ARAVIND
2019102074

DIP Lecture-6

3/09/2021

Spatial filtering

Image filtering

Process of transforming original image to get output with desirable properties

Filtering can be done in spatial / frequency domains

- Directly on pixels
- Apply FT and then perform filtering then later do inverse F.T

$$y(t) = x(t) * h(t)$$
$$Y(f) = X(f) \cdot H(f)$$

NOTE: No correct output, depends on user needs.

Filtering (Preprocessing) Biometrics (face, iris, fingerprint)

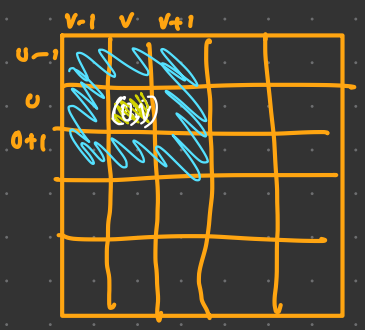
Based on type of face image, we do some transformations to obtain face feature for recognition

Mean / Average filter (Smoothing)

→ Take values from neighbors, take its average and assign to center
NOTE: Need to round off while assigning after finding average

→ Used to denoised and helps in tackling extremes

→ Essentially does blurring to the image



The average filter could be :

$$H = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$ → Sums up to 1

(3x3)

This is also known as weight mask or kernel

Formula:

$$I'(u,v) = \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j) \cdot H(i,j)$$

Effects:

→ Gets more blurred as mask size increases (more blurred

→ Lose edge information

Generally prefer odd, we can get center pixel. Hence even filter not very common.

Types of noise

→ Salt & Pepper : Random white, black

→ Impulse: Only random white

→ Gaussian: Random gray level variation

In terms of histogram perspective, it pushes pixel values towards median values

Can do repeated averaging using same smaller filter instead of taking larger filter

↳ Reason: → Computational efficiency & memory

↳ Lesser external noise (lesser neighbourhood is taken)

Weighted Averaging:

assign weights

↳ Rely more on some pixels & lesser on other pixels so accordingly

Modified formula:

$$I'(u,v) = \frac{\sum_{j=-a}^{a} \sum_{i=-b}^{b} I(u+i, v+j) \cdot H(i,j)}{\sum_{j=-a}^{a} \sum_{i=-b}^{b} H(i,j)}$$

Gaussian Function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \rightarrow \text{1D version}$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \rightarrow \text{2D}$$

→ Continous function (Assume a σ value)

For same filter size, as sigma increases, it'll be short & wide

as " decreases, it'll be tall & thin

Averaging vs Gaussian filtering:

Averaging ↓ Sharp decay

Gaussian filtering ↓ Smoother decay

Gaussian filter coeff:

S = Size of filter

S = round (σ)

↙ Heuristic

Usually use derivatives to compute edges as there is significant transition in pixel intensity as derivatives give rate of change

$$\frac{\delta f(x,y)}{\delta x} \sim f(x+1,y) - f(x,y)$$ → Digital Approx.

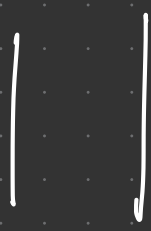
↘ Here $\Delta x = 1$ pixel

2nd derivatives:

$$\frac{\delta^2 f(x,y)}{\delta^2 x} \sim (f(x+1,y) - f(x,y)) - (f(x,y) - f(x-1,y))$$

In second derivative, if there's a zero crossing we can identify. It defines onset of image.

Image gradient & edges:



Gradient in x

Gradient in y

$$\nabla f = \left[\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}\right]$$ →

For vertical: $\left[\frac{\delta f}{\delta x}, 0\right]$

For horizontal: $\left[0, \frac{\delta f}{\delta y}\right]$

Prewitt Edge filters:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

↳ For vertical edges

$$G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

↳ For horizontal edges