C9

12. Noise

→ origins of noise: detector sensitivity, electrical noise, data transmission errors, air turbulance

1) Data drop-out noise = random bits "corrupted" or "lost" ("snow" on images)

↳ reduced by: threshold average filter; median filter

2) Fixed pattern noise → caused by sensors

↳ corrected by a "point by point" calibration of each sensor

↳ suitable Fourier filters

3) Detector or Shot Noise

↳ gaussian approximation

↳ low-contrast approximation

12. 2. Properties of Additive Noise

$$f(i,j) = s(i,j) + n(i,j) \qquad \sigma_f^2 = \sigma_s^2 + \sigma_n^2$$

$s(i,j)$: true signal; $n(i,j)$: noise

$\xRightarrow{\text{fourier}} F(k,l) = S(k,l) + N(k,l)$ → degrading the image (smoothened at the edges)

12.3. Signal to Noise ratio (SNR)

$$SNR = \sqrt{\frac{\sigma_f^2}{\sigma_n^2} - 1}$$

$$\sigma_s^2 = \langle | s(i,j) - \langle s(i,j) \rangle |^2 \rangle$$

$$\sigma_n^2 = \langle | n(i,j) |^2 \rangle$$

$$SNR = \frac{\sigma_s}{\sigma_n}$$

C10/11 13. Filtering

↳ filtering operations ← linear (convolution): $g(i,j) = h(i,j) \odot f(i,j)$ ↳ filter func. ↳ input
nonlinear (in real space)

13.1A. Linear filtering

1) Fourier space convolutions

$g(i,j) = \mathcal{F}^{-1}\{F(k,l)H(k,l)\}$ → fourier on both functions, multiply, inverse fourier
↳ of $f(i,j)$ ↳ of $h(i,j)$

- computational cost doesn't depend on filter f

2) Real space convolutions
- by direct application of the shift and multiply def. of convolution

$$g(i,j) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}-1} \sum_{n=-\frac{M}{2}}^{\frac{M}{2}-1} h(m,n) \cdot f(i-m, j-n)$$

- computational cost proportional w/ filter f

13.2 Fourier space filters

a) Low-pass filters: → attenuates only high spatial freq.
→ ideal (creates ringing effects): $H(k,l) = \begin{cases} 1, & k^2+l^2 \le w_0^2 \\ 0, & \text{else} \end{cases}$
→ smooth cut-off filter: Gaussian: $H(k,l) = \exp\left(-\frac{w}{w_0}\right)^2$

b) High-pass filters: → attenuates low spatial frequencies
→ enhancing the edges in images
→ ideal: $H(k,l) = \begin{cases} 0, & k^2+l^2 \le w_0^2 \\ 1, & \text{else} \end{cases}$
→ smooth cut-off filter: Gaussian: $H(k,l) = 1 - \exp\left(-\frac{w}{w_0}\right)^2$

c) Band-pass filters = low-pass + high-pass: $H(k,l) = H_L(k,l) + H_H(k,l)$

13.3 Real space filters

↳ finite range of filtering mask

a) Real space averaging: $\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$ → 5 pixel average $\quad \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$ → 9 pixel average

b) Real space differentiation ← x differential → effect on enhancing vertical edges
y differential → — || — horizontal edges

13.4. Uses of linear filters

↳

13.1B Real space non-linear filtering

! real = shift & multiply
=> output as a convolution

↳ involves the replacement of the summations by the specified non-linear operator to give

13.2. Shrink and expand filters

Shrink: the non-linear operator = min
Expand: O = max

=> remove all bright regions smaller than the filter size while leaving large regions almost unaltered

13.3. Treshold average filter:

↳ forming a local average but not including the central pixel; this average is then compared w/ the central pixel value; if it differs by more than a present treshold, then the central pixel is considered corrupted and replaced by the average

↳ removal of snow effect

$$g(i,j) = A \text{ for } |A - f(i,j)| > T$$
$$= f(i,j) \text{ else}$$

13.3. Median filters

$f(i) = 61, 10, 9, 11, 9$

2 values (9 & 9) less than 3
2 values (61, 11) less than 3

=> 10 = median

↳ median is defined as the middle value

13.4 Homomorphic filtering

↳ for correction of illumination variation

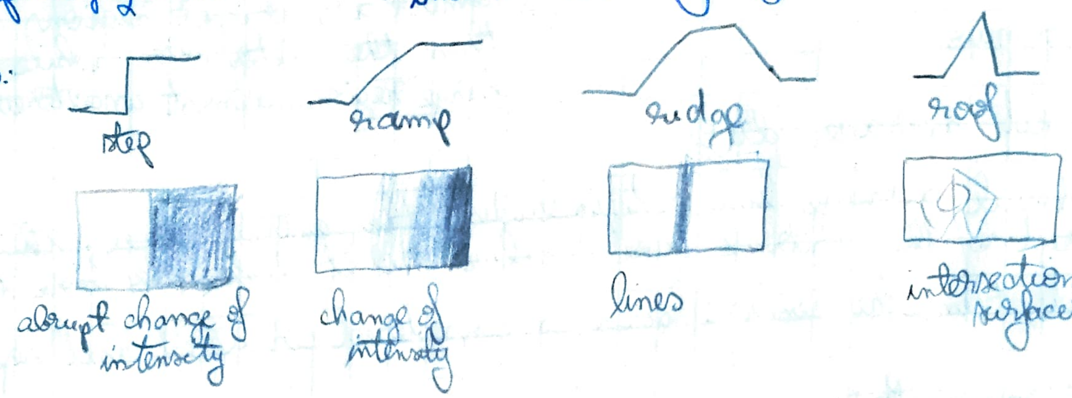
C 12/13 # 14. Edge based segmentation

14.1 Edge detection

↳ edges are generated by: - changes in the surface reflectance / orientation
- an object being partially covered by another
- indirect covering by shadows

high spatial frequencies

↳ Edges:



↳ steps in edge segmentation:
- preprocessing
- edge pixels detection and identification
- post processing - improve results, extract edges, close contours

↳ edge detection:
↳ enhance and mark edge pixels and extract them
↳ need some type of high-pass filter → 1st or 2nd order differentials

I. First order differentials

↳ $\left|\frac{\partial f(x,y)}{\partial x}\right|$ → vertical edge ↳ $\left|\frac{\partial f(x,y)}{\partial y}\right|$ → horizontal edge

↳ but we want both ⇒ $\frac{\partial f(x,y)}{\partial x}\cdot\cos\theta + \frac{\partial f(x,y)}{\partial y}\cdot\sin\theta$

↳ modulus of the gradient: $|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$

$|\nabla f(x,y)| > T \Rightarrow$ Edge

$|\nabla f(x,y)| < T \Rightarrow$ no Edge

$$\frac{\partial f(i,j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \circ f(i,j)$$

$$\frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \circ f(i,j)$$

Sobel filter

$$\frac{\partial f(i,j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \circ f(i,j)$$

$$\frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \circ f(i,j)$$

↳ most common 1st order differential edge detector

Problems:

↳ arbitrary treshold value → avoid by setting T such that x% of the image is classified as edge
↳ thick edges → if T is too low
→ avoid by edge thinning techniques (reduce to single pixels while keep edge connected)
↳ broken edges → avoid by range or edge-joining techniques
↳ noise points → avoid by modified treshold filter to remove isolated points

II. Second order differentials

↳ Laplacian: $\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$

⇒ a single convolution of: $\nabla^2 f(x,y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \circ f(i,j)$

Pre-processing

↳ to reduce the noise, smooth the image before Laplacian
(↳ ex. nine point average)

Problems:

↳ thin edges
↳ closed loops
↳ noise problems: Laplacian is a high pass filter, so enhances high frequencies

14.2. Methods based on filtering followed by differential operators

1D situation: – edge = step signal