LEC5 **DSA5203**

→ **Summary for PART 06, 07, 08**

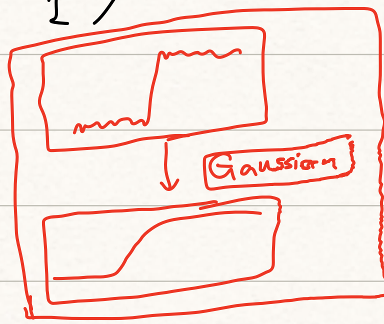
① Edge Detector → **basic idea** is: **image gradient**

a) Sobel Operator : (Prewitt operator)

- Gaussian Smooth
- Discrete Gradient

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

b) **LoG** (Laplacian of Gaussian) **Operator**
(also called **Marr Operator**)



$\boxed{\text{Zero-Crossing}}$ to compensate for the $\boxed{\text{margin-enlarging}}$ when using **Gaussian Smoothing**

$$LoG = \nabla G = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Note: $LoG \otimes I \Longleftrightarrow \nabla (G \otimes I)$

equivalent

Gaussian Smooth

$\boxed{\text{Laplacian Operator}}$

$$\nabla F = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) F$$

(3).计算零交叉(Zero crossing)

零交叉的实现较为简单，由于零交叉点意味着至少两个相邻的像素点的像素值异号，一共有四种需要检测的情况：左右，上下，两个对角，其中如果滤波后的图像g(x, y)的任意像素p处的四种情况其中一组的差值的绝对值超过了设定的**阈值**，则我们可以称p为一个零交叉像素，示例如下：

```
[rx, cx] = find( b(rr, cc) < 0 & b(rr, cc+1) > 0 ...
    & abs( b(rr, cc)-b(rr, cc+1) ) > thresh );   % [- +]
e((rx+1) + cx*m) = 1;
```

图 3-10

此为Marr-Hildreth其中一小部分，检测[- +]这一情况是否满足，其中thresh为提到的阈值

到这里我们就学习了两种最为流行且经典的先进边缘检测算法与思想，接下来说的是一些经验与算法的选择参考

c) **Canny Edge Detector** → **most successful one**

1. Gaussian Smooth

2\. Calculate $\begin{cases} \text{Gradient Strength} \\ \text{Gradient Direction} \end{cases}$

$$\begin{cases} \text{Strength}(m,n) = \text{sqrt}\left(f_x^2(m,n) + f_y^2(m,n)\right) \\ \text{direction}(m,n) = \arctan\left(\dfrac{f_y(m,n)}{f_x(m,n)}\right) \end{cases}$$

Sobel Operator

3\. NMS → Non-Maximal Supression

for one pixel (m,n), guarantee it has the largest gradient strength!

→ may also use Laplacian of Gaussian → zero-crossing

a) in its 8 neighbours
b) in its gradient direction

4\. 2-threshold about Gradient Strength

set to 1 ← Gradient Strength $(m,n) \geq T_{upper}$

set to 0 ← Gradient Strength $(m,n) \leq T_{lower}$

② PART 06 ⟶ $\boxed{\text{De-noise}}$

1. model

$$\underbrace{f_\eta}_{\text{obser.}} = \underbrace{f}_{\text{GT}} + \underbrace{\eta}_{\text{noise}}$$

Goal is to : find $\hat{f}$ from f_η to $\boxed{\text{approximate } f}$

$\boxed{\text{one measure is : } \min_{f'} MSE(f', f)}$

2. Method Outline

LINEAR
- a) Low-Pass filter ⟶ shrink variance
- b) Wiener Filter ⟶ consider in $\boxed{\text{Frequency-Domain}}$

NON-LINEAR ⟵ c) Median ....

- d) $\boxed{\text{Wavelet Transform}}$

3 a). Low-Pass Filter

idea: suppose
- $f[n]$
- $\eta[n]$ is i.i.d distributed

after Low-Pass filter, we attain : $\frac{1}{3}$ variance !

$$\hat{f}[n] = \frac{1}{3}\left(f[n-1] + f[n] + f[n+1]\right) + \boxed{\hat{\eta}[n]}$$

is not accurate though

3b) Wiener Filter (Optimal Linear Filter)

idea: find the best $\hat{f}_n = h \otimes f_n$ for some h

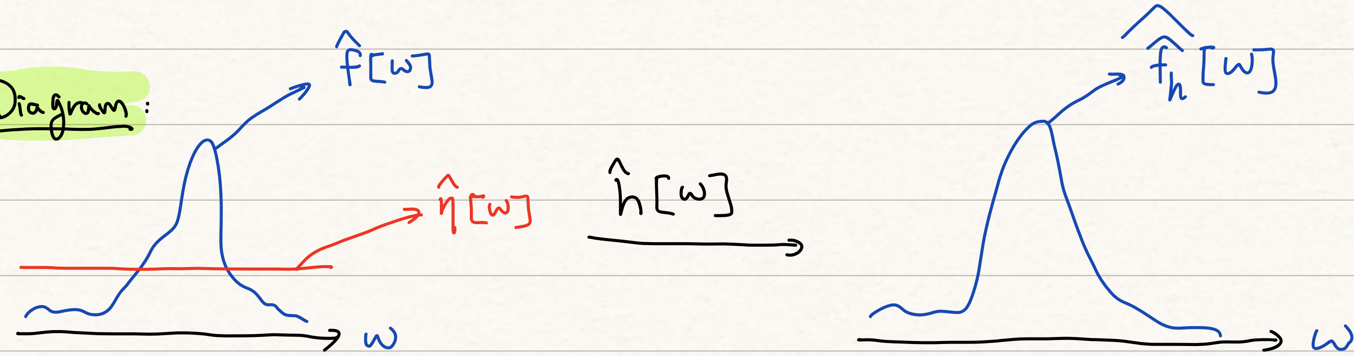
↳ in the image-domain

→ Q: What will happen in Frequency-Domain?

Answer: $\hat{\hat{f}}_n[w] = \hat{h}[w] \times \hat{f}_n[w]$

$$= \hat{h}[w] \times \left(\hat{f}[w] + \hat{\eta}[w] \right)$$

Diagram:



Remark: From this diagram, intuitively, we want the filter h

to
- shrink more on High-Frequency area
- shrink less on Low-Frequency area

↳ Low-Pass Filter!

↓

Another Perspective to show why low-pass filter works!

To conclude: $\hat{\hat{f}}_h[w] = \hat{h}[w] \cdot (\hat{f}[w] + \hat{\eta}[w])$

$$\hat{\hat{f}}_h[w] \approx \hat{f}[w]$$

$$\Rightarrow h := \underset{h}{\operatorname{argmin}}\ \mathbb{E}_{\hat{\eta}[w], \hat{f}[w]} \left[\| \hat{\hat{f}}_h[w] - \hat{f}[w] \|_2^2 \right]$$

$$= \underset{h}{\operatorname{argmin}}\ \mathbb{E}_{\hat{\eta}[w], \hat{f}[w]} \left[\| (1 - h[w]) \hat{f}[w] - h[w] \hat{\eta}[w] \|_2^2 \right]$$

(for each w)

under some ASSUMPTION:

$$\boxed{\hat{h}[n] = \frac{\mathbb{E}[\| \hat{f}[n] \|_2^2]}{\mathbb{E}[\| \hat{f}[n] \|_2^2] + \mathbb{E}[\| \hat{\eta}[n] \|_2^2]}}$$

↓ iDFT

$\boxed{h[n]}$ → $\boxed{\text{filter we want}}$

→ give us some Guideline to design linear filter h

↓

Limitation → cannot calculate in real life

3C) Median Filter

↓

$\boxed{\text{not very interesting}}$

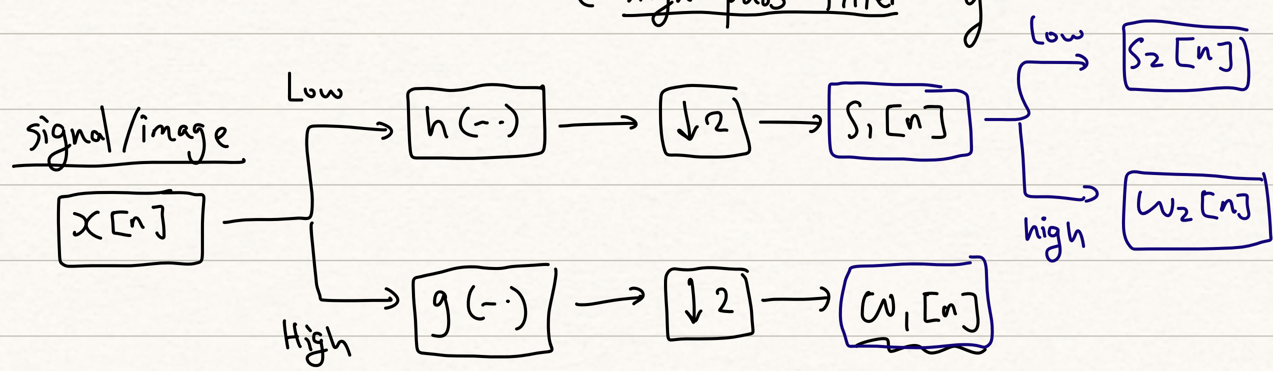
3d) Wavelet Filter

↓

Discrete Wavelet Filter (DFT) → Wavelet Decomposition

inverse DFT (iDFT) → Wavelet Reconstruction

1. DFT → require
   - low-pass filter h
   - high-pass filter g



2. iDFT