- DFT for 1D: $F(k) = \sum_{x=0}^{N-1} f(x)\, e^{-j\frac{2\pi kx}{N}}$ where $k \in [0, N-1]$.
- inverse DFT for 1D: $f(x) = \frac{1}{N} \sum_{k=0}^{N-1} F(k)\, e^{j\frac{2\pi kx}{N}}$ where $k \in [0, N-1]$.
- ~~inverse DFT for 2D: $f(x,y) = \frac{1}{m \cdot N} \sum_{u=0}^{m-1} \sum_{v=0}^{N-1} F(u,v)\, e^{j2\pi(ux/m + vy/N)}$~~
- DFT for 2D: $F(u,v) = \sum_{x=0}^{m-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{xu}{m} + \frac{yv}{N}\right)}$
- inverse DFT for 2D: $f(u,v) = \frac{1}{m \cdot N} \sum_{u=0}^{m-1} \sum_{v=0}^{N-1} F(u,v)\, e^{-j2\pi\left(\frac{xu}{m} + \frac{yv}{N}\right)}$
- DFT for 1D kernel: $F(k) = \text{kernel} * f(x)$
- DFT for 2D kernel: $f(k,l) = \text{kernel} * f(x,y) * \text{kernel}^T$.

Q. Compute DFT of $f(x) = \{1, 0, 0, 1\}$ ← use x values not f(x) values. ($x = 0, 1, 2, 3$)

$$F[k] = \sum_{x=0}^{3} f(x)\, e^{-j\frac{2\pi kx}{N}} = f(0)e^{0} + f(1)e^{-j2\pi k/4} + f(2)e^{-j\pi k} + f(3)e^{-j\frac{3\pi k}{2}}$$

$$= 1 + 0 + 0 + e^{-j3\pi k/2} = 1 + e^{-j3\pi k/2}.$$

When $k=0$, $F[0] = 1 + e^{0} = 1+1 = 2$

When $k=1$, $F[1] = 1 + e^{-j\frac{3\pi}{2}} = 1+j$

When $k=2$, $F[2] = 1 + e^{-j3\pi} = 1-1 = 0$

When $k=3$, $F[3] = 1 + e^{-j\frac{9\pi}{2}} = 1-j$

$\therefore F(K) = 2, 1+j, 0, 1-j$ ← DFT.

- $e^{j\pi} = -1$, $e^{-j\pi} = -1$, $e^{j\pi/2} = j$, $e^{-j\pi/2} = -j$
- $e^{-j2\pi} = 1$, $e^{-j3\pi} = -1$, $e^{-j3\pi/2} = j$, $e^{-j9\pi/2} = -j$
- $e^{-j\theta} = \cos\theta - j\sin\theta$, $e^{j\theta} = \cos\theta + j\sin\theta$

Q. Compute IDFT of $F(x) = \{6, -2+2j, -2, -2-2j\}$. use for IDFT

$$f(n) = \frac{1}{4} \sum_{k=0}^{3} X(K)\, e^{j\frac{2\pi}{4}k \cdot n} = \frac{1}{4}\left[x[0] + x[1]e^{j\frac{2\pi}{4}\cdot n} + x[2]e^{j\frac{2\pi}{4}\cdot 2n} + x[3]e^{j\frac{2\pi}{4}\cdot 3n}\right]$$

When $n=0$: $f(0) = \frac{1}{4}(x[0] + x[1] + x[2] + x[3]) = \frac{1}{4}(6 + (-2+2j) + (-2) + (-2-2j)) = 0$

When $n=1$: $f(1) = \frac{1}{4}\left[x[0] + x[1]e^{j\frac{2\pi}{4}} + x[2]e^{j\frac{2\pi}{4}\cdot 2} + x[3]e^{j\frac{2\pi}{4}\cdot 3}\right]$.

$$= \frac{1}{4}\left[6 + (-2+2j)(j) + (-2)(-1) + (-2-2j)(-j)\right] = 1$$

When $n=2$: $f(2) = \frac{1}{4}\left[x[0] + x[1]\cdot e^{j\frac{2\pi}{4}\cdot 2} + x[2]e^{j\frac{2\pi}{4}\cdot 4} + x[3]e^{j\frac{2\pi}{4}\cdot 6}\right]$.

$$= \frac{1}{4}\left[6 + (-2+2j)(-1) + (-2)(1) + (-2-2j)(-1)\right] = 2.$$

When $n=3$: $f(3) = \frac{1}{4}\left[x[0] + x[1]e^{j\frac{2\pi}{4}\cdot 3} + x[2]e^{j\frac{2\pi}{4}\cdot 6} + x[3]e^{j\frac{2\pi}{4}\cdot 9}\right]$

$$= \frac{1}{4}\left[6 + (-2+2j)(-j) + (-2)(-1) + (-2-2j)(j)\right] = 3$$

Q. kernel of 4 points $\{0, 1, 2, 3\}$ using given matrix.

$$\text{kernel} * f(x) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ -2+3j \\ -2 \\ -2-2j \end{bmatrix}$$

for 1D kernel.

Q. Compute 2D DFT for given kernel

$F(k,l) = \text{kernel} * f(x,y) * (\text{kernel})^T$

* Spatial dom $\xrightarrow{DFT}$ Freq domain
* Freq dom $\xrightarrow{IDFT}$ Spatial dom
* DCT → used for img compression & separation.

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

DCT for 1D: $x[k] = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left\{\frac{(2n+1)\pi k}{2N}\right\}$ where $0 \le k \le N-1$

& $\alpha(k)$: $\alpha(u), \alpha(v) = \begin{cases} \sqrt{1/N} & \text{if } k=0 \\ \sqrt{2/N} & \text{if } k \neq 0 \end{cases}$

* DFT uses exponential function
* DCT uses real valued function.

↳ DCT for 2D: $\alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$

kernel of DCT = $\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.653 & 0.2705 & -0.2705 & -0.653 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2705 & -0.653 & 0.653 & -0.2705 \end{bmatrix}$

Haar Transform

kernels.

For $n=2$, $H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ For $n=4$, $H_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & \sqrt{2} \end{bmatrix}$

Q. Given $f(x,y) = \begin{bmatrix} 4 & -1 \\ 2 & 3 \end{bmatrix}$ & $H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 4 & -1 \\ 2 & 3 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ -1 & 3 \end{bmatrix}$

Haar Wavelet Transform.

forward transform: $a = (x+y)/2$, $d = (x-y)/2$.

inverse transform: $x = (a-d)/2$, $y = (a+d)/2$

Eg: $\begin{bmatrix} 100 & 50 & 60 & 15 \\ 20 & 60 & 40 & 30 \\ 50 & 90 & 70 & 82 \\ 74 & 66 & 90 & 58 \end{bmatrix}$ → $a_1 = (100+50)/2 = 75$, $d_1 = (100-50)/2 = 25$

$a_2 = (60+150)/2 = 105$, $d_2 = (60-150)/2 = -45$

After transform: $[75 \cdot 105 \quad 25 \quad -45]$

Similarly perform for all the 3 rows

↳ after transformation $\begin{bmatrix} 75 & 105 & 25 & -45 \\ 40 & 35 & -20 & 5 \\ 70 & 76 & -20 & -6 \\ 70 & 74 & 4 & 16 \end{bmatrix}$

next

for inverse transform, do column by column

$a_1 = (75+40)/2 = 57.5$, $d_1 = (75-40)/2 = 17.5$

$a_2 = (70+70)/2 = 70$, $d_2 = (70-70)/2 = 0$.

after transform: $\begin{bmatrix} 57.5 & 70 & 25 & -45 \\ 70 & 75 & -20 & 5 \\ 17.5 & 35 & -20 & -6 \\ 0 & 1 & 4 & 16 \end{bmatrix}$

Repeat for 2nd. columns:

Q. Compute 2D DFT for given kernel

$F(k,l) = \text{kernel} * f(x,y) * (\text{kernel})^T$

* Spatial dom $\xrightarrow{DFT}$ Freq domain
* Freq dom $\xrightarrow{IDFT}$ Spatial dom
* DCT → used for img compression & separation.

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

DCT for 1D: $x[k] = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left\{\frac{(2n+1)\pi k}{2N}\right\}$ where $0 \le k \le N-1$

& $\alpha(k)$: $\sqrt{1/N}$ if $k=0$; $\sqrt{2/N}$ if $k \neq 0$

$a(u), a(v)$

* DFT uses exponential function
* DCT uses real valued function.

↳ DCT for 2D: $a(u)\,a(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$

kernel of DCT =
$$\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.653 & 0.2705 & -0.2705 & -0.653 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2705 & -0.653 & 0.653 & -0.2705 \end{bmatrix}$$

Haar Transform

kernels.

For $n=2$, $H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

For $n=4$, $H_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & \sqrt{2} \end{bmatrix}$

Q. Given $f(x,y) = \begin{bmatrix} 4 & -1 \\ 2 & 3 \end{bmatrix}$ & $H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 4 & -1 \\ 2 & 3 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ -1 & 3 \end{bmatrix}$$

Haar Wavelet Transform:

forward transform: $a = (x+y)/2$, $d = (x-y)/2$.

inverse transform: $x = (a-d)/2$, $y = (a+d)/2$

Eg:
$$\begin{bmatrix} 100 & 50 & 60 & 15 \\ 20 & 60 & 40 & 30 \\ 50 & 90 & 70 & 82 \\ 74 & 66 & 90 & 58 \end{bmatrix}$$

→ $a_1 = (100+50)/2 = 75$, $d_1 = (100-50)/2 = 25$

$a_2 = (60+150)/2 = 105$, $d_2 = (60-150)/2 = -45$

After transform: [75 105 25 -45]

Similarly perform for all the 3 rows

↳ after transformation
$$\begin{bmatrix} 75 & 105 & 25 & -45 \\ 40 & 35 & -20 & 5 \\ 70 & 76 & -20 & -6 \\ 70 & 74 & 4 & 16 \end{bmatrix}$$

next

~~for inverse transform~~, do column by column

$a_1 = (75+40)/2 = 57.5$, $d_1 = (75-40)/2 = 17.5$

$a_2 = (70+70)/2 = 70$, $d_2 = (70-70)/2 = 0$.

after transform:
$$\begin{bmatrix} 57.5 & 70 & 25 & -45 \\ 70 & 35 & -20 & 5 \\ 17.5 & 35 & -20 & -6 \\ 0 & 1 & 4 & 16 \end{bmatrix}$$

Repeat for 2nd columns:

SMOOTHING FILTER

- **Ideal Low Pass:** $H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$

where $D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$

- **Butterworth Low pass:** $H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$

- defining edges & reduce ringing.
- if n increases, img sharper but ringing increases

- **Gaussian Low pass:** $H(u,v) = e^{-D^2(u,v)/2D_0^2}$

- removes low freq noise & ringing effect.

Numerical on Ideal Low Pass: Convert the given spatial domain using Fourier transform & perform Ideal Low pass.

A) I/p img ① multiply I/p img by $(-1)^{x+y}$ to center transform.

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

$(-1)^{0+0} = 1, (-1)^{0+1} = -1, (-1)^{0+2} = 1, (-1)^{0+3} = -1$

$(-1)^{1+0} = 1, (-1)^{1+1} = -1, (-1)^{1+2} = 1, (-1)^{1+3} = -1$

Similarly $(-1)^{2+0} = 1, (-1)^{2+1} = -1, (-1)^{2+2} = 1, (-1)^{2+3} = -1, (-1)^{3+0} = -1, (-1)^{3+1} = 1, (-1)^{3+2} = -1, (-1)^{3+3} = 1$

$$\therefore \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 \end{bmatrix}.$$

② Compute DFT

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = F(u,v).$$

③ Multiply $F(u,v)$ with the filter $H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$

Compute $D(u,v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2} = \sqrt{\left(u - \frac{4}{2}\right)^2 + \left(v - \frac{4}{2}\right)^2} = \sqrt{(u-2)^2 + (v-2)^2}$

(Calculating distance from centre (2,2) where M=4, N=4).

Distance of each value of (u,v) from (2,2)

$(0,0) = \sqrt{(0-2)^2 + (0-2)^2} = \sqrt{8} = 4.24$. Similarly. $(1,0) = 2.23, (1,1) = 1.41, (1,2) = 1$.

$(0,1) = \sqrt{(0-2)^2 + (1-2)^2} = \sqrt{5} = 2.23$ $(1,3) = 1.41, (2,0) = 2, (2,1) = 1, (2,2) = 0$,

$(0,2) = \sqrt{(0-2)^2 + (2-2)^2} = 2$ $(2,3) = 1, (3,0) = 2.23, (3,1) = 1.41$,

$(0,3) = \sqrt{(0-2)^2 + (3-2)^2} = \sqrt{5} = 2.23$. $(3,2) = 1, (3,3) = 1.41$.

$$\therefore D(u,v) = \begin{bmatrix} 4.24 & 2.23 & 2 & 2.23 \\ 2.23 & 1.41 & 1 & 1.41 \\ 2 & 1 & 0 & 1 \\ 2.23 & 1.41 & 1 & 1.41 \end{bmatrix}$$

Given $\therefore H(u,v) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ Given $D_0 = 0.5$.

$$G(u,v) = F(u,v) \times H(u,v) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

④ Compute the inverse DFT of: kernel for IDFT = $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$

$$\therefore g(u,v) = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 8 & 8 & 8 & 8 \\ 8 & -8 & -8 & -8 \\ 8 & 8 & 8 & 8 \\ -8 & -8 & -8 & -8 \end{bmatrix}$$

⑤ Multiply with $(-1)^{x+y}$.

$$g(x,y) = \frac{8}{16}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \end{bmatrix}\begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

SHARPENING FILTER

- Ideal high pass filter: $H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \le D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$
  - Disadvtg – blurred edges
- Butterworth high pass filter: $H(u,v) = \frac{1}{1+[D_0/D(u,v)]^{2n}}$
  - Transition to higher freq. is much smoother than ideal HPS.
- Gaussian High Pass Filter: $H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$
  - Results are much smoother & finer than IHPS & BHPS.
- Laplacian filter: $\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$

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

$$\therefore \nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y).$$

0	1	0
1	-4	1
0	1	0

$$\text{or } g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \leftarrow \text{centre of mask is -ve} \\ f(x,y) + \nabla^2 f(x,y) & \leftarrow \text{centre of mask is +ve} \end{cases}$$

- High boost filter: $A \ge 1$.

If $A = 1 \rightarrow$ standard Laplacian sharpening.

0	-1	0
-1	A+4	-1
0	-1	0

- Unsharp masking: $f_s(x,y) = f(x,y) - \bar{f}(x,y)$.

(enhances edges).

	Spatial Domain	Freq Domain
① Computation Cost	Low	High.
② Quality	High control.	Low control.
③ Time complexity	~~Loo~~ Less.	More
④ Capacity	~~Less~~ High	Low
⑤ Robustness	Fragile.	more robust
⑥ Complexity	Low	High.

NOISE MODELS

- **Gaussian Noise** (grayscaling) $p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\bar{z})^2/2\sigma^2}$ ↙ $\sigma^2 = \frac{\Sigma(x-\bar{x})^2}{N}$
- **impulse Noise / Salt & pepper Noise** $p(z) = \begin{cases} P_a \text{ for } z=a \\ P_b \text{ for } z=b \\ 0 \text{ otherwise} \end{cases}$ > gray levels.
  $a=0$ (black), $b=255$ (white)
- **poisson Noise** $p(z) = \frac{(np)^z e^{-np}}{z!}$ n = total no. of pixels, p = noise pixel / total pixel noise.
- **exponential Noise** (illumination apps). $p(z) = \begin{cases} a e^{-az} & \text{for } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$ | mean = $1/a$, Variance = $1/a^2$.
- **Erlang / gamma noise** (illumination applications). $p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{otherwise.} \end{cases}$ | mean = b/a, var = b/a^2
- **rayleigh Noise** (remote sensing applications) $p(z) = \begin{cases} \frac{2}{b}(z-a) e^{\frac{-(z-a)^2}{b}} & \text{for } z \geq a \\ 0 & \text{otherwise} \end{cases}$ | mean = $a + \sqrt{\frac{\pi b}{4}}$, var = $b\frac{(4-\pi)}{4}$
- **Uniform Noise** (quantization). (random no. generator). $p(z) = \begin{cases} 1/(b-a) & \text{for } a \leq f(x,y) \leq b \\ 0 & \text{otherwise} \end{cases}$ | mean = $(a+b)/2$, Var = $(b-a)^2/12$

IMAGE RESTORATION IN PRESENCE OF NOISE

MEAN FILTERS:

- **Arithmetic Mean:** $f'(x,y) = \frac{1}{mn} \Sigma g(s,t)$.
- **Geometric Mean:** $f'(x,y) = \left[\prod g(s,t)\right]^{\frac{1}{mn}}$ (better smoothing but loss of details → blurring). π = multiplication.
- **Harmonic Mean:** $f'(x,y) = \frac{mn}{\Sigma \frac{1}{g(s,t)}}$ ($Q=0$) (works for salt noise but not pepper noise).
- **Contra harmonic Mean:** $f'(x,y) = \frac{\Sigma g(s,t)^{Q+1}}{\Sigma g(s,t)^{Q}}$ ($Q=-1$) (both salt & pepper noise)

$Q>0$ → elimination of pepper noise, $Q<0$ → elimination of salt noise.

ORDER STATISICC FILTERS: (pepper, salt)

- ~~Median Filter~~ • Max / Min filter • Midpoint filter $= \frac{1}{2}[\max - \min]$
- **Alpha trimmed Mean Filter:** $f(x,y) = \frac{1}{mn-d} \Sigma g(s,t)$

124	126	123
120	(150)	125
115	119	123

Neighbourhood of (x,y) = 124, 126, 123, 120, 150, 125, 115, 119, 123
Sorted values: ~~115~~, 119, 120, 123, 124, 125, 126, ~~123~~, ~~150~~.
Remaining value: 120, 123, 124, 125, 126. eg d=4

$f'(x,y) = \frac{1}{3\cdot3-4}(120+123+124+125+126) = \frac{618}{5} \approx 124$

(removes both salt + pepper noise).

ADAPTIVE MEDIAN FILTERING: (remove salt pepper noise, provide smoothing)

a) if σ^2 is 0, filter should return $g(x,y)$

b) if local variance is high relative to σ^2, filter returns a value close to $g(x,y)$.

c) if 2 variances are equal, filter returns arithmetic mean of S_{xy}.

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_n^2}{\sigma_L^2}\left[g(x,y) - m_L\right]$$

Level A: $A_1 = Z_{med} - Z_{min}$, $A_2 = Z_{med} - Z_{max}$.

if $A_1 > 0$ AND $A_2 < 0$, go to level B else increase window size.

if window size $\leq$ repeat S_{max} level A. else output Z_{med}.
$3\times3 \rightarrow 4\times4$.

Level B: $B_1 = Z_{xy} - Z_{min}$, $B_2 = Z_{xy} - Z_{max}$

if $B_1 > 0$ AND $B_2 < 0$, output Z_{xy} else output Z_{med}.

BAND PASS FILTERS

- Ideal band pass filter: $H(u,v) = \begin{cases} 0 & \text{if } D(u,v) < D_0 - \frac{W}{2} \\ 1 & \text{if } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ 0 & \text{if } D(u,v) > D_0 + \frac{W}{2} \end{cases}$

- Butterworth band pass filter: $H(u,v) = \dfrac{\left[\frac{D(u,v)\,W}{D^2(u,v) - D_0^2}\right]^{2n}}{1 + \left[\frac{D(u,v)\,W}{D^2(u,v) - D_0^2}\right]^{2n}}$

- Gaussian band pass filter:
$H(u,v) = e^{-\frac{1}{2}\left[\frac{D^2(u,v) - D_0^2}{D(u,v)\,W}\right]^2}$

BAND REJECT FILTERS

- Ideal band reject: $H(u,v) = \begin{cases} 1 & \text{if } D(u,v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ 1 & \text{if } D(u,v) > D_0 + \frac{W}{2} \end{cases}$

- Butterworth band reject: $H(u,v) = \dfrac{1}{1 + \left[\frac{D(u,v)\,W}{D^2(u,v) - D_0^2}\right]^{2n}}$

- Gaussian band reject: $H(u,v) = 1 + e^{-\frac{1}{2}\left[\frac{D^2(u,v) - D_0^2}{D(u,v)\,W}\right]^2}$

INVERSE FILTERING

Estimated original img $F'(u,v) = \dfrac{\text{degraded img } G(u,v)}{\text{degraded func } H(u,v)}$

$G(u,v) = F(u,v)\,H(u,v) + N(u,v)$ ← Unknown Noise

$\therefore F'(u,v) = F(u,v) + \dfrac{N(u,v)}{H(u,v)}$] if $H(u,v) = 0$, $F'(u,v) =$ infinity.

so $H'(u,v) = \begin{cases} 1/H(u,v) & \text{if } H > 0 \\ 1 & \text{if } H < 0 \end{cases}$

[WIENER FILTERING] $MSE = E\{f(x,y) - f'(x,y)\}^2$

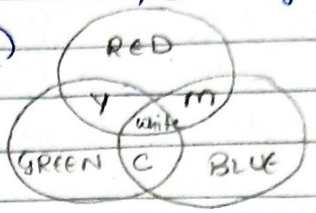
where MSE = Mean Square Error, E = expectation operation.

$$F'(u,v) = \frac{1}{H(u,v)}\left[\frac{|H(u,v)|^2}{|H(u,v)|^2 + k}\right] g(u,v) \quad \text{where } k = \frac{S_n(u,v)}{S_f(u,v)}$$

where $S_f(u,v)$ = Power spectrum of original img, $S_n(u,v)$ = Power spectrum of noise

[COLOR MODELS] — 0 represents darker shades, 255 → brighter shad

- RGB (Red Green Blue) [100, 010, 001]. digital camera
- CMY (Cyan Magenta Yellow) ← color printer
- HSI (Hue Saturation Intensity) human eye (decouples colors & grayscales them).



RGB model → addition, CMY model - subtraction

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \quad \& \quad \begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Q. RGB value are (35, 98, 156). Convert into CMY

Normalization: $R' = 35/255 = 0.137$, $G' = 98/255 = 0.384$, $B' = 156/255 = 0.6$

C.yan $= 1 - R = 1 - 0.137 = 0.863$, $M = 1 - G' = 1 - 0.384 = 0.616$, $Y = 1 - 0.611 = 0.389$.

In HSI model:

- Hue → color in the form of an angle b/w 0 to 360°
- Saturation → how much the color is diluted with white light. It varies b/w 0 & 1
- Intensity → 0 → black, 1 → white.

Q. Convert RGB(29, 104, 215) to HSI values.

$R' = 29/255 = 0.113$, $G' = 104/225 = 0.407$, $B' = 215/255 = 0.843$.

Intensity $I = \frac{1}{3}(R+G+B) = \frac{1}{3}(0.113 + 0.407 + 0.843) = 0.454$.

$$\text{Saturation } S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)]$$

$$= 1 - \frac{3}{(0.113+0.407+0.843)} \min\{(0.113, 0.407, 0.843)\} = 0.752.$$

Hue $H = \begin{cases} \theta & \text{if } B < G \\ 360 - \theta & \text{if } B > G. \end{cases}$ → In given problem $B > G$ (after normalization) so $H = 360 - \theta$.

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

$$= \cos^{-1}\left\{\frac{\frac{1}{2}\{(0.113-0.407)+(0.113-0.843)\}}{\sqrt{(0.113-0.407)^2+(0.113-0.407)(0.407-0.843)}}\right\} = 0'.$$

$H = 360 - \theta'$.

In RGB model, R(1,0,0), G(0,1,0), B(0,0,1), Black (0,0,0), W(1,1,1)
Yellow = R+G = (1,1,0), Cyan = G+B = (0,1,1), Magenta = R+B = (1,0,1)

Pixel depth: no. of bits used to represent each pixel.
for RGB, d = 8+8+8 bits = 24 bits, total colors = 2^{24}.

In CMY model, C(1,0,0), M(0,1,0), Y = (0,0,1), W(0,0,0), Black (1,1,1),
R(0,1,1), Blue (1,1,0), G (0,0,1).

8. Converting HSI to RGB

① RG sector ($0 \le H \le 120$)

$B = I(1-S)$, $R = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$, $G = 3I - (R+B)$

② GB Sector ($120^\circ \le H \le 240^\circ$)

$H = H - 120^\circ$, $R = I(1-S)$, $G = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$, $B = 3I - (R+G)$

③ BR sector ($240^\circ \le H \le 360^\circ$).

$H = H - 240^\circ$, $G = I(1-S)$, $B = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$, $R = 3I - (G+B)$

COLOUR IMG SMOOTHING $C(x,y) = \begin{bmatrix} 1/n \sum R(x,y) \\ 1/n \sum G(x,y) \\ 1/n \sum B(x,y) \end{bmatrix}$.

COLOR IMG SHARPENING $\nabla^2[C(x,y)] = \begin{bmatrix} \nabla^2 R(x,y) \\ \nabla^2 G(x,y) \\ \nabla^2 B(x,y) \end{bmatrix}$.

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

~~Post~~

Inverse filtering

- Simple & fast approach.
- provide blurring by simple linear system
- sensitive to noise
- amplifies high freq. noise

(VS)

Wiener Filtering

- advanced approach to remove noise & enhance imgs.
- designed to minimize MSE b/w original & estimated img.
- require prior knowledge abt. noise & img statistics
- computationally expensive.
- not suitable for real time img processing.