Chapter 1
Introduction

Image Processing and Computer Vision

LE Thanh Sach
Faculty of Computer Science and Engineering
Ho Chi Minh University of Technology, VNU-HCM

Overview

Linear Filters

Non-Linear Filters

Noise's Model

Sources of noise

Sources of noise

1. Environmental conditions during image acquisition.
   - Light level
   - Sensor temperature
2. Transmission
   - Signal interference
   - Quality of transmission channels

Gaussian Noise

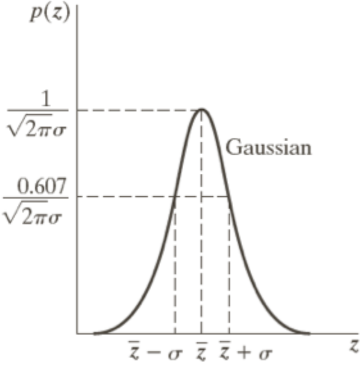
Gaussian probability density function

For each pixel in $I(u, v)$, the noise value z is drawn from a Gaussian probability density function.

$$p(z) = \frac{1}{\sqrt{(2\pi)}\sigma} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

1. μ : the mean of noise values, i.e., z
2. σ : the standard deviation.
3. σ^2 : the variance of z

Gaussian Noise



Properties

1. $[\mu - \sigma, \mu + \sigma]$: contains approximately 70% of noise values
2. $[\mu - 2\sigma, \mu + 2\sigma]$: contains approximately 95% of noise values

Rayleigh Noise

Rayleigh probability density function

For each pixel in $I(u,v)$, the noise value z is drawn from a Rayleigh probability density function.

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{for } z \geq a \\ 0 & \text{for } z < a \end{cases}$$

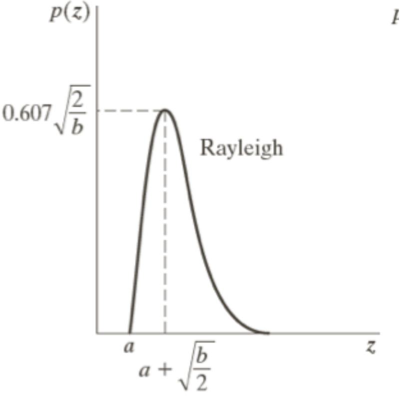
1. **mean of noises**:

$$\mu = a + \sqrt{\pi b/4}$$

2. **variance of noises**:

$$\sigma^2 = \frac{b(4-\pi)}{4}$$

Rayleigh Noise



Properties

1. The minimum noise value is a
2. The density is skewed to the right
3. $b > 0$

Erlang (Gamma) Noise

Erlang probability density function

For each pixel in $I(u, v)$, the noise value z is drawn from a Erlang probability density function.

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$

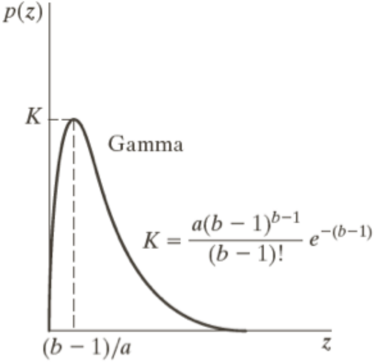
1. **mean of noises**:

$$\mu = \frac{b}{a}$$

2. **variance of noises**:

$$\sigma^2 = \frac{b}{a^2}$$

Erlang Noise



Properties

1. The minimum noise value is 0
2. $a > 0$
3. b is a positive integer

Exponential Noise

Exponential probability density function

For each pixel in $I(u, v)$, the noise value z is drawn from a exponential probability density function.

$$p(z) = \begin{cases} ae^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$

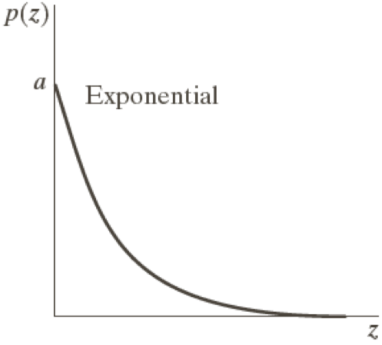
1. **mean of noises**:

$$\mu = \frac{1}{a}$$

2. **variance of noises**:

$$\sigma^2 = \frac{1}{a^2}$$

Exponential Noise



Properties

1. The minimum noise value is 0
2. $a > 0$
3. Exponential noise is a special case of Erlang noise, with $b = 1$

Uniform Noise

Probability density function of uniform noise

For each pixel in $I(u, v)$, the noise value z is drawn from a uniform probability density function.

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

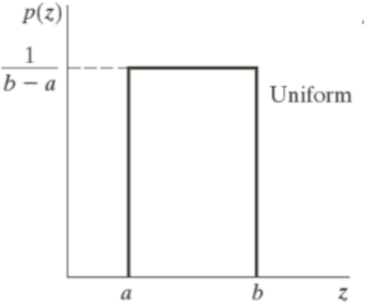
1. **mean of noises**:

$$\mu = \frac{a+b}{2}$$

2. **variance of noises**:

$$\sigma^2 = \frac{(b-a)^2}{12}$$

Uniform Noise



Properties

1. Noise values range from a to b
2. $a \leq b$
3. Probability of noise values are equal $(= \frac{1}{b-a})$.

Impulse (pepper-and-salt) Noise

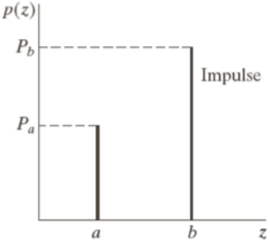
Probability density function of uniform noise

For each pixel in $I(u, v)$, the noise value z is drawn from a bipolar impulse probability density function.

$$p(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & otherwise \end{cases}$$

1. **Bipolar impulse**: both of P_a and P_b are not zero.
2. **Unipolar impulse**: either P_a or P_b is zero.
3. **Light vs dark dot**: $b > a$, gray-level b will appear as a light dot, gray-level a will appear as a dark dot.

Impulse Noise



Properties

1. Impulse corruption is large; so, impulse noise should be digitalized as extreme (pure black or white)
2. For signed numbers: negative impulse $\rightarrow$ black; positive impulse $\rightarrow$ white
3. For unsigned 8-bit numbers: $a = 0$ (black), $b = 255$ (white)

Images with noise

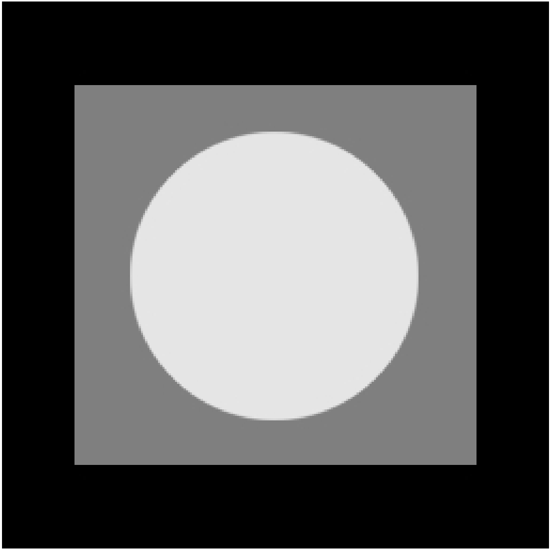


Figure: Pattern without noise

Images with noise

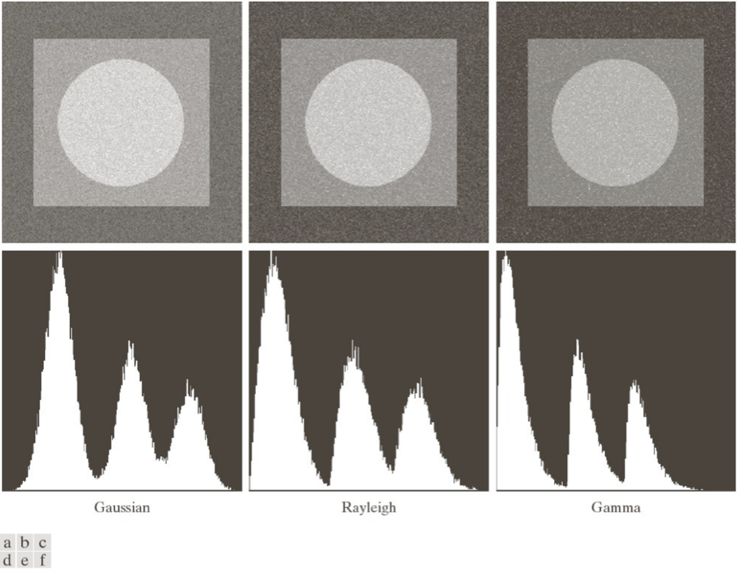


FIGURE 5.4 Images and histograms resulting from adding Gaussian, Rayleigh, and gamma noise to the image in Fig. 5.3.

Images with noise

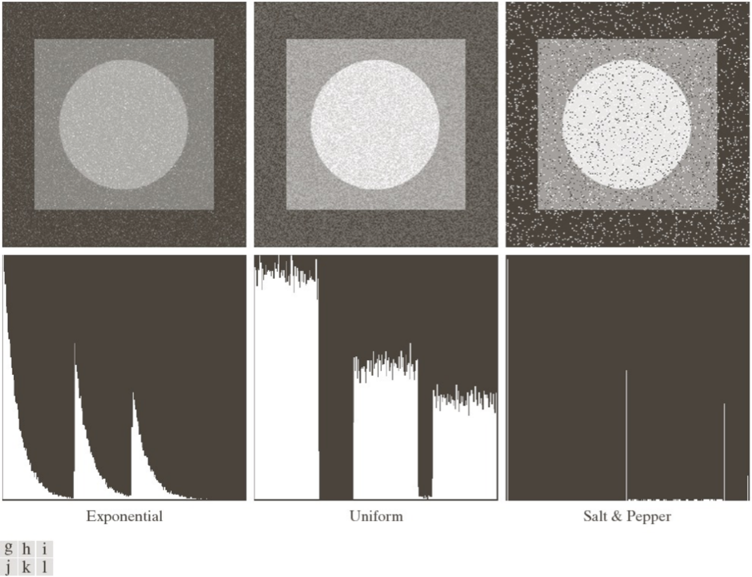


FIGURE 5.4 *(Continued)* Images and histograms resulting from adding exponential, uniform, and salt and pepper noise to the image in Fig. 5.3.

Periodic Noise

Periodic Noise

1. **Cause**: during image acquisition
2. **Type**: This is spatially dependent noise

Example

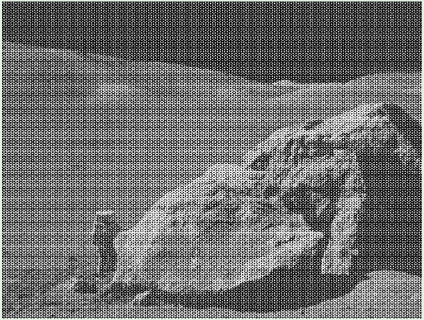


Figure: Example of periodic noise

Periodic Noise

Example

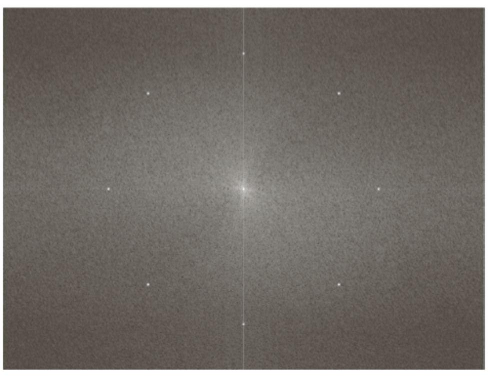


Figure: Frequency spectrum of previous image

Periodic Noise

Properties

1. Image shows periodic noise (sinusoidal) spatially.
2. So, frequency spectrum has some bight dot at some frequencies.
3. Therefore, this noise can be removed effectively in frequency domain.

Noise generation

Why do we need to generate noise?

- Noise is unwanted
- However, we need to generate them and to model them for research purpose.

Problem

Assume that we have following

1. $p_x(x)$: a probability density function of a random variable x
2. $p_z(z)$: a probability density function of a random variable z

Can we generate variable z if we have x, $p_x(x)$, and $p_z(z)$?

Noise generation

Let $c_x(x)$ and $c_z(z)$ be distribution function of variables x and z.
$c_x(x)$ and $c_z(z)$ are cumulative density functions of x and z. These functions can be computed as follows:

$$c_x(x) = \sum_{x=-\infty}^{\infty} p_x(x)$$

$$c_z(z) = \sum_{z=-\infty}^{\infty} p_z(z)$$

Noise generation

Let $z = f(x)$ be a function that map one-to-one between x and z. We have to discover this function.

Discovering $z = f(x)$

- Assume that we have $z_1 = f(x_1)$
- Then, $c_z(z_1) = c_x(x_1)$. This is because of one-one mapping
  - A value $x < x_1$ will be mapped into $z < z_1$
- Define $w \equiv c_x(x_1)$, i.e., $w \equiv c_z(z_1) \equiv c_x(x_1)$

Therefore,

- $z_1 = c_z^{-1}(w) \equiv c_z^{-1}(c_x(x_1))$

Mapping function $f(x)$

$$z = f(x) = c_z^{-1}(c_x(x))$$

Noise generation

Mapping function $f(x)$: Meaning

$$z = f(x) = c_z^{-1}(c_x(x))$$

- We can generate noise value z distributed with PDF $p_z(z)$, if we has input value x distributed with PDF $p_x(x)$

Why?

Because,

- If we know PDF $p_x(x)$, we can compute $c_x(x)$.
- If we know PDF $p_z(z)$, we can compute $c_z(z)$.
- From $c_z(z)$, we can obtain $c_z^{-1}(z)$ in term of closed-form or in term of a look-up table.

This is idea is also applied to Histogram equalization and matching

Noise generation

Some random number generators

Almost programming languages provide function **rand()** and **randn()** to generate number distributed with uniform or Gaussian probability density function respectively. Therefore, We can use these function to generate x and then from x to generate noise z with other kinds distributions.

Noise generation

How do we compute generate y in practice?

1. Generate random variable x by uniform distribution. Existing function $rand(.)$ in many programming language can do this task
2. Compute $w = c_x(x)$. Please note that $c_x(x)$ is uniform distribution function.
3. If we have closed-form of $z = c_z^{-1}(w)$ then can use this function to determine z.
4. If we do not have closed-form of $c_z^{-1}(z)$
   - Create a lookup table at the beginning for mapping $z \rightarrow c_z^{-1}(p)$, for discrete values p. We can do this because we know $c_z(z)$ in advance. This task is equal to rasterize $p = c_z(z)$ and store pairs into lookup table
   - Determine z according to the lookup table.

Noise generation: some popular PDF and CDF functions

TABLE 5.1 Generation of random variables.

Name	PDF	Mean and Variance	CDF	Generator†
Uniform	$p_z(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \le z \le b \\ 0 & \text{otherwise} \end{cases}$	$m = \frac{a+b}{2}, \quad \sigma^2 = \frac{(b-a)^2}{12}$	$F_z(z) = \begin{cases} 0 & z < a \\ \frac{z-a}{b-a} & a \le z \le b \\ 1 & z > b \end{cases}$	MATLAB function `rand`
Gaussian	$p_z(z) = \frac{1}{\sqrt{2\pi}b} e^{-(z-a)^2/2b^2}$ $-\infty < z < \infty$	$m = a, \quad \sigma^2 = b^2$	$F_z(z) = \int_{-\infty}^{z} p_z(v)\,dv$	MATLAB function `randn`
Salt & Pepper	$p_z(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$ $b > a$	$m = aP_a + bP_b$ $\sigma^2 = (a-m)^2 P_a + (b-m)^2 P_b$	$F_z(z) = \begin{cases} 0 & \text{for } z < a \\ P_a & \text{for } a \le z < b \\ P_a + P_b & \text{for } b \le z \end{cases}$	MATLAB function `rand` with some additional logic
Lognormal	$p_z(z) = \frac{1}{\sqrt{2\pi}bz} e^{-[\ln(z)-a]^2/2b^2}$ $z > 0$	$m = e^{a+(b^2/2)}, \sigma^2 = [e^{b^2} - 1]e^{2a+b^2}$	$F_z(z) = \int_0^z p_z(v)\,dv$	$z = ae^{bN(0,1)}$
Rayleigh	$p_z(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & z \ge a \\ 0 & z < a \end{cases}$	$m = a + \sqrt{\pi b/4}, \sigma^2 = \frac{b(4-\pi)}{4}$	$F_z(z) = \begin{cases} 1 - e^{-(z-a)^2/b} & z \ge a \\ 0 & z < a \end{cases}$	$z = a + \sqrt{-b\ln[1 - U(0,1)]}$
Exponential	$p_z(z) = \begin{cases} ae^{-az} & z \ge 0 \\ 0 & z < 0 \end{cases}$	$m = \frac{1}{a}, \quad \sigma^2 = \frac{1}{a^2}$	$F_z(z) = \begin{cases} 1 - e^{-az} & z \ge 0 \\ 0 & z < 0 \end{cases}$	$z = -\frac{1}{a}\ln[1 - U(0,1)]$
Erlang	$p_z(z) = \frac{a^b z^{b-1}}{(b-1)!} e^{-az}$ $z \ge 0$	$m = \frac{b}{a}, \quad \sigma^2 = \frac{b}{a^2}$	$F_z(z) = \left[1 - e^{-az}\sum_{n=0}^{b-1}\frac{(az)^n}{n!}\right]$ $z \ge 0$	$z = E_1 + E_2 + \cdots + E_b$ (The E's are exponential random numbers with parameter a.)

† $N(0,1)$ denotes normal (Gaussian) random numbers with mean 0 and a variance of 1. $U(0,1)$ denotes uniform random numbers in the range $(0, 1)$.

Noise Integration

Integration model

1. Additive Noise
   - Noise values will be **added** to free-noise image to create image with noise.
   - Model: $g(x,y) = f(x,y) + \eta(x,y)$
2. Multiplicative Noise
   - Noise values will be **multiplied** to free-noise image to create image with noise.
   - Model: $g(x,y) = f(x,y) \times \eta(x,y)$

Each pixel in noise image $z = \eta(x,y)$ is generated according to previous algorithm to follow PDF $p_z(z)$

Noise generation

Exercise

1. Write a program to add noise (additive and multiplicative) to input image

Noise Estimation

Reasons of estimation

- Noise will be removed more effectively, if we know noise model in advance.
- So, noise reduction needs estimation of noise model

Noise Estimation: Methods

Method of estimating noise model

1. Select a small strip (called S) in input images. This trip contains reasonably constant gray level.
   - "Reasonably" means gray levels are not too dark or bright
2. Compute histogram of values inside of the strip
3. Observe the histogram to determine noise model
4. Estimate the mean and the variance of PDF of noise model.
5. Use the estimated mean and variance to estimate other parameters, for examples, a and b in other types of models different to Gaussian.

Noise Estimation: Methods

Input data

1. S : a small strip in input image
2. $p(z_i)$: normalized histogram computed from S

Mean and Variance Estimation

1. **Mean:**

$$\mu = \sum_{z_i \in S} z_i p(z_i)$$

2. **Variance:**

$$\sigma^2 = \sum_{z_i \in S} (z_{i-\mu)^2} p(z_i)$$

Noise Estimation: Example

Example

a b c

FIGURE 5.6 Histograms computed using small strips (shown as inserts) from (a) the Gaussian, (b) the Rayleigh, and (c) the uniform noisy images in Fig. 5.4.

Mean Filters

Mean filters: Arithmetic mean filter

Mathematical model

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t)$$

Properties

1. Smooth local variations in an image.
2. Can reduce the following noises
   - Additive Gaussian noise with zero mean
   - Additive Uniform noise with zero mean
3. Result blurred image, especially, at edges, with large S_{xy}.

Mean filters: Geometric mean filter

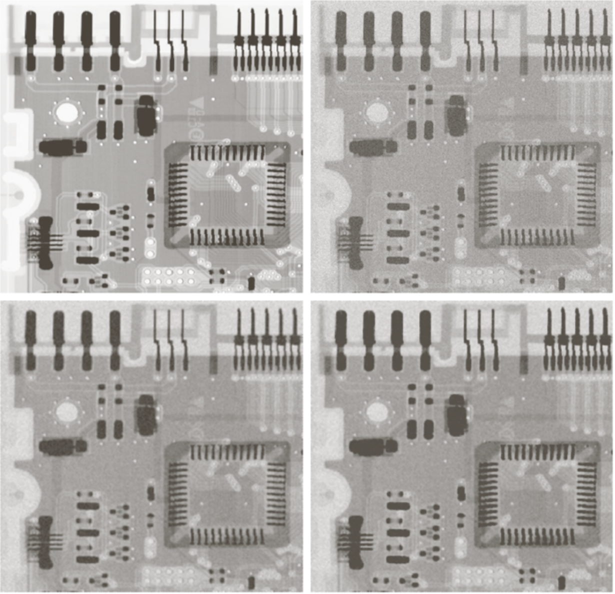
Mathematical model

$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

Properties

1. Smooth local variations in an image. Do smoothing comparable to the arithmetic mean filter
2. Tend to lose less image detail
3. Can reduce the following noises
   - Additive Gaussian noise with zero mean
   - Additive Uniform noise with zero mean
4. Result blurred image, especially, at edges, with large S_{xy}.

Mean filters: Examples



a	b
c	d

FIGURE 5.7 (a) X-ray image. (b) Image corrupted by additive Gaussian noise. (c) Result of filtering with an arithmetic mean filter of size 3×3. (d) Result of filtering with a geometric mean filter of the same size. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Mean filters: Harmonic mean filter

Mathematical model

$$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

Properties

1. Can reduce the following noises
   - Additive Gaussian noise with zero mean
   - Additive Uniform noise with zero mean
   - Salt noise
2. Can not reduce pepper noise (black)

Mean filters: Contraharmonic mean filter

Mathematical model

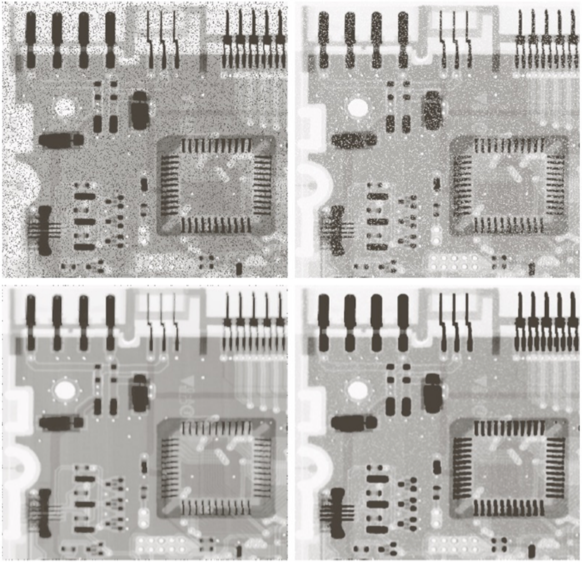
$$\hat{f}(x, y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

- Q: order of the filter
- $Q = 0$: Contraharmonic $\rightarrow$ Arithmetic
- $Q = 1$: Contraharmonic $\rightarrow$ Harmonic

Properties

1. Can reduce pepper-and-salt noise
   - $Q > 0$: reduce pepper noise
   - $Q < 0$: reduce salt noise
2. Can not reduce pepper and salt noise simultaneously

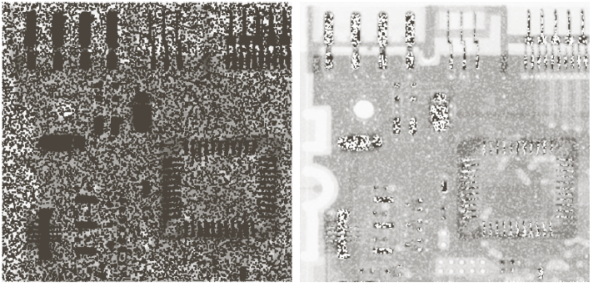
Mean filters: Examples



a b
c d

FIGURE 5.8 (a) Image corrupted by pepper noise with a probability of 0.1. (b) Image corrupted by salt noise with the same probability. (c) Result of filtering (a) with a 3×3 contraharmonic filter of order 1.5. (d) Result of filtering (b) with $Q = -1.5$.

Mean filters: Examples



a b

FIGURE 5.9 Results of selecting the wrong sign in contraharmonic filtering. (a) Result of filtering Fig. 5.8(a) with a contraharmonic filter of size 3×3 and $Q = -1.5$. (b) Result of filtering 5.8(b) with $Q = 1.5$.

Order-Statistics Filters

Order-Statistics Filters: Median filter

Mathematical model

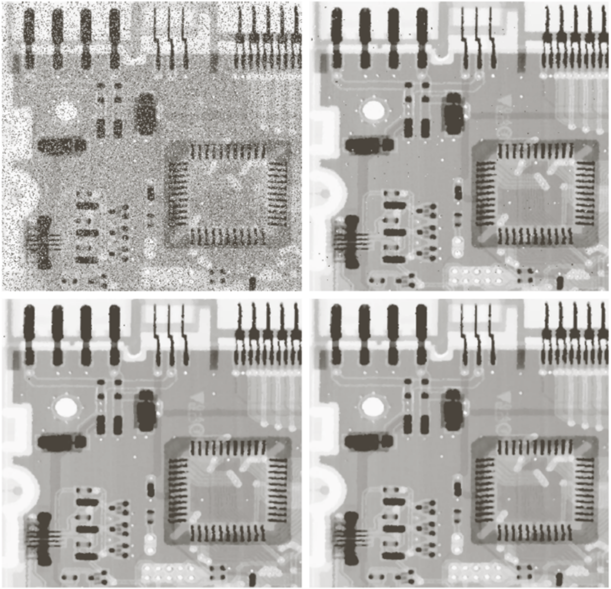
$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\textbf{median}}\{g(s,t)\}$$

- Assign to the output image $\hat{f}(x,y)$ the median value of gray levels in the neighborhood of (x,y)

Properties

1. Effectively reduce both of bipolar and unipolar impulse noise, i.e., salt-and-pepper noise
2. Produce less blurring images compared to linear
3. Can not work with Gaussian noise

Order-Statistics Filters: Examples



a	b
c	d

FIGURE 5.10 (a) Image corrupted by salt-and-pepper noise with probabilities $P_a = P_b = 0.1$. (b) Result of one pass with a median filter of size 3×3. (c) Result of processing (b) with this filter. (d) Result of processing (c) with the same filter.

Order-Statistics Filters: Max and Min filter

Mathematical model

$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}}\{g(s,t)\}$$
$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}}\{g(s,t)\}$$

- **Max filter**: Assign to the output image $\hat{f}(x,y)$ the maximum value of gray levels in the neighborhood of (x,y)
- **Min filter**: Assign to the output image $\hat{f}(x,y)$ the maximum value of gray levels in the neighborhood of (x,y)

Order-Statistics Filters: Max and Min filter

Mathematical model

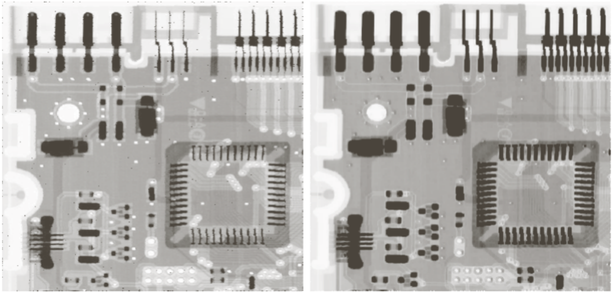
$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}}\{g(s,t)\}$$

$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}}\{g(s,t)\}$$

Properties

1. **Max filter**:
   - Finds the brightest points in image
   - Remove pepper noise
2. **Min filter**:
   - Finds the darkest points in image
   - Remove salt noise

Order-Statistics Filters: Examples



a b

FIGURE 5.11 (a) Result of filtering Fig. 5.8(a) with a max filter of size 3×3. (b) Result of filtering 5.8(b) with a min filter of the same size.

Order-Statistics Filters: Midpoint filter

Mathematical model

$$\hat{f}(x,y) = \frac{1}{2} \times \left(\max_{(s,t)\in S_{xy}}\{g(s,t)\} + \min_{(s,t)\in S_{xy}}\{g(s,t)\} \right)$$

Properties

1. Can reduce randomly distributed noise
   - Additive Gaussian noise with zero mean
   - Additive uniform noise with zero mean

Order-Statistics Filters: Alpha-trimmed mean filter

Mathematical model

$$\hat{f}(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$$

- Delete $d/2$ lowest and $d/2$ highest gray values in neighborhood of (x,y) to obtain $g_r(s,t)$ of $mn-d$ gray values.
- Assign the average of $g_r(s,t)$ to $\hat{f}(x,y)$
- $d=0$: Alpha-trimmed $\rightarrow$ Arithmetic mean filter
- $d=(mn-1)/2$: Alpha-trimmed $\rightarrow$ Median filter

Properties

1. Useful in situations involving multiple types of noise, in combination of salt-and-pepper and Gaussian noise.

Order-Statistics Filters: Examples

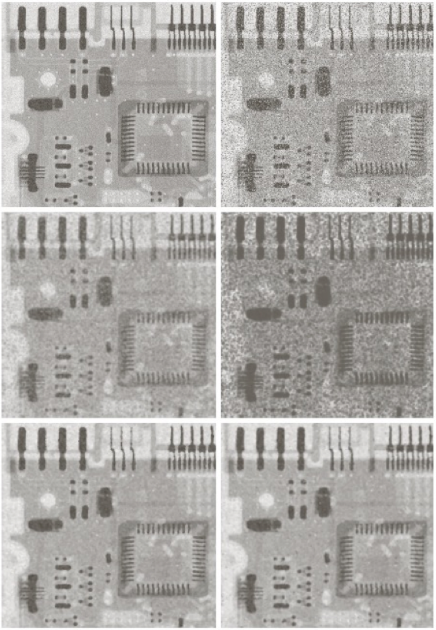


FIGURE 5.12 (a) Image corrupted by additive uniform noise. (b) Image additionally corrupted by additive salt-and-pepper noise. Image (b) filtered with a 5×5; (c) arithmetic mean filter; (d) geometric mean filter; (e) median filter; and (f) alpha-trimmed mean filter with $d = 5$.

Image Restoration: Model

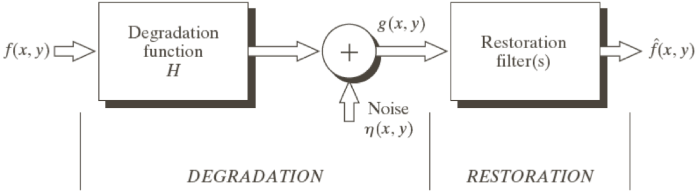


Figure: Model of degradation and restoration process

- $f(x, y)$: input image
- $\eta(x, y)$: noise at (x, y)
- $\hat{f}(x, y)$: restored image

Image Restoration: Model

Linear, Position-Invariant Degradation

1. Model in space domain:

$$g(x, y) = h(x, y) * f(x, y) + \eta(x, y)$$

1. Model in frequency domain:

$$G(u, v) = H(u, v)F(u, v) + N(u, v)$$

$G(u, v)$, $H(u, v)$, $F(u, v)$, and $N(u, v)$ are **Fourier transforms** of $g(x, y)$, $h(x, y)$, $f(x, y)$ and $\eta(x, y)$ respectively.

Inverse Filtering

- Let $\hat{F}(u, v$ be estimate of Fourier transforms of $f(x, y)$

Ideal Restoration

$$\begin{aligned}\hat{F}(u, v) &= \frac{G(u, v)}{H(u, v)} \\ &= F(u, v) + \frac{N(u, v)}{H(u, v)}\end{aligned}$$

Problems

1. **Problem 1**: even you know $H(u, v)$, you can not recover $f(x, y)$ exactly because you do not know $N(u, v)$.
2. **Problem 2**: At some (u, v), $H(u, v) = 0$ will cause $N(u, v)/H(u, v)$ to dominate $\hat{F}(u, v)$.

Inverse Filtering

Inverse Filtering

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

Tools for existing problems

1. **Problem 1**: Assume that there is no noise.
2. **Problem 2**: At some (u,v), $H(u,v) = 0$:
   1. **Replacement:** Replace $H(u,v) =$ a certain value at (u,v) where $H(u,v) = 0$:
   2. **Cut-off:** Filter $G(u,v)/H(u,v)$ with Butterworth lowpass function of some order, e.g., order $= 10$, with some radius, e.g., $40, 70$, etc - dependent on image size.
   3. **Finding radius:** Go from the origin to outside radially, find the first (u,v) that $H(u,v) = 0$. Limit the filter frequencies from the origin to this radius.

Inverse Filtering: Demonstration

(a) Original image

(b) Cut-off, R=full

Inverse Filtering: Demonstration

(a) Cut-off, $R = 40$

(b) Cut-off, $R = 85$

Inverse Filtering: Demonstration

Figure: Cut-off, $R = 70$

Inverse Filtering: Demonstration

Exercise

1. Write a program to simulate the atmospheric turbulence phenomenon, modeled by $H(u,v)$ in frequency domain as in the following. Take a look at Gonzalez's Book, page 258.

$$H(u,v) = e^{-k(u^2+v^2)^5/6}$$

   - Some k: $k = 0.001, 0.0025, 0.00025$

2. Write a program to remove the atmospheric turbulence phenomenon from images (generated from previous question.)

Wiener Filtering

Names and Wiener Filter's objective

- Other name: Minimum Mean Square Error Filter
- Objective: to minimize the mean square error between uncorrupted image f and its estimate $\hat{f}$.
- $\equiv$ Minimize

$$e^2 = E\{(f - \hat{f})^2\}$$

Wiener Filtering

Mathematical Model

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)} \times \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right] G(u,v)$$

- $H(u,v)$: degradation function (assume that it has been estimated).
- $G(u,v)$: Fourier transforms of degraded image $g(x,y)$, can be computed.
- $|H(u,v)|^2 = H^*(u,v)H(u,v)$: power spectrum of degradation function, can be computed
- $S_\eta(u,v) = |N(u,v)|^2$: power spectrum of noise.
- $S_f(u,v) = |F(u,v)|^2$: power spectrum of uncorrupted image. **This seldom is known**.

Wiener Filtering

Mathematical Model

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)} \times \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right] G(u,v)$$

Wiener filter to Inverse filter

Sepcial case: There is no noise. $N(u,v) = 0$.
Wiener filtering becomes Inverse filtering.

Wiener Filtering

A special case - White noise

We do not know:

1. $S_\eta(u,v)$: power spectrum of noise
2. $S_f(u,v)$: power spectrum of input signal

For white noise, we hope that, at a specific frequency (u,v), the power of noise is proportional to the power of input signal. It means $S_\eta(u,v) = K \times S_f(u,v)$

$$\frac{S_\eta(u,v)}{S_f(u,v)} = K$$

Remind

White noise is a type of noise that affects on all frequencies.

Wiener Filtering

Mathematical Model for white noise

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)} \times \frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right] G(u,v)$$

Wiener Filtering: Demonstration

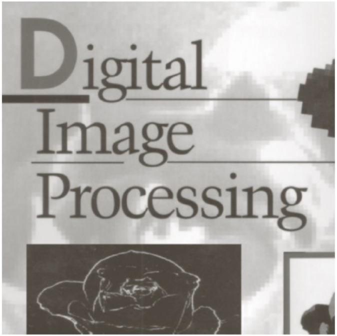
Demonstration's model of degradation

The demonstration for Wiener filtering in some consecutive slides uses **motion blur** degradation model, as shown below.

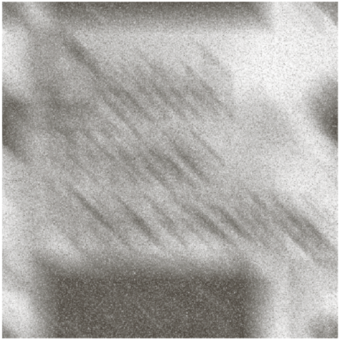
$$H(u,v) = \frac{T}{\pi(ua+vb)} sin\left[\pi(ua+vb)\right] e^{-j\pi(ua+vb)}$$

- $a = b = 0.1$
- $T = 1$

Wiener Filtering: Demonstration



(a) Original image



(b) Degraded image

Degradation method

1. Blurring the original with motion model
2. Corrupting heavily with additive Gaussian noise, zeros mean, variance of 650

Wiener Filtering: Demonstration

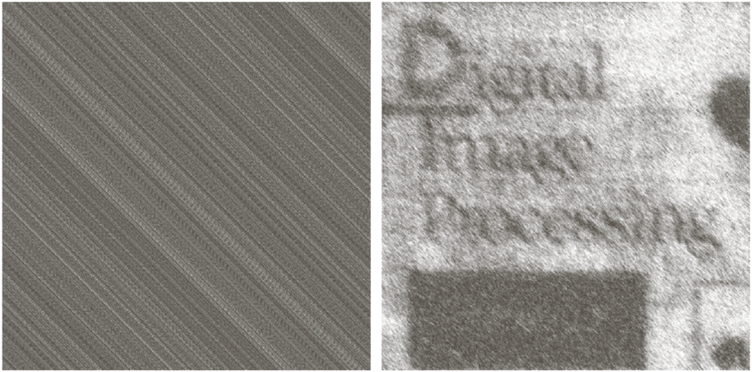
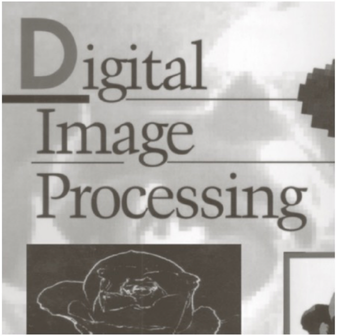


Figure: Restored images. Left: Inverse Filtering, Right: Wiener Filtering, K is selected to for best result

Wiener Filtering: Demonstration



(a) Original image

(b) Degraded image

Degradation method

1. Blurring the original with motion model
2. Corrupting additive Gaussian noise with smaller variance compared to the previous case, zeros mean.

Wiener Filtering: Demonstration

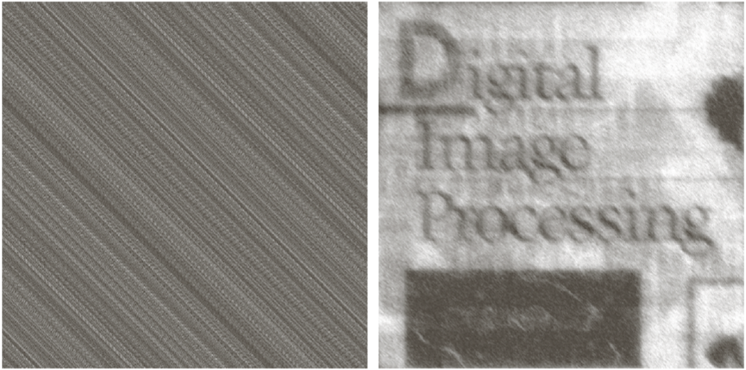
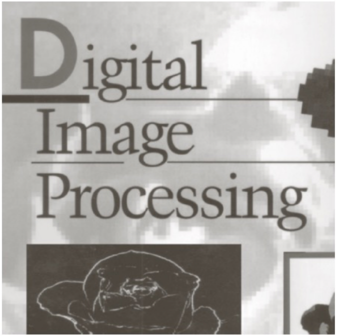


Figure: Restored images. Left: Inverse Filtering, Right: Wiener Filtering, K is selected to for best result

Wiener Filtering: Demonstration



(a) Original image

(b) Degraded image

Degradation method

1. Blurring the original with motion model
2. Corrupting additive Gaussian noise with smaller variance compared to the previous case, zeros mean.

Wiener Filtering: Demonstration

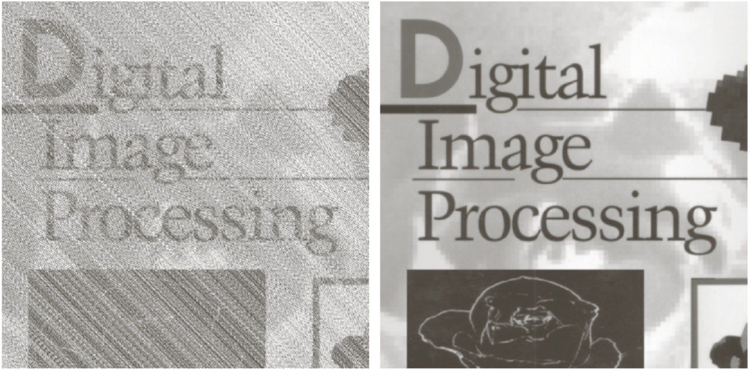


Figure: Restored images. Left: Inverse Filtering, Right: Wiener Filtering, K is selected to for best result

- The result of Wiener filter in this case shows excellent quality!

Wiener Filtering: Exercises

Exercise

1. Write a program to degrade input images with motion blur model and additive Gaussian noise with zeros mean.
2. Write a program to restore degraded images with Wiener and Inverse filtering