BIL 717
Image Processing
Mar. 14, 2016

Modern Image Smoothing

Erkut Erdem
Hacettepe University
Computer Vision Lab (HUCVL)
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 filtering
- Non-local means denoising
- LARK filter
Today

• Bilateral filtering
• Non-local means denoising
• LARK filter

Acknowledgement: The slides are 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/).
Notation and Definitions

- **Image** = 2D array of pixels

- **Pixel** = intensity (scalar) or color (3D vector)

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

- $F[I]$ = 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 $\sim$ average of its neighbors
Box Average

input → square neighborhood → 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 \( q \)
- Normalized box function
- Intensity at pixel \( q \)
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
Gaussian blur
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

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

- \( \sigma \): size of the window
- small \( \sigma \): limited smoothing
- large \( \sigma \): strong smoothing

Input
How to set $\sigma$

- 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.
Bilateral Filter

No Averaging across Edges

The kernel shape depends on the image content.

[Aurich 95, Smith 97, Tomasi 98]
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**

**space** weight

**not new**

**range** weight

Normalization factor
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]

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

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

reproduced from [Durand 02]
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 \( \sigma_s \): 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

$\sigma_s = 2$

$\sigma_r = 0.1$

$\sigma_s = 6$

$\sigma_r = 0.25$

$\sigma_s = 18$

$\sigma_r = \infty$

(Gaussian blur)
Varying the Range Parameter

\[ \sigma_s = 2 \]
\[ \sigma_s = 6 \]
\[ \sigma_s = 18 \]

\[ \sigma_r = 0.1 \]
\[ \sigma_r = 0.25 \]
\[ \sigma_r = \infty \text{ (Gaussian blur)} \]
\[ \sigma_r = 0.1 \]
$\sigma_r = \infty$  
(Gaussian blur)
Varying the Space Parameter

\[ \sigma_s = 2 \]

\[ \sigma_s = 6 \]

\[ \sigma_s = 18 \]

\[ \sigma_r = 0.1 \]

\[ \sigma_r = 0.25 \]

\[ \sigma_r = \infty \]

(Gaussian blur)
$\sigma_s = 2$
$\sigma_s = 6$
$\sigma_s = 18$
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 $\sim 2\sigma_s$
- 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.
2 iterations
4 iterations
Bilateral Filtering Color Images

For gray-level images

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

For color images

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

input  
output
Hard to Compute

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

- 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

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

• Bilateral filtering
• Non-local means denoising
• LARK filter

Acknowledgement: The slides are 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/).
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 \( \mathbf{V}_p \): a list of neighboring pixel values.

\[
\mathbf{V}_p = \begin{bmatrix}
0.74 \\
0.32 \\
0.41 \\
0.55 \\
\ldots \\
\ldots \\
\ldots \\
\ldots 
\end{bmatrix}
\]


‘Similar’ pixels \( p, q \)

\[ \rightarrow \text{SMALL} \]
vector distance;

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

‘Dissimilar’ pixels \( p, q \)

\( \rightarrow \) LARGE

vector distance;

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

‘Dissimilar’ pixels $p, q$

→ LARGE

vector distance;

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

Filter with this!

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

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

Filter with this:

No spatial term!

\[ NLMF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(\| \vec{V}_p - \vec{V}_q \| ^2) I_q \]
NL-Means Method:
Buades (2005)

pixels \( p, q \) neighbors
Set a vector distance:

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

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

\[
NLMF[ I ]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_r}(\| \vec{V}_p - \vec{V}_q \| ^2) I_q
\]
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).
Fig. 9. NL-means denoising experiment with a natural image. Left: Noisy image with standard deviation 20. Right: Restored image.

- Noisy source image:

• Gaussian Filter

Low noise,
Low detail

• Anisotropic Diffusion

(Note ‘stairsteps’: ~ piecewise constant)

- Bilateral Filter

(better, but similar ‘stairsteps’)

- NL-Means:
  - Sharp,
  - Low noise,
  - Few artifacts.

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.

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

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

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

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

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

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

denoised

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

• Bilateral filtering
• Non-local means denoising
• LARK filter

Acknowledgement: The slides are adapted from the ones prepared by P. Milanfar.
From pixels to patches and to images

Similarities can be defined at different scales..
Pixelwise similarity metrics

- To measure the similarity of two pixels, we can consider:
  - Spatial distance
  - Gray-level distance
Euclidean metrics

- Natural ways to incorporate the two $\Delta$s:
  - Bilateral Kernel [Tomasi, Manduchi, '98] (pixelwise)
  - Non-Local Means Kernel [Buades, et al. '05] (patchwise)
Bilateral Kernel (BL) [Tomasi et al. ‘98]

\[ K(x_l, x, y_l, y) = \exp \left\{ -\frac{\|y_l - y\|^2}{h_r^2} - \frac{\|x_l - x\|^2}{h_d^2} \right\} \]

**Pixels**

**Spatial similarity**
Non-local Means (NLM) [Buades et al. ‘05]

\[ K(x_l, x, y_l, y) = \exp \left\{ -\frac{||y_l - y||^2}{h_r^2} - \frac{||x_l - x||^2}{h_d^2} \right\} \]

Patches

Patch similarity

Spatial similarity

Smoothing effect
Beyond Euclidean metrics

- Better similarity measures
- More effective ways to combine the two $\Delta$s:
  - LARK Kernel [Takeda, et al. ‘07]
  - Beltrami Kernel [Sochen, et al.’98]

```
Riemannian Metric
```

```
"Signal-induced" distance
```

```
"Riemannian Metric"
```
Non-parametric Kernel Regression

- The data fitting problem

\[ y_i = z(x_i) + \varepsilon_i, \quad i = 1, 2, \ldots, P \]

- The regression function

- Zero-mean, i.i.d noise (No other assump.)

- The number of samples

- The sampling position

- Given samples

- The particular form of \( z(x) \) may remain unspecified for now.
Locality in Kernel Regression

• The data model

\[ y_i = z(x_i) + \varepsilon_i, \quad i = 1, 2, \ldots, P \]

• Local representation (N-term Taylor series expansion)

\[
\begin{align*}
    z(x_i) &\approx z(x) + z'(x)(x_i - x) + \frac{1}{2!}z''(x)(x_i - x)^2 \\
            &\quad + \cdots + \frac{1}{N!}z^{(N)}(x)(x_i - x)^N \\
    &= \beta_0 + \beta_1(x_i - x) + \beta_2(x_i - x)^2 \\
    &\quad + \cdots + \beta_N(x_i - x)^N.
\end{align*}
\]

• Note that with a polynomial basis, we only need to estimate the first unknown \( \beta_0 \)
Locality in Kernel Regression

- The data model
  \[ y_i = z(x_i) + \varepsilon_i, \quad i = 1, 2, \ldots, P \]

- Local representation (N-term Taylor series expansion)
  \[ z(x_i) = z(x) + [\nabla z(x)]^T (x_i - x) + \frac{1}{2!} (x_i - x)^T \{\mathcal{H}z(x)\} (x_i - x) + \cdots \]
  \[ = \beta_0 + \beta_1^T (x_i - x) + \beta_2^T \text{vech} \left\{ (x_i - x) (x_i - x)^T \right\} + \cdots, \]

- Note that with a polynomial basis, we only need to estimate the first unknown \( \beta_0 \)
Finding the unknowns via optimization

- We have a local representation with respect to each sample:
  \[ y_1 = \beta_0 + \beta_1^T (x_1 - x) + \beta_2^T \text{vech} \left\{ (x_1 - x) (x_1 - x)^T \right\} + \cdots + \epsilon_1, \]
  \[ y_2 = \beta_0 + \beta_1^T (x_2 - x) + \beta_2^T \text{vech} \left\{ (x_2 - x) (x_2 - x)^T \right\} + \cdots + \epsilon_2, \]
  \[ \vdots \]
  \[ y_P = \beta_0 + \beta_1^T (x_P - x) + \beta_2^T \text{vech} \left\{ (x_P - x) (x_P - x)^T \right\} + \cdots + \epsilon_P, \]

- Estimate the parameters \( \{ \beta_n \}_{n=0}^N \) from the data while giving the nearby samples higher weight than samples farther away.
  \[
  \min_{\{ \beta_n \}} \sum_{i=1}^P \left[ y_i - \beta_0 - \beta_1 (x_i - x) - \beta_2 (x_i - x)^2 \right.
  \]
  \[ - \cdots - \beta_N (x_i - x)^N \left\{ \frac{x_i - x}{h} \right\}^2 \frac{1}{h} K \left( \frac{x_i - x}{h} \right) \]
Finding the unknowns via optimization

- We have a local representation with respect to each sample:

\[
\begin{align*}
y_1 &= \beta_0 + \beta_1^T (x_1 - x) + \beta_2^T \text{vech} \{(x_1 - x)(x_1 - x)^T\} + \cdots + \varepsilon_1, \\
y_2 &= \beta_0 + \beta_1^T (x_2 - x) + \beta_2^T \text{vech} \{(x_2 - x)(x_2 - x)^T\} + \cdots + \varepsilon_2, \\
&\vdots \\
y_P &= \beta_0 + \beta_1^T (x_P - x) + \beta_2^T \text{vech} \{(x_P - x)(x_P - x)^T\} + \cdots + \varepsilon_P,
\end{align*}
\]

- Optimization

\[
\min_{\{\beta_n\}_{n=0}^N} \sum_{i=1}^P \left[ y_i - \beta_0 - \beta_1^T (x_i - x) - \beta_2^T \text{vech} \{(x_i - x)(x_i - x)^T\} - \cdots \right]^2 K(x_i - x)
\]

This term gives the estimated pixel value \(z(x)\).

The choice of the kernel function is open, e.g. Gaussian.

\[
\hat{z}(x) = \sum_{i=1}^P W_i(x, K, h, N) y_i
\]
Defining Data-Adaptive Kernels

• **Classic Kernel:** Locally Linear Filter:

\[ \hat{z}(x) = \hat{\beta}_0 = \sum_i W(x_i, x, N) y_i \]

Uses distance x-x_i

• **Data-Adaptive Kernel:** Locally Non-Linear Filter:

\[ \hat{z}(x) = \hat{\beta}_0 = \sum_i W(x_i, x, y_i, y, N) y_i \]

Uses x-x_i and y-y_i
Recall - Beyond Euclidean metrics

- Better similarity measures
- More effective ways to combine the two $\Delta$s:
  - LARK Kernel [Takeda, et al. ‘07]
  - Beltrami Kernel [Sochen, et al.‘98]
LARK Kernels

\[ K(C_l, x_l, x) = \exp \left\{ -(x_l - x)'C_l(x_l - x) \right\} \]
LARK Kernels

Locally Adaptive Regression Kernel: LARK

\[ K(C_l, x_l, x) = \exp \left\{ -(x_l - x)'C_l(x_l - x) \right\} \]

\[ C_l = \sum_{k \in \Omega_l} \begin{bmatrix} z_{x1}^2(x_k) & z_{x1}(x_k)z_{x2}(x_k) \\ z_{x1}(x_k)z_{x2}(x_k) & z_{x2}^2(x_k) \end{bmatrix} \]

“Structure tensor”
Gradient Covariance Matrix and Local Geometry

Gradient matrix over a local patch:

\[ C_l = \sum_{k \in \Omega_l} \begin{bmatrix} z_{x1}^2(x_k) & z_{x1}(x_k)z_{x2}(x_k) \\ z_{x1}(x_k)z_{x2}(x_k) & z_{x2}^2(x_k) \end{bmatrix} \]

\[ C_l = G^T G \]

\[
G = USV^T = U \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T
\]

Capturing locally dominant orientations.
Image as a Surface Embedded in the Euclidean 3-space

\[ S(x_1, x_2) = \{x_1, x_2, z(x_1, x_2)\} \in \mathbb{R}^3 \]

Arclength on the surface

\[
\begin{align*}
    ds^2 &= dx_1^2 + dx_2^2 + dz^2 \\
    &= dx_1^2 + dx_2^2 + (z_{x_1}dx_1 + z_{x_2}dx_2)^2 \\
    &= (1 + z_{x_1}^2)dx_1^2 + 2z_{x_1}z_{x_2}dx_1dx_2 + (1 + z_{x_2}^2)dx_2^2 \\
    &= (dx_1 \quad dx_2) \begin{pmatrix} 1 + z_{x_1}^2 & z_{x_1}z_{x_2} \\ z_{x_1}z_{x_2} & 1 + z_{x_2}^2 \end{pmatrix} \begin{pmatrix} dx_1 \\ dx_2 \end{pmatrix}
\end{align*}
\]

\[ (x_l - x)^T(C_l + I)(x_l - x) \quad \text{Riemannian metric} \]

Regularization term

\[ K(C_l, x_l, x) = \exp \left\{ -(x_l - x)'C_l(x_l - x) \right\} \]
(Dense) LARK Kernels as Visual Descriptors [Seo and Milanfar ‘10]

\[ K(C_l, x_l, x) = \exp \left\{ -(x_l - x)'C_l(x_l - x) \right\} \]

Measure the similarity of pixels using the metric implied by the local structure of the image
Robustness of LARK Descriptors

Original image  Brightness change  Contrast change  WGN sigma = 10

①

②

③
A Variant Better-suited for Restoration [Takeda et al. ’07]

\[ K(C_l, x_l, x) = \sqrt{\text{det} C_l} \exp \left\{ -(x_l - x)'C_l(x_l - x) \right\} \]
Film Grain Reduction (Real Noise)

Noisy image
Film Grain Reduction (Real Noise)

LARK
Film Grain Reduction (Real Noise)
Adaptive Sharpening/Denoising

• Sharpening the LARK Kernel

\[ S = K - \kappa L \otimes K \]
LARK-based Simultaneous Sharpening/Deblurring/Denoising

• Net effect:
  – aggressive denoising in “flat” areas
  – Selective denoising and sharpening in “edgy” areas

Locally adaptive denoise/deblur filters
Examples

original image

LARK

state-of-the-are methods
Examples
Examples