Chapter 4 Image Filtering

Image enhancement for elliminating undesirable characteristics

4.1 Histogram Modification

Unevenly distributed gray values => Histogram equalization
help threshold selection
subjective quality
good for human viewer

* scaling
graylevel; [a, b] -> [z1 , zk]

$$z' = \frac{z_k - z_1}{b-a}(z-a) + z_1$$
$$= \frac{z_k - z_1}{b-a} z + \frac{z_1 b - z_k a}{b-a}$$

problems; empty bins

* given histogram

pi at zi in original histogram
qi at zi in desired histogram

$$\sum_{i=1}^{k_1-1} p_i \leqq q_1 < \sum_{i=1}^{k_1} p_i$$ => z1, z2, ...zk1-1 into z1

$$\sum_{i=1}^{k_2-1} p_i \leqq q_1 + q_2 < \sum_{i=1}^{k_2} p_i$$ => zk1, ,,, zk2-1 into z2

* randomly distribute

new gray level; $k + [n * r]$

n; desired histogram interval
r; [0, 1) uniform random number

accumulating uniform number => Gaussian distribution

* probability theory ; assign

4.2 Linear Systems

Image processing operations; Linear system

Linear Space Invariant (LSI) system
irrespective of the position of the input pulse

$$a \cdot f_1(x, y) + b \cdot f_2(x, y) => a \cdot h_1(x, y) + b \cdot h_2(x, y)$$

Convolution;

$$\begin{aligned} h(x,y) &= f(x,y) * g(x,y) \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')g(x-x',y-y')dx'dy' \end{aligned}$$

For discrete functions;

$$\begin{aligned} h[i,j] &= f[i,j] * g[i,j] \\ &= \sum_{k=1}^{n}\sum_{l=1}^{m} f[k,l]g[i-k,j-l] \end{aligned}$$

weighted sum of image pixels

$h[i,j] = A\ p_1 + B\ p_2 + C\ p_3 + D\ p_4 + E\ p_5 + F\ p_6 + G\ p_7 + H\ p_8 + I\ p_9$

g[i,j] ; convolution mask
linear operation, spatially invariant since same filter weights used

Fourier Transform

$$\begin{aligned} F(u,v) &= \mathcal{F}\{f[k,l]\} \\ &= \sum_{k=1}^{n}\sum_{l=1}^{m} f[k,l]e^{-jku}e^{-jlv} \end{aligned}$$

amplitude and phase component of each frequency
time consuming expensive

in machine vision; nonlinear, spatially varying algorithm
not use FFT

4.3 Linear Filters

noise; salt and pepper noise; random white and black
impulse noise; random white
Gaussian noise; variation in intensity, Gaussian distribution or uniform distribution,
good model for sensor noise

Linear smoothing filter; good for removing Gaussian noise, etc.
design by assigning weights => convolution mask
saptially invariant,

Nonlinear filter; not weighted sum of pixels
can be spatially invariant
ex; median filter

* Mean filter
local averaging filter

$$h[i,j] = \frac{1}{M} \sum_{(k,l) \in N} f[k,l]$$

M; number of pixels in the neighborhood N

Ex;

$$h[i,j] = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{k=j-1}^{j+1} f[k,l]$$

N; controls the amount of filtering
larger N, greater degree of filtering, loss of image detail

Linear filter design; single peak
symmetry
step change -> gradual change

4.4 Median filter
linear filter; blur sharp image

replace each pixel value as median of the gray level in the local neighborhood
effective; salt and pepper, impulse noise

(1) take local window
(2) sort the pixels into ascending order by graylevel
(3) select the value of the middle pixel as the new value for pixel [i,j]

4.5 Gaussian Smoothing

weighting function; similar to Gaussian function

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

for image processing

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

* comment

effective low-pass filter; spatial and frequency domains

efficient to implement

- rationally symmetric in two dimension; edge in random direction
- single lobe; weight decrese monotonically with distance

edge is local feature

- Fourier transform of a Gaussian is also Gaussian

image is not corrupted by undesirable high frequency

- larger σ => wider Gaussian smoothing

adjust degree of smoothing (too much or too little)

- separable

two dim. Gaussian convolution;

one dim. Gaussian then one dim. Gaussian in orthogonal direction

4.5.1 Rotational symmetry

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

in polar coord. $r^2 = i^2 + j^2$

$g(r,\theta) = e^{-\frac{r^2}{2\sigma^2}}$; does'nt dependent on Θ

4.5.2 Fourier Transform Property

$$\mathcal{F}\{g(x)\} = \int_{-\infty}^{\infty} g(x)e^{-j\omega x}dx$$
$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} e^{-j\omega x} dx$$
$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} (\cos\omega x + j\sin\omega x) dx$$
$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \cos\omega x dx + j\int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \sin\omega x dx$$
$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \cos\omega x dx$$
$$= \sqrt{2\pi}\,\sigma\; e^{-\frac{\omega^2}{2\nu^2}}, \quad \nu^2 = \frac{1}{\sigma^2}$$

narrow spatial domain; less smoothing

pass more of high frequency and larger bandwidth

4.5.3 Gaussian Separability

$$g[i,j]*f[i,j] = \sum_{k=1}^{m}\sum_{l=1}^{n} g[k,l]f[i-k,j-l]$$
$$= \sum_{k=1}^{m}\sum_{l=1}^{n} e^{-\frac{(k^2+l^2)}{2\sigma^2}} f[i-k,j-l]$$
$$= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} \{\sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} f[i-k,j-l]\}$$

convolution is associative and commutative

using horizontal Gaussian mask,

(1) apply horizontal Gaussian mask,

(2) transpose the (1) step result, store tempory memory

(3) apply horizontal Gaussian mask to the (2) step tempory memory

(4) transpose the result

4.5.4 Cascading Gaussians

$$g(x)*g(x) = \int_{-\infty}^{\infty} e^{-\frac{\xi^2}{2\sigma^2}} e^{-\frac{(x-\xi)^2}{2\sigma^2}} d\xi$$
$$= e^{-\frac{x^2}{4\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{\xi^2}{\sigma^2}} d\xi$$
$$= \sqrt{\pi}\,\sigma e^{-\frac{x^2}{2(\sqrt{2}\sigma)^2}}$$

4.5.5 Designing Gaussian Filters

(1) Approximation using binomial expansion

$$(1+x)^n = \binom{x}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n$$

Pascal's triangle

ex)

1	4	6	4	1

(2) Approximation of Gaussian filter

$$g[i,j] = ce^{-\frac{(i^2+j^2)}{2\sigma^2}} \quad => \quad \frac{g[i,j]}{c} = e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

ex) $\sigma^2 = 2$, $n = 7$

approximate and make integer

[i,j]	-3	-2	-1	0	1	2	3
-3	1	4	7	10	7	4	1
-2	4	12	26	33	26	12	4
-1	7	26	55	71	55	26	7
0	10	33	71	91	71	33	10
1	7	26	55	71	55	26	7
2	4	12	26	33	26	12	4
3	1	4	7	10	7	4	1

normalize

$$\sum_{i=-3}^{3}\sum_{j=-3}^{3} g[i,j] = 1115$$

$$=> \quad h[i,j] = \frac{1}{1115}(f[i,j] * g[i,j])$$

4.5.6 Discrete Gaussian Filters

nXn discrete Gaussian filter => mXm Gaussian filter

=>> (n+m-1)X(n+m-1) Gaussian filter