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)

- $I_p = \text{value of image } I \text{ at position: } p = (p_x, p_y)$

- $F[I] = \text{output of filter } F \text{ applied to image } I$
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

square neighborhood

input

output

average
Equation of Box Average

\[ BA[I]_p = \sum_{q \in S} B_\sigma(p - q) I_q \]

- result at pixel \( p \)
- sum over all pixels \( q \)
- intensity at pixel \( q \)

normalized box function
Square Box Generates Defects

- Axis-aligned streaks
- Blocky results
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

Input \* per-pixel multiplication \* average \* output
box average
Equation of Gaussian Blur

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

\[
GB[I]_p = \sum_{q \in S} G_\sigma(||p - q||) I_q
\]
Spatial Parameter

\[ GB[I]_p = \sum_{q \in S} G_q \left( \| p - q \| \right) I_q \]

- \( p \) is the position of the point of interest.
- \( S \) is the set of points within the window size.
- \( G_q \) is a Gaussian kernel centered at \( q \).
- \( I_q \) is the intensity at point \( q \).

- **input**
- **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

\[
GB[I]_p = \sum_{q \in S} G_\sigma(\|p - q\|) I_q
\]
Blur Comes from Averaging across Edges

Same Gaussian kernel everywhere.
The kernel shape depends on the image content.
Bilateral Filter Definition: an Additional Edge Term

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

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

- **new** 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**

$$GB[I]_p = \sum_{q \in S} G_\sigma(\|p - q\|) I_q$$

**Bilateral filter**

[Aurich 95, Smith 97, Tomasi 98]

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

Normalization space range
Bilateral Filter on a Height Field

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

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

- **space** \( S_s \): spatial extent of the kernel, size of the considered neighborhood.

- **range** \( S_r \): “minimum” amplitude of an edge
Influence of Pixels

Only pixels close in space and in range are considered.
Exploring the Parameter Space

$s_r = 0.1$

$s_r = 0.25$

$s_r = \infty$

(Gaussian blur)

$s_s = 2$

$s_s = 6$

$s_s = 18$

input
Varying the Range Parameter

$s_s = 2$

$s_s = 6$

$s_s = 18$

$s_r = 0.1$

$s_r = 0.25$

$s_r = \infty$

(Gaussian blur)
$s_r = 0.1$
$s_r = 0.25$
$$s_r = \infty$$

(Gaussian blur)
Varying the Space Parameter

\[ s_s = 2 \]

\[ s_s = 6 \]

\[ s_s = 18 \]

\[ s_r = 0.1 \]

\[ s_r = 0.25 \]

\[ s_r = \infty \] (Gaussian blur)

input
\[ s_s = 2 \]
$s_s = 6$
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 $\sim 2s_s$
• Desirable for smoothing: more pixels = more robust
• Different from diffusion that stops at thin lines
Iterating the Bilateral Filter

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

- Generate more piecewise-flat images
- Often not needed in computational photo.
1 iteration
2 iterations
Bilateral Filtering Color Images

For gray-level images

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

intensity difference

For color images

\[ BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_t}(\| C_p - C_q \|) C_q \]

color difference

3D vector (RGB, Lab)

input

output
Hard to Compute

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

- 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.

- Bilateral filter
- Local mode filtering
- Robust statistics
- Partial differential equations

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*!

- 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.

**NL-Means Method:**

Buades (2005)

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

‘Similar’ pixels $p, q$

$\rightarrow$ SMALL vector distance;

$$\| V_p - V_q \|^2$$

‘Dissimilar’ pixels $p, q$

→ LARGE vector distance;

$\| V_p - V_q \|^2$

‘Dissimilar’ pixels \( p, q \)

\( \rightarrow \text{LARGE} \)

vector distance;

\[ \| \mathbf{V}_p - \mathbf{V}_q \|^2 \]

Filter with this!

\( p, q \) neighbors define a vector distance;

**Filter with this:**

\[ \| \mathbf{V}_p - \mathbf{V}_q \|^2 \]

**No spatial term!**

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

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

$\| \mathbf{V}_p - \mathbf{V}_q \|^2$
Vector Distance to $p$ sets
weight for each pixel $q$

$$NLMF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma^2}(\| \mathbf{V}_p - \mathbf{V}_q \|^2) I_q$$
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)

- 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)

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/
NL-Means Filter (Buades 2005)

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)

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/
NL-Means Filter (Buades 2005)

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

Input Image

Structure

Texture

+
Region Covariances as Region Descriptors
Tuzel et al., ECCV 2006

\[ F(x, y) = \phi(I, x, y) \]

\[ F(x, y) = \begin{bmatrix} I(x, y) & \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} & \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial y^2} & x & y \end{bmatrix}^T \]

\[ C_R = \frac{1}{n-1} \sum_{i=0}^{n} (z_k - \mu)(z_k - \mu)^T \]
Main motivation

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

\[ I = S + T \]

\[ S(p) = \frac{1}{Z_p} \sum_{q \in N(p,r)} w_{pq} I(q) \]

- 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

\[ C = LL^T \quad \text{Cholesky Decomposition} \]

\[ S = \{ s_i \} \quad \text{Sigma Points} \]

\[ s_i = \begin{cases} 
\alpha \sqrt{d} \bar{l}_i & \text{if } 1 \leq i \leq d \\
-\alpha \sqrt{d} \bar{l}_i & \text{if } d + 1 \leq i \leq 2d 
\end{cases} \]

- First order statistics can be easily incorporated to the formulation.
Model 1

\[ \Psi(C) = (\mu, s_1, \ldots, s_d, s_{d+1}, \ldots, s_{2d})^T \]

Final representation

\[ w_{pq} \propto \exp \left( -\frac{\|\Psi(C_p) - \Psi(C_q)\|^2}{2\sigma^2} \right) \]

Resulting kernel
Model 2

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

\[ d(p, q) = \sqrt{(\mu_p - \mu_q)C^{-1}(\mu_p - \mu_q)^T} \]

\[ C = C_p + C_q \]

Resulting kernel

\[ w_{pq} \propto \exp \left( -\frac{d(p, q)^2}{2\sigma^2} \right) \]
Illustrative Example

Input

Model 1
\[ \sigma = 0.1 \quad k = 9 \]

Model 2
\[ \sigma = 0.3 \quad k = 9 \]
Illustrative Example

Model2 Structure
Illustrative Example

Model2 Texture

Input

Structure

Texture
Multiscale Decomposition

\[ I(p) = \sum_{i=0}^{n} T_i(p) + S_n(p) \]
Multiscale Decomposition

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

\[S_1(k = 5)\]
Multiscale Decomposition

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

\[ S_2(k = 7) \]
A naive implementation of our structure preserving image smoothing can be alternatively defined as follows:

The extracted structure component at an iteration as an input for the smoothing process at the subsequent iteration: After the smoothing operation is finished on the input image, we smooth the input image by increasing the patch size (by increasing the scale of analysis) at each iteration and by using the intensity information and taking the effect of varying the size of the neighborhood window.

Both of our models have two main parameters, the scale parameter \( k \) controls the level of smoothing as it implicitly determines the size of the neighborhood window. For small values of \( k \), the structure component is more preserved and the texture component is extracted less. On the other hand, the parameter \( k \) causes blurriness. As demonstrated in the experiments, we empirically set the neighborhood size to \( k = 9 \).

In the experiments, we handle color images by computing the patch of an image pixel is represented with a 7-dimensional feature vector:

\[
\begin{pmatrix}
I_x & I_y & \mu_x & \mu_y & \sigma_x & \sigma_y & \mu_
\end{pmatrix}
\]

where \( I_x, I_y \) correspond to the gradient components, \( \mu_x, \mu_y \) are the mean values, and \( \sigma_x, \sigma_y \) are the standard deviations of image patch centered at pixels \( x, y \). Using the set of the covariance matrix can be obtained by simply concatenating the mean value and standard deviations of each patch.

Using Eq. 5, the covariance matrix for a 7-dimensional feature vector can be extended to include other features, like for example RGB components to the feature set does not change the results much but increases the running times.

As an alternative way to measure the similarity between two image pixels with respect to first and second-order feature statistics, we use the following distance measure is defined as:

\[
d(p, q) = \exp\left( -\sum_{k=1}^{10} \frac{(c_k(p) - c_k(q))^2}{2\sigma_k^2} \right)
\]

where \( c_k(p) \) refers to the value of feature \( k \) at pixel \( p \).

The proposed models effectively separate structure and texture, leading to structures to be perceived as fine details. In that respect, it is more important for structure-preserved smoothing models to suppress artifacts and accordingly the local structure information to be captured. As demonstrated in the output images, the proposed smoothing models fully separated texture from structure, with Model 2 slightly better at preserving structure than the intensity values in Equation 5. We empirically found that the output image contains various textured regions with different characteristics, which are well preserved by the proposed models.
Model 2 + Model 1

Model2 Structure
Model 2 + Model 1

Model2 Texture
Model 2 + Model 1
Model 2 + Model 1

- Input
- Model2 Texture
- Model2+Model1