Continental
The Future in Motion

Filtering 2D image processing

① Image can be thought of as a function (f). $f(x,y)$ gives the intensity at (x,y)

② Color image → three functions pasted together. Also called a vector valued function.

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

③ Filtering → can be defined as a process of transforming an image. Filtering can be used to either enhance the image (Eg. Contrasting) or extract the features from it (Eg. edge detection).
⇒ A filter defines a new image (G) in terms of existing image f.

Application of filtering:
Goal → improve image contents. ✓
→ extract information ✓

④ Image denoising → reducing noise in an image.
source of noise → CCD / cmos electronics.

Image filtering methods:
- Spatial domain
- Frequency domain (fourier transform)

$f(x,y) \rightarrow$ [Operation R] $\rightarrow g(x,y)$

$f(x,y) \rightarrow$ [Transform] $\xrightarrow{T(u,v)}$ [Operation R] $\xrightarrow{R[T(u,v)]}$ [inverse transform] $\rightarrow g(x,y)$

⑤ Spatial domain methods → For spatial domain of image filtering, we have two methods i.e. Point processing and area / mask processing.

Point processing method → works on a single pixel. The value of single pixel i.e. $f(x,y)$ is transformed (T) into an enhanced image
Eg. Negative, contrast streching, Histogram Equalization.
$$g(x,y) = T[f(x,y)]$$

Area / mask processing method → Works on a pixel and its neighbouring pixels.
$$g(x,y) = T[f(x,y)]$$ Need to define: Area shape & size operation.

⑥ Point processing method Examples → making negative of a picture
→ Contrast streching
→ Thresholding (changing pixel value - either 0 or 255)

Histogram equalization → changing → Histogram Equalization.
the histogram of the image so that it is spread equally. We have to find a function that will redistribute the image from limited spread to an extended / equalised spread.

Probability of Intensity $p(i) = \frac{n_i}{n}$ [no. of pixels with intensity i / no. of pixels (total)]

processing method → Need to define area shape and size and also the type of operation. for eg. rectangular mask of 3×3, 5×5.

size means that how much area (or pixels) around our pixel do we influence

Operation → by defining weight pixel value

simple averaging filter. (sharpening filter)
→ accentuates diff. with local average.

w(-1,-1)	w(-1,0)	w(-1,1)
w(0,-1)	w(0,0)	w(0,1)
w(1,-1)	w(1,0)	w(1,1)

} applying weighted to the whole image.

→ All these filters are considered to be linear filters i.e. shift invariant linear filters which means that the operator behaves in the same everywhere i.e. the value of the output depends upon the pattern in the image neighbourhood, not the position of the neighbourhood.

Linear → Eg. superposition → $h*(f1+f2) = (h*f1) + (h*f2)$
scaling → $h*(K,f) = K(h*f)$

Correlation filtering → lets say, we have a window size of $(2K+1) \times (2K+1)$, after generalisation, we have

$$G(i,j) = \sum_{u=-K}^{K} \sum_{v=-K}^{K} H(u,v)\, F[i+u, j+v]$$

H → mask, F → image. } Dot product

$G = H \otimes F$
cross-correlation.

idea → replace each pixel with a linear combination of its neighbours.

definition → cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to other. (for template matching)

$$(f \star g)(\tau) \triangleq \int_{-\infty}^{\infty} \overline{f(t)}\, g(t+\tau)\, dt$$ → continuous space

$\overline{f(t)}$ → complex conjugate of f(t)

$$(f \star g)[n] \triangleq \sum_{m=-\infty}^{\infty} \overline{f[m]}\, g[m+n]$$ → discrete space.

Convolution (for image filtering) →

$$G(i,j) = \sum_{u=-K}^{K} \sum_{v=-K}^{K} H(u,v)\, F[i-u, j-v]$$

$G = H * F$
convolution.

Convolution is same as correlation except, that the mask is flipped, both horizontally and vertically

$$g(i,j) = \sum_{k=-\frac{n_h}{2}}^{n_h} \sum_{l=-\frac{n_h}{2}}^{n_h} h(k,l)\, f(i-k, j-l) = \sum_{k=-\frac{n_h}{2}}^{n_h} \sum_{l=-\frac{n_h}{2}}^{n_h} h(i-k, j-l)\, f(k,l)$$

1	2	3
4	5	6
7	8	9

H → | 7 | 8 | 9 |
| 4 | 5 | 6 |
| 1 | 2 | 3 |

V → | 9 | 8 | 7 |
| 6 | 5 | 4 |
| 3 | 2 | 1 |

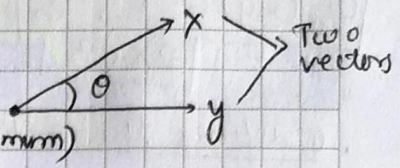
note → for symmetric masks, $(h(i,j) = h(-i,-j)$
correlation and convolution are similar.

In Convolution, we flip the matrix by 180°

Properties → linear and shift invariance
commutative $f*g = g*f$
Associative $(f*g)*h = f*(g*h)$
Identity
differentiation $\frac{\partial}{\partial x}(f*g) = \frac{\partial f}{\partial x} * g$.

→ **Correlation and Template matching** → measurement of similarity b/w 2 signals or sequences.

→ Geometric interpretation. $x \cdot y = |x||y| \cos(\theta)$



if x & y are perpendicular then $x \cdot y = 0$ (maximum)

$$(c)(\theta) = \frac{x \cdot y}{|x||y|}$$ → Cross correlation

Normalised Cross-Correlation (i.e. dividing by lengths)

$$g[t] = \frac{(h-\bar{h}) \cdot (f_t - \bar{f}_t)}{|h-\bar{h}| \cdot |f_t - \bar{f}_t|}$$

Zero-mean Normalised cross-correlation. (ZNCC)

→ Index
① Substract template average
② Substract patch average.

using the statistical term σ (standard deviation)

$$g[t] = \frac{(h-\bar{h}) \cdot (f_t - \bar{f}_t)}{n \cdot \sigma_h \cdot \sigma_{f_t}} \simeq g[t] = \frac{cov(h, f_t)}{\sigma_h \cdot \sigma_{f_t}}$$

Cross-Correlation of a template $h(x,y)$ with a sub-image $f(x,y)$ is:

n → no. of pixels in $h(x,y)$ & $f(x,y)$
$\bar{f}$ → average of f
σ_f → standard deviation of f

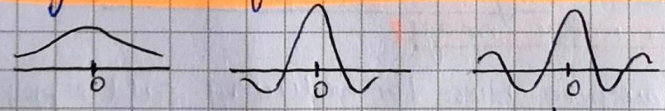
Disadvantage → Correlation is not rotation and scale invariant. i.e. the template must have the right scale and orientation in order to be detected in the image. Also not shape invariant.

⇒ **Convolution** → It can be defined as the mathematical operation on 2 functions to produce a 3rd function that expresses how the shape of one is modified by the other.

$$f(t) * g(t) \triangleq \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

→ According to applications, we choose the mask weights. It is done usually by sampling certain functions and their derivatives.

Eg.



Gaussian function — 1st derivative — 2nd derivative

→ Normalisation of weights → $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$ } Positive weights → sum = 1 negative weights

Smoothing → The basic idea of smoothing is to replace each pixel by average of its neighbours. useful for reducing noise and unimportant details. Bigger the mask, greater is the smoothing. When smoothing is done, blurring and loss of detail occurs.

→ When we apply box filter for smoothing, it creates edges in the smoothed image because of a rectangular function.

→ A solution is to use "fuzzy blob". i.e. the weight contribution of neighbourhood pixels is set according to their closeness to the center.

→ This is done by **Gaussian smoothing**. The idea behind gaussian smoothing is that each pixel is replaced by the weighted average of its neighbouring pixels. The mask weight is calculated by sampling a Gaussian function.

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- In the Gaussian mask, weight is highest at the centre and decreases with distance as we go away from the center.
- σ defines the extent of smoothing. Larger σ → more smoothing
  smaller σ → less smoothing.

→ **Properties of Gaussian** → convolution of 2 Gaussian functions leads to another gaussian function.

$$G_{\sigma_1}(x) * G_{\sigma_2}(x) = G_{\sqrt{\sigma_1^2+\sigma_2^2}}(x)$$

Kernels can be seperated → A 2D Gaussian can be ~~separated~~ expressed as the product of 2 1-D Gaussians

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}}\right)$$

called "separability of filters"

instead of taking the whole kernel,  , it is separated into  and  first horizontal operation, then vertical operation.

→ **Noise**
- Salt & pepper → random occurances of black/white pixels.
- Impulse → random occurances of white pixels
- Gaussian → variation of intensity drawn from Gaussian Normal distribution.

→ **Gaussian Noise** → Adding a noise process to the original image. It consists of several independent factors and is good for small standard deviations.

$$f(x,y) = \underset{\text{(image)}}{\bar{f}(x,y)} + \underset{\text{(noise process)}}{\eta(x,y)} \qquad \eta(x,y) \sim N(M, \sigma)$$

M → mean, σ → standard deviation.

If we perform smoothing operations with larger standard deviations, although it reduces the noise, it increases the blur in the image.

→ **Salt and pepper noise** → we cannot use gaussian blur for reducing salt & pepper noise because sep noise is not normally distributed. We can consider them as outliers. The problem is the pattern of the noise. Applying a wrong filtering technique would not solve it. It is done by median filtering.

→ **Median filtering** → median filtering is a non-linear operation used in image processing.

idea → use median of all the pixels in kernel area, instead of mean

5	10	2	20	5
6	9	2	12	14
8	7	20	19	18
2	1	10	10	9

image

Taking the median → [2, 2, 5, 6, **7**, 8, 9, 10, 20] replacing 9 with 7.

Advantage → Robust to outliers.

→ **Sharpening filters** → to make details more clear.

$$f + \alpha(f - f*g) = (1+\alpha)f - \alpha f*g = f*((1+\alpha)e - g)$$

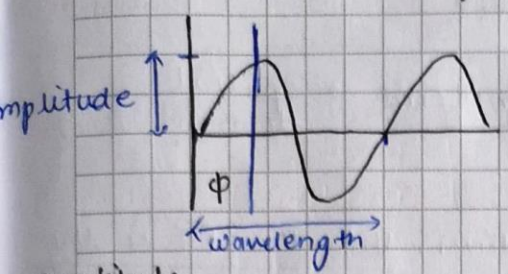
f: image, f*g: blurred image, e: unit impulse

→ Image sharpening → compute the intensity differences in local image regions. useful for emphasizing transitions in intensity.

High pass sharpening filters →
① useful for highlighting fine details.
② Elements of the mask contains both positive and negative weights.
③ Sum of mask elements is 0.

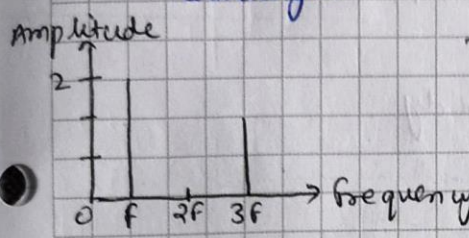
Filtering in frequency domain

① Based on fourier Transform whose basic idea is that, any periodic function can be written as the sum of sine and cosine functions.



$A \sin(\omega x + \phi)$ — ω: angular frequency $2\pi f$; ϕ: phase; A: Amplitude

we can add enough functions to get any signal $f(x)$.



Frequency Spectra

→ for 1D signals we will use the enhanced (Complex) version of sines and cosines

$$e^{ix} = \cos(x) + i\sin(x)$$
$$e^{i2\pi\nu t} = \cos(2\pi\nu t) + i\sin(2\pi\nu t)$$

$\nu \rightarrow$ frequency; Period $\rightarrow \frac{1}{\nu}$

→ The problem with fourier transform is that, the basis is complex, but the signal is real. The solution is to combine a pair of conjugate basis element.

$$B_{N-k}(n) = e^{\frac{2\pi i(N-k)n}{N}} = e^{2\pi i n - \frac{2\pi i k n}{N}} = e^{-\frac{2\pi i k n}{N}} = B_{-k}(n)$$

✓ The trick is to consider $B_{-N/2}$ to $B_{N/2}$ as the basis elements.

⇒ Real signals will have the same coefficients for B_k and B_{-k}.

$B_0 \rightarrow e^{2\pi i \, 0 \, n/N} = 1$ — coefficient at 0 acts as "Constant Bias" (gives the offset). Also called "DC component" "frequency-zero component".

⇒ 1D fourier transform → let $f(x)$ be a continuous function, FT is the function $F(u)$ given by

$$FT\{f(x)\} = F(u) = \int_{-\infty}^{+\infty} f(x)\, e^{-i2\pi ux}\, dx$$

Real Part / Imaginary Part

$$F(u).Re = \int_{-\infty}^{+\infty} f(x)\cos(-2\pi ux)\,dx \qquad F(u).Im = \int_{-\infty}^{+\infty} f(x)\sin(-2\pi ux)\,dx$$

→ Fourier Transform stores the magnitude & phase at each frequency.
Magnitude → how much signal is there at a particular frequency.
Phase → spatial information.

$$\text{Amplitude } A = \sqrt{R(w)^2 + I(w)^2}$$
$$\text{Phase } \Phi = \tan^{-1}\frac{I(w)}{R(w)}$$

Discrete Fourier Transform → The discrete FT for a discrete f(x) signal with N values is given by:-

$$F(u) = \frac{1}{N} \sum_{x=0 \dots N-1} f(x)\, e^{-i\pi 2ux/N}$$

Real Part

$$F(u) = \frac{1}{N} \sum_{x=0 \dots N-1} f(x) \cos(-2\pi ux/N)$$

Imaginary Part

$$F(u) = \frac{1}{N} \sum_{x=0 \dots N-1} f(x) \sin(-2\pi ux/N)$$

→ $$F(u) = \int_{-\infty}^{+\infty} f(x)\, e^{-i\pi 2ux}\, dx$$ ⇒ Fourier Transform → frequency

$$f(x) = \int_{-\infty}^{+\infty} F(u)\, e^{i\pi 2ux}\, du$$ ⇒ Inverse fourier Transform → spatial domain

2D fourier Transform → Let f(x,y) be a continuous function, then the FT is the function f(u,v) given by:

$$FT(f(x,y)) = F(u,v) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\, e^{-i\pi 2(ux+vy)}\, dx\, dy$$

$$F(u,v).re = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y) \cos(-2\pi(ux+vy))\, dx\, dy$$

$$F(u,v).im = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y) \sin(-2\pi(ux+vy))\, dx\, dy$$

→ **Removing high frequencies** → usually noise can be categorised as high frequency. So, if we remove high frequencies (putting them to 0) from the image in which we did the fourier transform, we get a new spectrum. And if we do an inverse fourier transform of it, we get a noiseless image (although we have blurred edges)

→ **Removing low frequencies** → If, in an image, in which we did the fourier transform, we remove the low frequencies, we get high intensity points, high contours. Just like we did (kind of) edge detection.

→ **Removing periodic noise** → fourier transform can remove periodic noise.

→ **FT of a gaussian** → If we have a gaussian centred at origin

$$f(r) = \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2} \quad \{r^2 = x^2+y^2\}$$

$$F(u,v) = F(p) = e^{-2\pi p^2\sigma^2} \quad \{p^2 = u^2+v^2\}$$

* FT of a gaussian is a gaussian

Convolution theorem → $F(g * h) = F[g]\,F[h]$

$g * h = F^{-1}[F[g]\,F[h]]$

⇒ This property of fourier transform is important because it makes it easier to perform multiplication operation than just performing convolution. (Eg: in linear filtering op)

Derivation of convolution theorem

$$I(u) = \int_{-\infty}^{+\infty}\left(\int_{-\infty}^{+\infty} f(\tau)g(t-\tau)\,d\tau\right) e^{-2\pi iut}\, dt$$

$$= \int_{-\infty}^{+\infty}\left(\int_{-\infty}^{+\infty} f(\tau)g(\delta)\,d\tau\right) e^{-2\pi iu(\delta+\tau)}\, d\delta$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(\tau)e^{-2\pi iu\tau}\, g(\delta)e^{-2\pi iu\delta}\, d\tau\, d\delta$$

$\delta \to t-\tau$ } Imp.

$$= \underbrace{\int_{-\infty}^{+\infty} f(\tau) e^{-2\pi i u \tau} d\tau}_{F(u)} \cdot \underbrace{\int_{-\infty}^{+\infty} g(\delta) e^{-2\pi i u \delta} d\delta}_{G(u)}$$

$$\boxed{H(u) = F(u)\, G(u)}$$

→ **Importance of Convolution theorem:** It establishes the link between the operation in the frequency domain and the action of linear spatial filters.

→ There are 2 ways to carry out linear spatial filtering operations :-

① Spatial domain → Convolution with a spatial operator
② frequency domain → ① calculate FT of image and FT of filter.
② multiply both the FTs.
③ calculate their inverse FT

→ We can choose from both of these domains because filter may be simpler to compute in one domain
→ computational cost

Dirac delta and its transform

$$\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases}$$

$$\int_{-\infty}^{+\infty} \delta(x)\, dx = 1 \qquad \int_{-\infty}^{+\infty} f(x)\, \delta(x)\, dx = f(0)$$

$$F(u) = 1$$

→ we can add many linear functions → property of FT.

can be used for image recognition (in texture images)

→ the spatial function $f(x,y)$ $\left[f(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(u,v)\, e^{2\pi i (ux+vy)}\, du\, dv \right]$

can be decomposed into a weighted sum of 2D orthogonal basis functions in a similar manner to decomposing a vector onto a basis using scalar products.

---

operties

Linearity → $\int_{-\infty}^{+\infty} (af(x) + bg(x))\, e^{-2\pi i u x} dx = a\int_{-\infty}^{+\infty} f(x) e^{-2\pi i u x} dx + b\int_{-\infty}^{+\infty} g(x) e^{-2\pi i u x} dx$

$= a F(u) + b G(u)$

Time shift → $\int_{-\infty}^{+\infty} f(x - x_0)\, e^{-2\pi i u x} dx \qquad (x - x_0) \Rightarrow \delta$

$= \int_{-\infty}^{+\infty} f(\delta)\, e^{-2\pi i u (\delta + x_0)} d\delta = e^{-2\pi i u x_0} \cdot \int_{-\infty}^{+\infty} f(\delta)\, e^{-2\pi i u \delta} d\delta$

$= e^{-2\pi i u x_0} \cdot F(u)$

derivative $\frac{\partial}{\partial x} f(x) = u F(u)$

Integration $\int f(x)\,dx = \frac{F(u)}{u}$

Convolution $f(x) \otimes g(x) = F(u)\,G(u)$

★ The log-polar transformation is also known as "Fourier-mellin transformation