Image Noise and Filtering

CMPE 264: Image Analysis and Computer Vision
Winter’03, Hai Tao
1/14/03
1/16/03
Estimating acquisition noise

- Noise introduced by imaging system
- Simple model
  - Assumption: noise at each pixel is independent, characterized by its mean and standard deviation
  - Estimating the mean and the standard deviation $\sigma$ for each pixel

For each $i, j = 0,..., N - 1, let$

$$\bar{I}(i, j) = \frac{1}{N} \sum_{k=0}^{N-1} I_k(i, j)$$

$$\sigma(i, j) = \left(\frac{1}{N - 1} \sum_{k=0}^{N-1} (I_k(i, j) - \bar{I}(i, j))^2\right)^{1/2}$$

- For most imaging system, $\sigma \approx 2.5$
Estimating acquisition noise

- Estimating auto-covariance
  - In reality, the noise in neighboring pixels is not independent
  - The correlation is described by auto-covariance
  - If we assume auto-covariance of noise is the same everywhere in the image, then

\[
C_{II}(i', j') = \frac{1}{N^2} \sum_{i=0}^{N-i'-1} \sum_{j=0}^{N-j'-1} (I_k(i, j) - \bar{I}(i, j))(I_k(i + i', j + j') - \bar{I}(i + i', j + j'))
\]

- Example: How to compute \( C_{II}(2,1) \) for a 10x10 image?

\[
N_{i'} = 7, N_{j'} = 8, \text{for each } i' = 2, j' = 1, \text{compute}
\]

\[
C_{II}(2,1) = \frac{1}{10^2} \sum_{i=0}^{7} \sum_{j=0}^{8} (I_k(i, j) - \bar{I}(i, j))(I_k(i + 2, j + 1) - \bar{I}(i + 2, j + 1))
\]
Estimating acquisition noise

- Auto-covariance for a typical imaging system. Notice that the covariance along the horizontal direction, which is a characteristic often observed in CCD cameras.
Modeling image noise

- Additive noise model
  
  Random noise \( n(i, j) \) added to pixel value \( I(i, j) \)
  
  \[ \hat{I}(i, j) = I(i, j) + n(i, j) \]

- Signal-to-noise ratio (SNR), often expressed in decibel

  \[ SNR = \frac{\sigma_s}{\sigma_n} \]

  \[ SNR_{dB} = 10\log_{10}\left(\frac{\sigma_s}{\sigma_n}\right) \]

- 20 dB means \( \frac{\sigma_s}{\sigma_n} = 100 \)
Modeling image noise

- **Gaussian noise** - white Gaussian, zero-mean stochastic process
  - White – \( n(i,j) \) independent in both space and time
  - Zero-mean – \( \bar{I}(i, j) = 0 \)
  - Gaussian - \( n(i,j) \) is random variable with distribution
    \[
    p(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}
    \]

- **Impulsive noise** – also called peak, spot, or salt and pepper noise, caused by transmission errors, faulty CCD sites, etc.
  - \( I_{sp}(i, j) = \begin{cases} I(i, j) & x < l \\ I_{\min} + y(I_{\max} - I_{\min}) & x \geq l \end{cases} \)
  - \( x, y \in [0,1] \) are two uniformly distributed random variables
Modeling image noise

Example: Gaussian noise and salt and pepper noise, $l=0.99$

Figure 3.1  (a) Synthetic image of a 120 × 120 grey-level “checkerboard” and grey-level profile along a row. (b) After adding zero-mean Gaussian noise ($\sigma = 5$). (c) After adding salt and pepper noise (see text for parameters).
Filtering noise - denoise

- **Goal:** recover $I(i, j)$ from $\hat{I}(i, j) = I(i, j) + n(i, j)$
- **Methods:** linear filtering and nonlinear filtering
- **Linear filtering:** replace the original pixel by the weighted sum of its neighboring pixel. The weights are the filter coefficients

\[
I_A(i, j) = I \ast A = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} A(h, k) I(i-h, i-k)
\]
Mean filter – smoothing by averaging

The basic assumption is that the noise has higher frequency and the signal has lower frequency. Noise can be canceled by low-pass filtering.

A averaging filter with m=5 is

\[
A_{avg} = \frac{1}{25} \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Why does it work?

- Explanation in spatial domain: reduce standard deviation by 5
- Explanation in frequency domain: suppress high frequency components since \( F(I \ast A) = F(I)F(A) \)
Gaussian filter

The Fourier transform of a Gaussian kernel is also Gaussian, better low-pass filter than averaging filter

\[ G(h, k) = e^{-\frac{h^2+k^2}{2\sigma^2}} \]

Gaussian kernel is separable, which means 2D Gaussian filtering can be implemented as two 1D Gaussian filtering

\[
I_G = I * G = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} G(h, k) I(i-h, i-k)
\]

\[
= \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} e^{-\frac{h^2+k^2}{2\sigma^2}} I(i-h, i-k)
\]

\[
= \sum_{h=-m/2}^{m/2} e^{-\frac{h^2}{2\sigma^2}} \sum_{k=-m/2}^{m/2} e^{-\frac{k^2}{2\sigma^2}} I(i-h, i-k)
\]
Gaussian filter

**SEPAR_FILTER**
- Build a 1-D Gaussian filter $g$, of width $\sigma_g = \sigma_G$
- Convolve each row of $I$ with $g$, yielding a new image $I_r$
- Convolve each column of $I_r$ with $g$, yielding a new image $I_G$

**Construction of Gaussian filter**
- In order to subtend 98.76% of the area in the Gaussian function, for a given $\sigma$, the width of the filter need to be five times of $\sigma$, or $w = 5\sigma$
- Integer filter: Gaussian filter can be implemented efficiently by converting the real numbers into rational numbers with a common denominator: scale the filter so that the smallest components are 1s, replace the other entries with the closest integers
Gaussian filter

Example

- Filter image with Gaussian noise
- Filter image with salt and pepper noise

Figure 3.2  (a) Results of applying Gaussian filtering (kernel width 5 pixel, $\sigma = 1$) to the “checkerboard” image corrupted by Gaussian noise, and grey-level profile along a row. (b) Same for the “checkerboard” image corrupted by salt and pepper noise.
Nonlinear filtering

- Median filtering
  - Algorithm MED_FILTER
  - For each pixel \( I(i,j) \) and its \( n \times n \) neighborhood,
    - Sort its neighboring pixels \( \{ I(i + h, i + k) \mid h, k \in [-n/2, n/2] \} \)
    - Assign the median value to \( I(i,j) \)
  - Example: with window/mask size 3×3

\[
\begin{array}{cccccc}
22 & 17 & 102 & 105 & 106 & 102
\end{array}
\]
\[
\begin{array}{cccccc}
21 & 21 & 20 & 102 & 102 & 102
\end{array}
\]
\[
\begin{array}{cccccc}
19 & 22 & 20 & 102 & 102 & 102
\end{array}
\]
\[
\begin{array}{cccccc}
24 & 21 & 101 & 101 & 104 & 104
\end{array}
\]
\[
\begin{array}{cccccc}
19 & 18 & 101 & 108 & 101 & 101
\end{array}
\]
\[
\begin{array}{cccccc}
22 & 17 & 102 & 105 & 106 & 102
\end{array}
\]
\[
\begin{array}{cccccc}
21 & 21 & 99 & 102 & 102 & 102
\end{array}
\]
\[
\begin{array}{cccccc}
19 & 21 & 97 & 102 & 102 & 102
\end{array}
\]
\[
\begin{array}{cccccc}
24 & 21 & 97 & 101 & 104 & 104
\end{array}
\]
\[
\begin{array}{cccccc}
19 & 18 & 101 & 108 & 101 & 101
\end{array}
\]
Median filtering

- Example with Gaussian noise
  - Preserve discontinuity in the signal.
  - No contribution from pixels with large noise
  - Expensive to compute
Homework

Use Matlab to complete the following experiment

- Read in a gray-level image
- Add Gaussian noise to the image with $\sigma = 10$
- Implement
  - 5 by 5 Separable Gaussian filter with $\sigma = 0.8$
  - 5 by 5 Median filter
- Turn in
  - Matlab code for adding noise and the two filtering algorithms
  - The original image, the image with noise, and the filtering results
  - Derivation of the discrete Gaussian filter