Computer Vision

Correlation

Correlation is process of moving filter mask often referred to as kernel over the image and computing the sum of products at each location.

difference betn correlation & convolution

- Correlation is measurement of the similarity between two signals / sequences.
- Convolution is measurement of effect of one signal on the other signal.

* use of correlation

Often used in applications where we need to measure the similarity betn images or parts of images.
(eg template matching)

- Convolution (linear operator)

Similar to correlation except that the mask is first flipped both horizontally & vertically.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

$$g(i,j) = \omega(i,j) * f(i,j)$$

Convolution $= \sum_{s=-k/2}^{k/2} \sum_{t=-k/2}^{k/2} \omega(s,t)\, f(i-s, j-t)$

Correlation $\Rightarrow g(i,j) = \omega(i,j) \bullet f(i,j)$

$$= \sum_{s=-k/2}^{k/2} \sum_{t=-k/2}^{k/2} \omega(s,t)\, f(i+s, j+t)$$

Note if $\omega(i,j)$ is symmetric (ie $\omega(i,j) = \omega(-i,-j)$), then convolution is equivalent to correlation

Concept

For image → 1

padded →

0	0	0	0	0	
0	0	0	0	0	
0	0	1	0	0	
0	0	0	0	0	
0	0	0	0	0	

apply mask

1	2	3
4	5	6
7	8	9

Start from a position

1) Correlation

mask =

1	2	3
4	5	6
7	8	9

final image =

9	8	7
6	5	4
3	2	1

since 9 reach 1 first then mask goes down & other things happen

2) Convolution

mask becomes

9	8	7
6	5	4
3	2	1

so final image =

1	2	3
4	5	6
7	8	9

1 D continuous convolution

is defined as

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(a)\, g(x-a)\, da$$

* Think of $f(x)$ as image & $g(x)$ as mask although you can reverse their roles.

Convolution is commutative

$$f(x) * g(x) = g(x) * f(x)$$

Observations

1) Extent of $f(x) * g(x)$ is equal to extent of $f(x)$ plus extent of $g(x)$

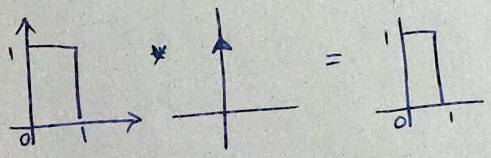
2) for every x, limits of integral are determined as follows:

- Lower limit : MAX (left limit of $f(x)$, left limit of $g(x-a)$)
- Upper limit : MIN (right limit of $f(x)$, right limit of $g(x-a)$)

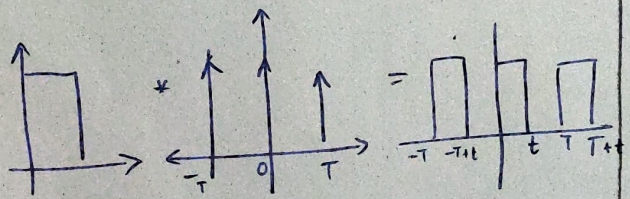
3) Convolution with an impulse (delta function)

$$f(x) * \delta(x) = \int_{-\infty}^{\infty} f(a)\, \delta(x-a)\, da = f(x)$$

(since $\delta(x-a) = 1$ if $a = x$)



4) convolution with train of impulse



5) Convolution Theorem

Convolution in spatial domain is equivalent to multiplication in the frequency domain

$$f(x) * g(x) \longleftrightarrow F(u)\,G(u)$$

6) Multiplication in spatial domain is equivalent to convolution in frequency domain

2D convolution

Defination

$$f(x,y) * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(a,b)\, g(x-a, y-b)\, da\, db.$$

2D convolution theorem

$$f(x,y) * g(x,y) \longleftrightarrow F(u,v)\, G(u,v)$$

$$f(x,y)\, g(x,y) \longleftrightarrow F(u,v) * G(u,v)$$

1D / 2D FT

Fourier Transform

- Fourier transform stores the magnitude & phase at each frequency.
- Magnitude encodes how much signal there is at particular frequency.
- Phase encodes spatial information (indirectly)

Amplitude $A = \sqrt{R(\omega)^2 + I(\omega)^2}$

Phase $\phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$

1) 1-D continious FT

$$F(\omega) = \frac{1}{\sqrt{2\omega}} \int_{-\infty}^{\infty} f(x) e^{-j2\pi\omega x}$$

$$g(\omega x) = e^{-j2\pi\omega x}$$

2) 1D - DFT of length N

$$F(u) = \frac{1}{N} \sum_{n=0}^{N-1} f(n) e^{-\frac{j2\pi un}{N}}$$

3) Forward transform

$$y(u) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) a(u,n)$$

↳ basis

4) Inverse transform

$$x(n) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} y(u) b(u,n)$$

5) basis

$$a(u,n) = e^{-\frac{j2\pi un}{N}}$$

$$= \cos\left(2\pi \frac{un}{N}\right) - j \sin\left(2\pi \frac{un}{N}\right)$$

2D DFT

DFT

$$y(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) e^{-\frac{j2\pi um}{M}} e^{-\frac{j2\pi vn}{N}}$$

IDFT

$$x(m,n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} y(u,v) e^{\frac{j2\pi um}{M}} e^{\frac{j2\pi vn}{N}}$$

Use of FT

1) filtering in frequency domain
2) Combining image (hybrid image)
   ↓
   Gaussian + laplacian

Edge & Corner detection

1) Image sub-sampling

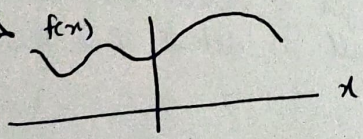
Throw away every other row & column to create a 1/2 size image.

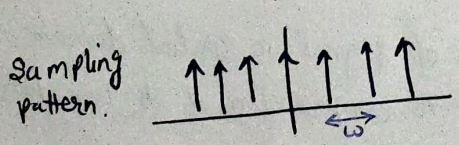
Sampling & Nyquist rate

- **Aliasing** can arise when you sample a continuous signal or image.
  - occurs when your sampling rate is not high enough to capture amount of details in your image.
  - Can give you wrong signal/image - an alias
  - formally image contains structure at diff. scales called frequencies in fourier domain.
  - Sampling rate must be high enough to capture the highest frequencies in image

To avoid aliasing
- Sampling rate $> 2 *$ max freq in image
- Min sampling rate is called Nyquist rate

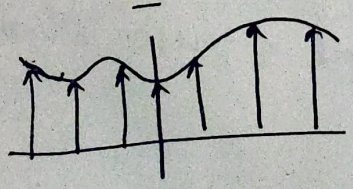
Sampling →





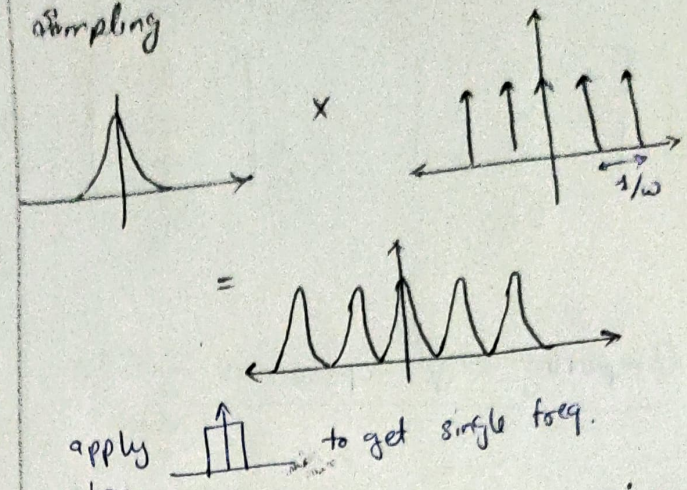
=

in spatial domain



in frequency domain

Sampling



apply box → to get single freq.

- Sinc function convolution in spatial domain is equivalent to box filter in freq domain.

Edge detection

Origin of edges

1. surface normal discontinuity
2. depth discontinuity
3. surface color discontinuity
4. illumination discontinuity.

We want edge operator that produces:-
- Edge magnitude
- Edge orientation
- High detection rate & good localization

Gradient

gradient eqn $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

- Represent direction of most rapid change in intensity.

eg  $\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

- Gradient direction $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$, edge strength $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

eg prewitt mask - Better approximation of gradient

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_x \qquad \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad S_y$$

Comparing edge operators

1) Gradient
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Good localization
Noise sensitive
Poor detection

2) Roberts (2×2)

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

3) Sobel (3×3)

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

4) Sobel (5×5)

$$\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -3 & 0 & 3 & 2 \\ -3 & -5 & 0 & 5 & 3 \\ -2 & -3 & 0 & 3 & 2 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 5 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -3 & -5 & -3 & -2 \\ -1 & -2 & -3 & -2 & -1 \end{bmatrix}$$

- Poor localization
- less noise sensitive
- Good detection

Effect of Noise

Noise is high freq component.
before edge detection remove noise
use smoothing filters like averaging filter, gaussian low pass filter.

- laplacian of gaussian can be approximated by difference betn 2 different gaussians.

$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{\frac{}{2\sigma^2}}$ Gaussian

$\frac{\partial}{\partial x} h_\sigma(u,v)$ Derivative of gaussian (0.5)

$\nabla^2 h_\sigma(u,v)$ Laplacian of gaussian

⇒ ∇^2 is laplacian operator

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

⇒ Canny Edge operator

1) Smooth image I with 2D gaussian : $G * I$

2) Find local edge normal directions for each pixel

$$\bar{n} = \frac{\nabla(G * I)}{|\nabla(G * I)|}$$

3) Compute edge magnitudes $|\nabla(G * I)|$

4) Locate edges by finding zero crossings along edge normal directions (non-maximum suppression)

$$\frac{\partial^2 (G * I)}{\partial \bar{n}^2} = 0$$

⇒ choice of σ depends on desired behavior
- Larger σ detects large scale edges
- Small σ detects fine features

Note

- expansion of signal in spatial domain is equivalent to compression of its transform in freq. domain.
- Difference of gaussian ≡ unsharp masking

Texture

1) What is texture?

- An image obeying some statistical property
- Similar structures repeated over & over again
- Often has some degree of randomness.

Aspects of texture

- Size / Granularity
- Directionality / orientation
- Random or regular

Texture Energy features

Use texture filters applied to the image to create filtered images from which texture features are computed.

Law's technique

→ Filter the input image using texture filters.
→ Compute texture energy by summing absolute value of filtering results in local neighborhoods around each pixel
→ Combine features to achive rotational invariance

Law's texture masks

L5 (Level) = [1 4 6 4 1]
E5 (Edge) = [-1 -2 0 2 1]
S5 (Spot) = [-1 0 2 0 -1]
R5 (Ripple) = [1 -4 6 -4 1]

(K5) (Gaussian) gives a center-weighted local average.
E5) (gradient) responds to row or col step edges
S5) (LOG) detects spot
R5) (Gabor) detects ripple

To create 2D masks multiply 2 1D masks

eg E5 × L5

$$\begin{bmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{bmatrix} \times [1\ 4\ 6\ 4\ 1]$$

- Using autocorrelation to detect texture

$$\rho(dr, dc) = \frac{\sum_{r=0}^{N}\sum_{c=0}^{N} I[r,c]\, I(r+dr, c+dc)}{\sum_{r=0}^{N}\sum_{c=0}^{N} I^2[r,c]}$$

$$= \frac{I[r,c] \circ I_d[r,c]}{I[r,c] \circ I[r,c]}$$

- Autocorrelation function computes the dot product (energy) of original image with shifted image for different shifts
- Can detect repetitive patterns of texels
- Captures fineness / coarseness of texture

Interpretation

- Regular texture → fun. will have peaks & valleys
- Random textures → only peak at [0,0], breadth of peak gives size of texture
- Coarse texture → function drops off slowly
- Fine texture → function drops off rapidly

⇒ Power spectrum of signal is fourier transform of autocorrelation function

Concentrated power → regularity
high freq. power → fine texture
Directionality → directional texture

Markov chains

- a sequence of random variables $x_1, x_2 \ldots x_n$
- x_t is state of model at time t
- Markov assumption: each state is dependent only on previous one
  - dependency given by conditional probability

$$P(x_t \mid x_{t-1})$$

- above is called first order Markov chain
- N^{th} order Markov chain:

$$P(x_t \mid x_{t-1}, \ldots, x_{t-N})$$

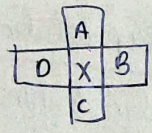
Markov Random field (MRF)

- generalization of markov chains to two or more dimensions

First order MRF:

Probability that pixel X takes a certain value given values of neighbours A, B, C & D

$$P(X \mid A, B, C, D)$$



⇒ neighbor pixels are highly correlated