Image Restoration

ENE 461 Class 7 (1/2008), August 1, 2008

Werapon Chiracharit, Ph.D., ENE, KMUTT
werapon.chi@kmutt.ac.th

Degradation and Noises

Linear Space-Invariant System

$$\text{Image}_{\text{recorded}}(x,y) = \text{Degrade}(\,\text{Image}_{\text{true}}(x,y)\,) + \text{Noise}(x,y)$$

$$\hat{f}(x,y) = f(x,y) * h(x,y) + n(x,y)$$

where $h(x,y)$ is Degradation Function
and Signal-to-Noise Ratio (SNR) = $\sigma_{\text{true}}/\sigma_{\text{noise}}$

- from Imaging
- from Digitization
- from Transmission

2

Noise Probability Distribution

- Uniform, $p(i) = 1/(i_{max}-i_{min})$ for $i_{min} \le i \le i_{max}$
  $= 0$ otherwise
- Gaussian, $p(i) = 1/\sigma\sqrt{2\pi}\ e^{-(i-\mu)^2/2\sigma^2}$
  for N(0,1) $\Rightarrow$ mean $\mu = 0$ and variance $\sigma^2 = 1$
- Poisson, $p(i) = \lambda^i e^{-\lambda} / i!$ where λ is mean
- Salt & Pepper, $p(i) = p_{pepper}$ for $i = i_{pepper}$
  $= p_{salt}$ for $i = i_{salt}$ and $i_{salt} > i_{pepper}$
  $= 0$ otherwise

3

Noise Distribution (Cont'd)

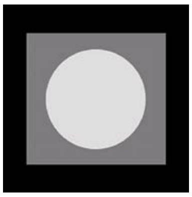
- Speckle or Multiplicative Noise,
  $$I_{recorded} = I_{true} + I_{true} \times N$$
  where N is randomly normal distribution with $\mu = 0$

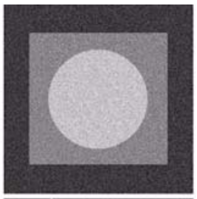
- Periodic Noise $\Leftarrow$ Frequency Filter (Band Reject)

4

Noise Image

Original Image | Gaussian Noise | Salt & Pepper Noise





Gaussian PDF | Impulse PDF

5

High-Frequency Noise

- An image contains mostly low frequency information.
- The noise is dominant for high frequencies.
- Its effects can be reduced by using some kind of spatial lowpass filter.

  (Mean Filter and Gaussian Filter)

6

Salt & Pepper Noise

Median Filter

- Sorting all pixel values from the surrounding neighborhood.
- Replacing the central pixel with the middle sorted value.

2, 10, 11, 12, **12**, 13, 15, 15, 16

13	12	15
10	2	12
16	15	11

13	12	15
10	**12**	12
16	15	11

7

Salt & Pepper Noise (Cont'd)

Original Image
with Salt & Pepper Noise

Mean Filter

Median Filter

Max & Min Filter

- Maximum Filter ⇒ Brightening
  I=nlfilter(I,[m n],'max(x(:))');
- Minimum Filter ⇒ Darkening
  I=nlfilter(I,[m n],'min(x(:))');


Max Filtered


Original Image


Min Filtered

9

Spatial Wiener Filter

- Adaptive Filter ⇒ characteristics are changed according to the grayscale values under the mask (local area).

$$I_{restored} = \mu_{mask} + \frac{\sigma^2_{mask}}{(\sigma^2_{mask}+\sigma^2_{entire})} (I_{recorded} - \mu_{mask})$$

High σ^2_{mask} → Detail or Edges, $I_{restored} \approx I_{recorded}$

Low σ^2_{mask} → Background, $I_{restored} \approx \mu_{mask}$

10

Spatial Wiener Filter (Cont'd)

$$I_{restored} = \mu_{mask} + \frac{\max\{0, \sigma^2_{mask} - \sigma^2_{average}\}}{\max\{\sigma^2_{mask}, \sigma^2_{average}\}} (I_{recorded} - \mu_{mask})$$

where $\sigma^2_{average} = \text{mean}(\sigma^2_{mask})$

- It is a kind of low-pass filter (blurring).

11

Lucy-Richardson Algorithm

For the convolution result for each i^{th} pixel,

$\hat{f}_i = \sum_j h_{ij} f_j$ $\Leftarrow$ Expected Value

and $\sum_j h_{ij} = 1$ $\Leftarrow$ Cumulative Probability (Lowpass Filter Mask)

Unbluring, $\tilde{f}_j = \sum_i g_{ij} \hat{f}_i$

- Probability, $P(\hat{f}_i \mid f_j) = h_{ij}$
- Likelihood, $g_{ij} = L(f_j) = P(f_j \mid \hat{f}_i)$

$$= P(\hat{f}_i \mid f_j)\, P(f_j) / P(\hat{f}_i)$$
$$= h_{ij} f_j / \hat{f}_i$$
$$= h_{ij} f_j / \textstyle\sum_j h_{ij} f_j$$

12

Lucy-Richardson Algorithm (Cont'd)

Therefore the iterative restoration,

$$f_j^{k+1} = \sum_i g_{ij}\hat{f}_i$$
$$= \sum_i h_{ij} f_j^k \hat{f}_i / \sum_j h_{ij} f_j^k$$
$$= f_j^k \sum_i \left(\frac{h_{ij}\hat{f}_i}{\sum_j h_{ij} f_j^k} \right)$$

Or $I_{k+1}(x,y) = I_k(x,y) \left(h(-x,-y) * \frac{\hat{I}(x,y)}{h(x,y) * I_k(x,y)} \right)$

h(x-1,y-1)	h(x-1,y)	h(x-1,y+1)
h(x,y-1)	h(x,y)	h(x,y+1)
h(x+1,y-1)	h(x+1,y)	h(x+1,y+1)

Mask

$*$

I(x-1,y-1)	I(x-1,y)	I(x-1,y+1)
I(x,y-1)	I(x,y)	I(x,y+1)
I(x+1,y-1)	I(x+1,y)	I(x+1,y+1)

Image

13

Lucy-Richardson Algorithm (Cont'd)

Conversed or not when $f_j^k \rightarrow f_j^{k+1}$

- Log-Likelihood, $\ln(L(f_j)) = \ln(h_{ij}) + \ln(f_j) - \ln(\sum_j h_{ij} f_j)$

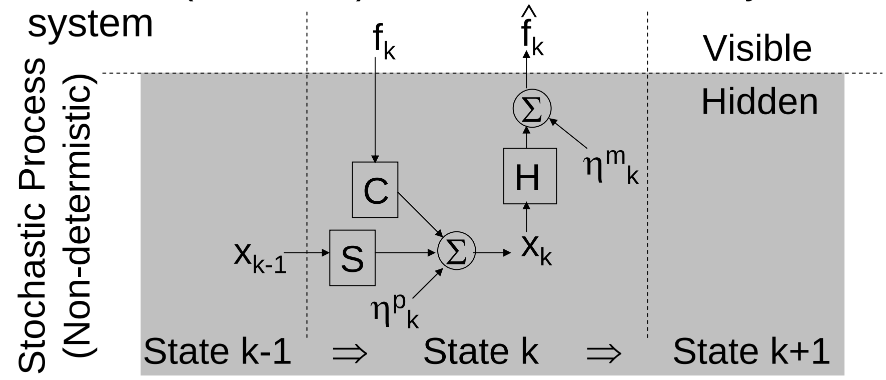
- $\partial \ln(L(f_j))/\partial f_j = 1/f_j - \sum_j h_{ij}/\sum_j h_{ij} f_j = 1/f_j - 1/\sum_j h_{ij} f_j = 0$
  (Stop at Critical Point when $f_j^{k+1} \approx f_j^k$)

- $\partial^2 \ln(L(f_j))/\partial f_j^2 = -1/f_j^2 + \sum_j h_{ij}/\sum_j h_{ij}^2 f_j^2$
  $= -1/f_j^2 + 1/\sum_j h_{ij}^2 f_j^2$
  < 0
  (Maximum Likelihood)

14

Kalman Filter

Recursive (feedback) estimator for linear dynamic system



State-Space, $x_k = Sx_{k-1} + Cf_k + \eta^p_k$

Current State; Previous State; State Transition Model; Control Input Model; Input; Process Noise ~ $N(0,v_p)$

Observation, $\hat{f}_k = Hx_k + \eta^m_k$

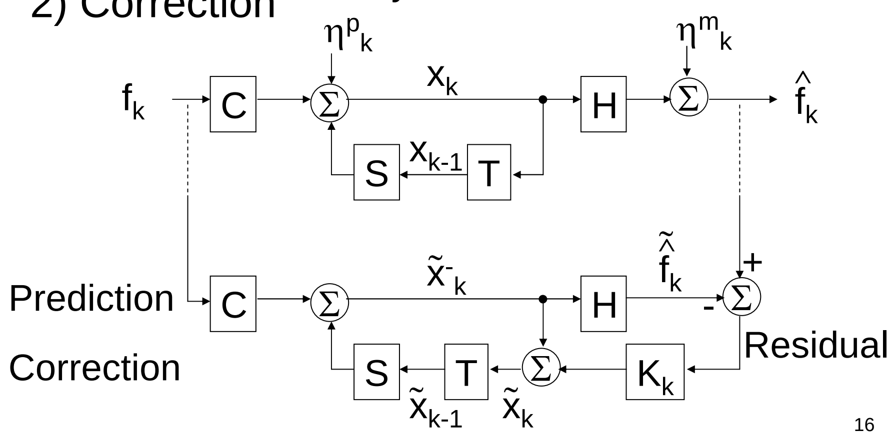
Measurement; Observation Model; Current State; Measurement Noise ~ $N(0,v_m)$

Kalman Filter (Cont'd)

It is a real time process. There are 2 steps

1) Prediction ⇒ Optimal (Min Mean-Square-Error) Noisy Data

2) Correction



Kalman Filter (Cont'd)

- Prediction

  Priori State Estimate, $\tilde{x}^-_k = S\tilde{x}_{k-1} + Cf_k$

  Priori Covariance, $V^-_k = S^2V_{k-1} + v_p$

- Correction

  Kalman Gain, $K_k = HV^-_k / (H^2V^-_k + v_m)$

  Priori State Estimate, $\tilde{x}_k = \tilde{x}^-_k + K_k(\hat{f}_k - H\tilde{x}^-_k)$

  Posteriori Covariance, $V_k = V^-_k(1-HK_k)$

Exercise: Prove these!

17

Kalman Filter (Cont'd)

1) Initialization

   Prediction Seed $\Rightarrow \tilde{I}_k = I_k$

   Error Seed $\Rightarrow \tilde{E}_k = v$

2) Correction

   Compute Kalman Gain, $K_k = \tilde{E}_k/(\tilde{E}_k+v)$

   Update Image Prediction with Measurement, $\hat{I}$

   $I_k = H\tilde{I}_k + (1-H)\hat{I} + K_k(\hat{I}-\tilde{I}_k)$

   Update Variance Estimate, $E_k = \tilde{E}_k(1-K_k)$

3) Prediction

   Next Image $\Rightarrow \tilde{I}_{k+1} = I_k$

   Next Variance $\Rightarrow \tilde{E}_{k+1} = E_k$

4) Update Value

   $\tilde{I}_k = \tilde{I}_{k+1}$

   $\tilde{E}_k = \tilde{E}_{k+1}$

5) Repeat 2) Until OK

18

Optimization Method

Gradient Descent (Newton's Method)

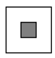
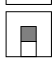
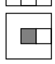
- Cost function to be minimized, $E(x)$
- Chain rule, $dE/dt = (dE/dx)(dx/dt)$
- for $t \rightarrow t+1$ then $E_t \rightarrow E_{t+1}$ and $x_t \rightarrow x_{t+1}$
  let $dx/dt = -dE/dx$
  $dE/dt = -(dE/dx)^2 \rightarrow$ Downhill
- $x_{t+1} - x_t = -dE/dx$
  $x_{t+1} = x_t - dE/dx \quad \rightarrow$ Update Rule
- Repeat until OK (no significantly changed)

19

Optimization Method (Cont'd)

- Initialize randomly the restored image, $I_{restored}$
- Create the cost function

$$
\begin{aligned}
E = \sum_{x,y} \{ & M(x,y) \times [\, I_{recorded}(x,y) - I_{restored}(x,y)\,]^2 \\
& + \lambda[\, I_{restored}(x+1,y) - I_{restored}(x,y)]^2 \\
& + \lambda[\, I_{restored}(x,y+1) - I_{restored}(x,y)]^2 \; \}
\end{aligned}
$$

- Restored image must be close to the observed image.
- Neighboring pixels should be similar (smoothness constraint).
- $M(x,y) = 1$ for data exists
  $= 0$ for no data (missing)

- Find $dE/dI_{restored}(x,y)$
- Update rule with α step-size,
  $I_{restored}^{k+1}(x,y) = I_{restored}^{k}(x,y) - \alpha \, dE/dI_{restored}^{k}(x,y)$

20

Something hidden?

(^ ^)v

$$
\begin{aligned}
dE/dI_{restored}(x,y) = & -2M(x,y)[\, I_{recorded}(x,y) - I_{restored}(x,y) \,] \\
& -2\lambda[\, I_{restored}(x+1,y) - I_{restored}(x,y)] \quad \text{(^\_^)} \\
& -2\lambda[\, I_{restored}(x,y+1) - I_{restored}(x,y)] \quad \text{(^ ^”)} \\
& +2\lambda[\, I_{restored}(x,y) - I_{restored}(x-1,y)] \quad \text{(o\_O”)} \\
& +2\lambda[\, I_{restored}(x,y) - I_{restored}(x,y-1)] \quad \text{(>\_<”)}
\end{aligned}
$$

(^ ^”) o O (OH BOY!)

Be Careful!

Optimization Method (Cont’d)

Original Image
(256×256 pixels)

Recorded Image
(50% Missing Data
with Gaussian Noise)

Restored Image
(5 Iterations, t = 10.38 s,
$E = 3.5367\times10^{7}$)

Restored Image
(15 Iterations, t = 32.25 s,
$E = 1.3502\times10^{7}$)

Restored Image
(30 Iterations, t = 62.56 s,
$E = 7.4913\times10^{6}$)

Restored Image
(45 Iterations, t = 93.76 s,
$E = 6.5085\times10^{6}$)

Optimization Method (Cont'd)

Gibbs Sampler or Simulated Annealing (Metropolis Monte Carlo)

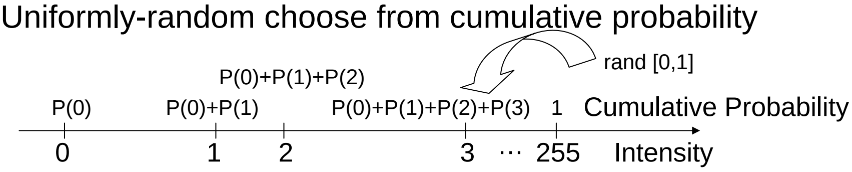
- Initialize randomly the restored image, $I_{restored}$
- Create the cost function, E
- For each pixel (x,y)

  For each state that $I_{restored}(x,y) = 0,1,2,\dots,255$

  -Find $E(I_{restored}(x,y))$ that is E(0),E(1),E(2),…,E(255)

  -Convert to their probability, P(0),P(1),P(2),…,P(255)

  where $P=e^{-E/T}$,T is temperature $\Rightarrow$ lower E, higher P
- Uniformly-random choose from cumulative probability



T too high $\rightarrow$ random walk, T too low $\rightarrow$ greedy

- Reduce T=T/k and repeat simulated annealing

Time consumption $\Rightarrow$ Find global E changing only at (x,y)

MATLAB Function

- imnoise(image,'type')
- medfilt2(image,[m,n])
- wiener2(I,[M N])
- rand(m,n), randn(m,n), unidrnd(N)
- cputime