Adaptive Filter

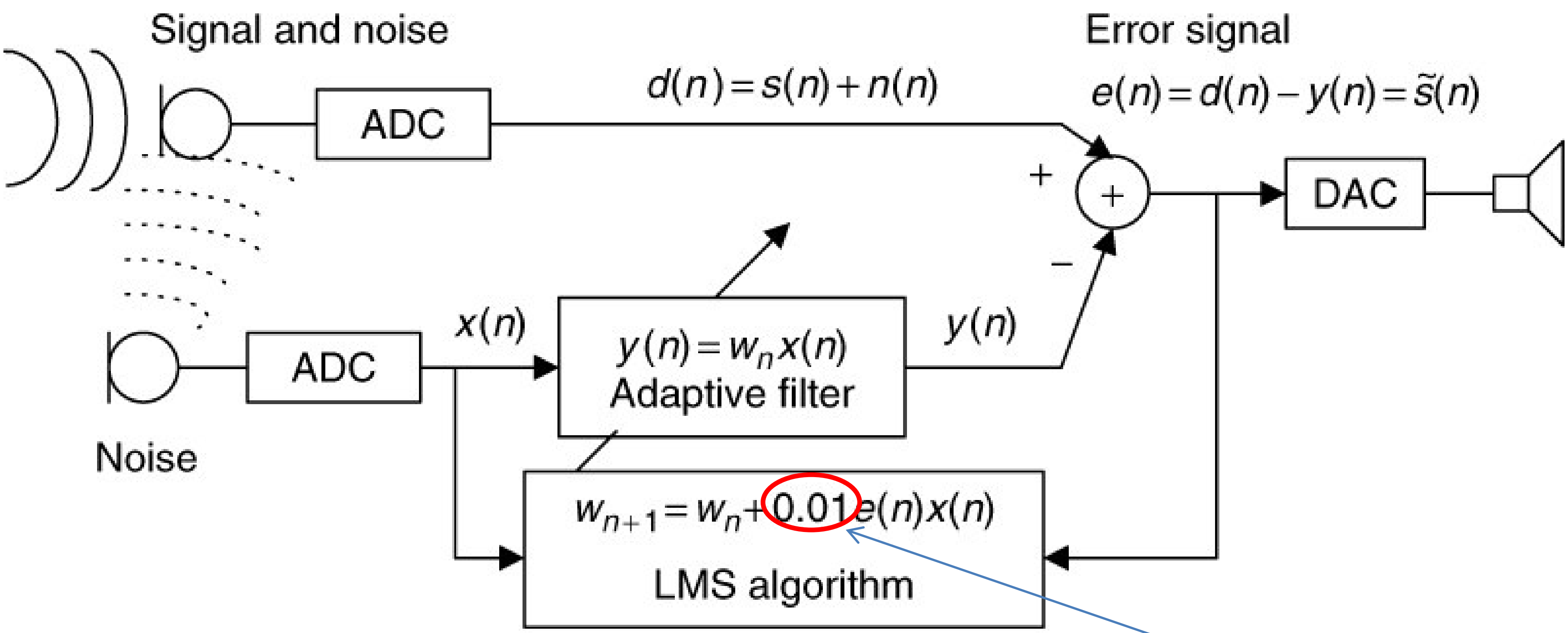
A digital filter that automatically adjusts its coefficients to adapt input signal via an adaptive algorithm.

Applications:

- **Signal enhancement**
- **Active noise control**
- **Noise cancellation**
- **Telephone echo cancellation**

Text: Digital Signal Processing
by Li Tan, Chapter 10

Simplest Noise Canceller



Controls speed of convergence

n(n) is a linear filtered (delayed) version of *x*(n).

Simplest Noise Canceller – contd.

$$y(n) = w_n x(n)$$
$$e(n) = d(n) - y(n)$$
$$w_{n+1} = w_n + 0.01 e(n) x(n).$$

Initial coefficient $w_0 = 0.3$

n	$d(n)$	$x(n)$
0	-0.2947	-0.5893
1	1.0017	0.5893

Measured

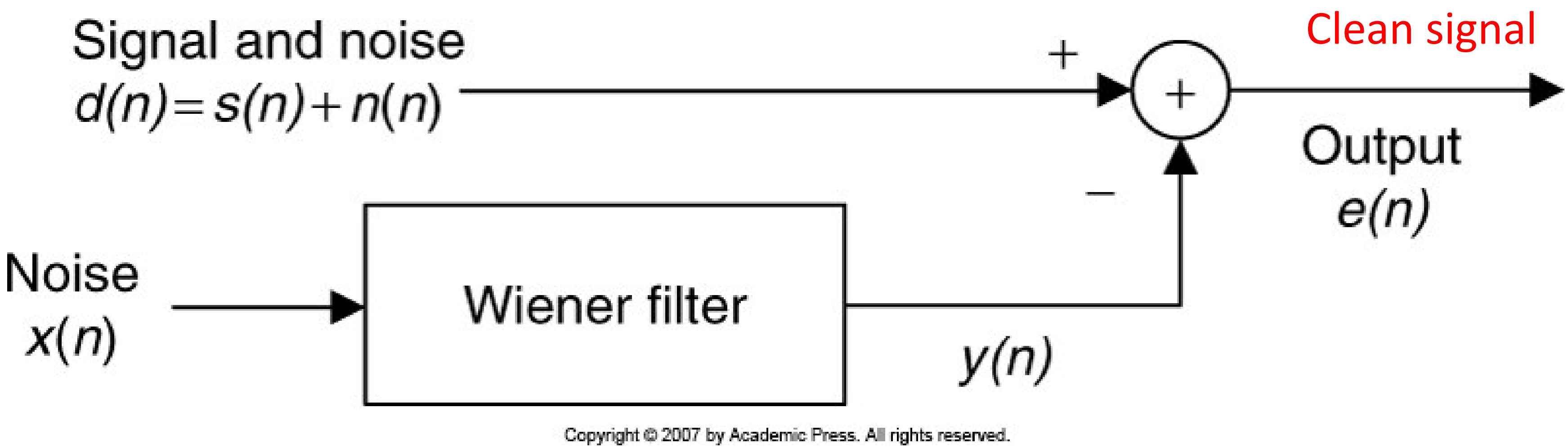
$n = 0,$ $y(0) = w_0 x(0) = 0.3 \times (-0.5893) = -0.1768$

$$e(0) = d(0) - y(0) = -0.2947 - (-0.1768) = -0.1179 = \tilde{s}(0)$$
$$w_1 = w_0 + 0.01 e(0) x(0) = 0.3 + 0.01 \times (-0.1179) \times (-0.5893) = 0.3007$$

$n = 1,$ $y(1) = w_1 x(1) = 0.3007 \times 0.5893 = 0.1772$

$$e(1) = d(1) - y(1) = 1.0017 - 0.1772 = 0.8245 = \tilde{s}(1)$$
$$w_2 = w_1 + 0.01 e(1) x(1) = 0.3007 + 0.01 \times 0.8245 \times 0.5893 = 0.3056$$

Wiener Filter & LMS Algorithm



Consider, single weight case, $y(n) = wx(n)$

Error signal, $e(n) = d(n) - wx(n)$

Now we have to solve for the best weight w*

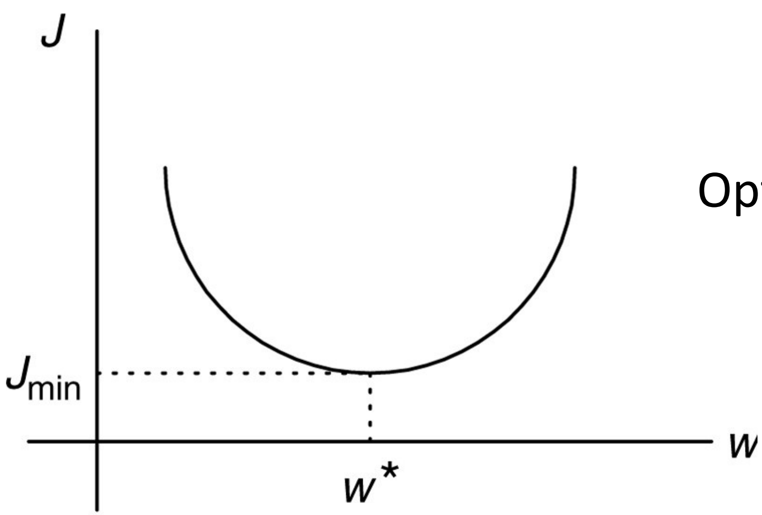
$$e^2(n) = (d(n) - wx(n))^2 = d^2(n) - 2d(n)wx(n) + w^2x^2(n)$$

LMS Algorithm

Taking Expectation of squared error signal

$$E\left(e^2(n)\right) = E\left(d^2(n)\right) - 2wE\left(d(n)x(n)\right) + w^2E\left(x^2(n)\right)$$

$$J = E(e^2(n)) = \text{MSE (mean squared error)}$$
$$\sigma^2 = E(d^2(n)) = \text{power of corrupted signal}$$
$$P = E(d(n)x(n)) = \text{cross-correlation between } d(n) \text{ and } x(n)$$
$$R = E(x^2(n)) = \text{autocorrelation}$$



For large N, $J = \sigma^2 - 2wP + w^2R$

Optimal w* is found when minimum J is achieved

$$\frac{dJ}{dw} = -2P + 2wR = 0.$$

$$w^* = R^{-1}P$$

LMS Algorithm - Example

Given MSE function for the Wiener filter:

$$\frac{dJ}{dw} = -20 + 10 \times 2w = 0$$

Solving for optimal, we get $w^* = 1$

Finally we get

$$J_{\min} = J|_{w=w^*} = 40 - 20w + 10w^2|_{w=1} = 40 - 20 \times 1 + 10 \times 1^2 = 30$$

large N

$$w^* = R^{-1}P$$

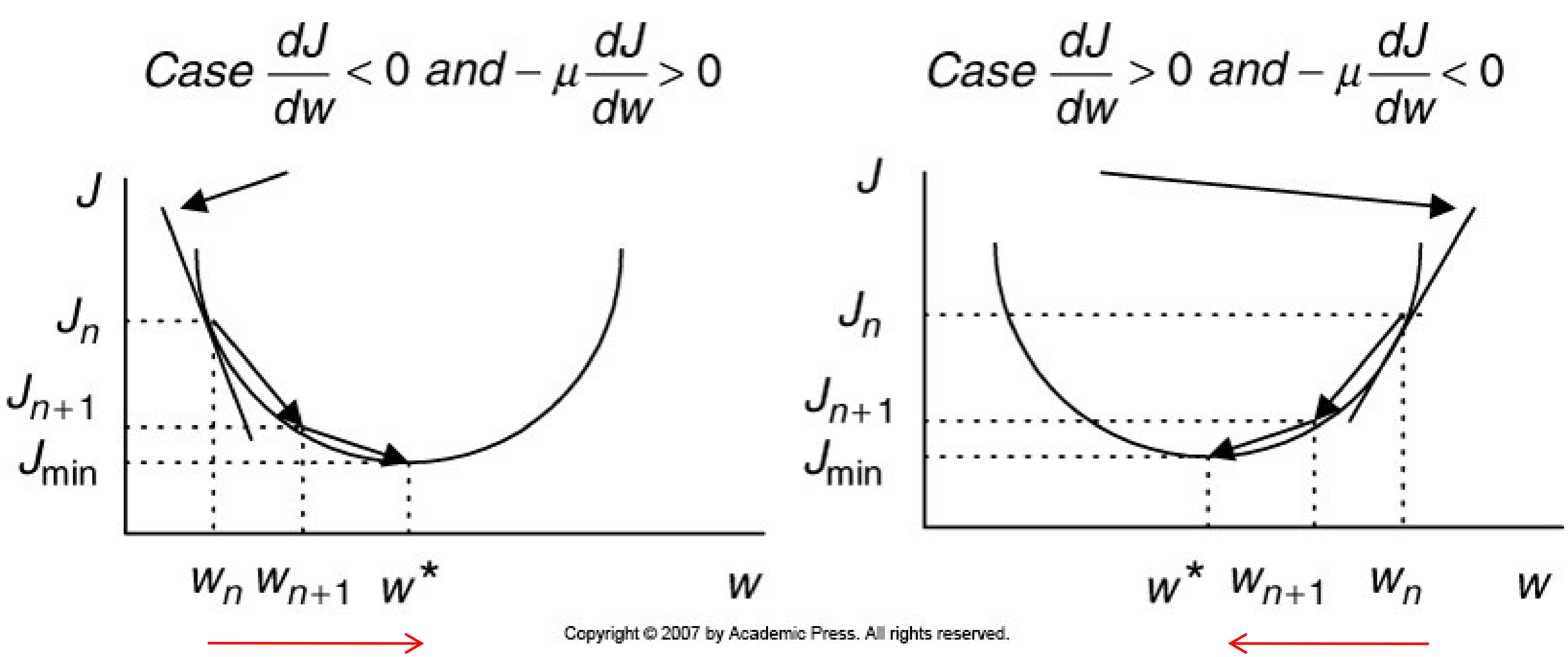
R^{-1}: Matrix inversion

Makes real-time implementation difficult

Steepest Decent Algorithm

$$w_{n+1} = w_n - \mu \frac{dJ}{dw}$$

μ = constant controlling the speed of convergence.



Steepest Decent Algorithm: Example

Given: $J = 40 - 20w + 10w^2$

$\mu = 0.04 \qquad w_0 = 0$

Iteration three times

Find optimal solution for w*

Solution:

$$\frac{dJ}{dw} = -20 + 10 \times 2w_n$$

For n = 0,

$$\mu \frac{dJ}{dw} = 0.04 \times (-20 + 10 \times 2w_0)|_{w_0=0} = -0.8$$

$$w_1 = w_0 - \mu \frac{dJ}{dw} = 0 - (-0.8) = 0.8$$

For n = 1,

$$\mu \frac{dJ}{dw} = 0.04 \times (-20 + 10 \times 2w_1)|_{w_1=0.8} = -0.16$$

$$w_2 = w_1 - \mu \frac{dJ}{dw} = 0.8 - (-0.16) = 0.96$$

For n = 2,

$$\mu \frac{dJ}{dw} = 0.04 \times (-20 + 10 \times 2w_2)|_{w_2=0.96} = -0.032$$

$$w_3 = w_2 - \mu \frac{dJ}{dw} = 0.96 - (-0.032) = 0.992.$$

$$w^* \approx w_3 = 0.992$$

$$J_{\min} \approx 40 - 20w + 10w^2|_{w=0.992}$$

$$= 30.0006$$

Steepest Decent Algorithm – contd1.

To make it sample-based processing, we need to take out estimation.

$$J = e^2(n) = (d(n) - wx(n))^2$$

$$\frac{dJ}{dw} = 2(d(n) - wx(n))\frac{d(d(n) - wx(n))}{dw} = -2e(n)x(n)$$

Updating weight $w_{n+1} = w_n + 2\mu e(n)x(n)$

For multiple tap FIR filter:

$$y(n) = w_n(0)x(n) + w_n(1)x(n-1) + \cdots + w_n(N-1)x(n-N+1)$$

for $i = 0, \ldots, N-1$

$$w_{n+1}(i) = w_n(i) + 2\mu e(n)x(n-i).$$

Choose convergence constant as $0 < \mu < \frac{1}{NP_x}$ P_x : maximum input power

Steepest Decent Algorithm – contd2.

Steps:

1. Initialize $w(0), w(1), \ldots, w(N-1)$ to arbitrary values.
2. Read $d(n)$, $x(n)$, and perform digital filtering:

$$y(n) = w(0)x(n) + w(1)x(n-1) + \cdots + w(N-1)x(n-N+1).$$

3. Compute the output error:

$$e(n) = d(n) - y(n).$$

4. Update each filter coefficient using the LMS algorithm:

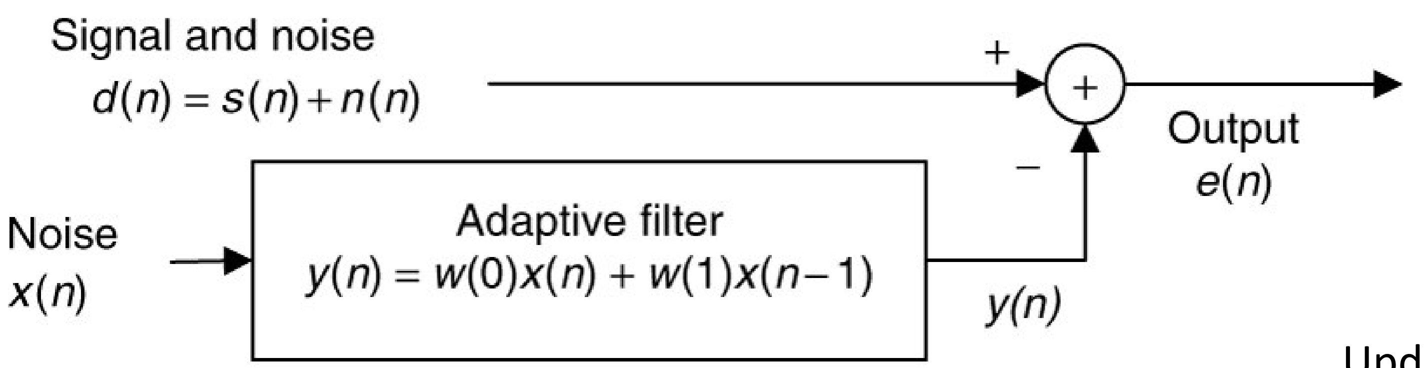
$$\text{for } i = 0, \ldots, N-1$$
$$w(i) = w(i) + 2\mu e(n)x(n-i).$$

Noise Canceller Using Adaptive Filter-1

Perform adaptive filtering to obtain outputs $e(n) = n = 0,\ 1,\ 2$

Given: $x(0) = 1,\ x(1) = 1x(2) = -1,$ $d(0) = 2,\ d(1) = 1,\ d(2) = -2$

Initial weights: $w(0) = w(1) = 0,$ $\mu = 0.1$



Solution:

Updating weights:

Filtering: $y(n) = w(0)x(n) + w(1)x(n-1)$

Output: $e(n) = d(n) - y(n)$

$$w(0) = w(0) + 2\mu e(n)x(n)$$

$$w(1) = w(1) + 2\mu e(n)x(n-1)$$

Noise Canceller Using Adaptive Filter-2

For $n = 0$

Digital filtering:

$$y(0) = w(0)x(0) + w(1)x(-1) = 0 \times 1 + 0 \times 0 = 0$$

Computing the output:

$$e(0) = d(0) - y(0) = 2 - 0 = 2$$

Updating coefficients:

$$w(0) = w(0) + 2 \times 0.1 \times e(0)x(0) = 0 + 2 \times 0.1 \times 2 \times 1 = 0.4$$
$$w(1) = w(1) + 2 \times 0.1 \times e(0)x(-1) = 0 + 2 \times 0.1 \times 2 \times 0 = 0.0$$

For $n = 1$

Digital filtering:

$$y(1) = w(0)x(1) + w(1)x(0) = 0.4 \times 1 + 0 \times 1 = 0.4$$

Computing the output:

$$e(1) = d(1) - y(1) = 1 - 0.4 = 0.6$$

Updating coefficients:

$$w(0) = w(0) + 2 \times 0.1 \times e(1)x(1) = 0.4 + 2 \times 0.1 \times 0.6 \times 1 = 0.52$$
$$w(1) = w(1) + 2 \times 0.1 \times e(1)x(0) = 0 + 2 \times 0.1 \times 0.6 \times 1 = 0.12$$

Noise Canceller Using Adaptive Filter-3

For $n = 2$

Digital filtering:

$$y(2) = w(0)x(2) + w(1)x(1) = 0.52 \times (-1) + 0.12 \times 1 = -0.4$$

Computing the output:

$$e(2) = d(2) - y(2) = -2 - (-0.4) = -1.6$$

Updating coefficients:

$$w(0) = w(0) + 2 \times 0.1 \times e(2)x(2) = 0.52 + 2 \times 0.1 \times (-1.6) \times (-1) = 0.84$$
$$w(1) = w(1) + 2 \times 0.1 \times e(2)x(1) = 0.12 + 2 \times 0.1 \times (-1.6) \times 1 = -0.2.$$

Output (noise-cleaned signal):

$$e(0) = 2,\ e(1) = 0.6,\ e(2) = -1.6$$

Practice: Textbook by Li Tan, Chapter 10.

10.3, 10.5, 10.7

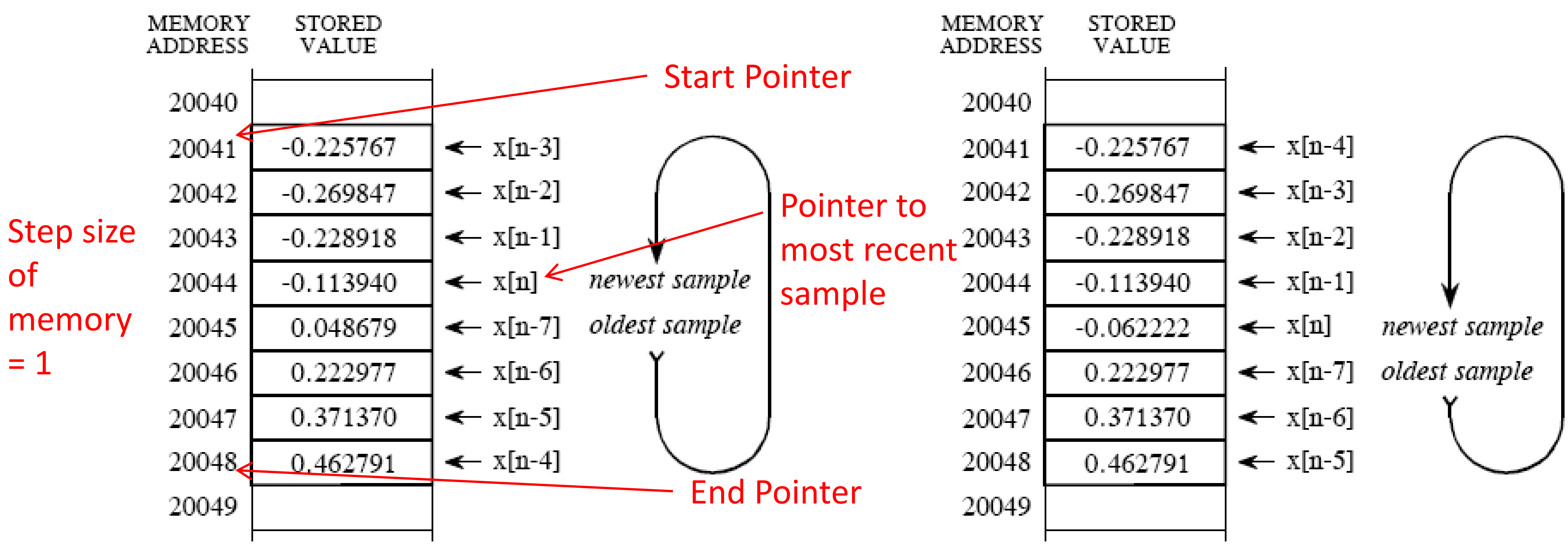
Digital Signal Processors - Introduction

Processors dedicated to DSP

Off-line processing: all the data needs to be in the memory at the same time. The output is produced after all the input data is in the memory.

On-line processing: the output is produced as the same time the input is coming. No delay or little delay.

Circular Buffer Operation:



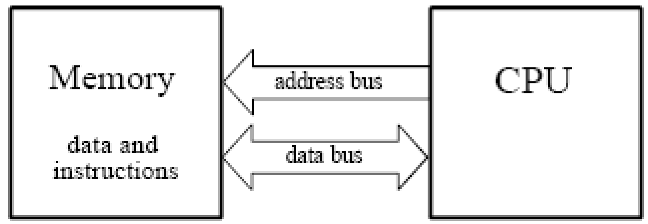
a. Circular buffer at some instant

b. Circular buffer after next sample

Microprocessors Architecture

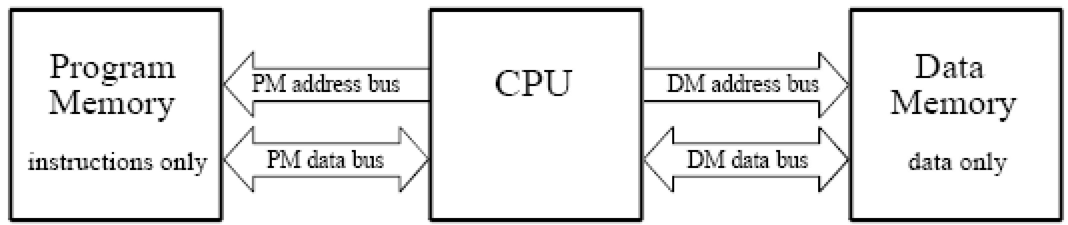
a. Von Neumann Architecture (*single memory*)

Normal microprocessors

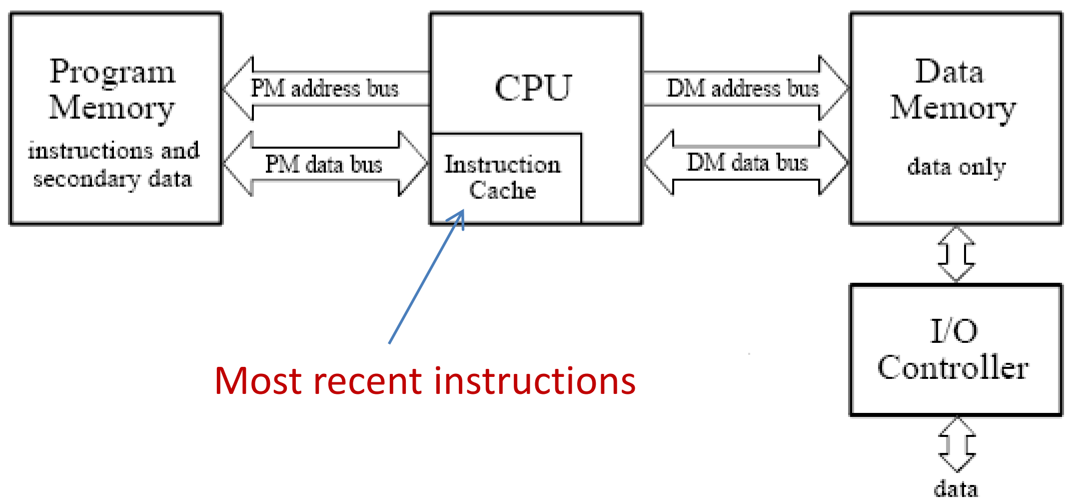


b. Harvard Architecture (*dual memory*)

DSP uses these ideas

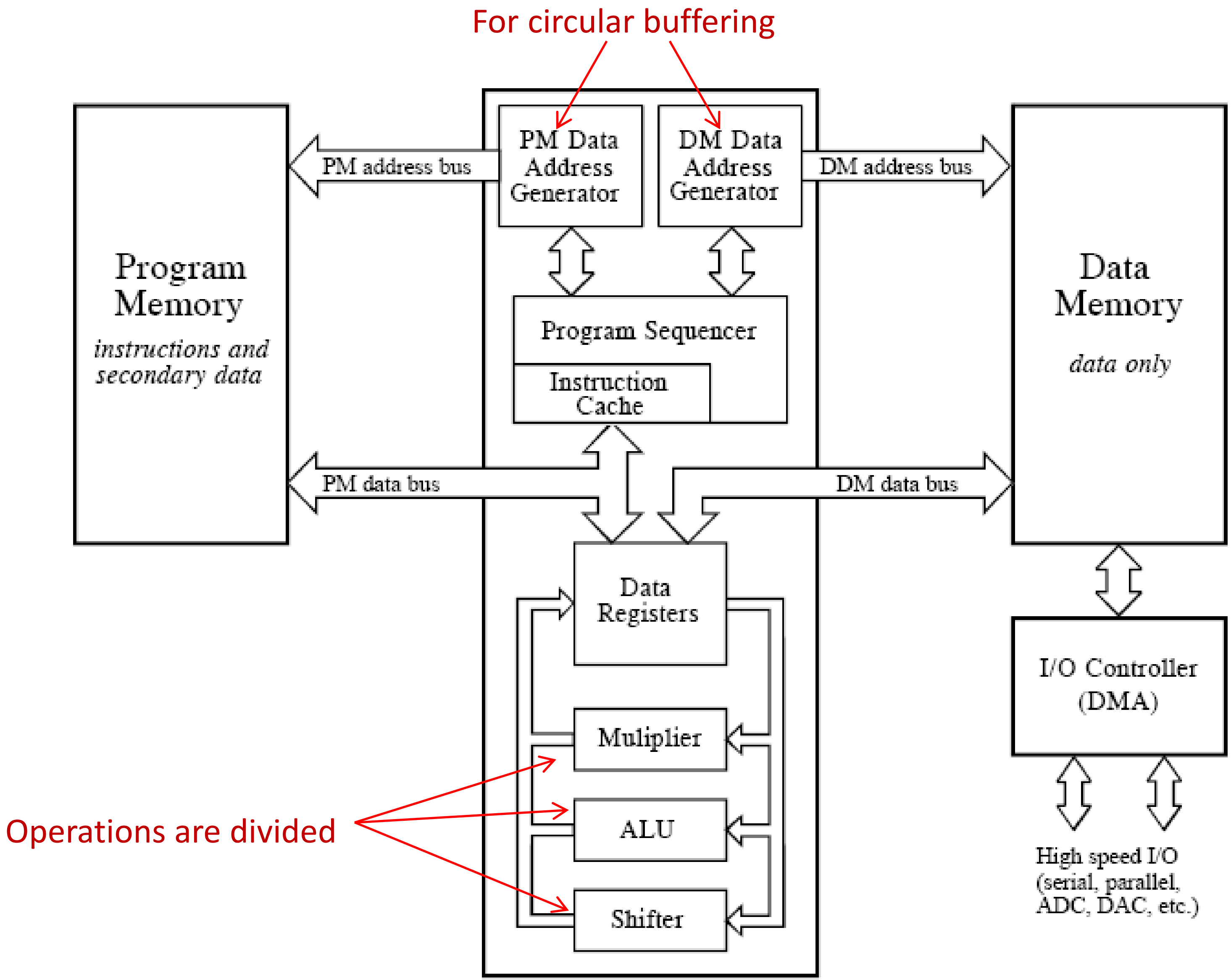


c. Super Harvard Architecture (*dual memory, instruction cache, I/O controller*)



Most recent instructions

Typical DSP Architecture

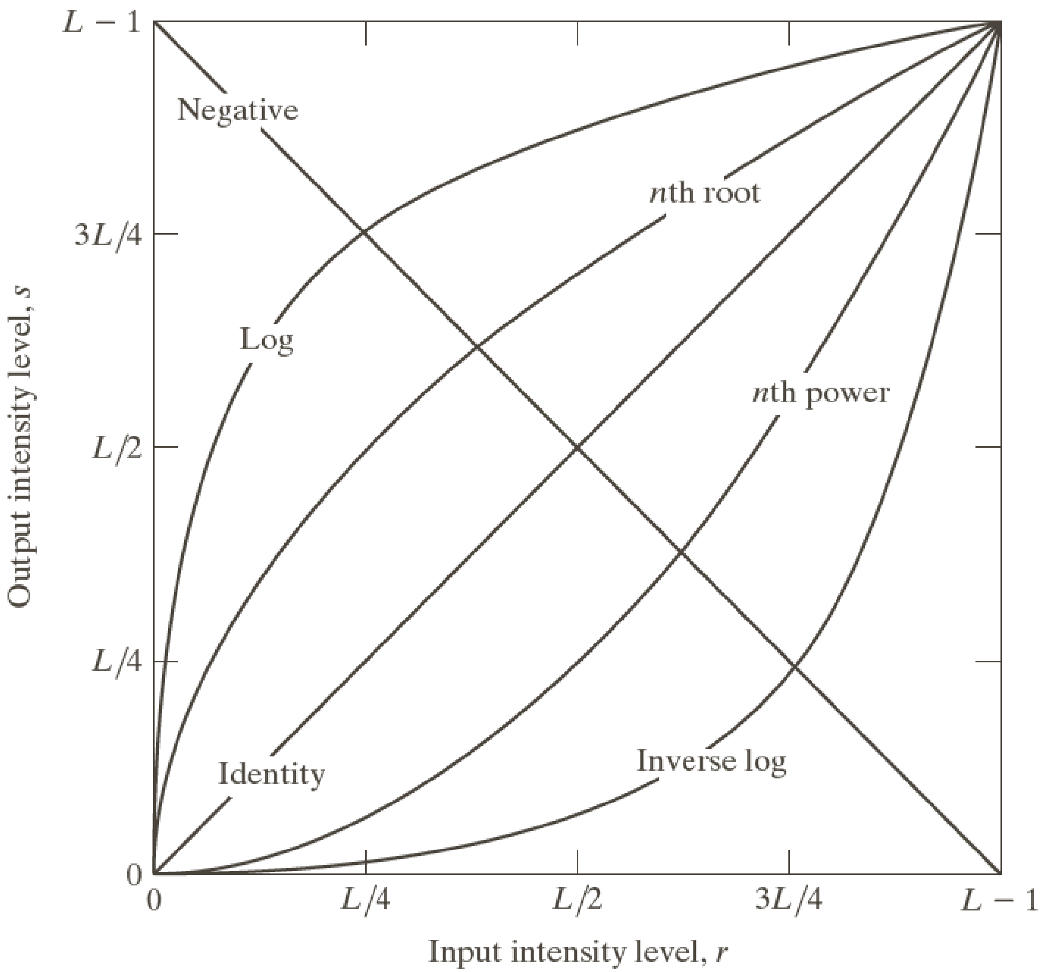


Application of DSP in Image Processing: Basic Intensity Transfer Functions

- Linear
- Logarithmic
- Power-law

Grey Image has intensity in the range [0 – 255] i.e. it uses 8 bits.

Intensity is transformed for better viewing.

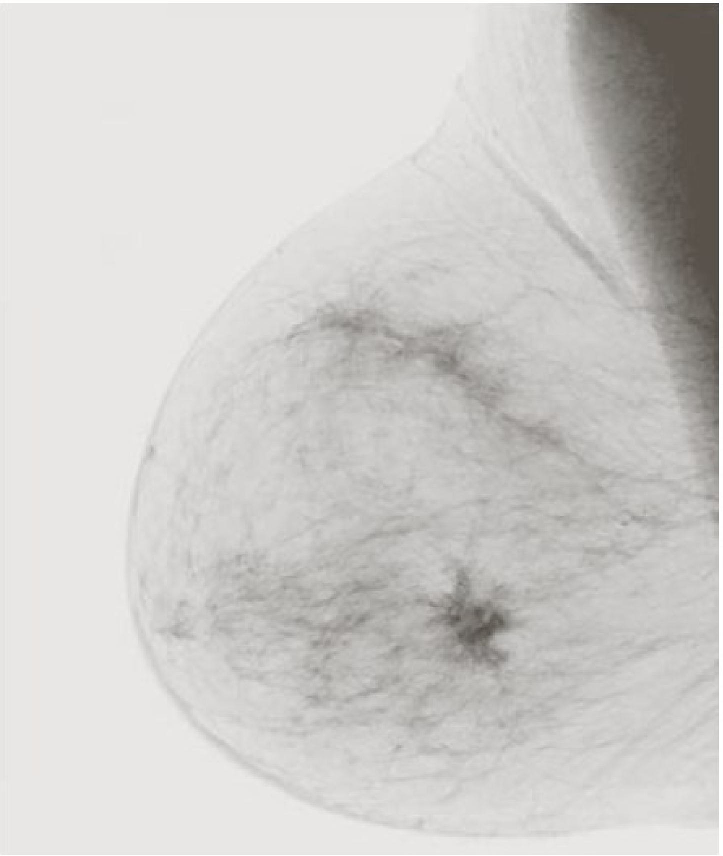


Negative Transformation

Intensity level: [0, *L* - 1]

$$s = L - 1 - r$$

Original Image



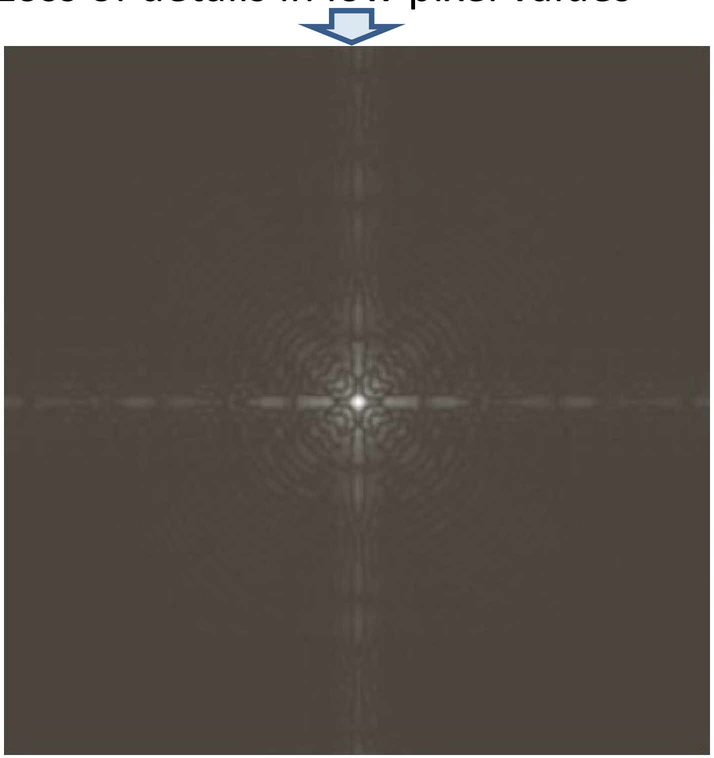
Negative Image

Log Transformation

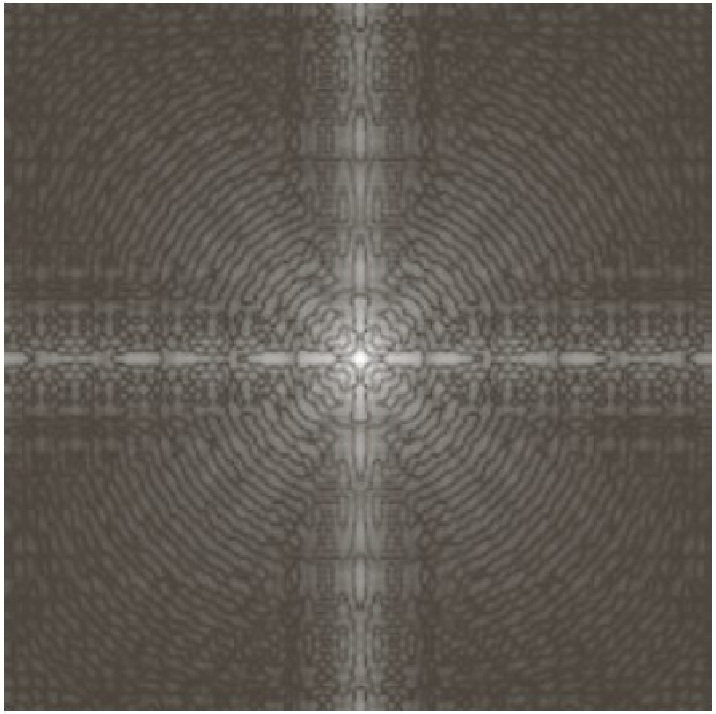
$$s = c\log(1 + r)$$

Compresses the dynamic range of images with large variations in pixel values.

Loss of details in low pixel values



Original Image (Fourier Spectrum)

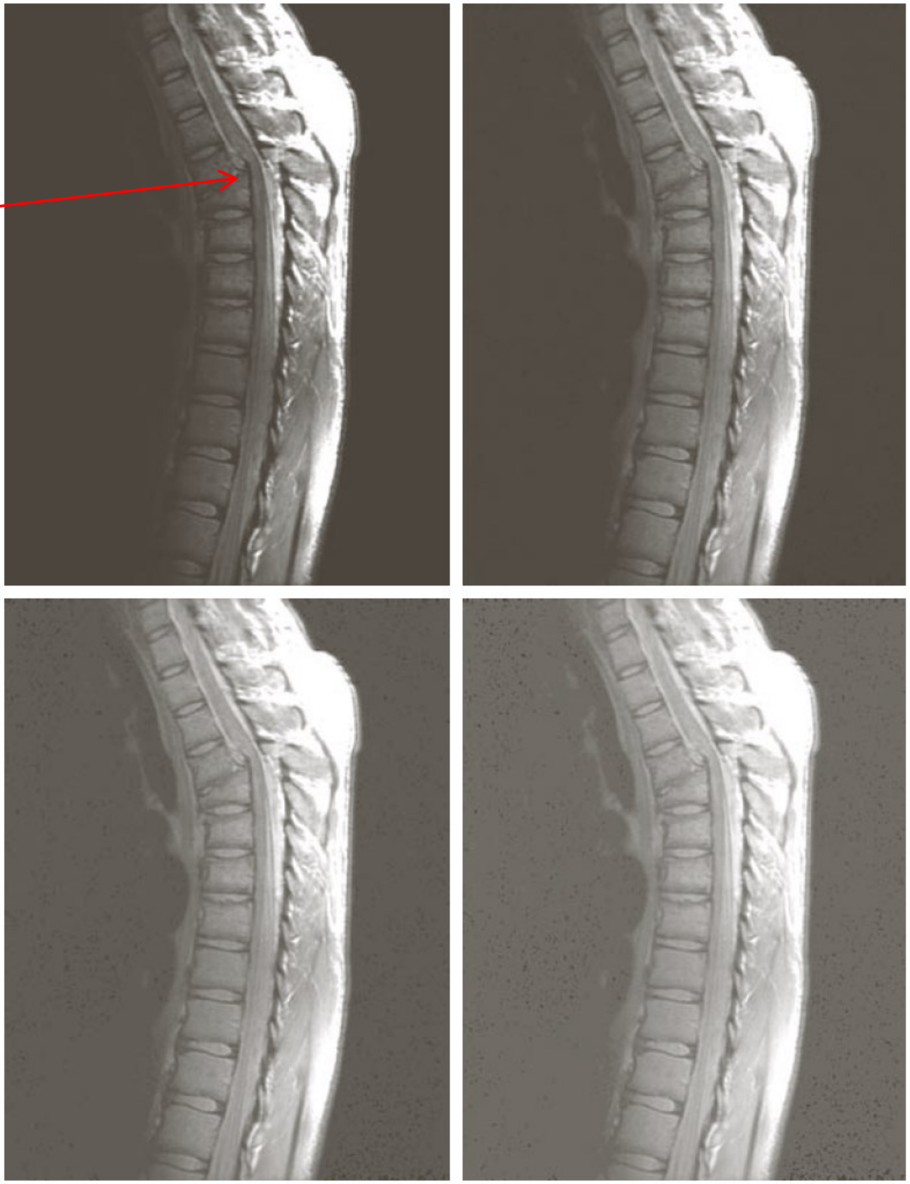


Log Transformed Image

Contrast manipulation: Power-Law

MRI of a fractured spine.

$$s = cr^{\gamma}$$



$\lambda = 0.6$

$\lambda = 0.4$

$\lambda = 0.3$

Best contrast

Washed out

Filter Mask

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{k=1}^{9} w_k z_k = \mathbf{w}^T \mathbf{z}$$

Smoothing Filter (low pass) mask:

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

Equal weight

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Weighted average

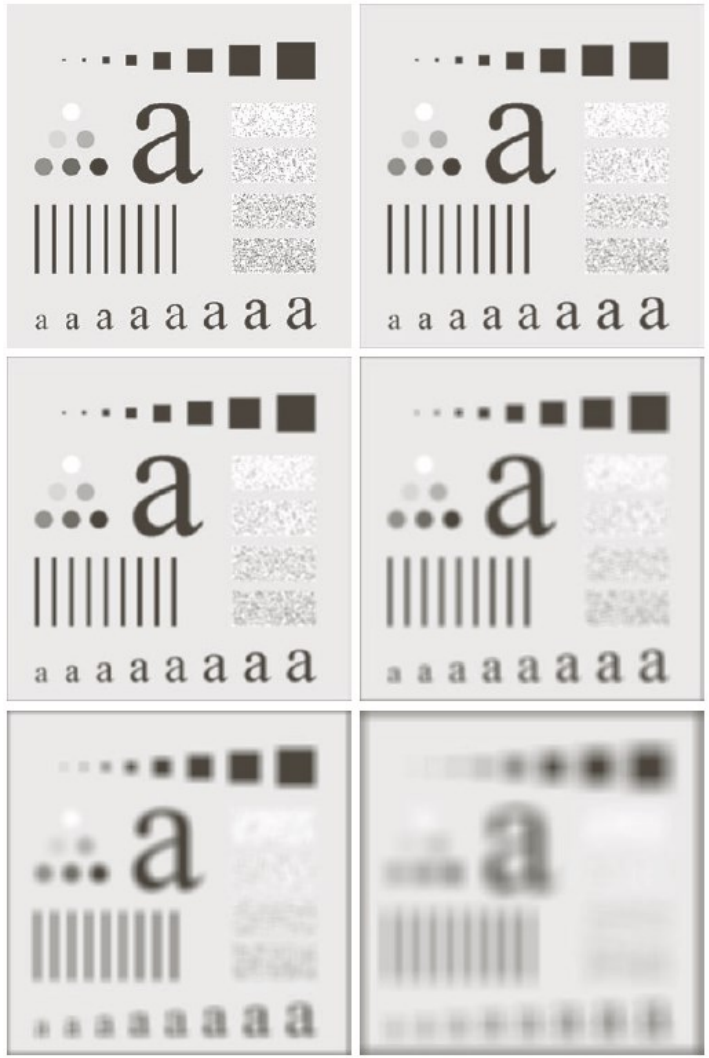
$$g(x,y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)}$$

Normalization factor

Smoothing Effect

Original image

5 X 5 mask



3 X 3 mask

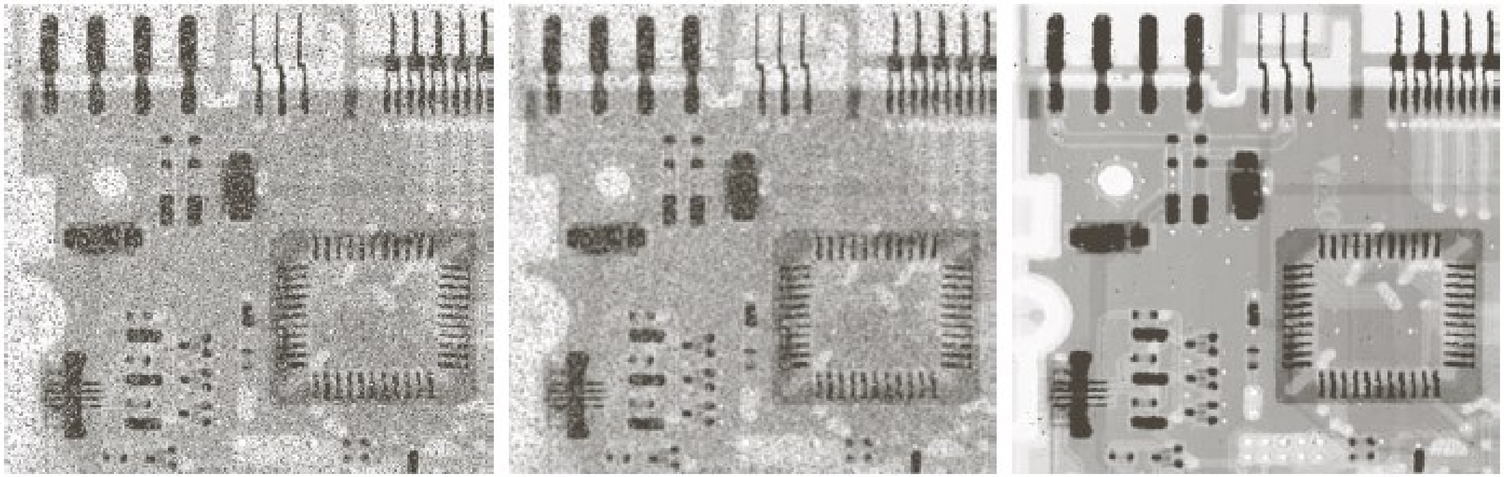
Noise is less pronounced.

35 X 35 mask

Completely blurred!

Median Filtering

Find the median in the neighborhood, then assign the center pixel value to that median.



a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Digital (Image) Watermarking

Inserting information (watermark) into images in such a way that the watermark is inseparable from the images.

- Copyright identification.
- User identification.
- Authenticity determination.
- Automated monitoring.
- Copy protection.

$$f_w = (1-\alpha) f + \alpha w$$

Watermarked image (f_w)

Original image (f)

Watermark (w)

Example of Watermarking - I

Visible

a
b c

FIGURE 8.50
A simple visible watermark: (a) watermark; (b) the watermarked image; and (c) the difference between the watermarked image and the original (non-watermarked) image.

α= 0.3

Textbook: Digital Image Processing by Gonzalez and Woods

Example of Watermarking - II

Invisible

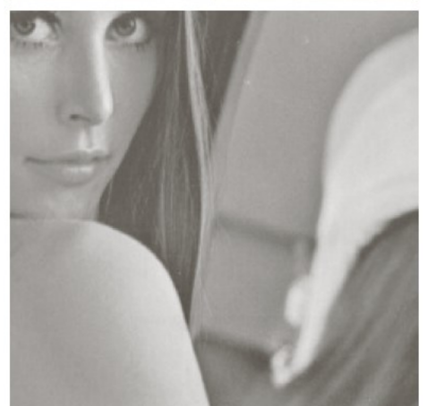
Watermark is inserted in image's two LSBs.

$$f_w = 4\left(\frac{f}{4}\right) + \frac{w}{64}$$

Sets two LSBs of image to 00.

Shifts two MSBs into two LSBs.

By zeroing 6 MSB and scaling the remains to full intensity level, we get



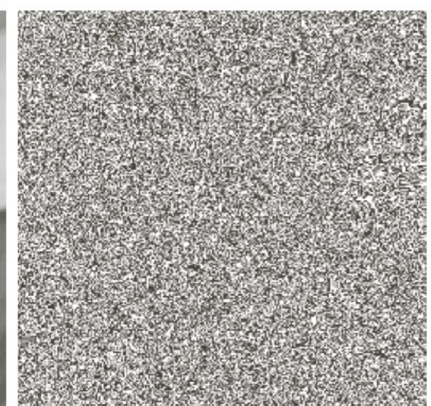
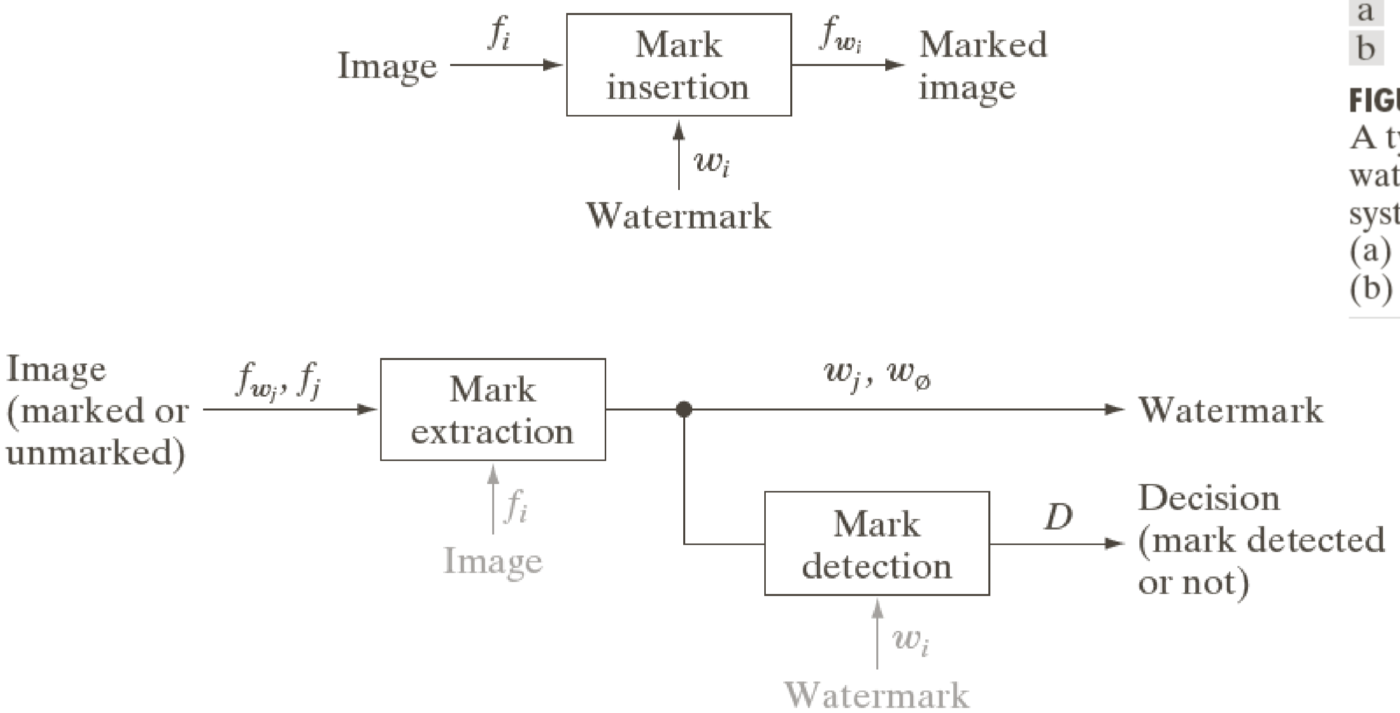


Image Watermarking System



a
b

FIGURE 8.52
A typical image watermarking system: (a) encoder; (b) decoder.

Private / restricted key system: f_i and w_i are used.

Public / unrestricted key system: f_i and w_i are unused.