Filtering Images in the Spatial Domain

Ross Whitaker
SCI Institute, School of Computing
University of Utah
Overview

• Correlation and convolution
• Linear filtering
  – Smoothing, kernels, models
  – Detection
  – Derivatives
• Nonlinear filtering
  – Median filtering
  – Bilateral filtering
  – Neighborhood statistics and nonlocal filtering
## Cross Correlation

- **Operation on image neighborhood and small ...**
  - “mask”, “filter”, “stencil”, “kernel”
- **Linear operations within a moving window**

### Input image

| 100 | 130 | 104 | 99 ... |
| 87  | 95  | 103 | 150 ... |
| 50  | 36  | 150 | 104 ... |
| 20  | 47  | 205 | 77  ... |

### Filter

| 0.0 | 0.1 | 0.0 |
| 0.1 | 0.6 | 0.1 |
| 0.0 | 0.1 | 0.0 |

### Output image

| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |

### Calculation

\[
0.0 \times 87 + 0.1 \times 95 + 0.0 \times 103 + 0.1 \times 50 + 0.6 \times 36 + 0.1 \times 150 + 0.0 \times 20 + 0.1 \times 47 + 0.0 \times 205 = 55.8 \, \| \, 134.8
\]
Cross Correlation

• 1D
  \[ g(x) = \sum_{s=-a}^{a} w(s) f(x + s) \]

• 2D
  \[ g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t) \]

\[
\begin{array}{cccc}
  w(-a, -b) & \cdots & \cdots & w(a, -b) \\
  \vdots & & & \vdots \\
  w(s, t) = & \cdots & w(0, 0) & \cdots \\
  \vdots & & & \vdots \\
  w(-a, b) & \cdots & \cdots & w(a, b)
\end{array}
\]
Correlation: Technical Details

- **Boundary conditions**
  - Pad image with amount \((a,b)\)
    - Constant value or repeat edge values
  - **Cyclical boundary conditions**
    - Wrap or mirroring
Correlation: Technical Details

• **Boundaries**
  - Can also modify kernel - no long correlation

• **For analysis**
  - Image domains infinite
  - Data compact (goes to zero far away from origin)

\[
g(x, y) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w(s, t) f(x + s, y + t)
\]
Correlation: Properties

- **Shift invariant**
  \[ g = w \circ f \quad g(x, y) = w(x, y) \circ f(x, y) \]
  \[ w(x, y) \circ f(x-x_0, y-y_0) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w(s, t) f(x-x_0+s, y-y_0+t) = g(x-x_0, y-y_0) \]

- **Linear**
  \[ w \circ (\alpha e + \beta f) = \alpha w \circ e + \beta w \circ f \]

**Compact notation**

\[ C_{wf} = w \circ f \]
Filters: Considerations

• Normalize
  - Sums to one
  - Sums to zero (some cases, later)

• Symmetry
  - Left, right, up, down
  - Rotational

• Special case: auto correlation

\[ C_{ff} = f \circ f \]
Examples 1

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
1/9 \times & 1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
Examples 2

1 1 1
1/9 * 1 1 1
1 1 1

1 1 1 1 1
1/25 * 1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
Smoothing and Noise

Noisy image

5x5 box filter
Noise Analysis

- Consider an a simple image $I()$ with additive, uncorrelated, zero-mean noise of variance $s$.
- What is the expected rms error of the corrupted image?
- If we process the image with a box filter of size $2a+1$ what is the expected error of the filtered image?

$$\text{RMSE} = \left( \frac{1}{|D|} \sum_{(x,y) \in D} \left( \tilde{I}(x,y) - I(x,y) \right)^2 \right)^{\frac{1}{2}}$$
Cross Correlation Continuous Case

- $f, w$ must be "integrable"
  - Must die off fast enough so that integral is finite

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(s, t) f(x + s, y + t) ds \, dt$$

- Same properties as discrete case
  - Linear
  - Shift invariant
Other Filters

- **Disk**
  - Circularly symmetric, jagged in discrete case

- **Gaussians**
  - Circularly symmetric, smooth for large enough stdev
  - Must normalize in order to sum to one

- **Derivatives - discrete/finite differences**
  - Operators
Pattern Matching/Detection

• The optimal (highest) response from a filter is the autocorrelation evaluated at position zero

\[
\max_{\bar{x}} C_{ff}(\bar{x}) = C_{ff}(0) = \int f(\bar{s})f(\bar{s})d\bar{s}
\]

• A filter responds best when it matches a pattern that looks itself

• Strategy
  - Detect objects in images by correlation with “matched” filter
Match Filter Example

Trick: make sure kernel sums to zero
Match Filter Example
Match Filter Example
Derivatives: Finite Differences

\[ \frac{\partial f}{\partial x} \approx \frac{1}{2h} (f(x + 1, y) - f(x - 1, y)) \]

\[ \frac{\partial f}{\partial x} \approx w_{dx} \circ f \quad w_{dx} = \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \]

\[ \frac{\partial f}{\partial y} \approx w_{dy} \circ f \quad w_{dy} = \begin{bmatrix} -\frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} \]
Derivative Example

\[
\begin{bmatrix}
0 & 0 & 0 \\
-1 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & -1 & 0 \\
0 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]
Convolution

- **Discrete**
  \[ g(x, y) = w(x, y) * f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t) \]

- **Continuous**
  \[ g(x, y) = w(x, y) * f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(s, t) f(x - s, y - t) ds dt \]

- **Same as cross correlation with kernel transposed around each axis**
- **The two operations (correlation and convolution) are the same if the kernel is symmetric about axes**

\[ g = w * f = w^* * f \]

\[ w^* \] reflection of \( w \)
Convolution: Properties

• Shift invariant, linear
• Commutative

\[ f \ast g = g \ast f \]

• Associative

\[ f \ast (g \ast h) = (f \ast g) \ast h \]

• Others (discussed later):
  - Derivatives, convolution theorem, spectrum...
Computing Convolution

- **Compute time**
  - MxM mask
  - NxN image

- **Special case: separable**

  Two 1D kernels

  \[ w = w_x \ast w_y \]

  \[ w \ast f = (w_x \ast w_y) \ast f = w_x \ast (w_y \ast f) \]

  \[ O(M^2N^2) \]

  \[ O(MN^2) \]

  “for” loops are nested 4 deep
Separable Kernels

• Examples
  - Box/rectangle
  - Bilinear interpolation
  - Combinations of partial derivatives
    • \( \frac{d^2 f}{dx dy} \)
  - Gaussian
    • Only filter that is both circularly symmetric and separable

• Counter examples
  - Disk
  - Cone
  - Pyramid
Nonlinear Methods For Filtering

- Median filtering
- Bilateral filtering
- Neighborhood statistics and nonlocal filtering
Median Filtering

- For each neighborhood in image
  - Sliding window
  - Usually odd size (symmetric) 5x5, 7x7, ...
- Sort the greyscale values
- Set the center pixel to the median
- Important: use “Jacobi” updates
  - Separate input and output buffers
  - All statistics on the original image
Median Filter

• **Issues**
  - **Boundaries**
    • Compute on pixels that fall within window
  - **Computational efficiency**
    • What is the best algorithm?

• **Properties**
  - Removes outliers (replacement noise – salt and pepper)
  - Window size controls size of structures
  - Preserves straight edges, but rounds corners and features
Median vs Gaussian

Original + Gaussian Noise

3x3 Median 3x3 Box
Replacement Noise

- Also: “shot noise”, “salt&pepper”
- Replace certain % of pixels with samples from pdf
- Best strategy: filter to avoid outliers
Smoothing of S&P Noise

• It’s not zero mean (locally)
• Averaging produces local biases
Median Filtering

Median 3x3  
Median 5x5
Median Filtering

Median 3x3

Median 5x5
Median Filtering

- Iterate

Median 3x3

2x Median 3x3
Median Filtering

- Image model: piecewise constant (flat)
Order Statistics

- Median is special case of order-statistics filters
- Instead of weights based on neighborhoods, weights are based on ordering of data

<table>
<thead>
<tr>
<th>Neighborhood</th>
<th>Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1, X_2, \ldots, X_N$</td>
<td>$X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(N)}$</td>
</tr>
</tbody>
</table>

Filter

$$F(X_1, X_2, \ldots, X_N) = \alpha_1 X_{(1)} + \alpha_2 X_{(2)} + \ldots + \alpha_N X_{(N)}$$

Neighborhood average (box)

$$\alpha_i = \frac{1}{N}$$

Median filter

$$\alpha_i = \begin{cases} 1 & i = (N + 1)/2 \\ 0 & \text{otherwise} \end{cases}$$

Trimmed average (outlier removal)

$$\alpha_i = \begin{cases} \frac{1}{M} & (N - M + 1)/2 \leq i \leq (N + M + 1)/2 \\ 0 & \text{otherwise} \end{cases}$$
Piecewise Flat Image Models

- Image piecewise flat -> average only within similar regions
- Problem: don’t know region boundaries
Piecewise-Flat Image Models

- Assign probabilities to other pixels in the image belonging to the same region

- Two considerations
  - **Distance**: far away pixels are less likely to be same region
  - **Intensity**: pixels with different intensities are less likely to be same region
Piecewise-Flat Images and Pixel Averaging

Distance (kernel/pdf)

\[ G(x_i - x_j) \]

Distance (pdf)

\[ H(f_i - f_j) \]

Prob pixel belongs to same region as \( i \)

Prob pixel belongs to same region as \( i \)
Bilateral Filter

- **Neighborhood** – sliding window
- **Weight contribution of neighbors according to:**
  \[ f_i \leftarrow k_i^{-1} \sum_{j \in N} f_j G(x_i - x_j) H(f_i - f_j) \]
  \[ k_i = \sum_{j \in N} G(x_i - x_j) H(f_i - f_j) \]
- **G** is a Gaussian (or lowpass), as is **H**, **N** is neighborhood,
  - Often use \( G(r_{ij}) \) where \( r_{ij} \) is distance between pixels
  - Update must be normalized for the samples used in this (particular) summation
- **Spatial Gaussian with extra weighting for intensity**
  - Weighted average in neighborhood with downgrading of intensity outliers
Bilateral Filtering

Gaussian Blurring

Bilateral
Bilateral Filtering

Gaussian Blurring  Bilateral
Nonlocal Averaging

• **Recent algorithm**
  - NL-means, Baudes et al., 2005
  - UINTA, Awate & Whitaker, 2005

• **Different model**
  - No need for piecewise-flat
  - Images consist of pixels with similar neighborhoods
    - Scattered around
      - General area of a pixel
      - All around

• **Idea**
  - Average pixels with similar neighborhoods
Nonlocal Averaging

- **Strategy:**
  - Average pixels to alleviate noise
  - Combine pixels with similar neighborhoods

- **Formulation**
  - $n_{i,j}$ - vector of pixels values, indexed by $j$, from neighborhood around pixel $i$

$$n_i - \text{vector} = n_{i,j}$$
Nonlocal Averaging Formulation

- **Distance between neighborhoods**
  
  \[ d_{i,k} = d(n_i, n_k) = \|n_i - n_k\| = \left( \sum_{j=1}^{N} (n_{i,j} - n_{k,j})^2 \right)^{\frac{1}{2}} \]

- **Kernel weights based on distances**
  
  \[ w_{i,j} = K(d_{i,j}) = e^{-\frac{d_{i,j}^2}{2\sigma^2}} \]

- **Pixel values:** \( f_i \)
Averaging Pixels Based on Weights

- For each pixel, i, choose a set of pixel locations
  - \( j = 1, \ldots, M \)
  - Average them together based on neighborhood weights

\[
g_i \leftarrow \frac{1}{\sum_{j=1}^{M} w_{i,j}} \sum_{j=1}^{M} w_{i,j} f_j
\]
Nonlocal Averaging
Some Details

- **Window sizes:** good range is 5x5->11x11

- **How to choose samples:**
  - Random samples from around the image
    - UINTA, Awate&Whitaker
  - Block around pixel (bigger than window, e.g. 51x51)
    - NL-means

- **Iterate**
  - UNITA: smaller updates and iterate
NL-Means Algorithm

• For each pixel, p
  - Loop over set of pixels nearby
  - Compare the neighborhoods of those pixels to the neighborhood of p and construct a set of weights
  - Replace the value of p with a weighted combination of values of other pixels

• Repeat... but 1 iteration is pretty good
Results

Noisy image (range 0.0-1.0)  Bilateral filter (3.0, 0.1)
Results

Bilateral filter (3.0, 0.1)

NL means (7, 31, 1.0)
Results

Bilateral filter (3.0, 0.1)  NL means (7, 31, 1.0)
Less Noisy Example
Less Noisy Example
Results

Original  Noisy  Filtered
Checkerboard With Noise

Original  Noisy  Filtered
Quality of Denoising

- $\sigma$, joint entropy, and RMS-error vs. number of iterations
MRI Head
MRI Head
Fingerprint
Fingerprint
Results

Original  Noisy  Filtered
Results

Original  Noisy  Filtered
Results

Original

Noisy

Filtered
Fractal

Original  Noisy  Filtered
Piecewise Constant

- Several 10s of Iterations
- Tends to obliterate rare events
Texture, Structure