ECE 484 Digital Image Processing
Lec 08 - Non Linear Filters

Zhu Li
Dept of CSEE, UMKC
Office Hour: Tu/Th 2:30-4pm, FH560E,
Email: lizhu@umkc.edu, Ph: x 2346.
http://l.web.umkc.edu/lizhu
Outline

- Recap of Freq Domain Filtering
- Non Linear Filters
- Summary
2-D FT illustrated

- FFT2 illustrated:
  - Real-valued $x(m, n)$
  - $a(u, v, m, n) = e^{-j2\pi\left(\frac{um}{N} + \frac{vn}{N}\right)}$

  - $y(u, v)$
    - Real part: $\text{real}(y(u, v))$
    - Imaginary part: $\text{imag}(y(u, v))$
FFT on Lena

- Lena in freq domain representation

```matlab
% lena fft analysis
ff = fft2(im);
figure(11); colormap('gray'); axis off;
subplot(1,3,1); imagesc(im); title('lena');
subplot(1,3,2); imagesc(log(abs(ff))); title('abs(fft2(im))');
subplot(1,3,3);
imagesc(angle(ff));title('angle(fft2(im)))');
```
Freq Domain Filtering

- Filtering in spatial domain is multiplication in freq domain.
- Can do Low Pass, High Pass, and Band Pass directly.

<table>
<thead>
<tr>
<th>Lowpass Filter</th>
<th>Mesh</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ideal:</strong> $H(u,v) = \begin{cases} 1 &amp; \text{if } D(u,v) \leq D_0 \ 0 &amp; \text{if } D(u,v) &gt; D_0 \end{cases}$</td>
<td><img src="image1" alt="Ideal Mesh" /></td>
<td><img src="image2" alt="Ideal Image" /></td>
</tr>
<tr>
<td><strong>Butterworth:</strong> $H(u,v) = \frac{1}{1 + \left(\frac{D(u,v)}{D_0}\right)^{2n}}$</td>
<td><img src="image3" alt="Butterworth Mesh" /></td>
<td><img src="image4" alt="Butterworth Image" /></td>
</tr>
<tr>
<td><strong>Gaussian:</strong> $H(u,v) = e^{-D^2(u,v)/2D_0^2}$</td>
<td><img src="image5" alt="Gaussian Mesh" /></td>
<td><img src="image6" alt="Gaussian Image" /></td>
</tr>
</tbody>
</table>
Bandpass/Notch Filter

- Removing repetitive patterns
  - Used to remove repetitive "Spectral" noise from an image
  - Act like a narrow highpass filter, but they "notch" out frequencies other than the dc component
  - Attenuate a selected frequency (and some of its neighbors) and leave other frequencies of the Fourier transform relatively unchanged

![Noisy Image](image1)
![Fourier Spectrum of Image](image2)

![Image after Butterworth notch filters](image3)
![Spectrum of image after Butterworth notch filters](image4)
basis images: DCT vs DFT

1D-DCT
\[
a(0, n) = \sqrt{\frac{1}{N}} \quad u = 0
\]
\[
a(u, n) = \sqrt{\frac{2}{N}} \cos \left(\frac{\pi(2n + 1)u}{2N}\right) \quad u = 1, 2, \ldots, N - 1
\]

N=32

1D-DFT
\[
a(u, n) = e^{-j2\pi \frac{un}{N}} = \cos \left(2\pi \frac{un}{N}\right) - j\sin \left(2\pi \frac{un}{N}\right)
\]
2D DCT

\[
C_x[k_1, k_2] = \begin{cases} 
\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} 4x[n_1, n_2] \cos \left( \frac{\pi}{2N_1} k_1 (2n_1 + 1) \right) \cos \left( \frac{\pi}{2N_2} k_2 (2n_2 + 1) \right), & 0 \leq k_1 < N_1, 0 \leq k_2 < N_2 \\
0 & \text{otherwise}
\end{cases}
\]

\[
x[n_1, n_2] = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} w_1[k_1] w_2[k_2] C_x[k_1, k_2] \cos \left( \frac{\pi}{2N_1} k_1 (2n_1 + 1) \right) \cos \left( \frac{\pi}{2N_2} k_2 (2n_2 + 1) \right), 0 \leq n_1 < N_1, 0 \leq n_2 < N_2 \\
\end{cases}
\]

with

\[
w_1[k_1] = \begin{cases} 
\frac{1}{2}, & k_1 = 0 \\
1, & 1 \leq k_1 < N_1
\end{cases}
\]

\[
w_2[k_2] = \begin{cases} 
\frac{1}{2}, & k_2 = 0 \\
1, & 1 \leq k_2 < N_2
\end{cases}
\]

1d 8-pt DCT basis

2d DCT 8x8 basis
DCT Energy Compaction

- Better Compaction Than DFT
  - real only representation
  - no sharp edge when construct DCT

- DCT is the basis for JPEG compression
Outline

- Recap of Lec 05
- HW-1 & Tutorials
- Non Linear Filters
- Summary
Denoising Problem Definition

\[ y = x + n \]

\[ \hat{n} = y - \hat{x} \]

\[ \text{MSE / PSNR} \]

Visual Quality

\( p \)

\( Z. \ Li, \ ECE484 \ Digital \ Image \ Processing, \ 2019 \)
Sources of Image Noises

- Metrics

  Mean Square Error
  \[ \text{MSE} = \| \hat{x} - x \|_2^2 \]

  Peak Signal-to-Noise ratio
  \[ \text{PSNR} = 20 \log_{10} \frac{255}{\sqrt{\text{MSE}}} \]

- Multiplicative Noise (shot noise)
- Additive Noise (read+amplifier noise)
Problem with Gaussian Smoothing

- Loss of fine/high freq details at texture rich region

\[ \hat{x}(i) = \frac{1}{C_i} \sum_j y(j) e^{-\frac{||i-j||^2}{2\sigma^2}} \]
Edge Preserving Filtering

- **Edges** ⇒ smooth only along edges
- **“Smooth”** regions ⇒ smooth isotropically
Non Linear Order Statistics Filters

Order stats filters

\[ g(x, y) = \text{median}\{f(s, t)\} \]

\[ g(x, y) = \max_{(s, t) \in W(x, y)} \{f(s, t)\} \]

\[ g(x, y) = \min_{(s, t) \in W(x, y)} \{f(s, t)\} \]
Denoising with Median Filter

- Media Filter (why non-linear?) performs better

<table>
<thead>
<tr>
<th>Noisy Image</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>97 103 83 82 81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>98 103 105 108 97</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99 255 102 101 95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101 103 107 255 93</td>
<td></td>
<td>101 97 103 83 82 81</td>
</tr>
<tr>
<td>93 101 112 108 107</td>
<td></td>
<td>98 103 105 108 97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>99 255 105 101 95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101 103 107 255 93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>93 101 112 108 107</td>
</tr>
</tbody>
</table>

- Results
Medialn Filter

- Problems with Median Filter

- good at removing outliers like pepper & salt noise
- certain structure is kept (e.g., straight edge)
- details lost, and structure is determined by the kernel size
More Denoising Filtering Results

- Mean, Gaussian vs Median

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3x3</strong></td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>5x5</strong></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>7x7</strong></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
</tbody>
</table>

- salt & pepper
- Gaussian noise
Denoising Filtering Challenges

- Mean: blurs image, removes simple noise, no details are preserved
- Gaussian: blurs image, preserves details only for small $\sigma$.
- Median: preserves some details, good at removing strong noise

Can we do better?
- Yes, Bilateral Filters
What Is Bilateral Filter?

- **Bilateral**
  - Affecting or undertaken by two sides equally

- **Property:**
  - Convolution filter
  - Smooth image but preserve edges
  - Operates in the domain (geometry) and the range (photometric) of image

Gaussian filter       Bilateral filter
It is clear that in weighting this neighborhood, we would like to preserve the step
Geometry Weights

\[ I(p) \]

\[ W_c(p) = e^{x p\left(-\frac{(u - p)^2}{2\sigma_c^2}\right)} \]
Filtered Output

- Weighted sum on the $W_c(p)$

\[ W_c(p) = \exp\left(-\frac{(u - p)^2}{2\sigma^2_c}\right) \]
Edge loss

- Edge is smoothed/lost

\[ W_c(p) = \exp\left(-\frac{(u - p)^2}{2\sigma_c^2}\right) \]
What is wrong here?

- Filter weights not reflecting affinity in attributes/pixel values

\[ W_c(p) = \exp\left(-\frac{(u - p)^2}{2\sigma_c^2}\right) \]
Cause of the problem

- Geometry derived weights not reflecting value affinity
Introducing Photometric weights

\[ W_s(p) = \exp\left(-\frac{(I(u) - I(p))^2}{2\sigma_s^2}\right) \]

\[ W_c(p) = \exp\left(-\frac{(u - p)^2}{2\sigma_c^2}\right) \]
Bilateral Filtering

- Filter Weights derived from both geometric and photometric distances

\[
I'(u) = \frac{\sum_{p \in \mathcal{N}(u)} e^{\frac{\|u-p\|^2}{2\sigma_c^2}} e^{\frac{|I(u)-I(p)|}{2\sigma_s^2}} I(p)}{\sum_{p \in \mathcal{N}(u)} e^{\frac{\|u-p\|^2}{2\sigma_c^2}} e^{\frac{|I(u)-I(p)|}{2\sigma_s^2}}}
\]

Denoise

Feature preserving

Normalization
Bilateral Kernel Property

- Bilateral filter kernels

\[ I'(u) = \frac{\sum_p W_c(p) * W_s(p) * I(p)}{\sum_p W_c(p) * W_s(p)} \]

- Per each sample, we can define a bilateral ‘kernel’ that averages its neighborhood
- This kernel changes from sample to sample, that is why it is non-linear.
- The sum of the kernel entries is 1 due to the normalization,
- The center entry in the kernel is the largest,
- Subject to the above, the kernel can take any form (as opposed to filters which are monotonically decreasing).
- Controlled by neighborhood size \( N(u) \), photometric and geometry kernels, \( \sigma_s, \sigma_c \)

https://users.soe.ucsc.edu/~manduchi/Papers/ICCV98.pdf
Illustration of bilateral filter changes

input
Bilateral Filter

- Illustration of bilateral filter changes
Bilateral Filter

input

$W_c$
Bilateral Filter

input

$W_c$

$W_s$
Bilateral Filter

input

$W_c$

$W_s$

$W_s * W_c$
Bilateral Filter

Filtering process

input

\( W_c \)

\( W_s \)

\( W_s * W_c \)

Output
Bilateral Filter Results

Original
Bilateral Filter Results

\[ \sigma_c = 3, \]
\[ \sigma_s = 3 \]
Bilateral Filter Results

\[ \sigma_c = 6, \quad \sigma_s = 3 \]
Bilateral Filter Results

$\sigma_c = 12,$
$\sigma_s = 3$
Bilateral Filter Results

\[ \sigma_c = 12, \quad \sigma_s = 6 \]
Bilateral Filter Results

\[ \sigma_c = 15, \quad \sigma_s = 8 \]
Redundancy in natural images

- Exploiting self similar patches in images to constrain the local filter kernels
  - Non Local Means (NLM) filters
Non Local Means (NLM)

- Baudes et al. (2005) use a weighted average based on similarity
  \[
  \hat{x}(i)_{BL} = \frac{1}{C_i} \sum_j y(j) e^{-\frac{|y(i) - y(j)|^2}{2\sigma^2} e^{-\frac{\|i-j\|^2}{2\rho^2}}}
  \]

- \[
  \hat{x}(i)_{NLM_{1x1}} = \frac{1}{C_i} \sum_j y(j) e^{-\frac{|y(i) - y(j)|^2}{2\sigma^2}}
  \]

- BL filter's spatial kernel is turned off
- intensity kernel reflects similarity between self similar patches: pixel \( j \) and its neighborhood \( y(j) \) is defined as say 5x5 block centered at \( j \).
Why NLM is Better?

- NLM filter weights precisely captures local edge structure as desired.

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).
NLM Denoising Results

- Gaussian Smoothing
- Anisotropic Filtering
- Bilateral Filtering
- Windowed Weiner
- Hard WT
- Soft WT
- NLM
Matlab Implementation

- `imnlmfilt()`
  - `[filteredImage, estDoS] = imnlmfilt(noisyImage);`
The bilateral filter is a powerful filter:
- Can work with any reasonable distances function $W_s$ and $W_c$,
- Easily extended to higher dimension signals, e.g. Images, video, mesh, animation data etc.
- Easily extended to vectored-signals, e.g. Color images, etc.

Bilateral Mesh Denoising

[ Fleishman et al, siggraph 03 ]

Point Cloud Attributes Coding - LOD based prediction
Non-Linear Filters

- Median, Mean and Gaussian filters have different performances in denoising images
- Bilateral filter reflects both pixel value affinity, as well as geometry/graph affinity
- A special case for graph signal processing
- Many applications.