30.08.2019

Digital Image Processing (CSE/ECE 478)

Lecture-10: Image Enhancement in Frequency Domain – FFT, Filtering in Frequency Domain

Ravi Kiran

Center for Visual Information Technology (CVIT), IIIT Hyderabad

Discrete Fourier Transform (DFT) – 1D

DFT

$$F[u] = \frac{1}{M}\sum_{x=0}^{M-1} f[x]e^{-j2\pi ux/M}$$

Inverse DFT

$$f[x] = \sum_{x=0}^{M-1} F[u]e^{j2\pi ux/M}$$

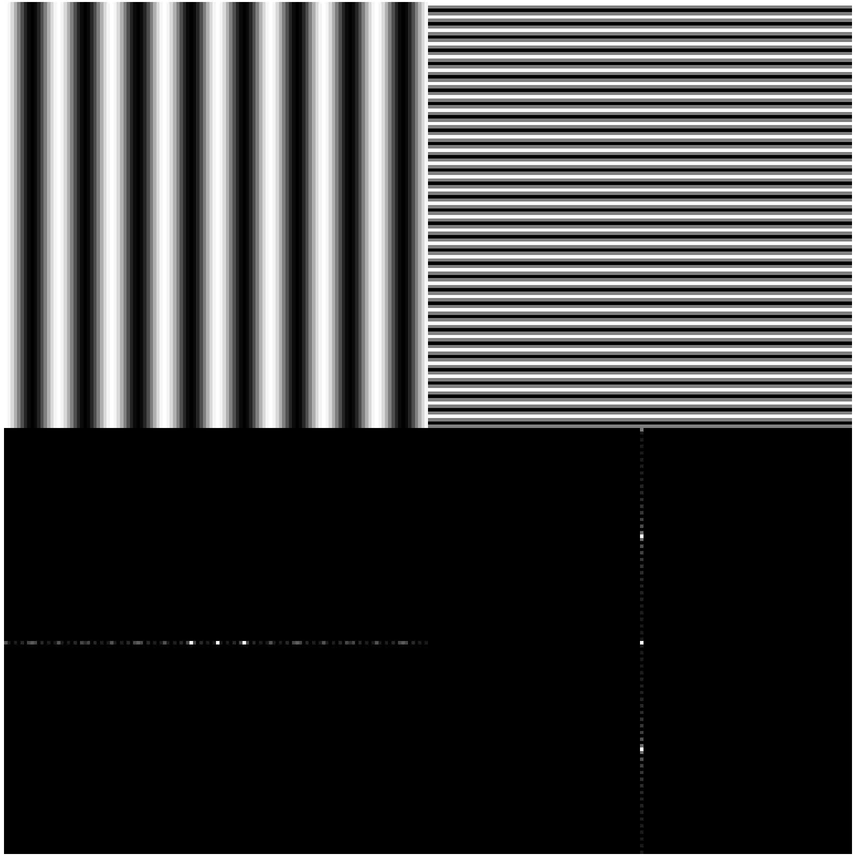
2D DFT

$$F(u,v) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} f(x,y)\, e^{-2\pi j \left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

$$f(x,y) = \frac{1}{MN} \sum_{v=0}^{N-1} \sum_{u=0}^{M-1} F(u,v)\, e^{2\pi j \left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

https://www.cs.unm.edu/~brayer/vision/fourier.html

DFT for simple ‘spatial’ patterns



Discrete Fourier Transform

1-D, Discrete Case:

Fourier Transform: $$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{-j2\pi ux/M} \qquad u = 0,\ldots,\text{M-1}$$

Inverse Fourier Transform: $$f(x) = \sum_{u=0}^{M-1} F(u)e^{j2\pi ux/M} \qquad x = 0,\ldots,\text{M-1}$$

F(*u*) can be written as:

$$F(u) = R(u) + jI(u)$$

Polar coordinate:

$$F(u) = |F(u)|e^{-j\phi(u)}$$

where

$$|F(u)| = \sqrt{R(u)^2 + I(u)^2}$$

magnitude

$$\phi(u) = \tan^{-1}\left(\frac{I(u)}{R(u)}\right)$$

phase

Magnitude and Phase Spectra

Figure 4a

Original

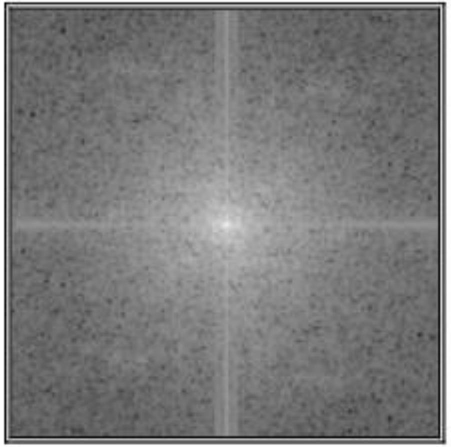


Figure 4b

$\log(|A(\Omega,\Psi)|)$

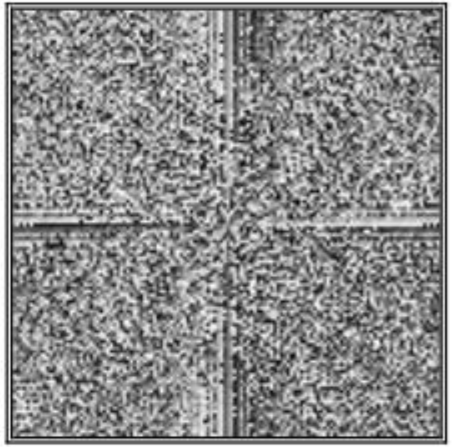


Figure 4c

$\varphi(\Omega,\Psi)$

https://i.imgur.com/YidmGcQ.png

Separability of 2D DFT

$$F(u,v) = \sum^{N-1} \sum^{M-1} f(x,y)\, e^{-2\pi j\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

$$= \sum_{y=0}^{N-1} e^{-2\pi j\left(\frac{vy}{N}\right)} \underbrace{\sum_{x=0}^{M-1} e^{-2\pi j\left(\frac{ux}{M}\right)} f(x,y)}$$

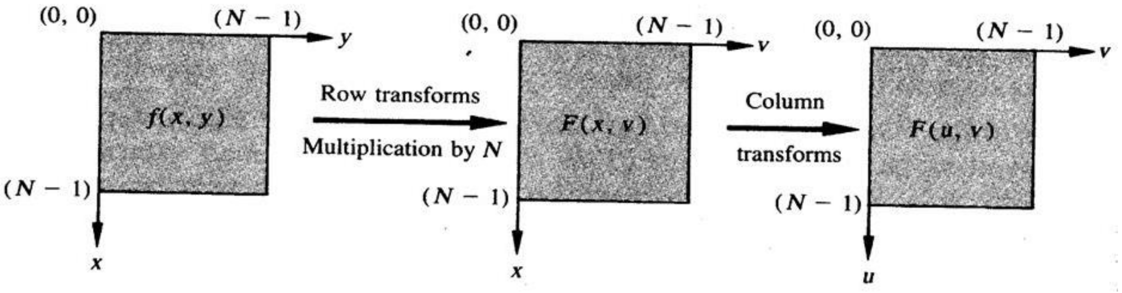
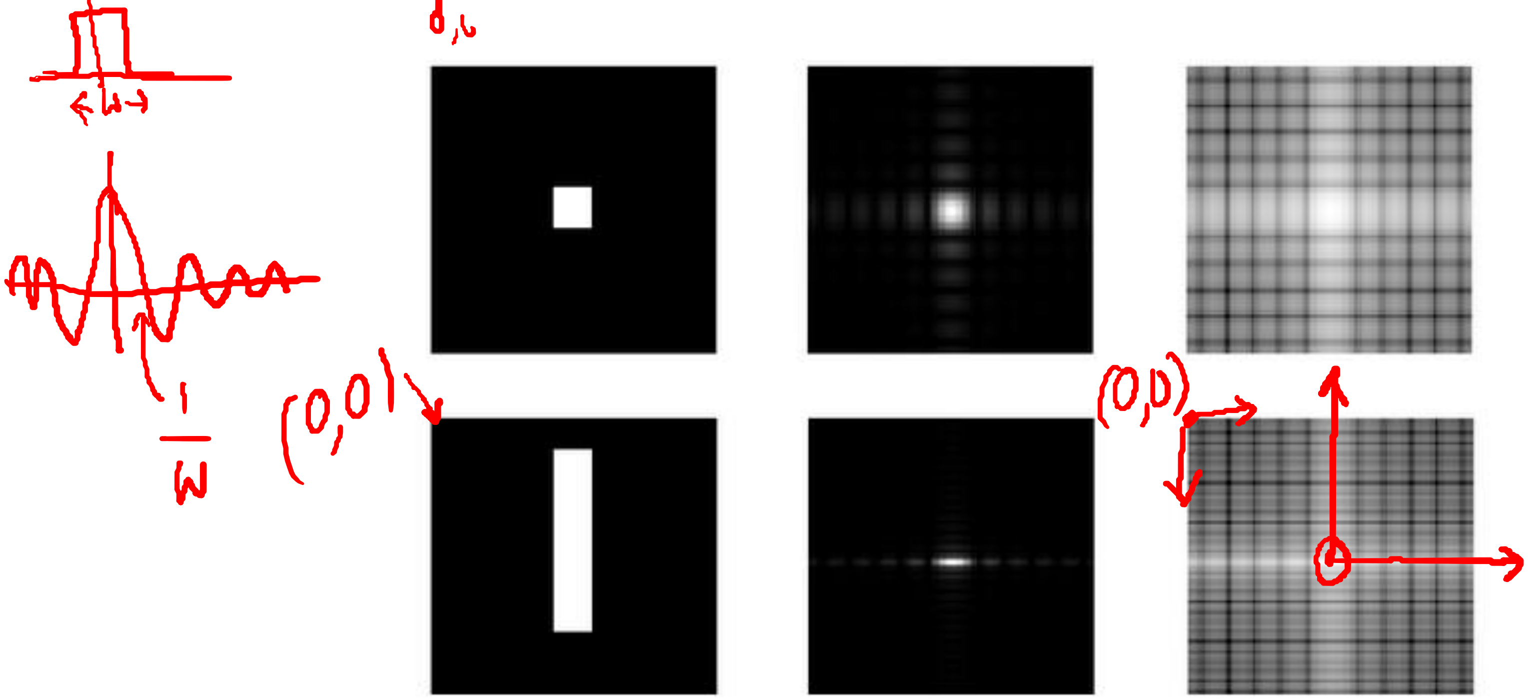


Figure 3.7 *Computation of the 2-D Fourier transform as a* $\boxed{\textit{series of 1-D transforms.}}$

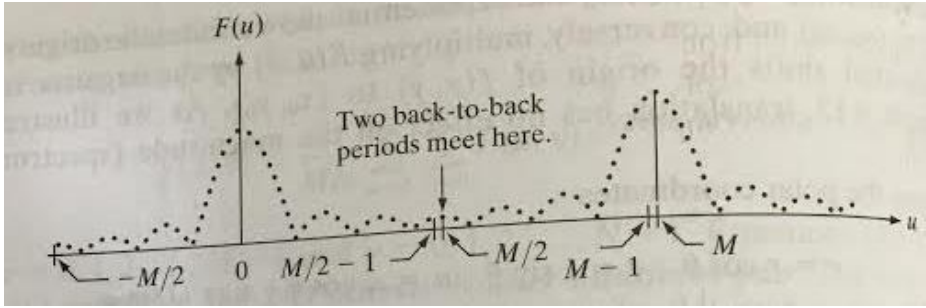
https://slideplayer.com/slide/5375929/

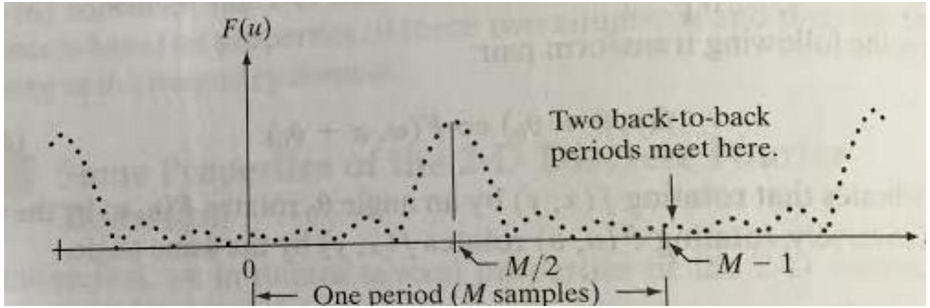
Centered FT Magnitude Images



Shifting origin

1-D

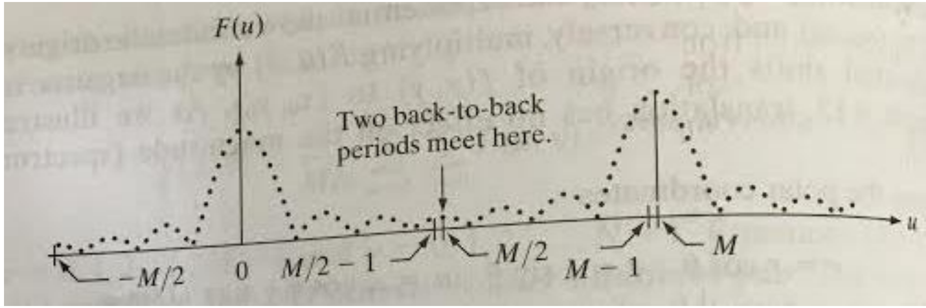


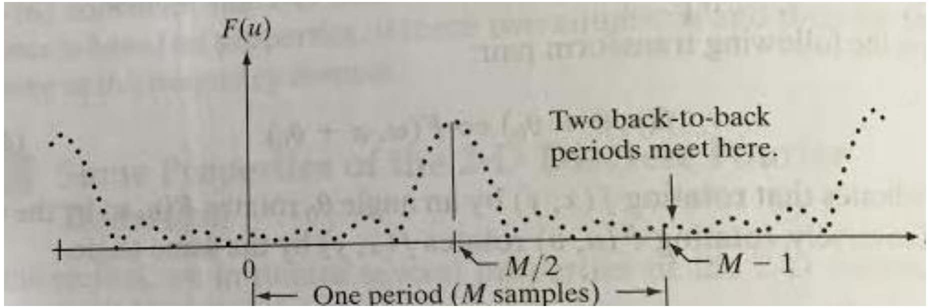


$$f[x]e^{\frac{j2\pi u_o x}{M}} \leftrightarrow F(u - u_o)$$

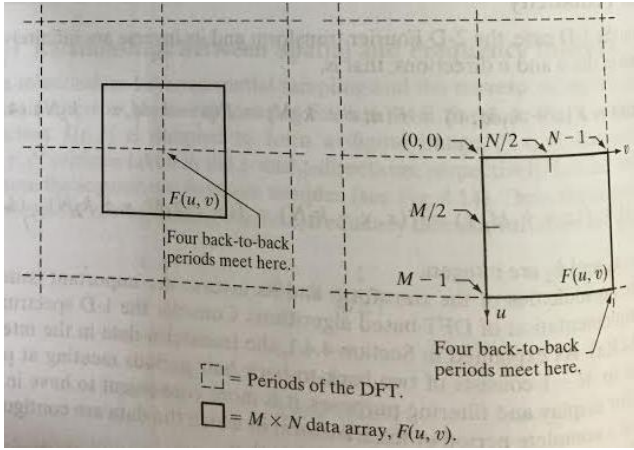
Shifting origin

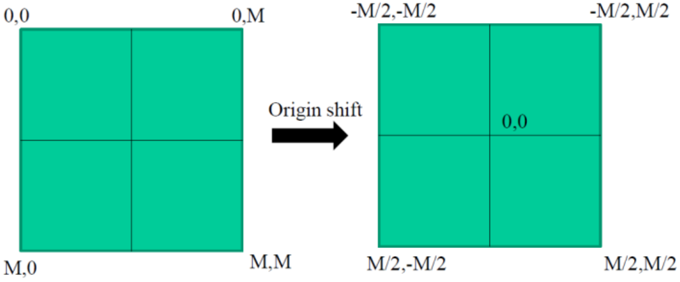
1-D





2-D

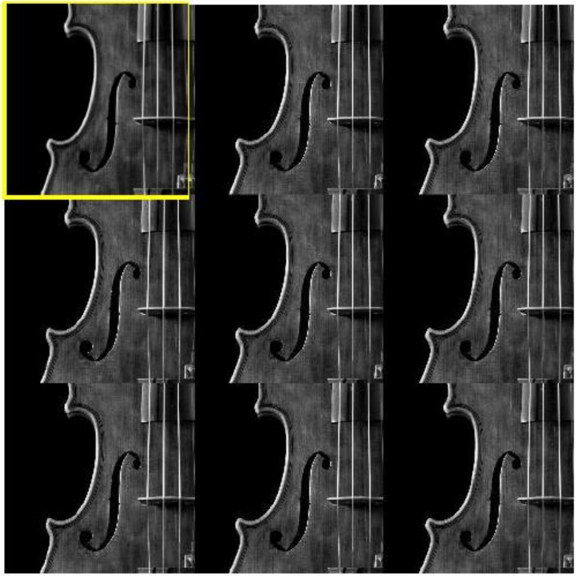


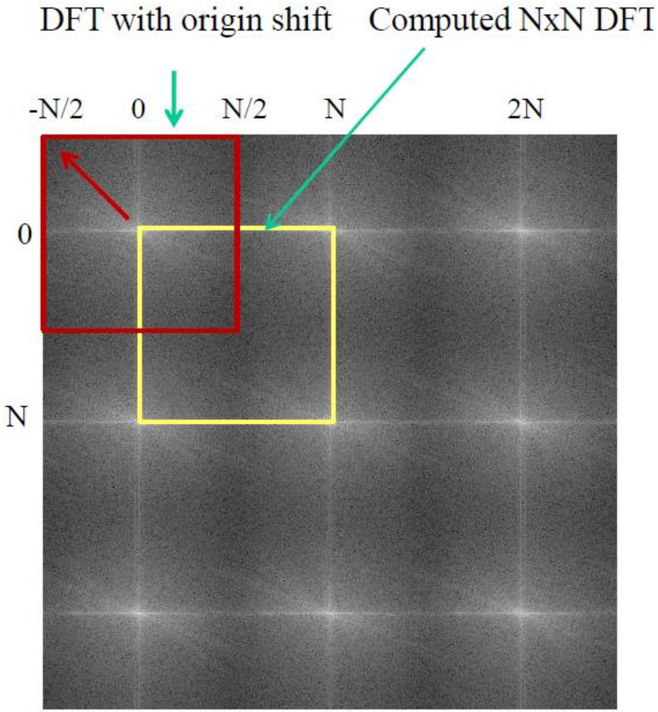


$$f[x,y]e^{j2\pi\left(\frac{u_O x}{M}+\frac{v_O y}{M}\right)} \leftrightarrow F(u-u_o, v-v_0)$$

$f[x,y](-1)^{x+y}$

Shifting origin





Complexity

DFT

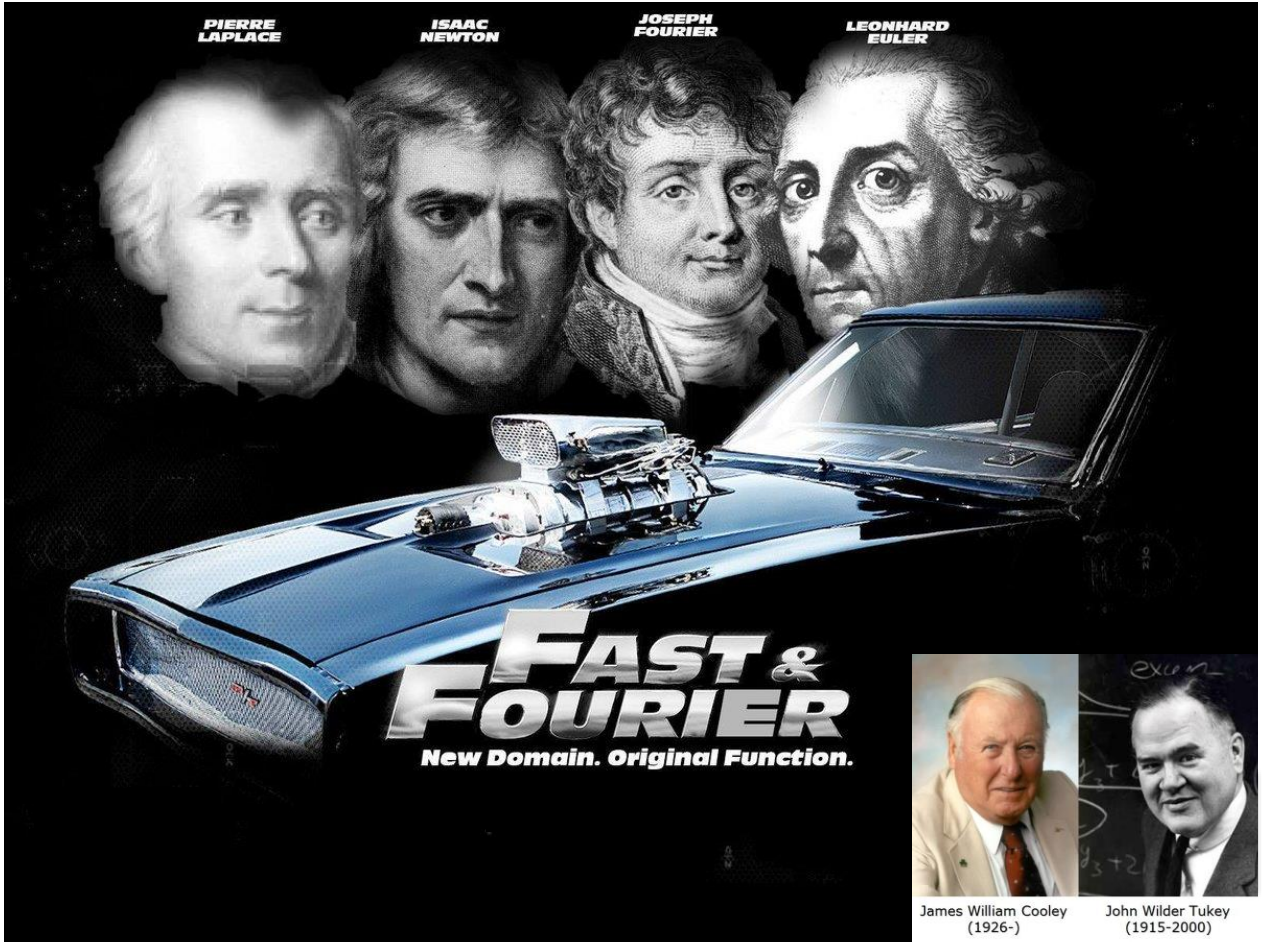
$$F[u] = \frac{1}{M}\sum_{x=0}^{M-1} f[x]e^{-j2\pi ux/M}$$

$O(M)$

$u = 0, 1, 2 \cdots (M-1)$

Inverse DFT

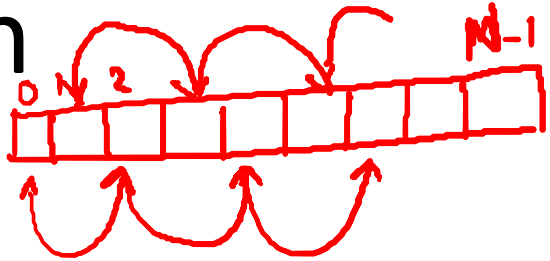
$$f[x] = \sum_{x=0}^{M-1} F[u]e^{j2\pi ux/M}$$



James William Cooley (1926-)

John Wilder Tukey (1915-2000)

FFT : Motivation



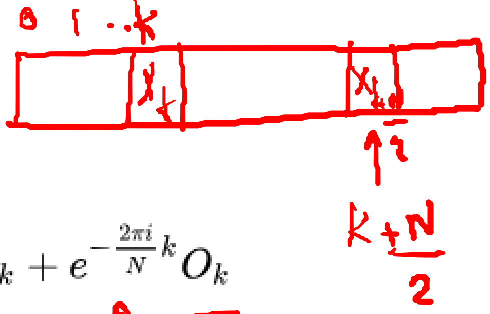
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk}$$

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k}$$

$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of even-indexed part of } x_n} + e^{-\frac{2\pi i}{N} k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of odd-indexed part of } x_n} = E_k + e^{-\frac{2\pi i}{N} k} O_k$$

https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm

FFT : Motivation



$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2}mk}}_{\text{DFT of even-indexed part of } x_n} + e^{-\frac{2\pi i}{N}k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2}mk}}_{\text{DFT of odd-indexed part of } x_n} = E_k + e^{-\frac{2\pi i}{N}k} O_k$$

$$X_{k+\frac{N}{2}} =$$

https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm

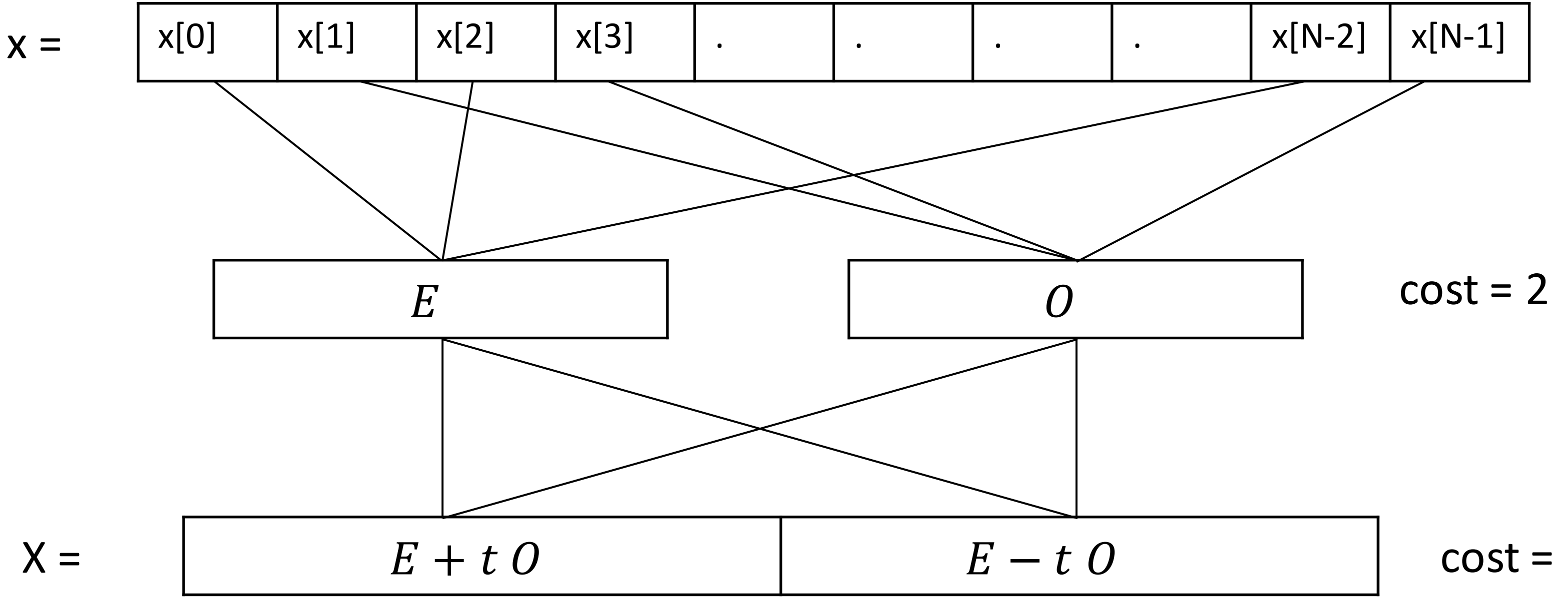
Fast Fourier Transform

FFT(n, [$a_0, a_1, \ldots, a_{n-1}$]):
  if n=1: return a_0
  F_{even} = FFT(n/2, [$a_0, a_2, \ldots, a_{n-2}$]) ← E_k
  F_{odd} = FFT(n/2, [$a_1, a_3, \ldots, a_{n-1}$]) ← O_k
  for k = 0 to n/2 – 1:
    $\omega^k = e^{2\pi ik/n}$
    $y^k = F_{even\ k} + \omega^k F_{odd\ k}$
    $y^{k+n/2} = F_{even\ k} - \omega^k F_{odd\ k}$
  return [$y_0, y_1, \ldots, y_{n-1}$]

$$O(N) = 2\,O\left(\frac{N}{2}\right) + O(N)$$

$$X_k = E_k + e^{-\frac{2\pi i}{N}k} O_k$$

$$X_{k+\frac{N}{2}} = E_k - e^{-\frac{2\pi i}{N}k} O_k$$



cost = 2 $T_{N/2}$

cost = 2 N

DFT vs FFT computation times

n	$N = 2^n$	N^2	$N \log N$
10	1 024	1 048 576	10 240
12	4 096	16 777 216	49 152
14	16 384	268 435 456	229 376
16	65 536	4 294 967 296	1 048 576

- Fourier Transform → Projection onto a linear orthonormal basis of complex sinusoids

$$x = \sum_k (x, e_k) e_k$$

$$F[u] = \frac{1}{M} \sum_{x=0}^{M-1} f[x] e^{-j2\pi ux/M}$$

DFT matrix

$$x = \sum_k (x, e_k) e_k$$

$$F[u] = \frac{1}{M} \sum_{x=0}^{M-1} f[x] e^{-j2\pi ux/M}$$

$$X = Wx\,;$$

The transformation matrix W can be defined as $W = \left(\frac{\omega^{jk}}{\sqrt{N}}\right)_{j,k=0,\ldots,N-1}$, or equivalently

$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}, \begin{bmatrix} x[0] \\ x[1] \\ \vdots \end{bmatrix}$$

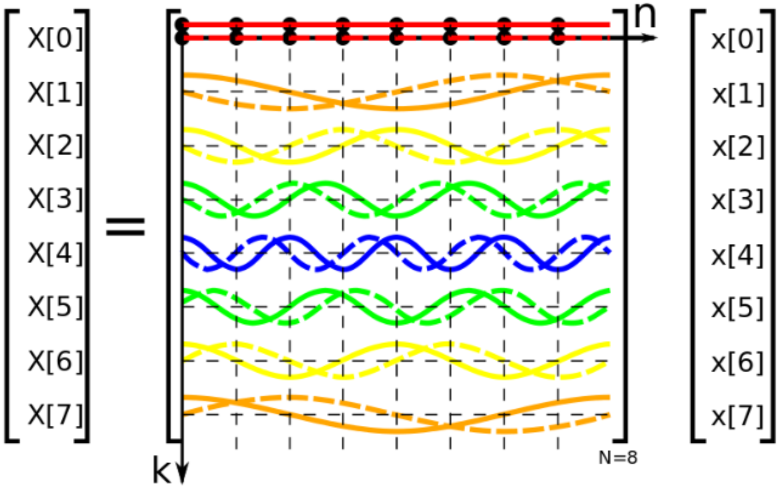
$\omega = e^{-2\pi i/N}$ is a primitive Nth root of unity

https://en.wikipedia.org/wiki/DFT_matrix

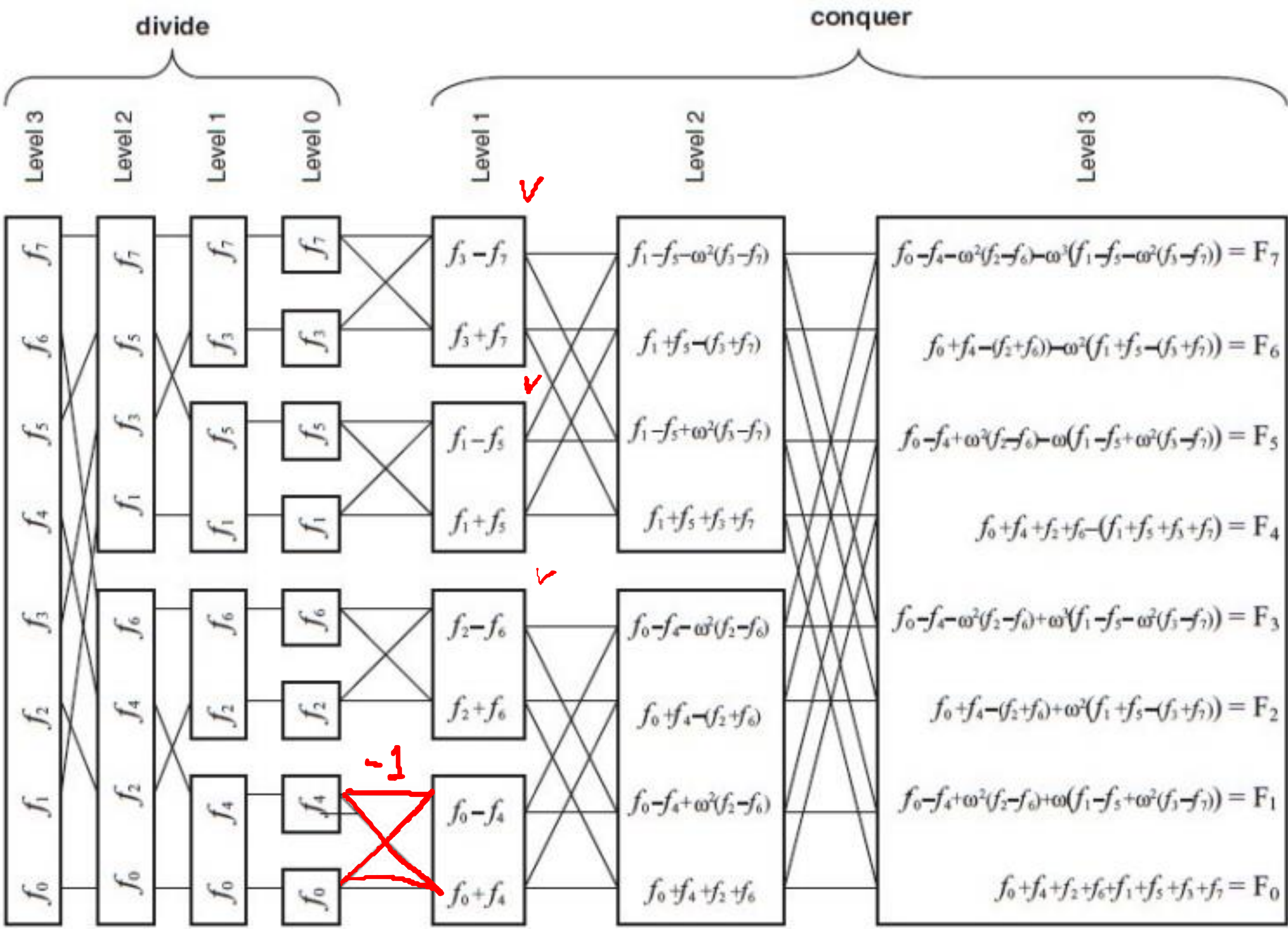
$$W = \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \omega^{10} & \omega^{12} & \omega^{14} \\ \omega^0 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} & \omega^{15} & \omega^{18} & \omega^{21} \\ \omega^0 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} & \omega^{20} & \omega^{24} & \omega^{28} \\ \omega^0 & \omega^5 & \omega^{10} & \omega^{15} & \omega^{20} & \omega^{25} & \omega^{30} & \omega^{35} \\ \omega^0 & \omega^6 & \omega^{12} & \omega^{18} & \omega^{24} & \omega^{30} & \omega^{36} & \omega^{42} \\ \omega^0 & \omega^7 & \omega^{14} & \omega^{21} & \omega^{28} & \omega^{35} & \omega^{42} & \omega^{49} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & -i & -i\omega & -1 & -\omega & i & i\omega \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & -i\omega & i & \omega & -1 & i\omega & -i & -\omega \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\omega & -i & i\omega & -1 & \omega & i & -i\omega \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & i\omega & i & -\omega & -1 & -i\omega & -i & \omega \end{bmatrix}$$

where

$$\omega = e^{-\frac{2\pi i}{8}} = \frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}}$$



```
FFT(n, [a_0, a_1, …, a_{n-1}]):
   if n=1: return a_0
   F_even = FFT(n/2, [a_0, a_2, …, a_{n-2}])
   F_odd  = FFT(n/2, [a_1, a_3, …, a_{n-1}])
   for k = 0 to n/2 – 1:
      ω^k = e^{2πik/n}
      y^k     = F_even k + ω^k F_odd k
      y^{k+n/2} = F_even k – ω^k F_odd k
   return [y_0, y_1, …, y_{n-1}]
```

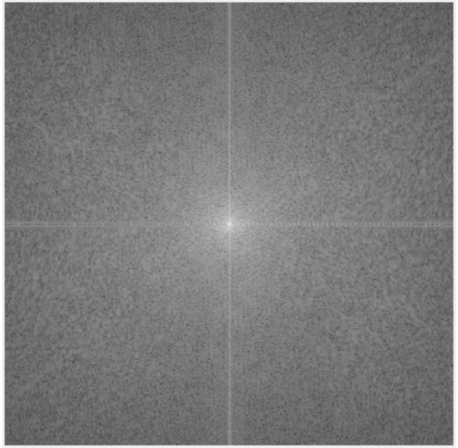


Correspondence to spatial filtering

```
f = rgb2gray(imread('boy.jpg'));
```

-1	0	1
-2	0	2
-1	0	1

```
h = [-1 0 1; -2 0 2; -1 0 1];
```



```
F = fft2(double(f), 402, 402);
```

```
H = fft2(double(h), 402, 402);
```

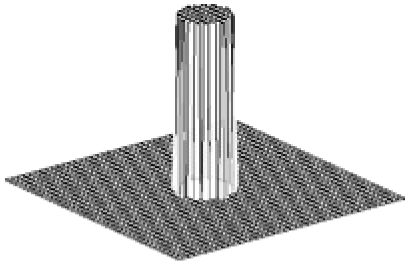
```
F_fH = fftshift(H).*fftshift(F);
ffi = ifft2(ifftshift(F_fH));
```

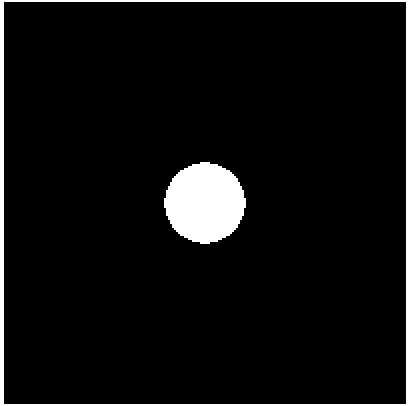
Correspondence to spatial filtering

```
%Sobel filter in frequency domain
f = rgb2gray(imread('boy.jpg'));
h = [-1 0 1; -2 0 2; -1 0 1];
F = fft2(double(f), 402, 402);
H = fft2(double(h), 402, 402);
F_fH = fftshift(H).*fftshift(F);
ffi = ifft2(ifftshift(F_fH));
```

Ideal Low Pass Filters

$$H(u,v) = \begin{cases} 1 \text{ if } D(u,v) \le D_0 \\ 0 \text{ if } D(u,v) > D_0 \end{cases}$$





where $D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$

$D_0 \rightarrow$ cut off frequency

Image courtesy: Gonzalez and Woods

Ideal Low Pass Filters

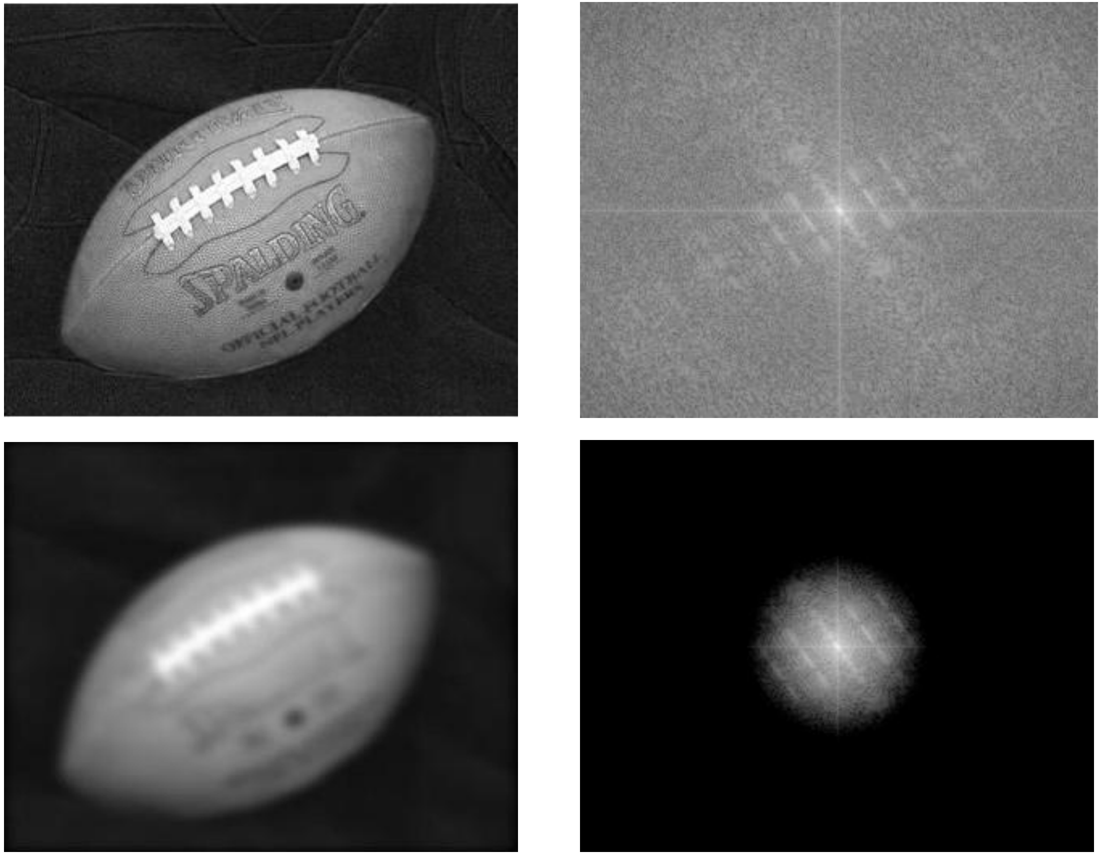
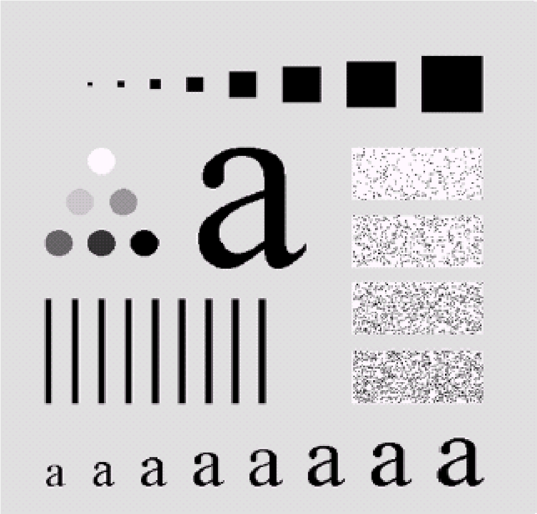
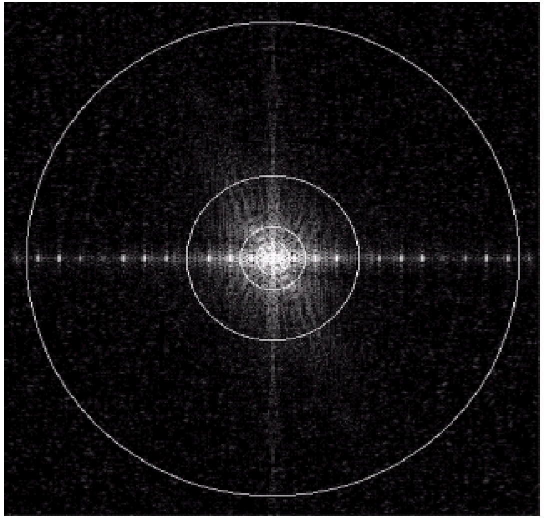


Image courtesy: cs.uregina.ca

Ideal Low Pass Filters

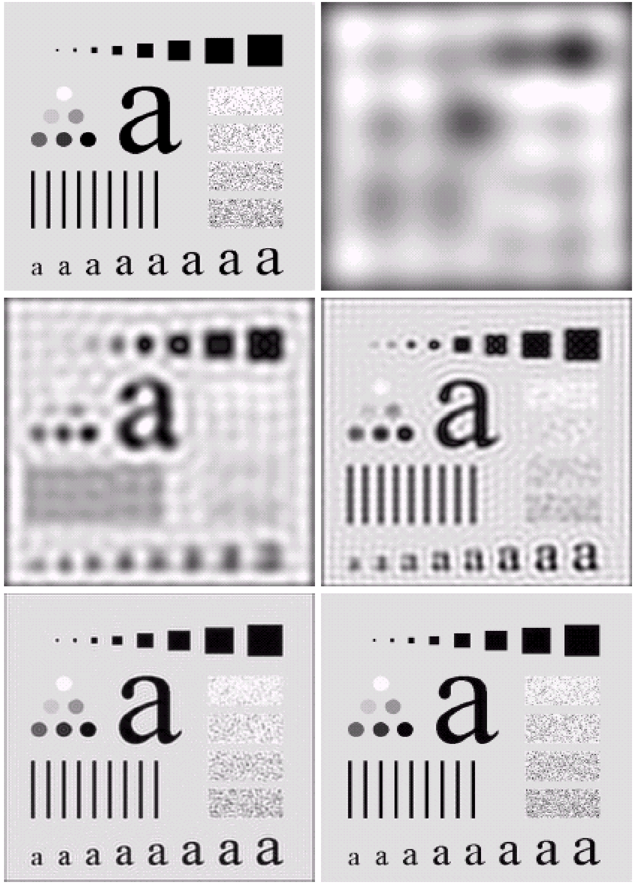




Radii 10,30,60,160 and 460 → power 87, 93.1, 95.7, 97.8 and 99..2

Image courtesy: Gonzalez and Woods

Ideal Low Pass Filters



ILPF radius 10

ILPF radius 30

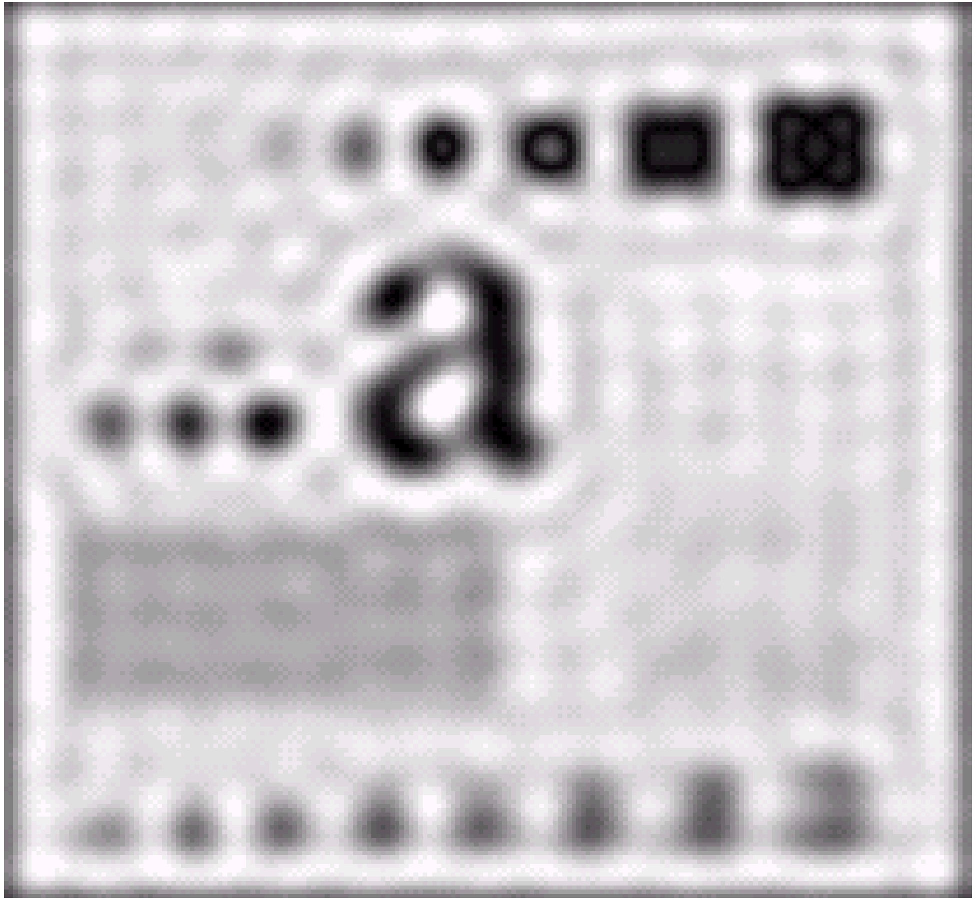
ILPF radius 60

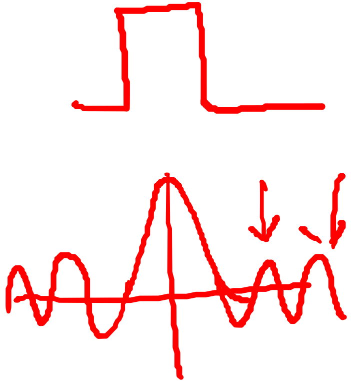
ILPF radius 160

ILPF radius 460

Image courtesy: Gonzalez and Woods

Ideal Low Pass Filters

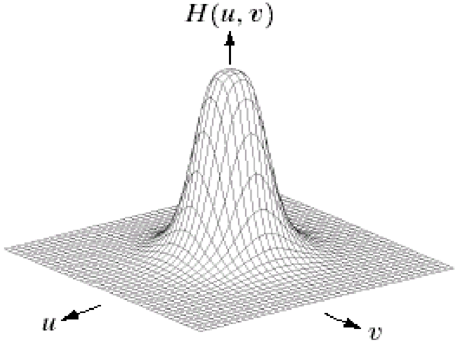


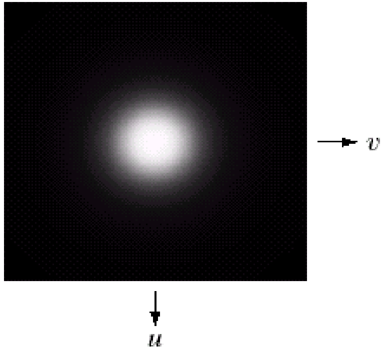


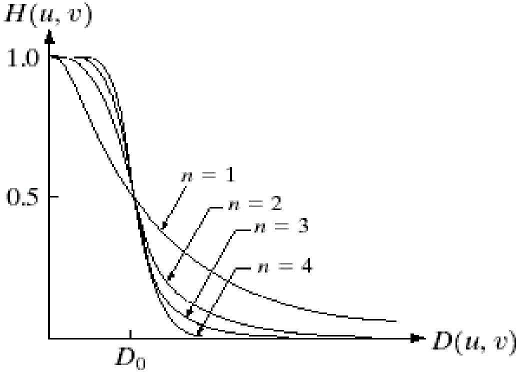
ILPF radius 30

Image courtesy: Gonzalez and Woods

Butterworth Low Pass Filters







$$H(u,v) = \frac{1}{1+[D(u,v)/D_0]^{2n}}$$ where $$D(u,v) = [(u-M/2)^2 + (v-N/2)^2]^{1/2}$$

Image courtesy: Gonzalez and Woods

Butterworth Low Pass Filters (BLPF)

Order two, i.e.
n=2

BLPF cut off frequency 10

BLPF cut off frequency 30

BLPF cut off frequency 60

BLPF cut off frequency 160

BLPF cut off frequency 460

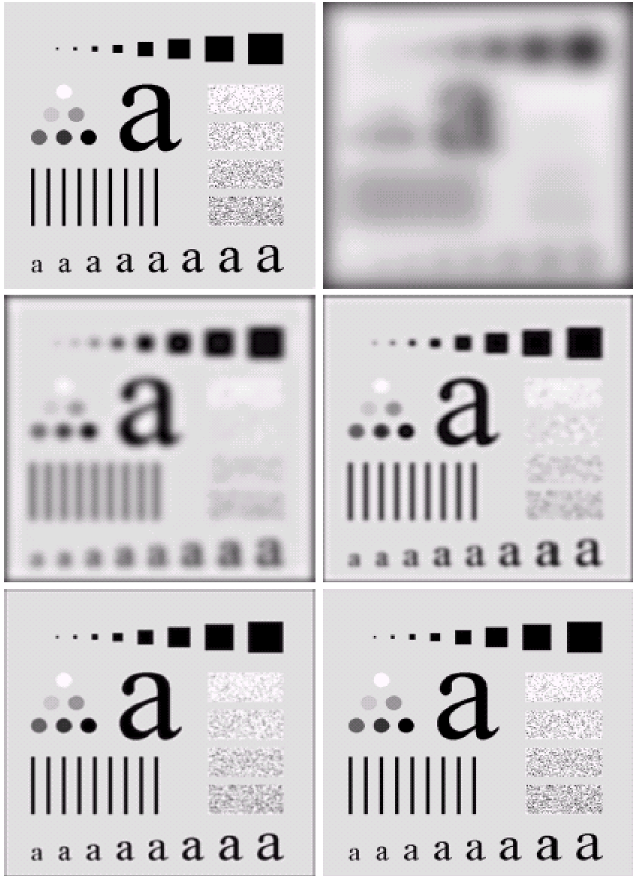
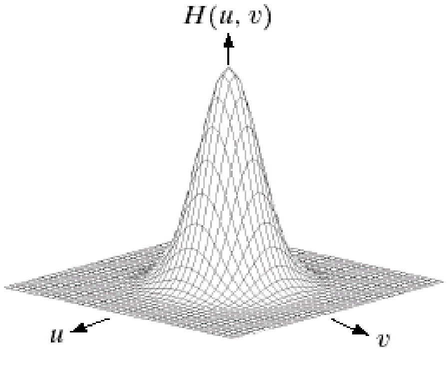
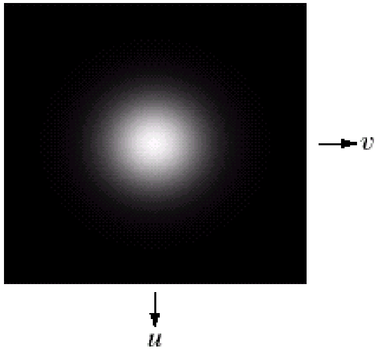
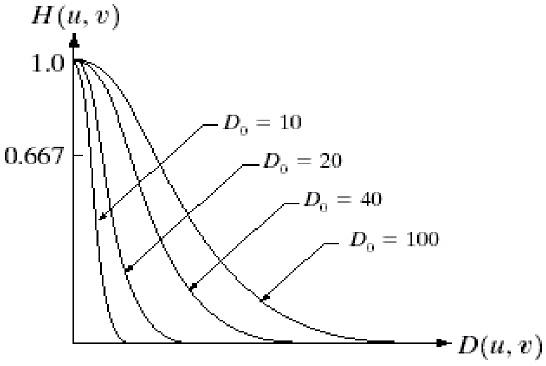


Image courtesy: Gonzalez and Woods

Gaussian Low Pass Filters



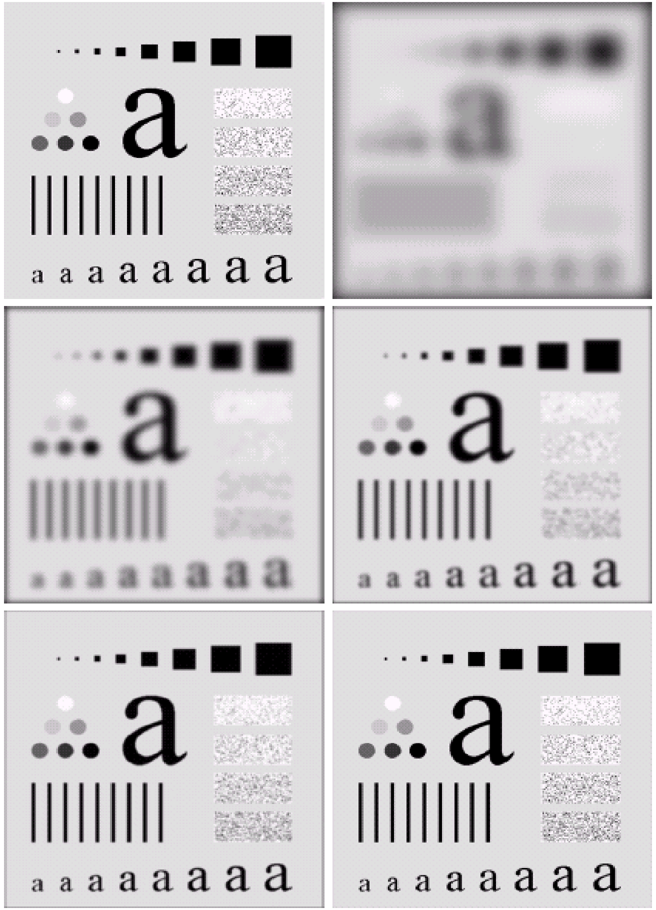




$$H(u,v)=e^{-D^2(u,v)/2D_0^{\,2}}$$

Image courtesy: Gonzalez and Woods

Gaussian Low Pass Filters (GLPF)



GLPF cut off frequency 10

GLPF cut off frequency 30

GLPF cut off frequency 60

GLPF cut off frequency 160

GLPF cut off frequency 460

Image courtesy: Gonzalez and Woods

Comparison (ILPF, BLPF, GLPF)

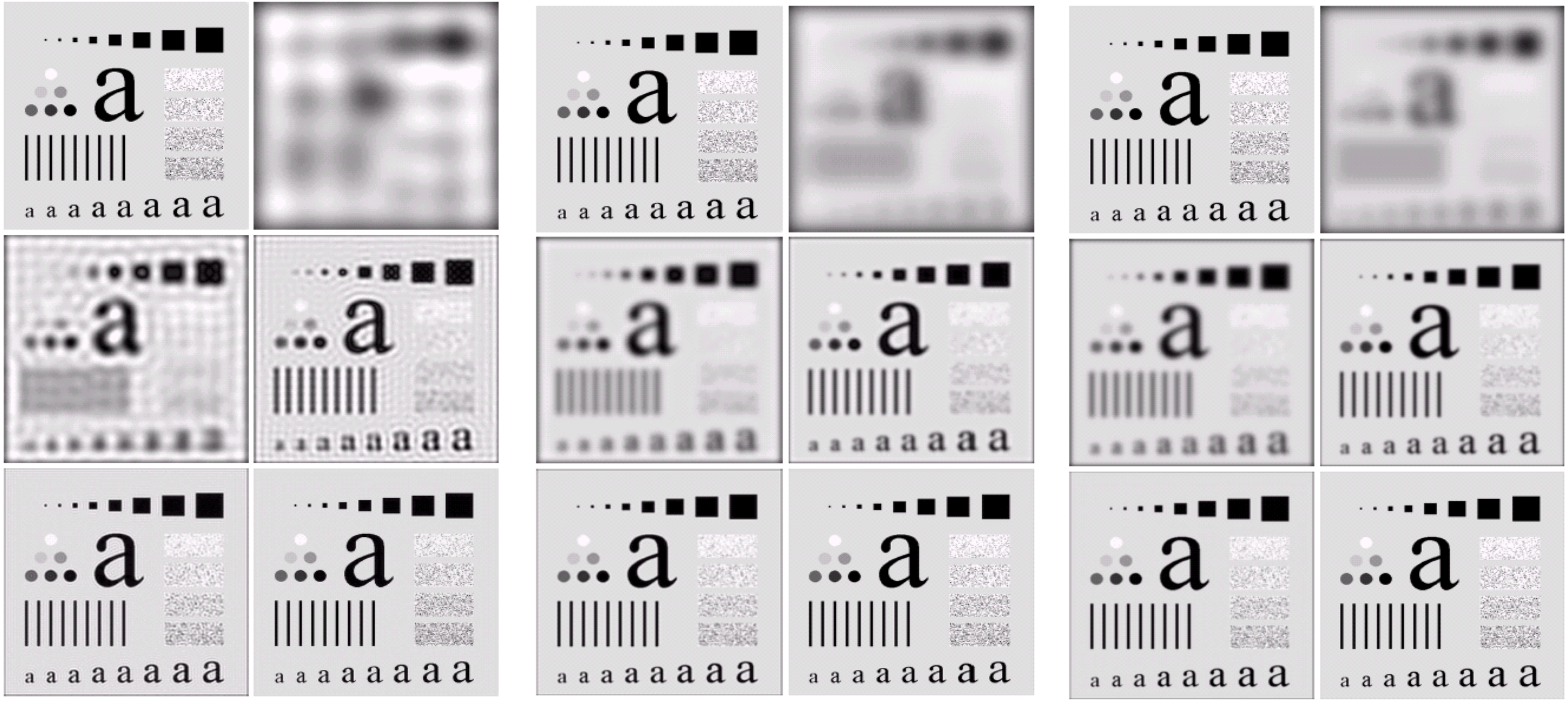


Image courtesy: Gonzalez and Woods

Low pass filtering application

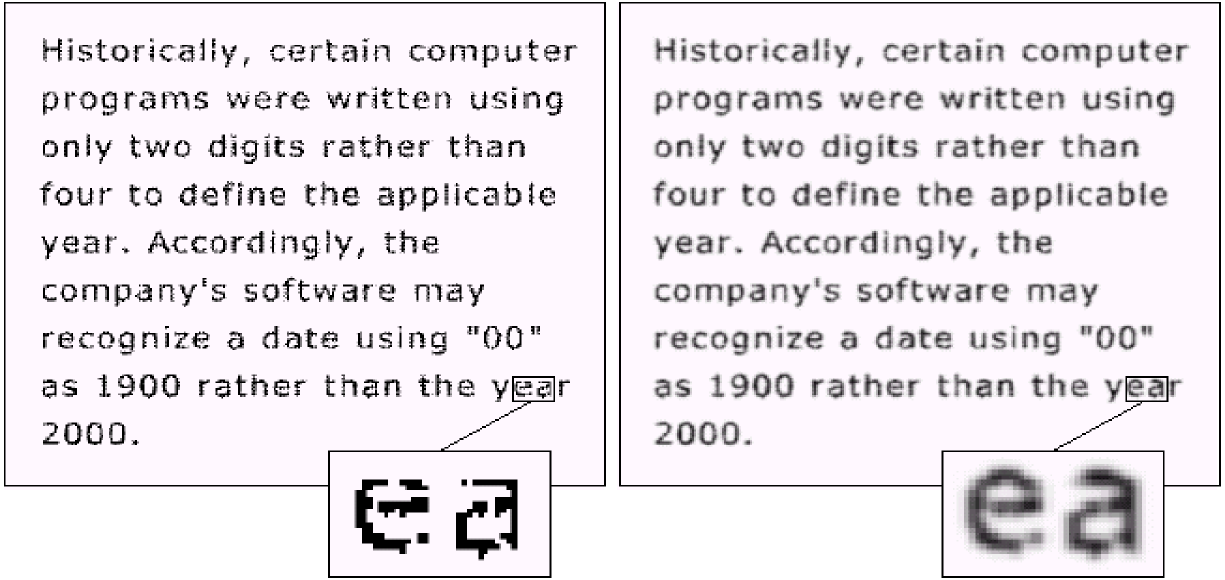


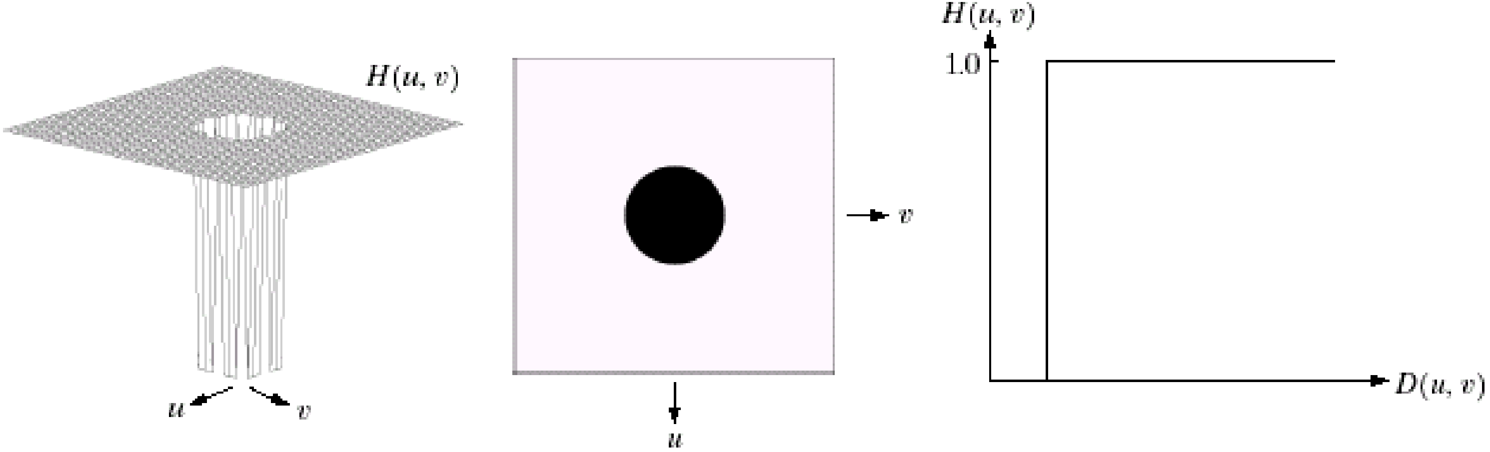
Image courtesy: Gonzalez and Woods

Image Sharpening in Frequency Domain

High Pass filter can be obtained from a given low pass filter:

$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

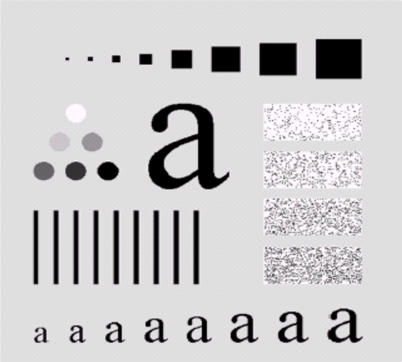
Ideal High Pass Filters

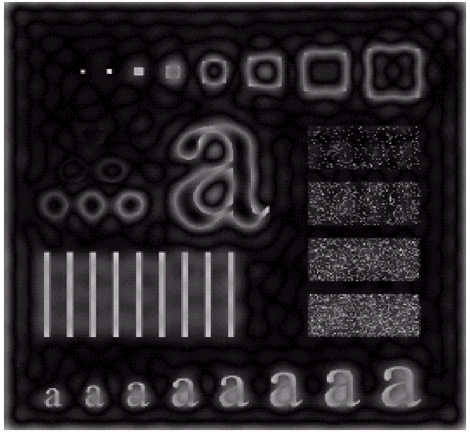


$$H(u,v)=\begin{cases}0 \text{ if } D(u,v)\le D_0\\ 1 \text{ if } D(u,v)>D_0\end{cases}$$

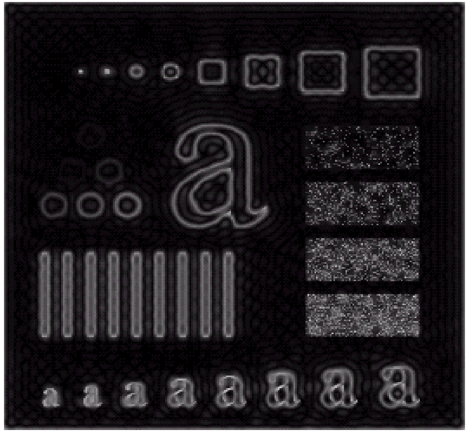
Image courtesy: Gonzalez and Woods

Ideal High Pass Filters

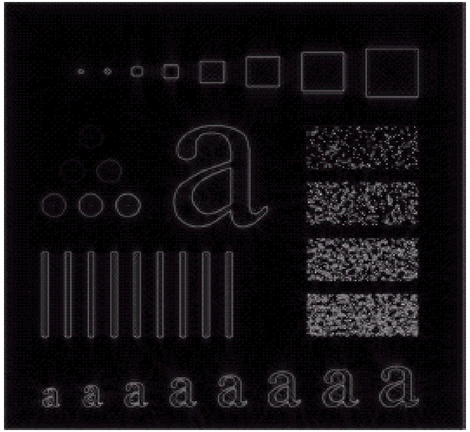




IHPL with $D_0 = 30$



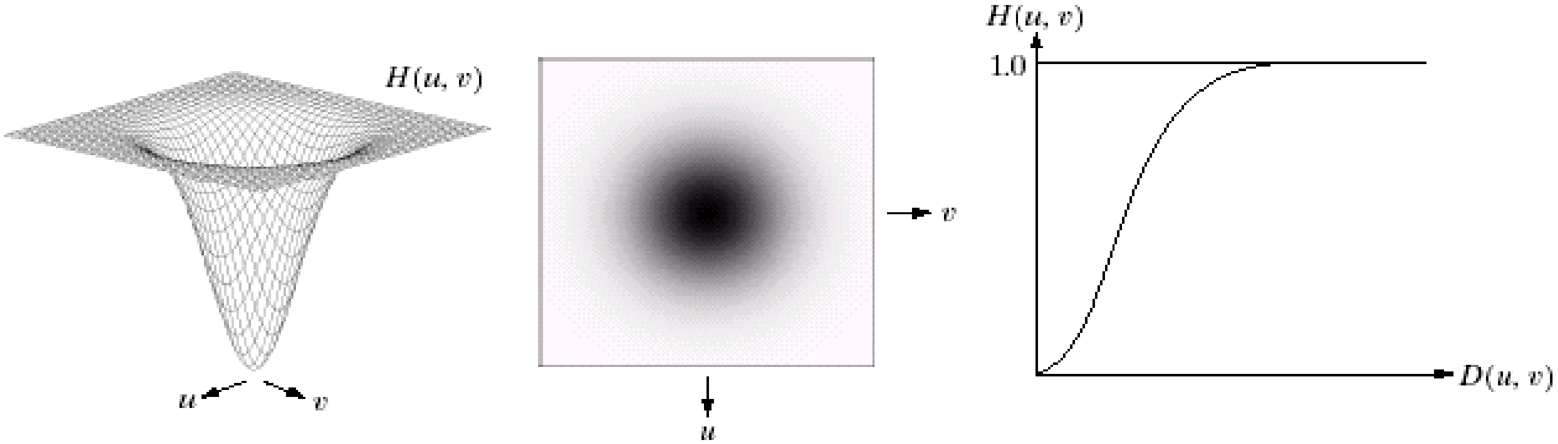
IHPF with $D_0 = 60$



IHPF with $D_0 = 160$

Image courtesy: Gonzalez and Woods

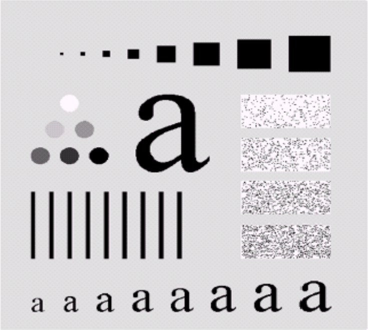
Gaussian High Pass Filters

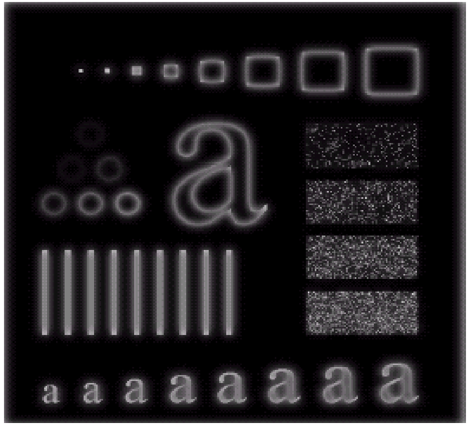


$$H(u,v) = 1 - e^{-D^2(u,v)/2{D_0}^2}$$

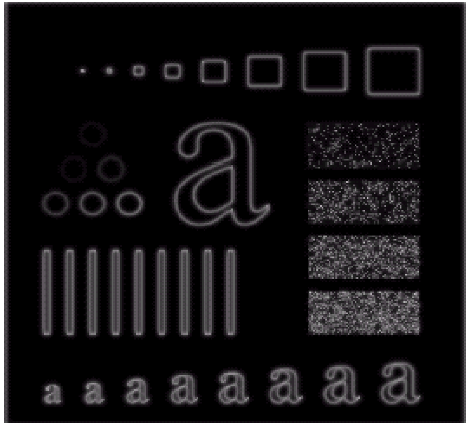
Image courtesy: Gonzalez and Woods

Gaussian High Pass Filters

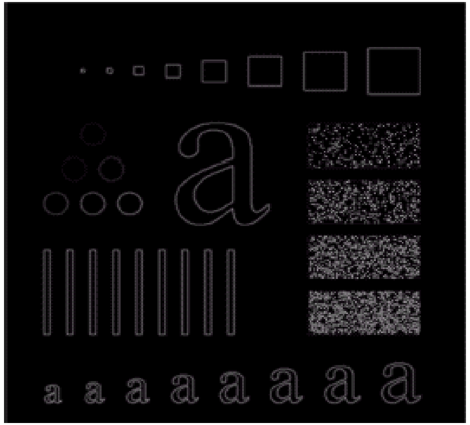




GHPL with D_0 = 30



GHPF with D_0 = 60



GHPF with D_0 = 160

Image courtesy: Gonzalez and Woods

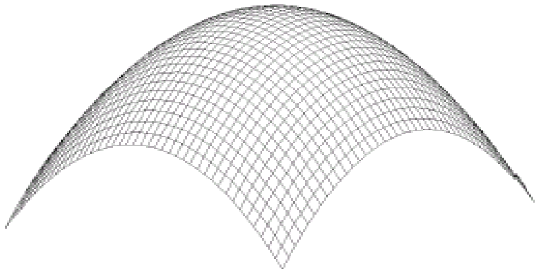
Laplacian in frequency domain

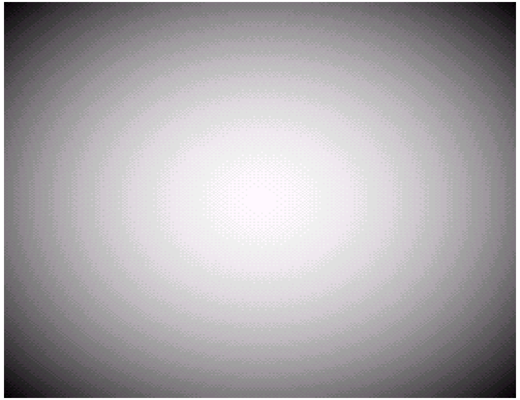
$$\Im[\frac{d^n f(x)}{dx^n}] = (ju)^n F(u)$$

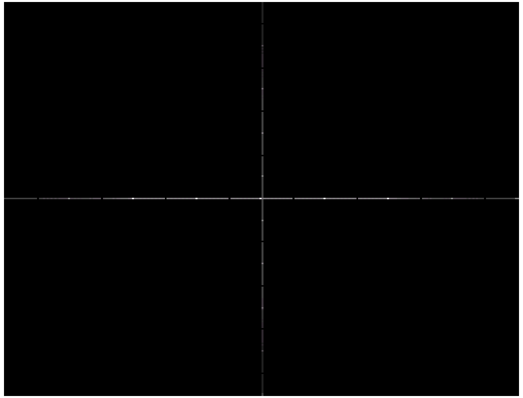
$$\Im[\frac{\partial^2(f(x,y))}{\partial x^2} + \frac{\partial^2(f(x,y))}{\partial y^2}] = (ju)^2 F(u,v) + (jv)^2 F(u,v)$$

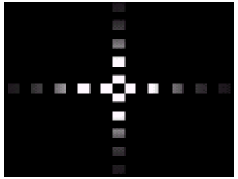
$$= -(u^2 + v^2) F(u,v)$$

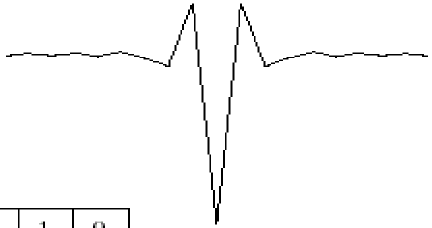
Laplacian in frequency domain











0	1	0
1	−4	1
0	1	0

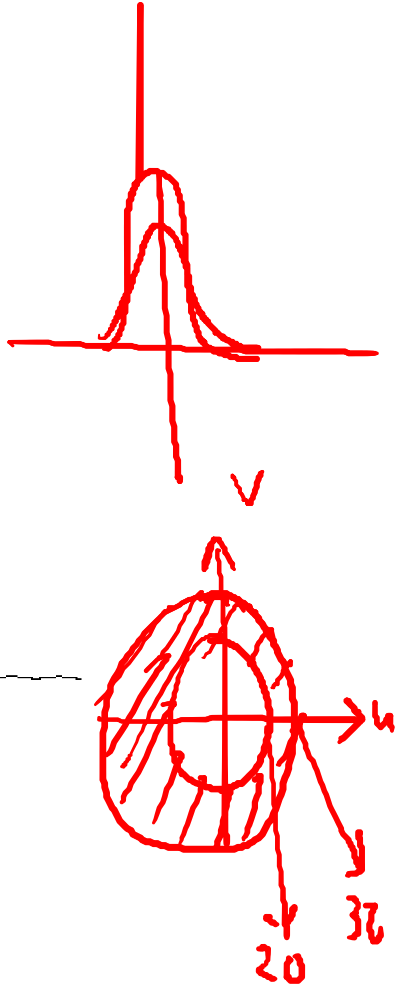


Image courtesy: Gonzalez and Woods

Notch Reject filter (Notch pass filter)

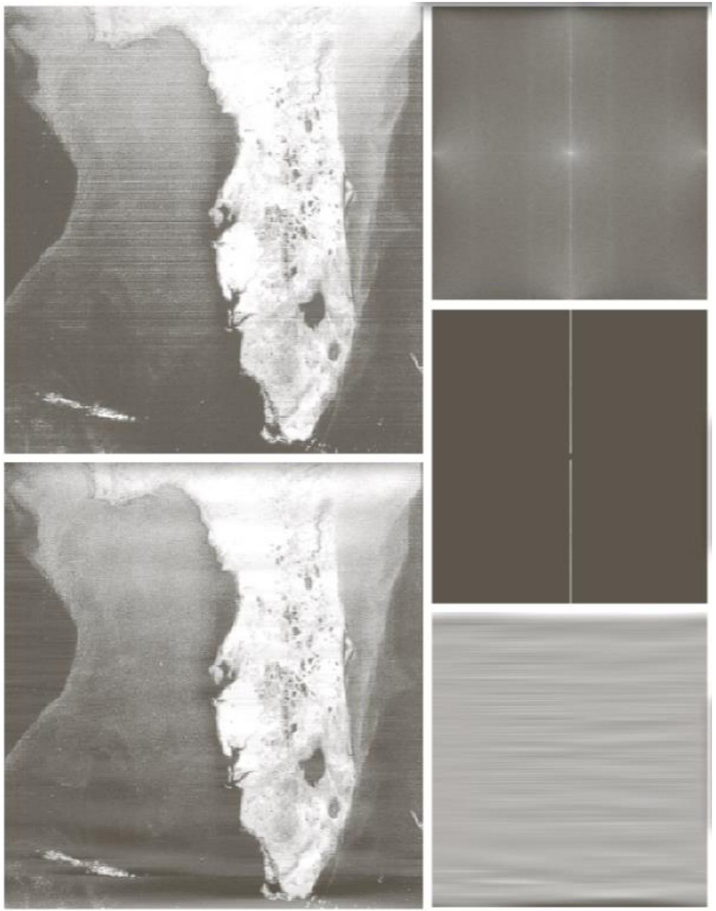


Image courtesy: Gonzalez and Woods

Filtering in frequency domain

- Band reject (Band pass filters)
- Unsharp Masking and High boost filtering
- Homomorphic filtering

$$\ln[i[x,y]] = \ln[I[x,y]\,R[x,y]]$$

$$= \ln[I[x,y]] + \ln[R[x,y]]$$

Additional considerations

- Circular convolution → Wraparound error
  - Zero padding
- Windowing the transform (apodizing)

Frequency Domain vs Spatial Domain Filtering

- Any linear spatial filter
- Guide the process of spatial filter design

Related Topics

- Gabor filters
- Wavelets
- Shape descriptors

References

- G & W (4.5.1, 4.5.2, 4.5.5, 4.6 – 4.11)