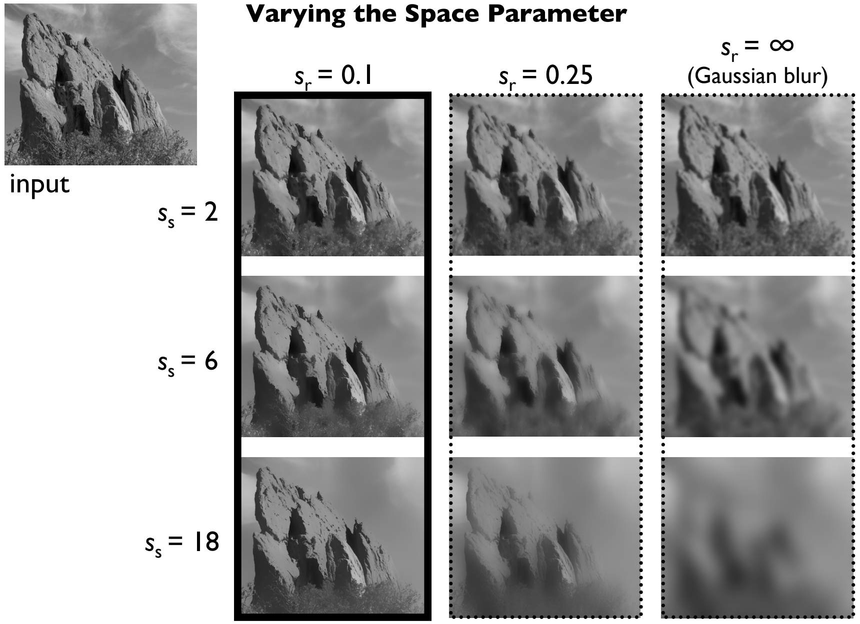

input


$s_r = 0.1$


$s_r = 0.25$


$s_r = \infty$
(Gaussian blur)

**Varying the Space Parameter**







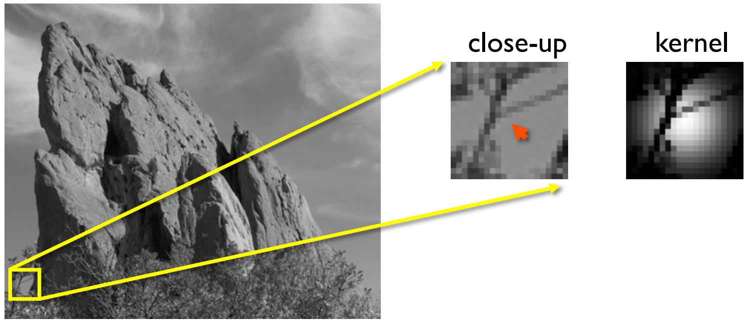
## How to Set the Parameters

Depends on the application. For instance:

- space parameter: proportional to image size
  - e.g., 2% of image diagonal
- range parameter: proportional to edge amplitude
  - e.g., mean or median of image gradients
- independent of resolution and exposure

## Bilateral Filter Crosses Thin Lines

- Bilateral filter averages across features thinner than ~$2s_s$
- Desirable for smoothing: more pixels = more robust
- Different from diffusion that stops at thin lines




## Iterating the Bilateral Filter

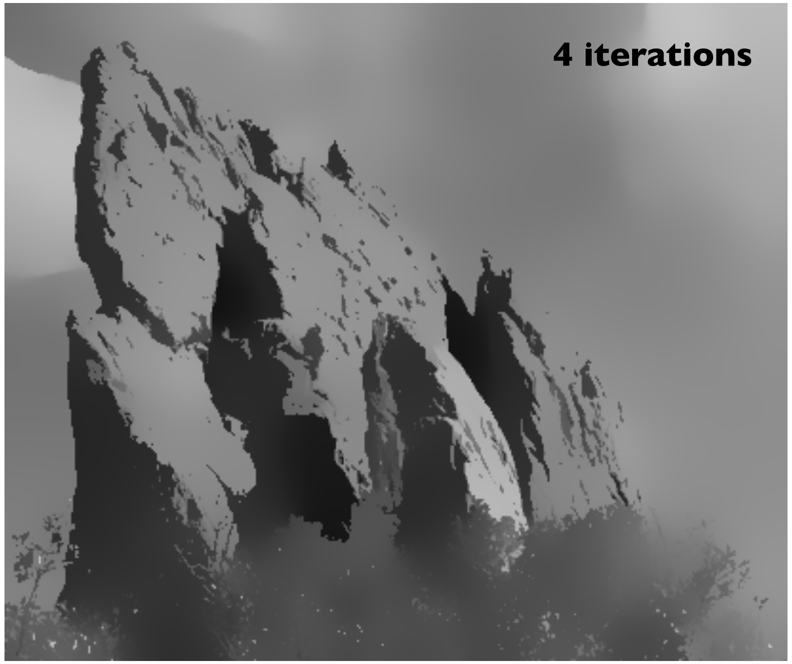
$$I_{(n+1)} = BF\left[I_{(n)}\right]$$

- Generate more piecewise-flat images
- Often not needed in computational photo.


input


1 iteration


2 iterations


4 iterations

## Bilateral Filtering Color Images

For gray-level images

intensity difference

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

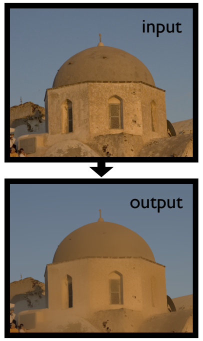
scalar

For color images

color difference

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

3D vector
(RGB, Lab)




## Hard to Compute

- Nonlinear $$BF[I]_{\mathbf{p}} = \boxed{\frac{1}{W_{\mathbf{p}}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|)\, \boxed{G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |)}\, I_{\mathbf{q}}$$
- Complex, spatially varying kernels
  - Cannot be precomputed, no FFT…

- Brute-force implementation is slow > 10min

## Basic denoising

Noisy input

Bilateral filter 7x7 window

## Basic denoising

Bilateral filter

Median 3x3

## Basic denoising

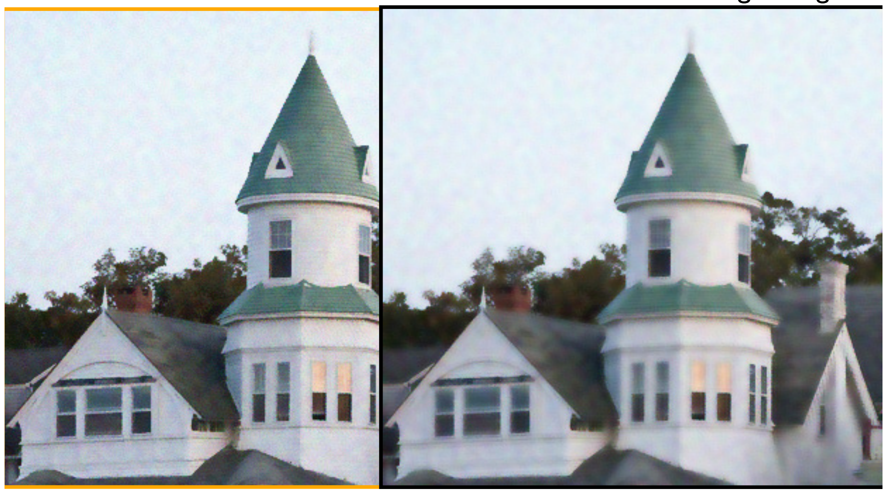
Bilateral filter | Median 5x5

## Basic denoising

Bilateral filter | Bilateral filter – lower sigma

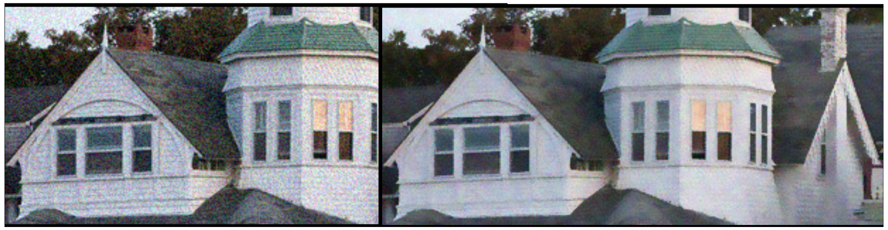
## Basic denoising

Bilateral filter | Bilateral filter – higher sigma

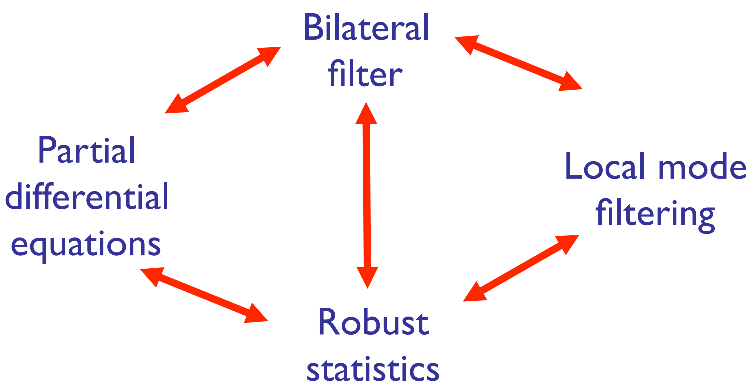


## Denoising

- Small spatial sigma (e.g. 7x7 window)
- Adapt range sigma to noise level
- Maybe not best denoising method, but best simplicity/quality tradeoff
  - No need for acceleration (small kernel)
  - But the denoising feature in e.g. Photoshop is better



Goal: Understand how does bilateral filter relates with other methods




more in BIL717 Image Processing graduate course..

## 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:**

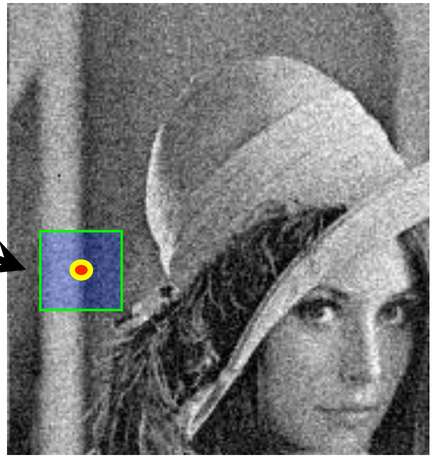
## NL-Means Method: Buades (2005)

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

– Define a small, simple fixed size neighborhood;

## NL-Means Method: Buades (2005)

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



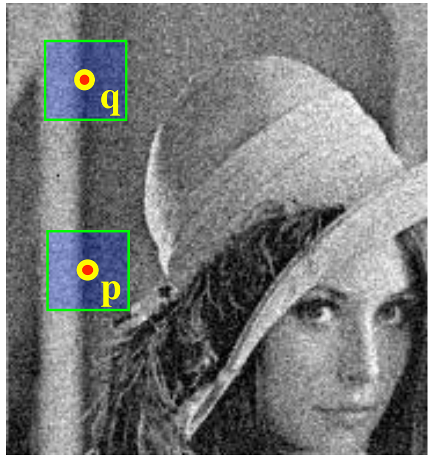
- For each and every pixel **p**:
  - Define a small, simple fixed size neighborhood;
  - Define vector $\mathbf{V_p}$: a list of neighboring pixel values.

## NL-Means Method: Buades (2005)

<u>'Similar'</u> pixels **p, q**

→ **SMALL** vector distance;

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

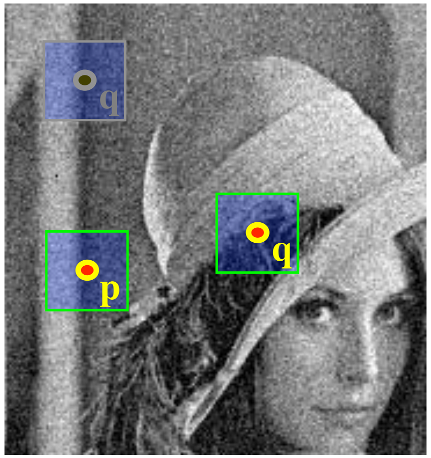



## NL-Means Method: Buades (2005)

<u>'Dissimilar'</u> pixels **p, q**

→ **LARGE** vector distance;

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



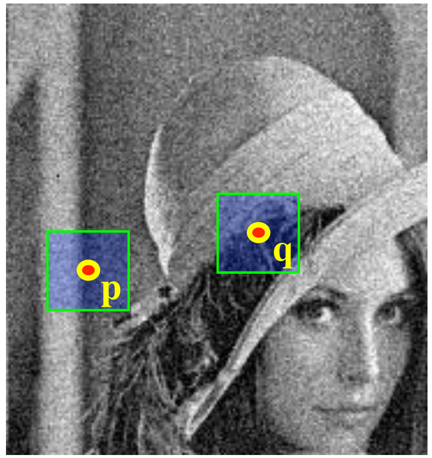

## NL-Means Method: Buades (2005)

<u>'Dissimilar'</u> pixels **p, q**

→ **LARGE** vector distance;

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

**Filter with this!**



## NL-Means Method: Buades (2005)

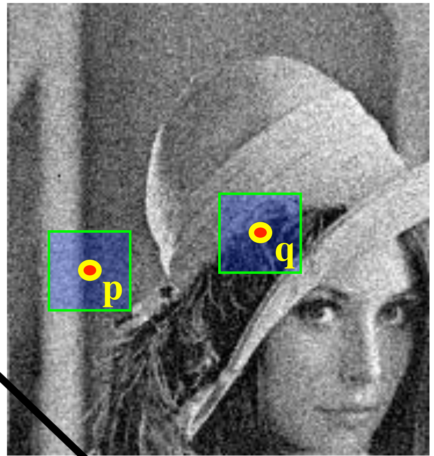
**p, q** neighbors define

a vector distance;

**Filter with this:**

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

**No spatial term!**



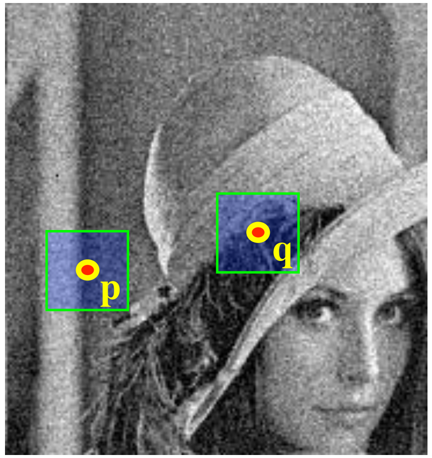

$$NLMF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} \cancel{G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|)} \, G_{\sigma_r}\left( \| \vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}} \|^2 \right) I_{\mathbf{q}}$$

## NL-Means Method: Buades (2005)

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

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

**Vector Distance to p sets weight for each pixel q**




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

## NL-Means Filter (Buades 2005)

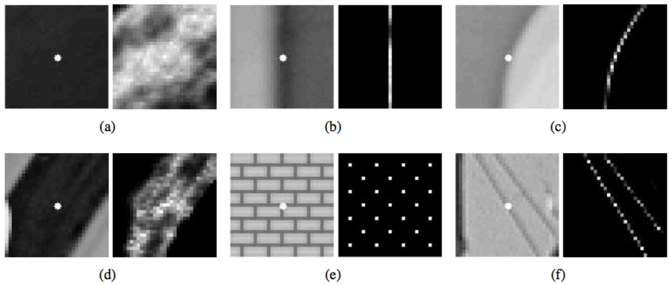



**Figure 2. Display of the NL-means weight distribution used to estimate the central pixel of every image. The weights go from 1(white) to zero(black).**

## NL-Means Filter (Buades 2005)

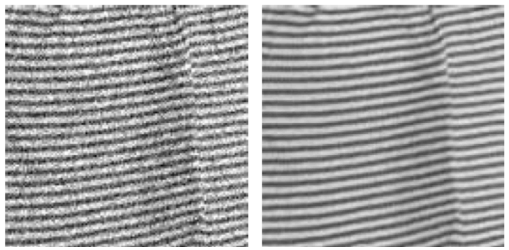


FIG. 9. *NL-means denoising experiment with a natural image. Left: Noisy image with standard deviation 20. Right: Restored image.*

## 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 (Buades 2005)

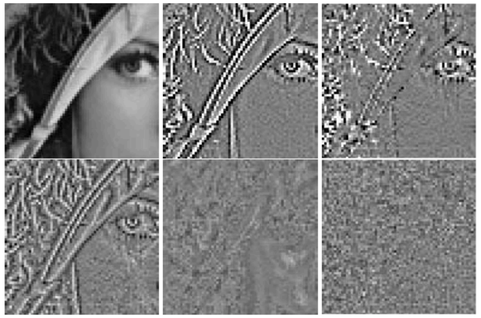


**Figure 4. Method noise experience on a natural image. Displaying of the image difference $u - D_h(u)$. From left to right and from top to bottom: original image, Gauss filtering, anisotropic filtering, Total variation minimization, Neighborhood filtering and NL-means algorithm. The visual experiments corroborate the formulas of section 2.**

# NL-Means Filter (Buades 2005)


http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

# NL-Means Filter (Buades 2005)


original

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

## NL-Means Filter (Buades 2005)

noisy

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

## NL-Means Filter (Buades 2005)

denoised

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

## NL-Means Filter (Buades 2005)

original

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

## NL-Means Filter (Buades 2005)

noisy

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

## NL-Means Filter (Buades 2005)

denoised

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

## Structure-Texture Decomposition
### Karacan et al., SIGGRAPH Asia 2013

Input Image

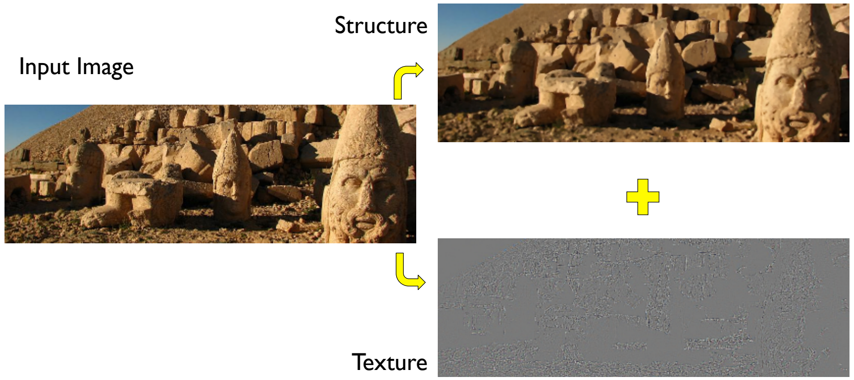
## Structure-Texture Decomposition
### Karacan et al., SIGGRAPH Asia 2013

Structure Component

## Structure-Texture Decomposition
### Karacan et al., SIGGRAPH Asia 2013

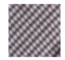
Texture Component

## Structure-Texture Decomposition
**Karacan et al., SIGGRAPH Asia 2013**

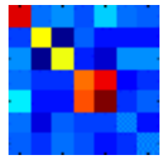



## Region Covariances as Region Descriptors
**Tuzel et al., ECCV 2006**


$$F(x,y) = \phi(I, x, y)$$



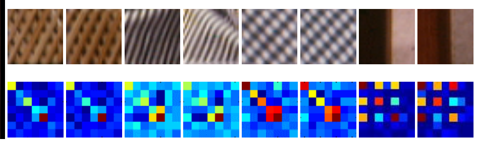
$$F(x,y) = \left[ I(x,y) \quad \left|\frac{\partial I}{\partial x}\right| \quad \left|\frac{\partial I}{\partial y}\right| \quad \left|\frac{\partial^2 I}{\partial x^2}\right| \quad \left|\frac{\partial^2 I}{\partial y^2}\right| \quad x \quad y \right]^T$$



$$\mathbf{C}_R = \frac{1}{n-1}\sum_{i=0}^{n}(\mathbf{z}_k - \mu)(\mathbf{z}_k - \mu)^T$$

## Main motivation

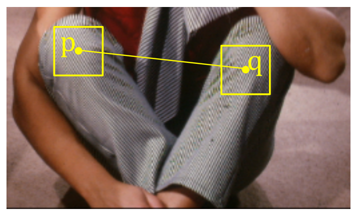

- Region covariances well capture local structure and texture information.
- Similar regions have similar statistics.



## Formulation

$$I = S + T$$

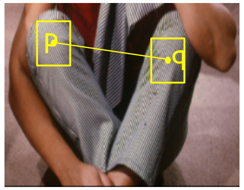
$$S(\mathbf{p}) = \frac{1}{Z_{\mathbf{p}}} \sum_{\mathbf{q} \in N(\mathbf{p}, r)} w_{\mathbf{pq}} I(\mathbf{q})$$




- Structure-texture decomposition via smoothing
- Smoothing as weighted averaging
- Different kernels ($w_{pq}$) result in different types of filters.
- Two novel patch-based kernels for structure-texture decomposition

## Model 1

- Covariance matrices do not live on Euclidean space.
- Hong et al.,CVPR'09 suggested a way to transform covariance matrices into Euclidean Space.
- Every covariance matrix has a unique Cholesky decomposition




$C = LL^T$ Cholesky Decomposition

$S = \{s_i\}$ Sigma Points

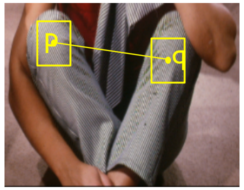
$$\mathbf{s}_i = \begin{cases} \alpha\sqrt{d}\mathbf{L}_i & \text{if } 1 \le i \le d \\ -\alpha\sqrt{d}\mathbf{L}_i & \text{if } d+1 \le i \le 2d \end{cases}$$

- First order statistics can be easily incorporated to the formulation.

## Model 1

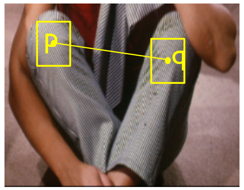
$$\Psi(\mathbf{C}) = (\mu, \mathbf{s}_1, \ldots, \mathbf{s}_d, \mathbf{s}_{d+1}, \ldots, \mathbf{s}_{2d})^T$$ Final representation

$$w_{\mathbf{pq}} \propto \exp\left(-\frac{\|\Psi(\mathbf{C_p}) - \Psi(\mathbf{C_q})\|^2}{2\sigma^2}\right)$$ Resulting kernel




## Model 2

- An alternative way is to use statistical measures.
- A Mahalanobis-like distance measure to compare to image patches




$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(\mu_\mathbf{p} - \mu_\mathbf{q})\mathbf{C}^{-1}(\mu_\mathbf{p} - \mu_\mathbf{q})^T}$$

$$\mathbf{C} = \mathbf{C_p} + \mathbf{C_q}$$

Resulting kernel

$$w_{\mathbf{pq}} \propto \exp\left(-\frac{d(\mathbf{p}, \mathbf{q})^2}{2\sigma^2}\right)$$

## Illustrative Example

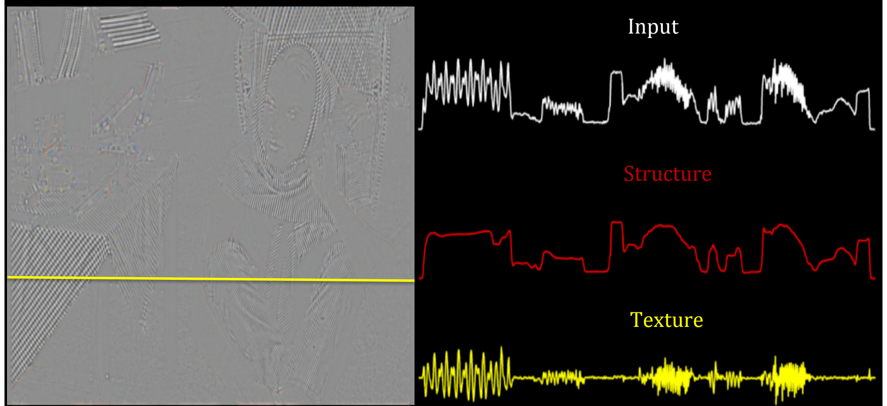
## Illustrative Example


Input

## Illustrative Example


Model2 Structure

## Illustrative Example

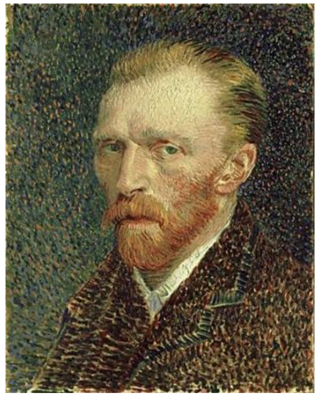



Model2 Texture

## Multiscale Decomposition

$$I(\mathbf{p}) = \sum_{i=0}^{n} T_i(\mathbf{p}) + S_n(\mathbf{p})$$

Input



## Multiscale Decomposition

$$I(\mathbf{p}) = \sum_{i=0}^{n} T_i(\mathbf{p}) + S_n(\mathbf{p})$$

$S_1 (k = 5)$

## Multiscale Decomposition

$$I(\mathbf{p}) = \sum_{i=0}^{n} T_i(\mathbf{p}) + S_n(\mathbf{p})$$

$S_2 (k = 7)$

## Multiscale Decomposition

$$I(\mathbf{p}) = \sum_{i=0}^{n} T_i(\mathbf{p}) + S_n(\mathbf{p})$$

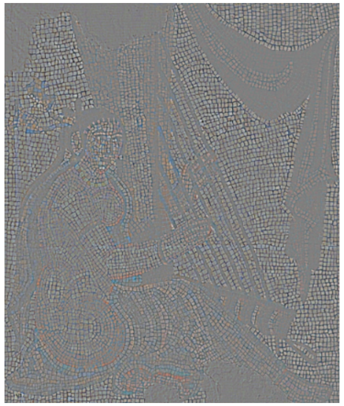
$S_3 (k = 9)$

## Model 2 + Model 1

**Model2 Structure**

## Model 2 + Model 1

Model2 Texture



## Model 2 + Model 1

Model2+Model1

## Model 2 + Model 1

Input

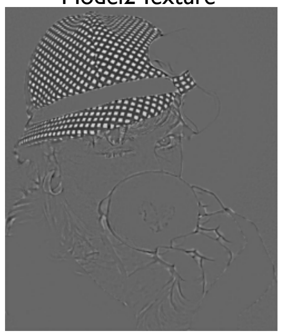
Model2

Model2+Model1

## Model 2 + Model 1

Input

Model2 Texture



Model2+Model1