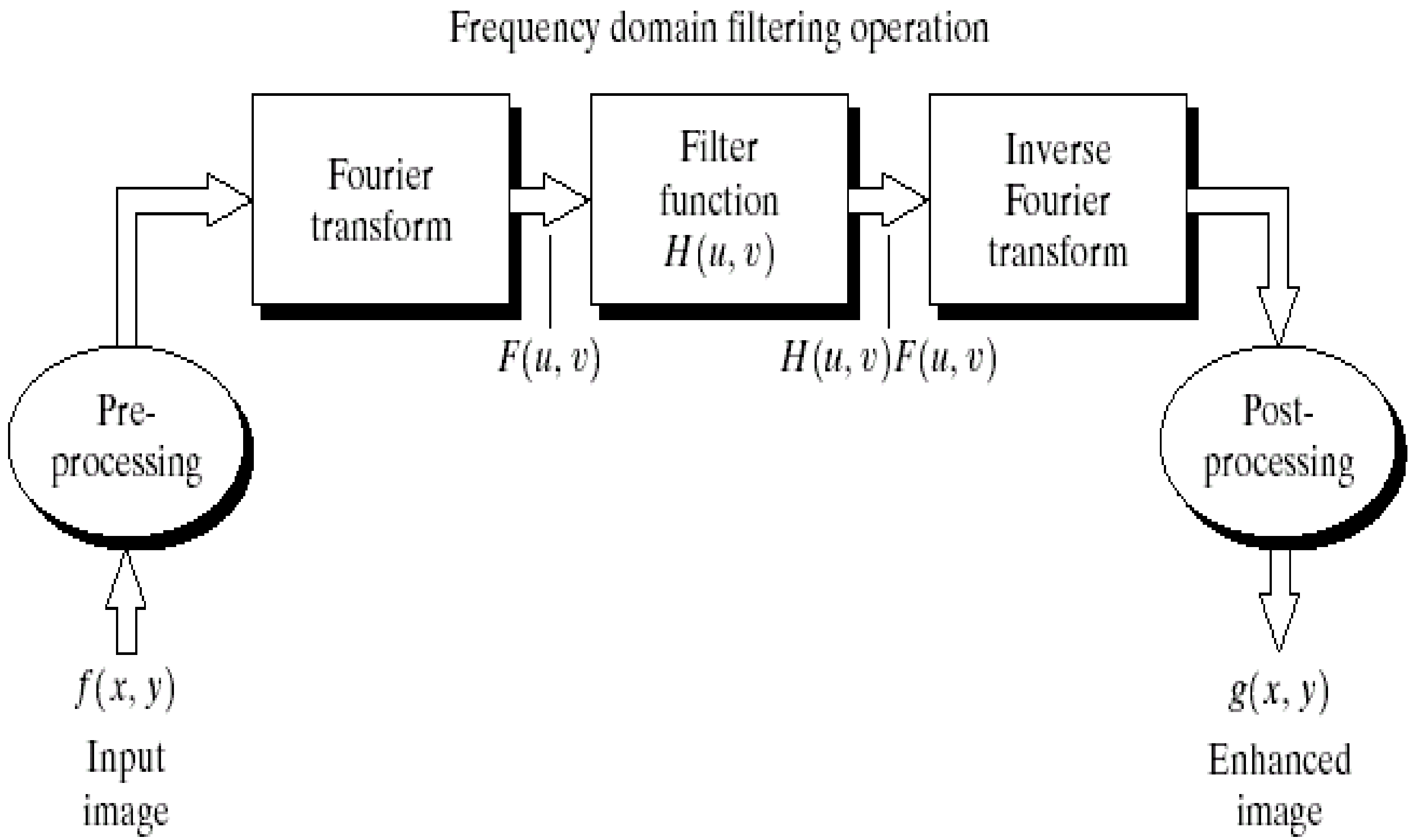
Filtering in Frequency Domain

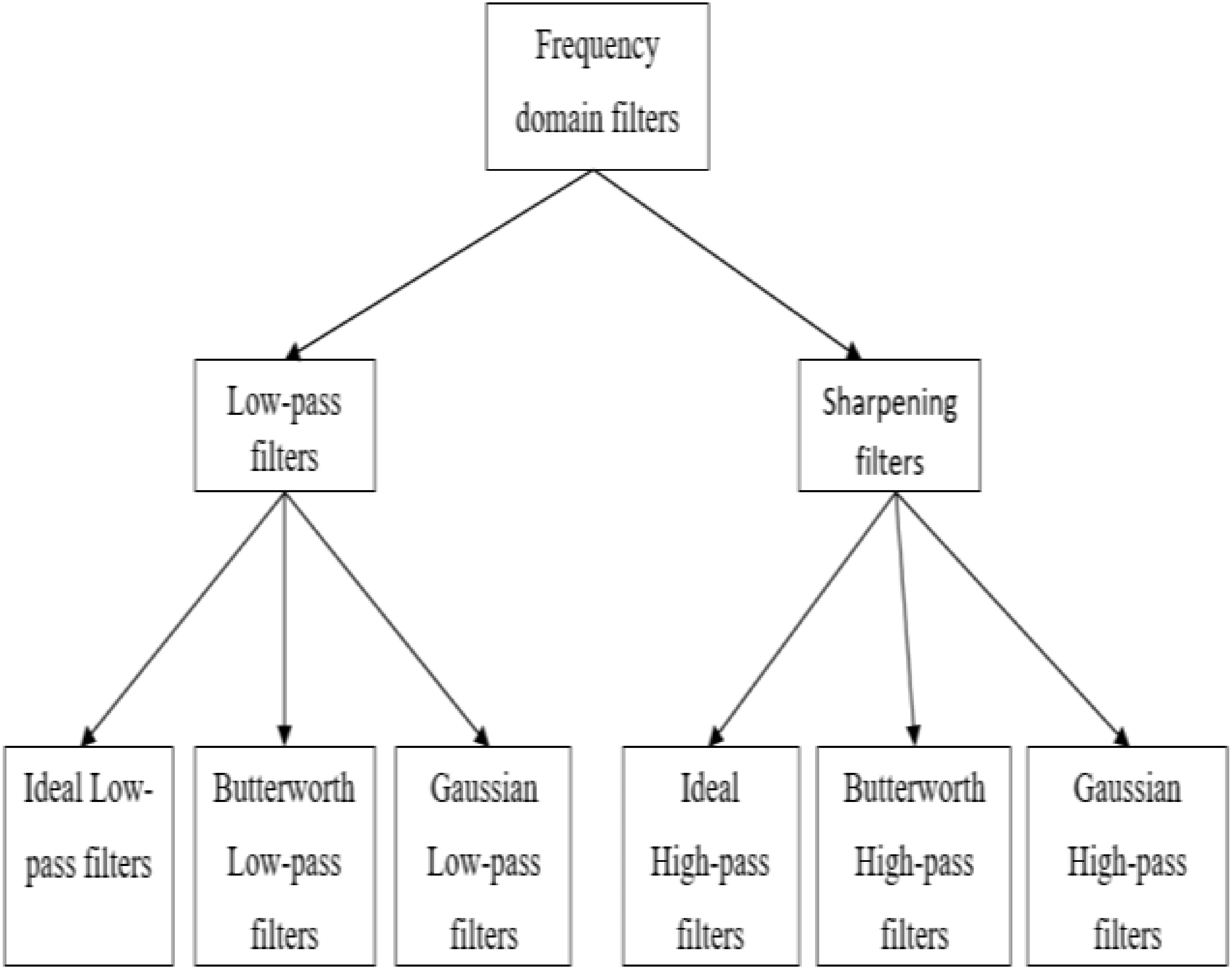
- one reason for using Fourier transform in image processing is due to convolution theorem
- Spatial convolution can be performed by element-wise multiplication of the Fourier transform by suitable "filter matrix"



- In spatial domain, we deal with images as it is. The value of the pixels of the image change with respect to scene.
- Whereas in frequency domain, we deal with the rate at which the pixel values are changing in spatial domain.
- Filtering in the frequency domain is generally:

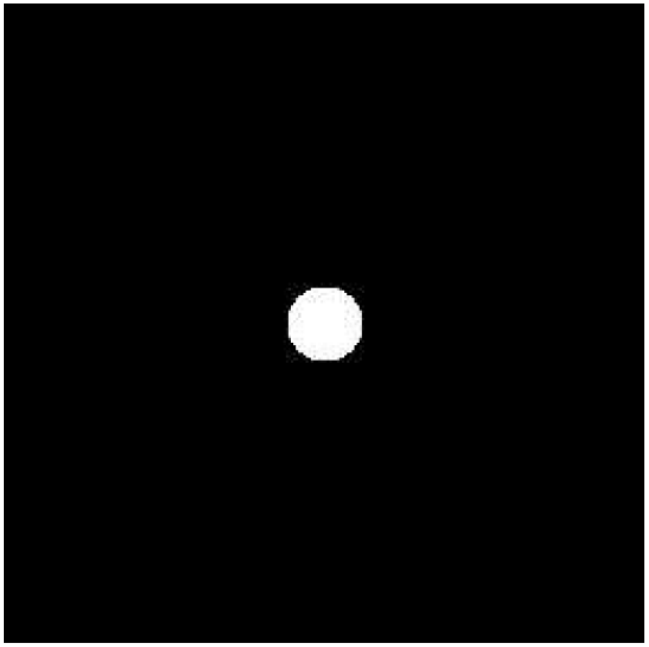
\- **computationally faster to perform two 2D Fourier transforms and a filter multiply than to perform a convolution in the image (spatial) domain**.

\- **gives control over the whole images, where you can enhance(eg edges) and suppress (eg smooth shadow) different characteristics of the image very easily**.


Frequency domain filters
Low-pass filters
Sharpening filters
Ideal Low-pass filters
Butterworth Low-pass filters
Gaussian Low-pass filters
Ideal High-pass filters
Butterworth High-pass filters
Gaussian High-pass filters

Ideal Low Pass Filtering

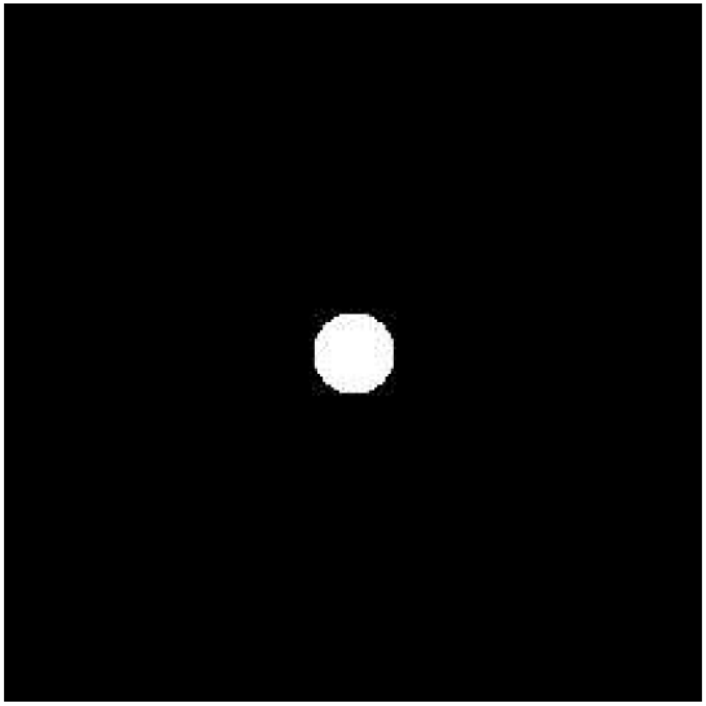
- **Low pass filter:** Keep frequencies **below** a certain frequency
- Low pass filtering causes **blurring**
- After DFT, DC components and low frequency components are towards center
- May specify frequency cutoff as circle c



Ideal Low Pass Filtering

- Multiply Image Fourier Transform *F* by some filter matrix *m*

$$m(x,y) = \begin{cases} 1 & \text{if } (x,y) \text{ is closer to the center than some value } D, \\ 0 & \text{if } (x,y) \text{ is further from the center than } D. \end{cases}$$



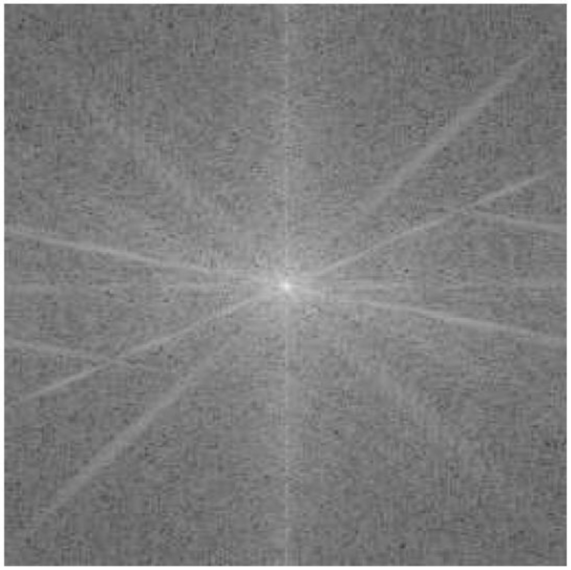
Ideal Low Pass Filtering

- Low pass filtered image is inverse Fourier Transform of product of *F* and *m*

$$\mathcal{F}^{-1}(F \cdot m)$$

- Example: Consider the following image and its DFT

Image



DFT

Ideal Low Pass Filtering

Applying
low pass filter
to DFT
Cutoff D = 15

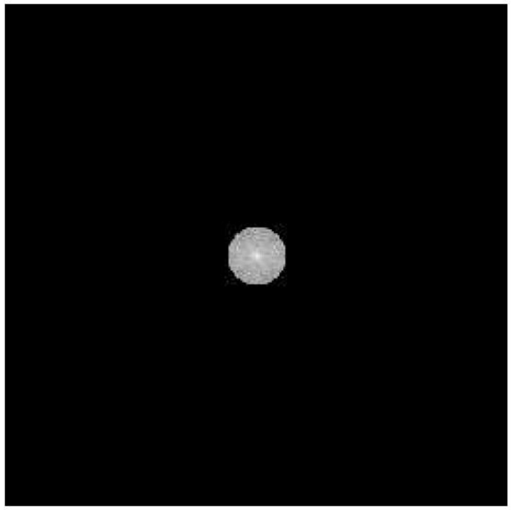
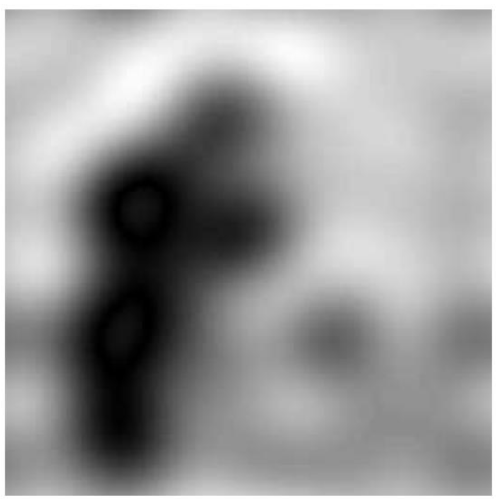


Image after
inversion

low pass filter
Cutoff D = 5

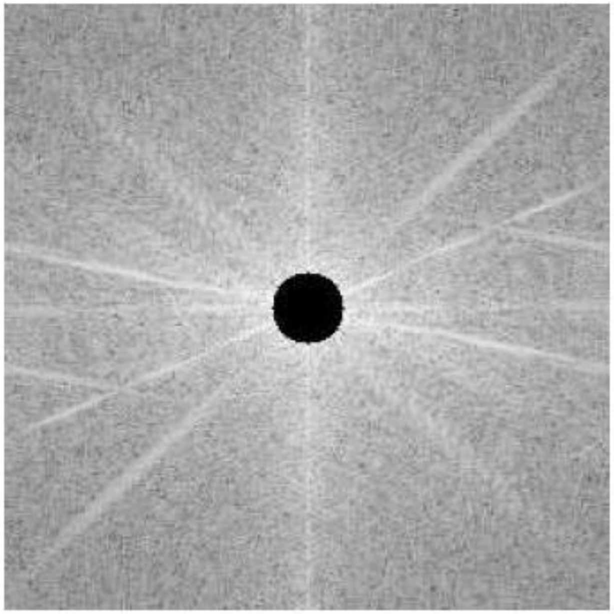
Note: Sharp filter
Cutoff causes
ringing



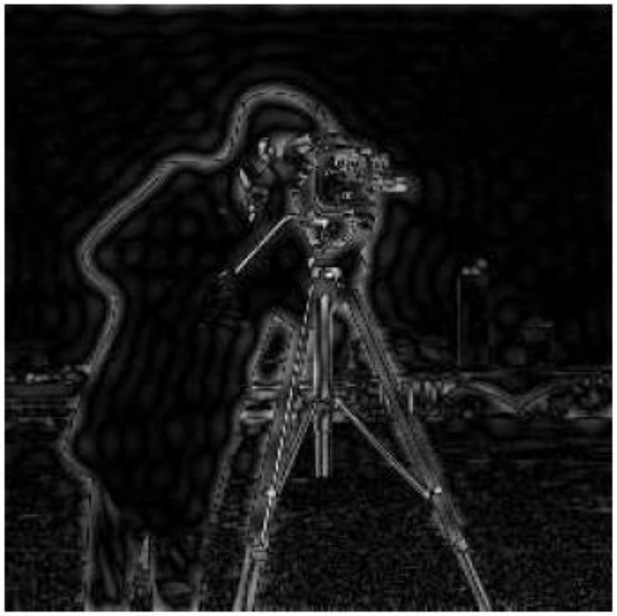
low pass filter
Cutoff D = 30

Ideal High Pass Filtering

- Opposite of low pass filtering: eliminate center (low frequency values), keeping others
- High pass filtering causes image **sharpening**
- If we use circle as cutoff again, size affects results
  - Large cutoff = More information removed



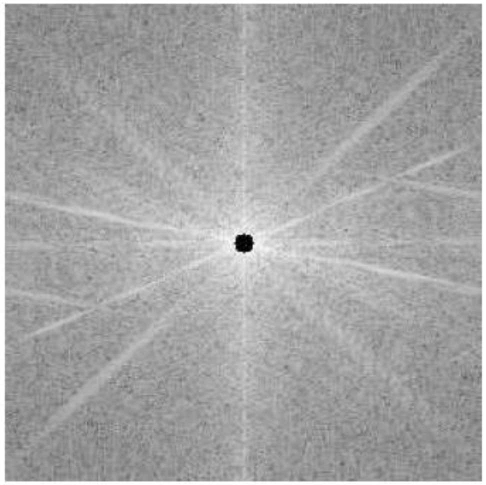
DFT of Image after high pass Filtering



Resulting image after inverse DFT

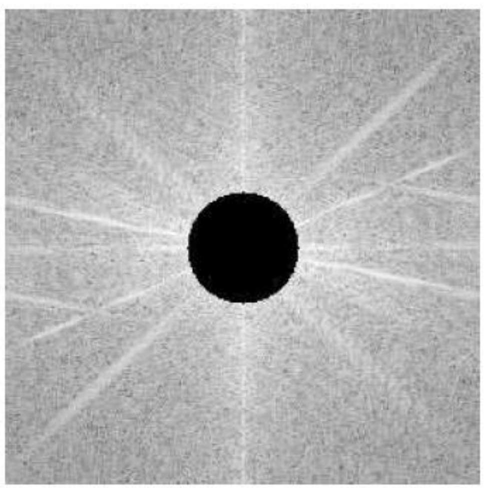
Ideal High Pass Filtering: Effect of Cutoffs

High pass filtering of DFT with filter Cutoff D = 5



Low cutoff frequency removes Only few lowest frequencies

High pass filtering of DFT with filter Cutoff D = 30

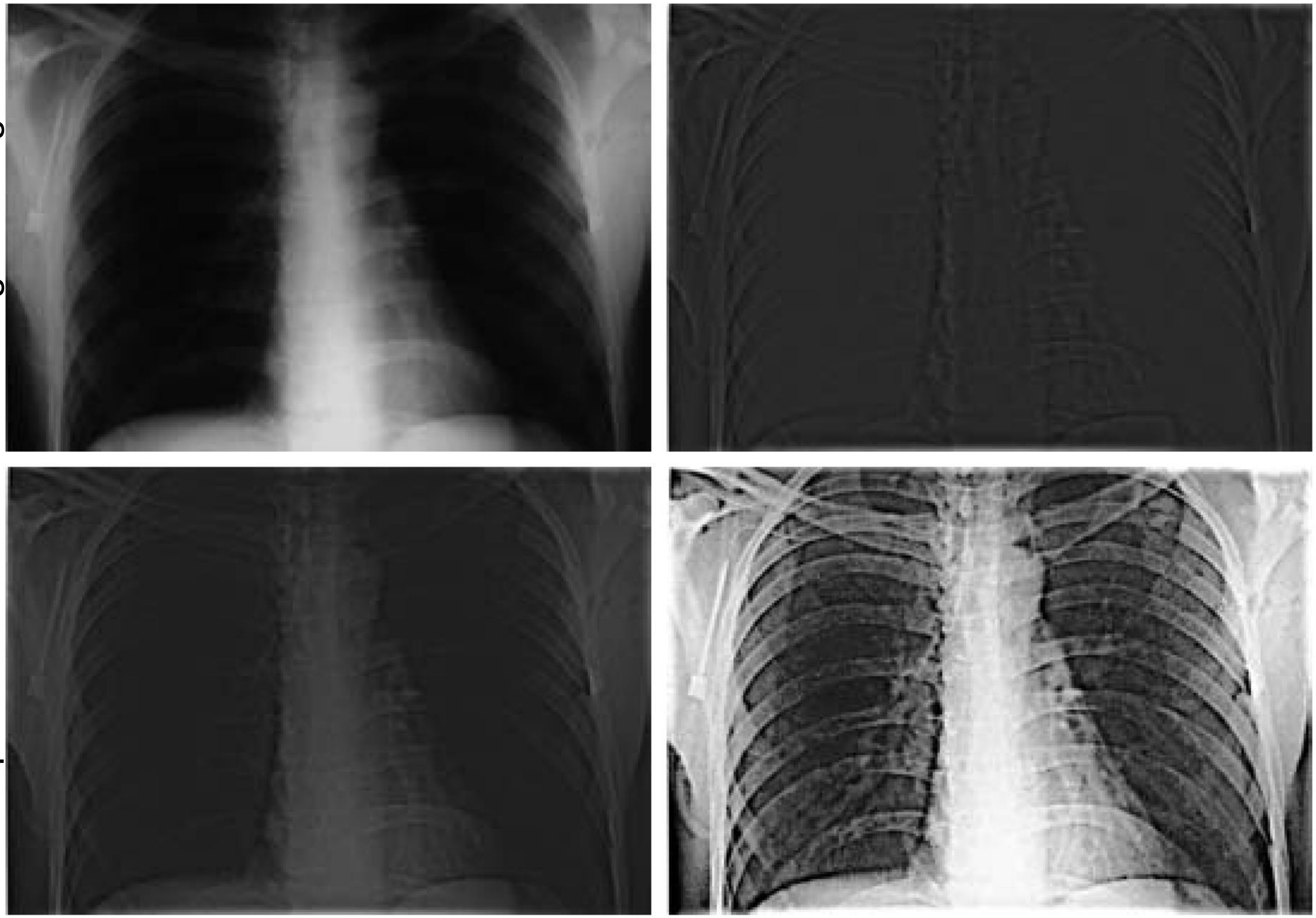


High cutoff frequency removes many frequencies, leaving only edges

Highpass Filtering Example

Original image

Highpass filtering result



High frequency emphasis result

After histogram equalisation

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Butterworth Filtering

The Butterworth filter has a parameter called the filter order.

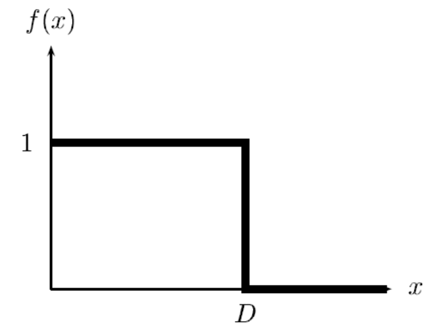
- For high order values, the Butterworth filter approaches the ideal filter. For low order values, Butterworth filter is more like a Gaussian filter.

- Thus, the Butterworth filter may be viewed as providing a transition between two “extremes”.

Butterworth Filtering (Formal)

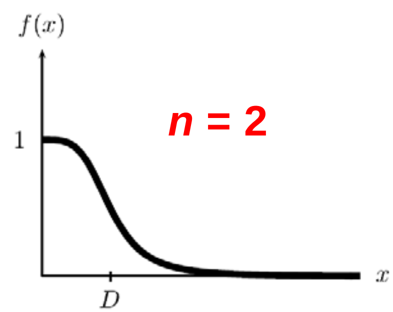
- Sharp cutoff leads to ringing
- To avoid ringing, can use circle with more gentle cutoff slope
- **Butterworth filters** have more gentle cutoff slopes
- Ideal low pass filter

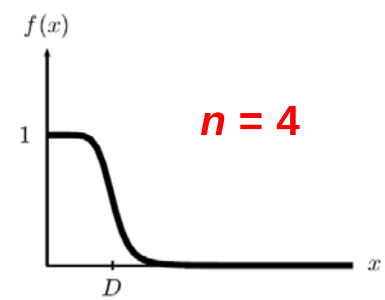
$$f(x) = \begin{cases} 1 & \text{if } x < D \\ 0 & \text{if } x \geq D \end{cases}$$



- Butterworth
  low pass filter

$$f(x) = \frac{1}{1 + (x/D)^{2n}}$$



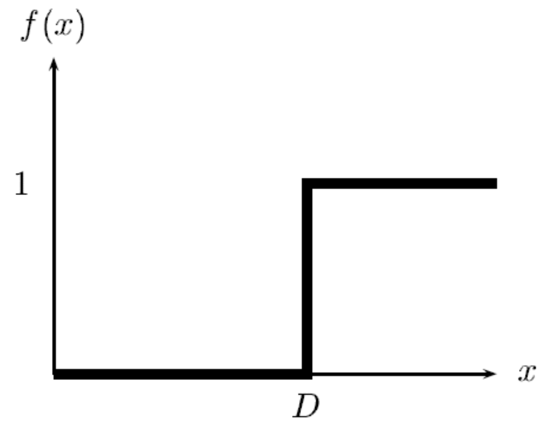


- *n* called **order** of the filter, controls sharpness of cutoff

Butterworth High Pass Filtering

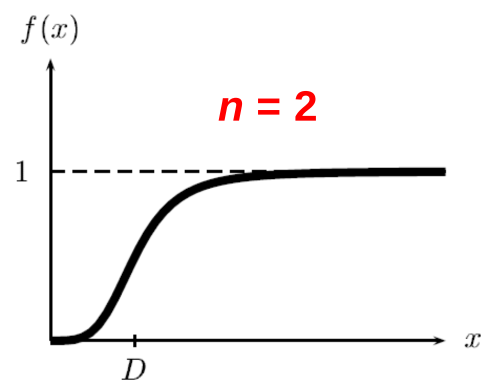
- Ideal high pass filter

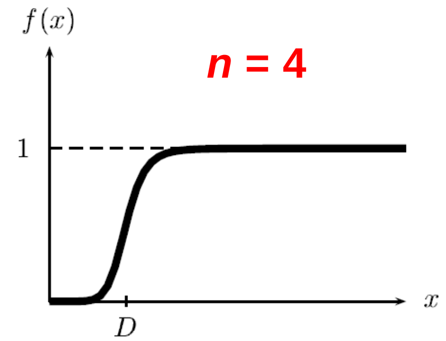
$$f(x) = \begin{cases} 1 & \text{if } x > D \\ 0 & \text{if } x \leq D \end{cases}$$



- Butterworth high pass filter

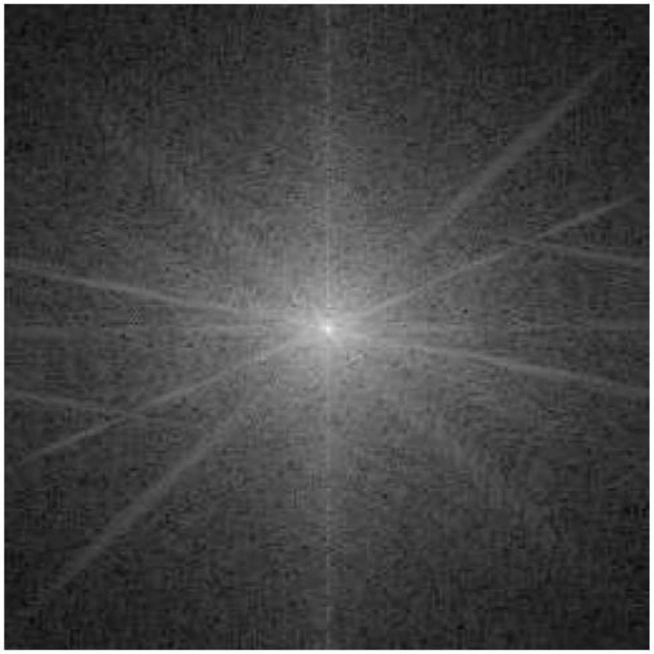
$$f(x) = \frac{1}{1 + (D/x)^{2n}}$$





Low Pass Butterworth Filtering

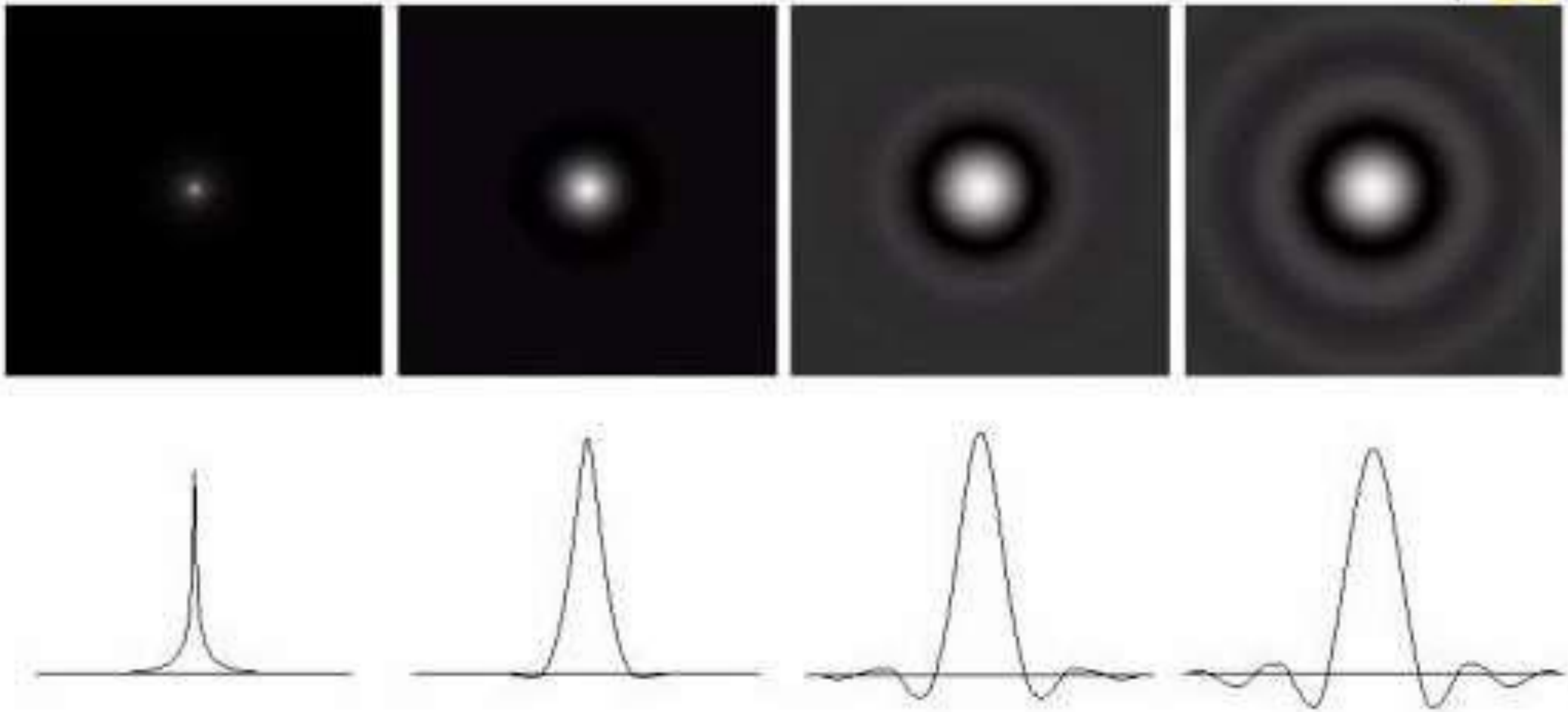
- Low pass filtering removes high frequencies, blurs image
- Gentler cutoff eliminates ringing artifact



DFT of Image after low pass Butterworth filtering

Resulting image after inverse DFT

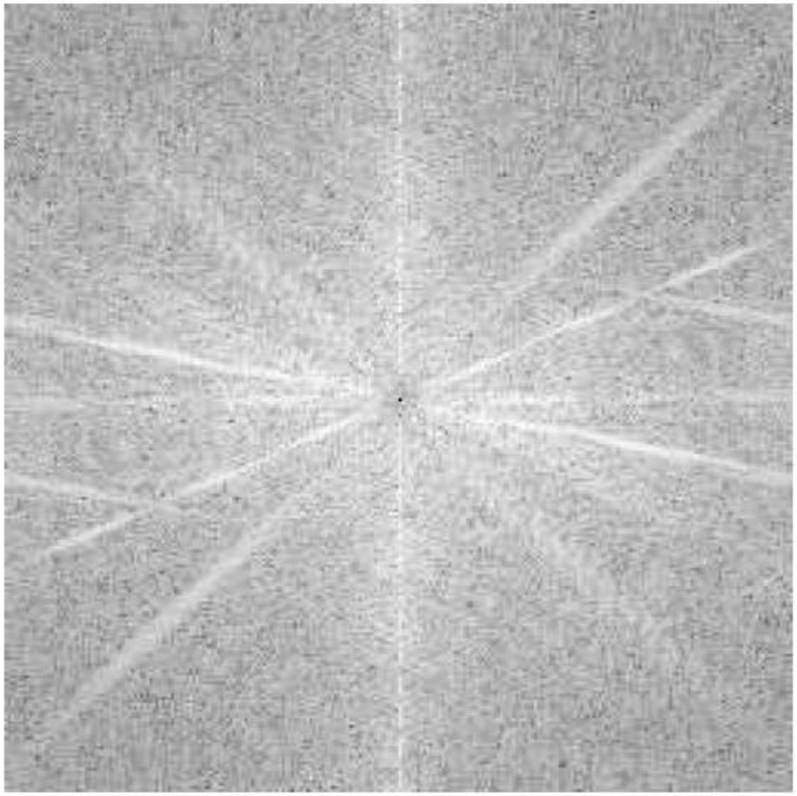
BUTTERWORTH LOWPASS FILTERS



a b c d

FIGURE 4.16 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding gray-level profiles through the center of the filters (all filters have a cutoff frequency of 5). Note that ringing increases as a function of filter order.

High Pass Butterworth Filtering



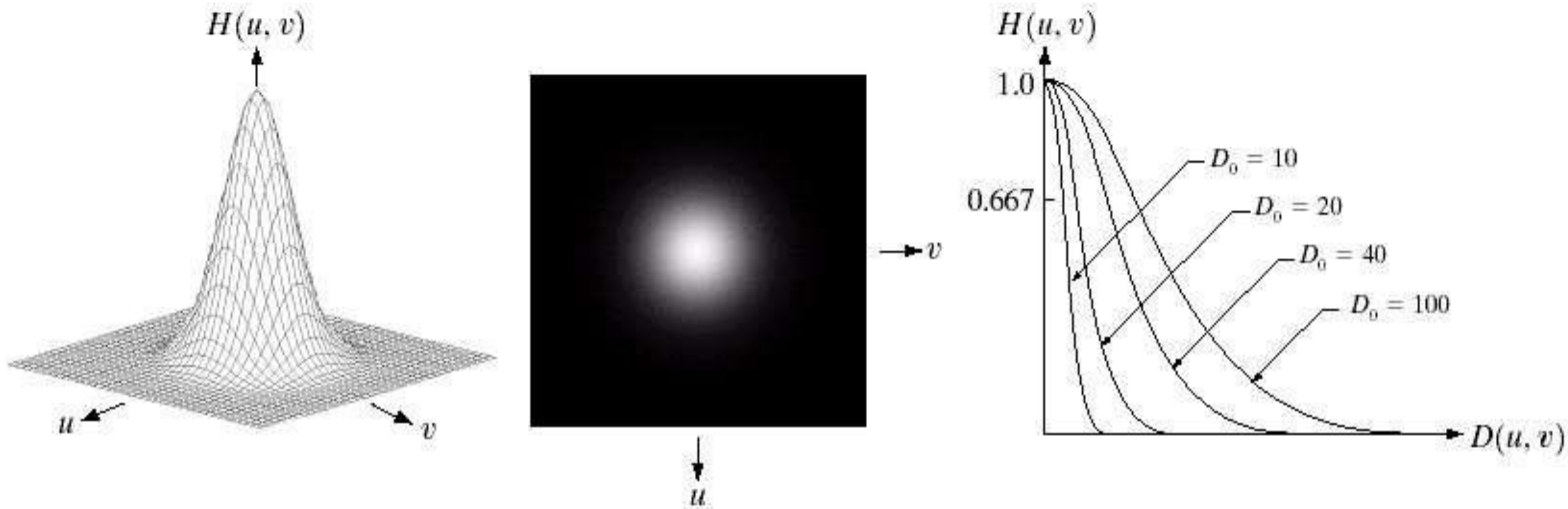
DFT of Image after high
pass Butterworth filtering

Resulting image
after inverse DFT

Gaussian Filtering

- Gaussian filters can be applied in frequency domain
- Same steps
  - Create gaussian filter
  - Multiply (**DFT of image**) by (**gaussian filter**)
  - Invert result
- **Note:** Fourier transform of gaussian is also a gaussian,
- Just apply gaussian multiply directly (no need to find Fourier transform)

GAUSSIAN LOWPASS FILTERS



A)Perspective plot of a GLPF transfer function

B) Filter displayed as an image

C) Filter radius cross section for various values of D_0

The concept of filtering and low pass remains the same, but only the transition becomes different and smoother.

Main advantage of a Gaussian LPF over a Butterworth LPF is that we are assured that there will be no ringing effects no matter what filter order we choose to work with.

GAUSSIAN LOWPASS FILTERS

The transfer function of a Gaussian lowpass filter is defined as:

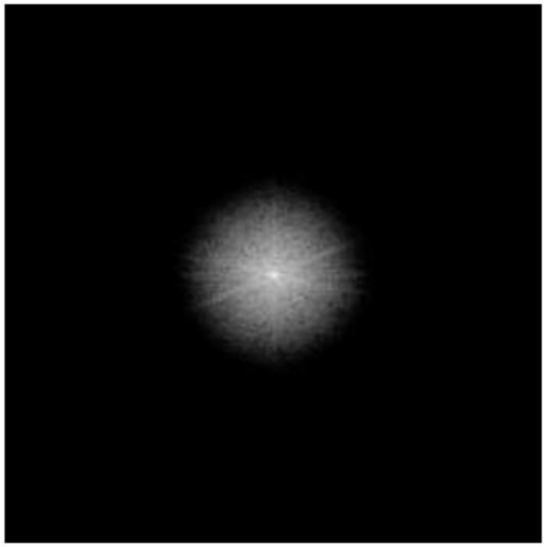
$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}$$

Here, σ is the standard deviation and is a measure of spread of the Gaussian curve.

- If we put $\sigma = D_0$ we get,

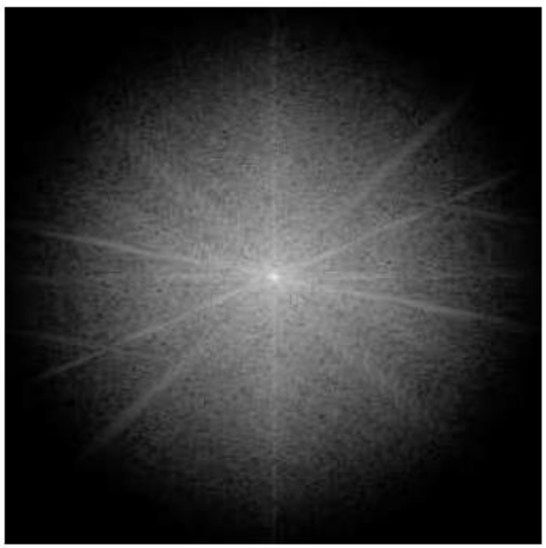
$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

Frequency Domain Low Pass Gaussian Filtering



(a) $\sigma = 10$

(b) Resulting image

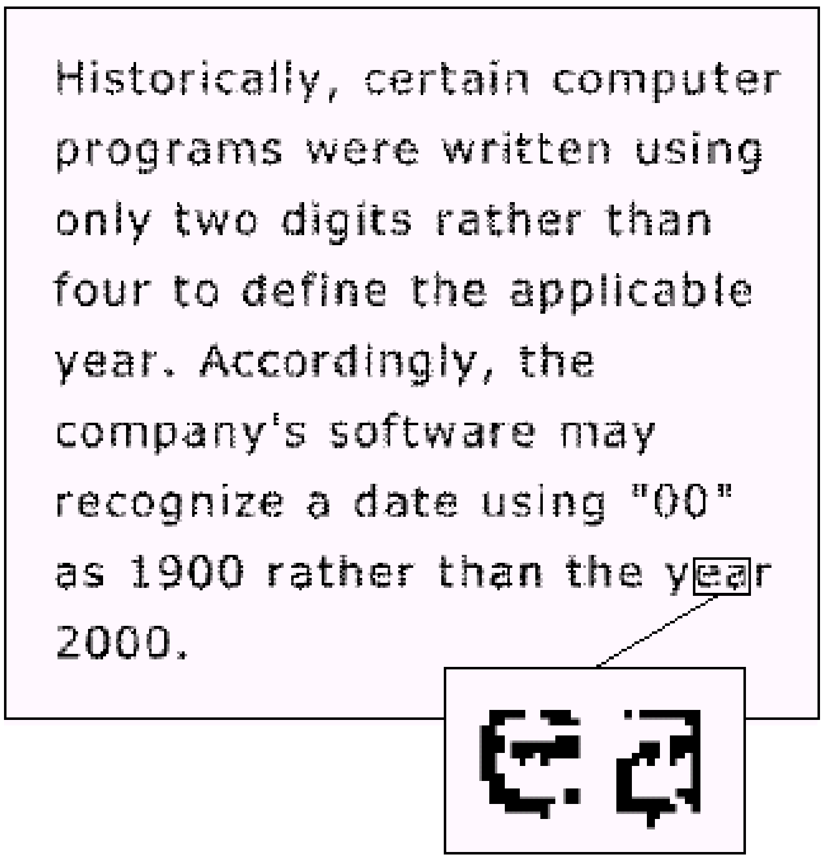


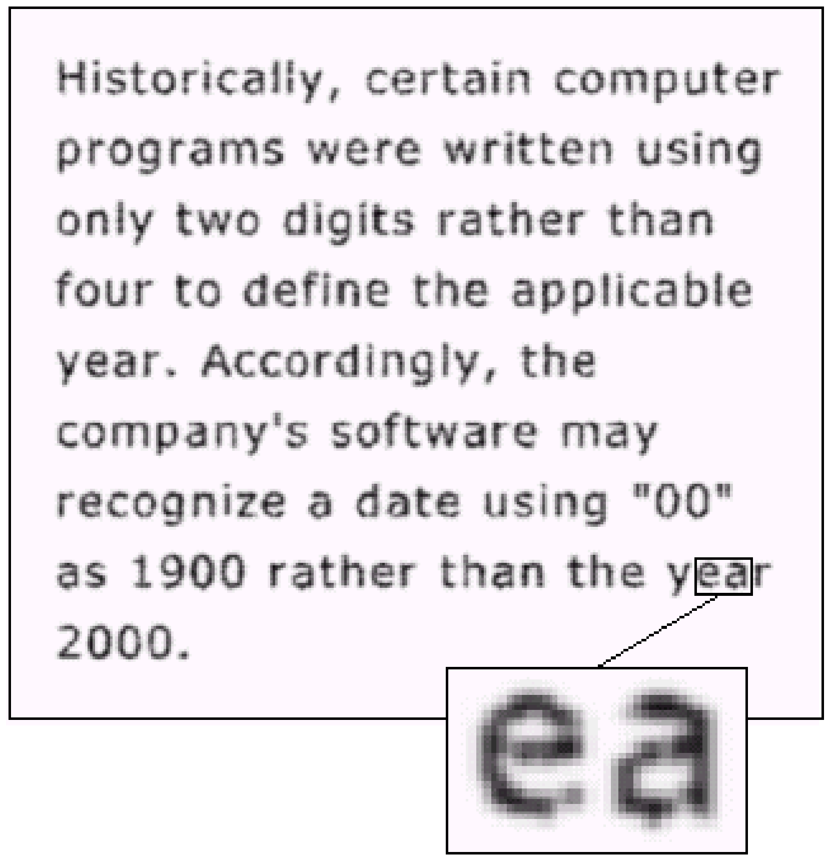
(c) $\sigma = 30$

(d) Resulting image

Lowpass Filtering Examples

A low pass Gaussian filter can be used to connect broken text


Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.


Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Lowpass Filtering Examples

Different lowpass Gaussian filters used to remove blemishes in a photograph

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Gaussian high pass filter

- Gaussian high pass filter has the same concept as ideal high pass filter, but again the **transition is more smooth** as compared to the ideal one.

Frequency Domain High Pass Gaussian Filtering

(a) Using $\sigma = 10$

(b) Using $\sigma = 30$

Frequency Domain
Removal of Periodic Noise

- **Recall:** periodic noise could not be removed in spatial domain
- Periodic noise can be removed in frequency domain
- Periodic noise shows up as spikes away from origin in DFT
- Higher frequency noise = farther away from origin

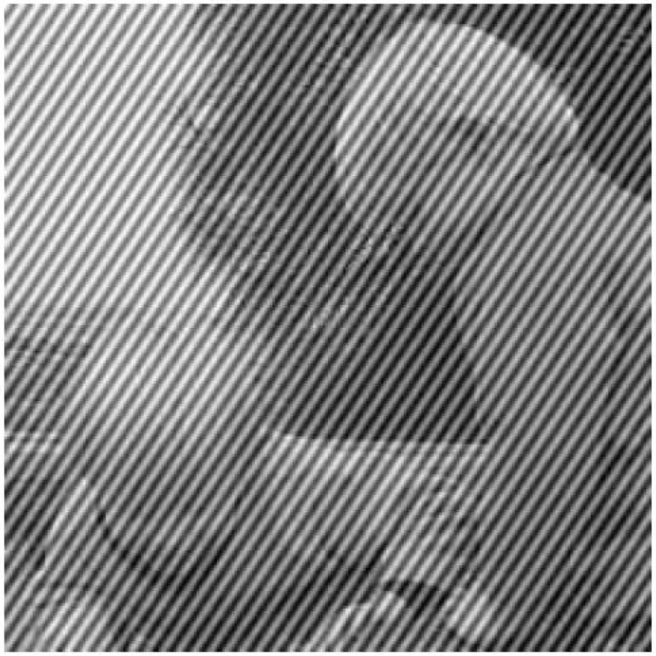
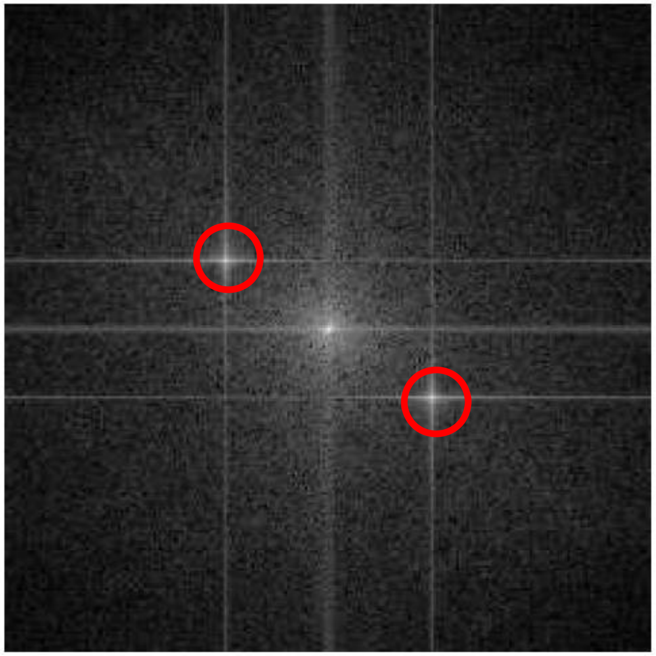


Image with periodic Noise

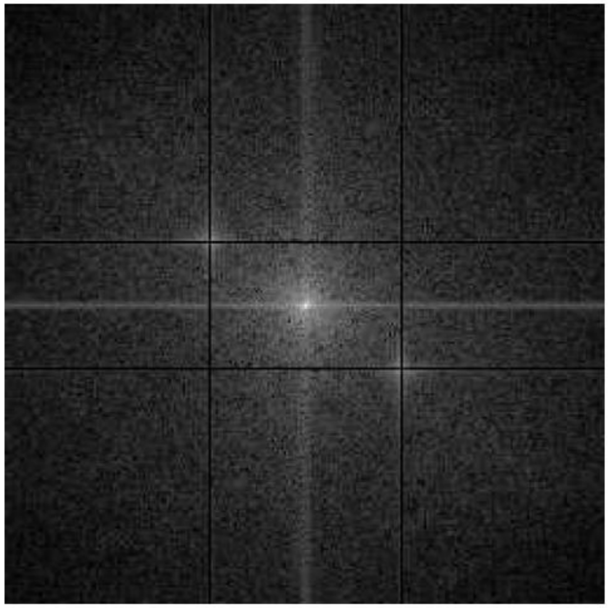


DFT of Image

Frequency Domain
Removal of Periodic Noise

- 2 ways to remove periodic noise in frequency domain
  - Notch Filter
  - Band reject filter

- Notch filter: Set rows, columns of DFT corresponding to noise = 0
- Removes much of the periodic noise

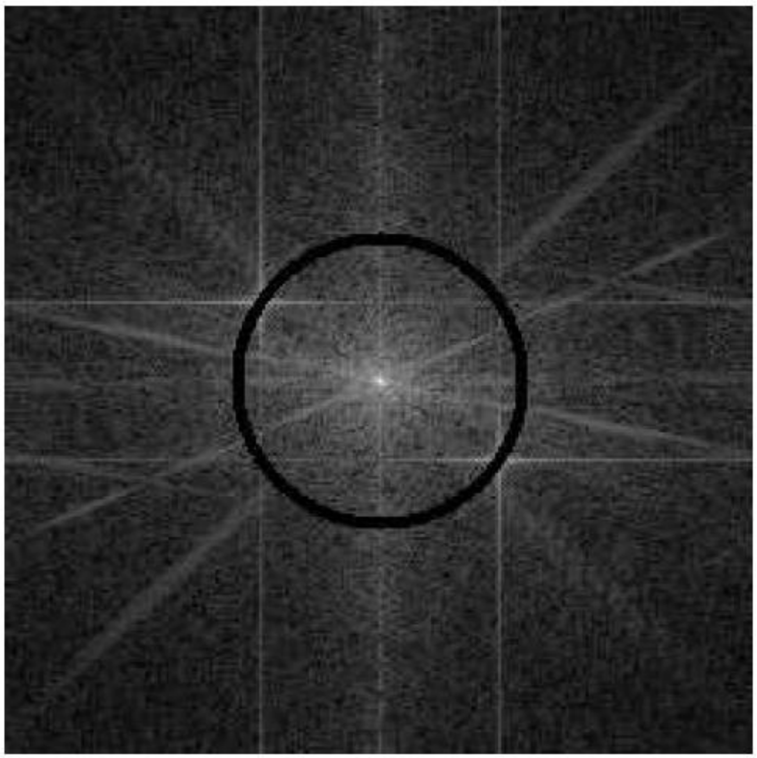


Notch Filter

Result after notch filter applied then inverted

Frequency Domain Removal of Periodic Noise: Band Reject Filter

- Create filter with 0's at radius of noise from center, 1 elsewhere
- Apply filter to DFT

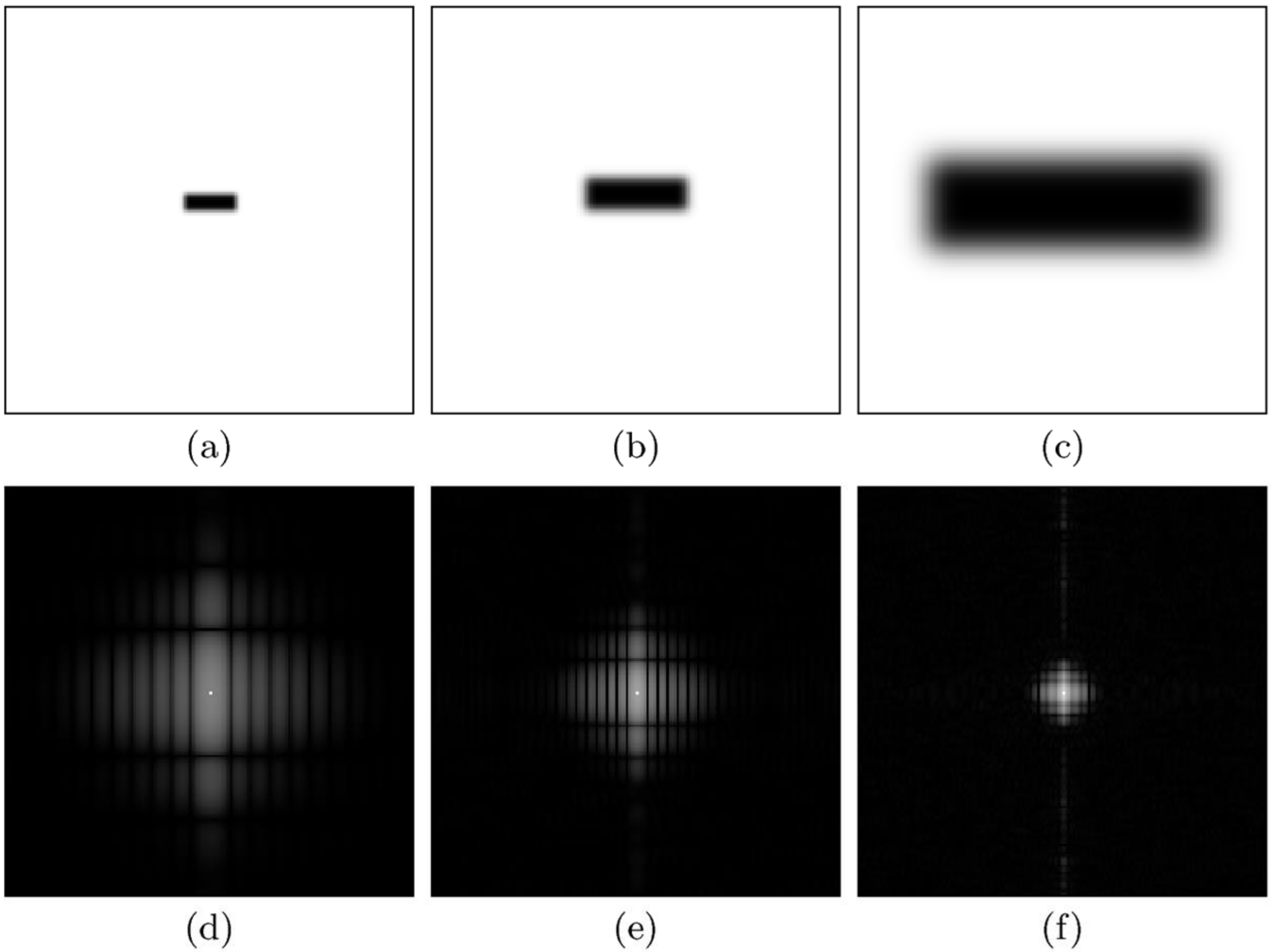


Band Reject Filter

Result after band reject filter applied then inverted

2D Fourier Transform Examples: Scaling

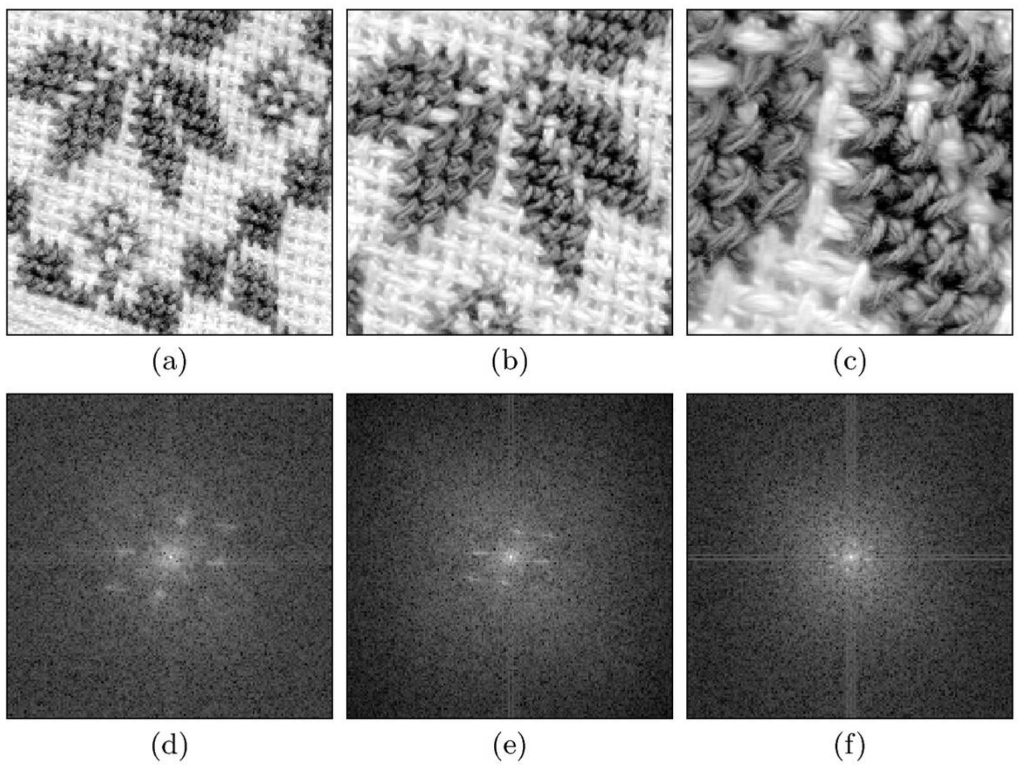
- Stretching image ➔ Spectrum contracts
- And vice a versa



(a) (b) (c) (d) (e) (f)

2D Fourier Transform Examples: Periodic Patterns

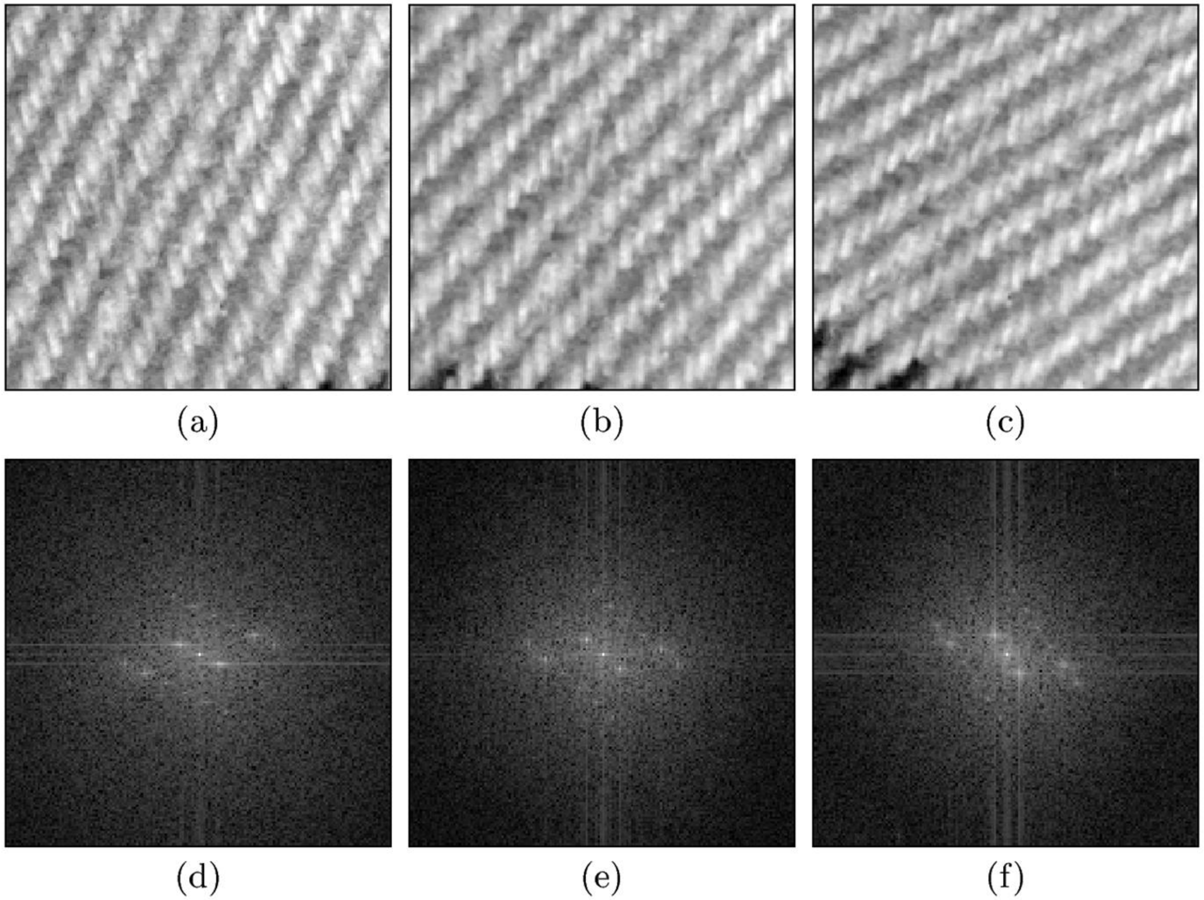
- Repetitive periodic patterns appear as distinct peaks at corresponding positions in spectrum



Enlarging image (c) causes Spectrum to contract (f)

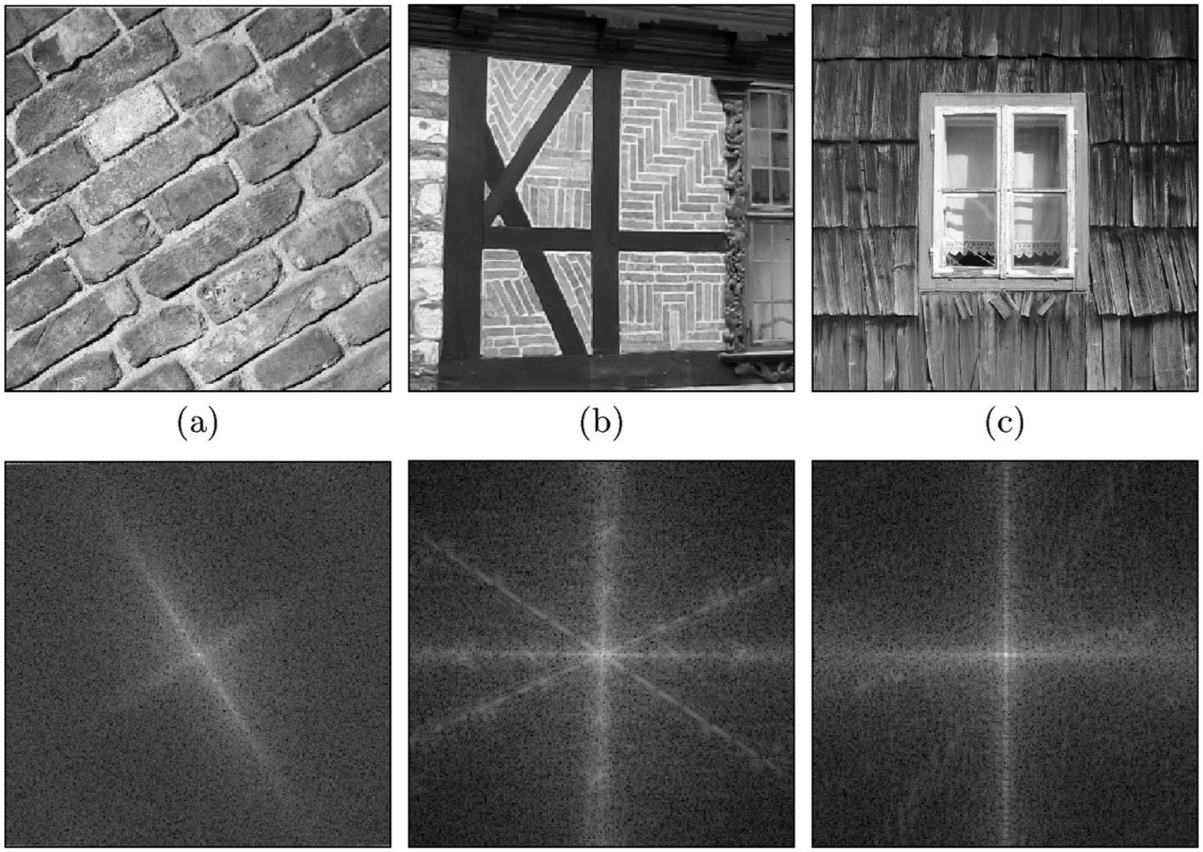
2D Fourier Transform Examples: Rotation

- Rotating image ➔ Rotates spectra by same angle/amount



2D Fourier Transform Examples: Oriented, elongated Structures

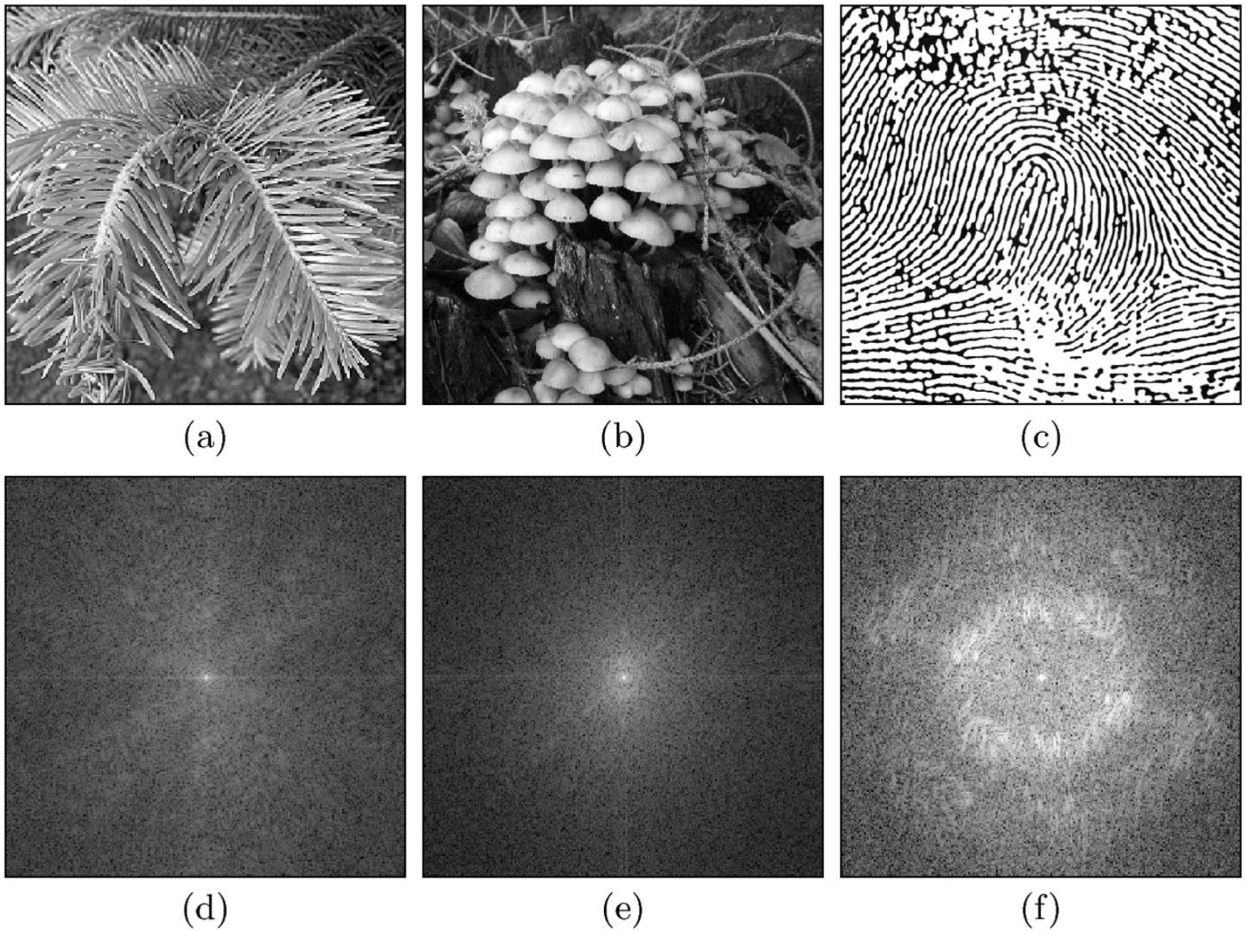
- Man-made elongated regular patterns in image => appear dominant in spectrum



(a) (b) (c) (d) (e) (f)

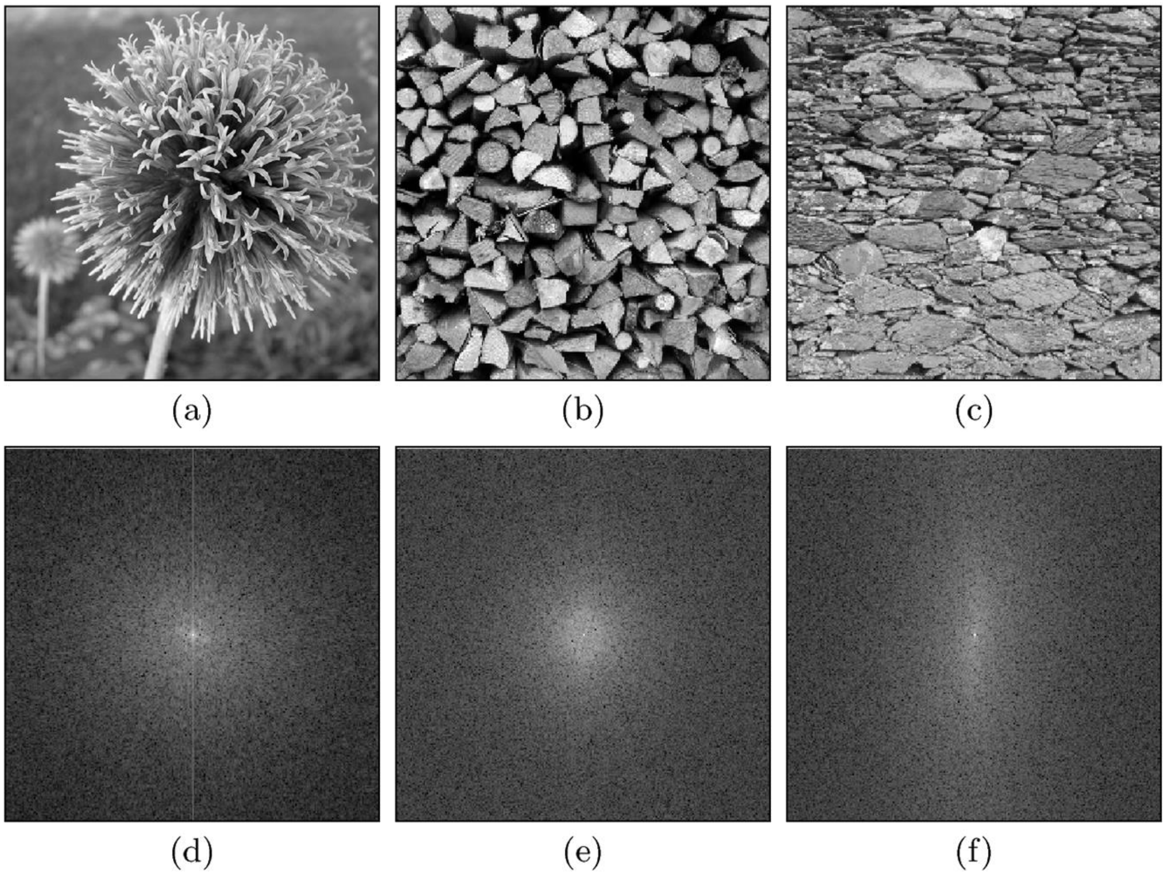
2D Fourier Transform Examples: Natural Images

- Repetitions in natural scenes ➔ less dominant than man-made ones, less obvious in spectra



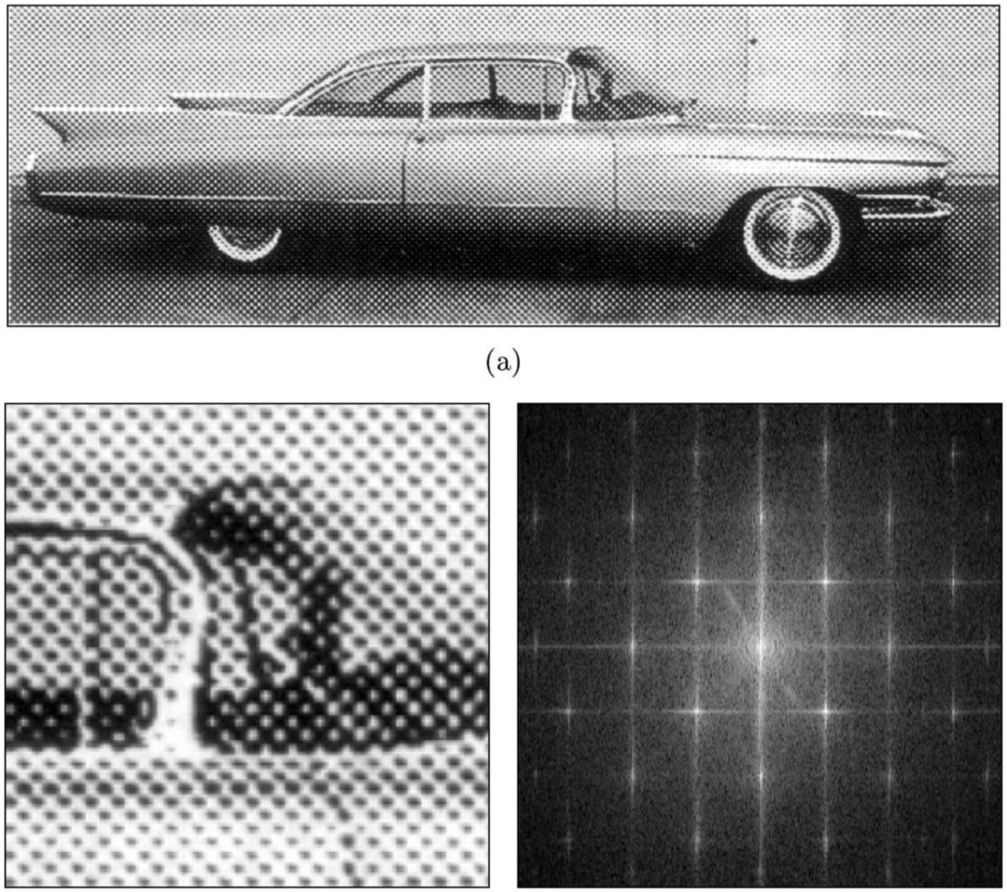
2D Fourier Transform Examples: Natural Images

- Natural scenes with repetitive patterns but no dominant orientation ➔ do not stand out in spectra



2D Fourier Transform Examples: Printed Patterns

- Regular diagonal patterns caused by printing => Clearly visible/removable in frequency spectrum.



(a) (b) (c)

FREQUENCY DOMAIN

Watermark attacks	**Effect in spatial domain**	**Effect in frequency domain**
Gaussian smoothing attack	Reduces the variation in image pixel values.	Acts as low pass filter
Gaussian noise attack	Increases the variation in pixel values	Similar effect to a high pass filter
Salt & pepper noise attack	Same as Gaussian noise attack	Same as Gaussian noise attack.
Median filter attack	Similar, albeit much smaller, effect as with Gaussian smoothing attack.	Similar effect to a high pass filter
Histogram equalization attack	Reduces the number of unique grayscale values and make the histogram more uniformly distributed	Similar, albeit more moderate, effect as Gaussian smoothing attack
Sharpen attack	Reduces the overall image intensity and amplifies differences around edges	Acts as high pass filter
JPEG Compression attack	Reduces the variation in image pixel values and creating blocks, or uniform regions, in the image.	Similar effect to a high pass filter.

Restoration Filters:

- Filter used in restoration is different from the filter used in enhancement process
- used for operation of noisy image and estimating the clean and original image

Types of Restoration Filters:

i) Inverse Filter

ii) Pseudo Inverse Filter

iii) Wiener Filter

i) Inverse Filter: Inverse Filtering is the process of receiving the input of a system from its output.

- It is the simplest approach to restore the original image once the degradation function is known.

```
H'(u, v) = 1 / H(u, v)


Let,
F'(u, v) -> Fourier transform of the restored image
G(u, v) -> Fourier transform of the degraded image
H(u, v) -> Estimated or derived or known degradation function
then F'(u, v) = G(u, v)/H(u, v)
where, G(u, v) = F(u, v).H(u, v) + N(u, v)
and F'(u, v) = f(u, v) - N(u, v)/H(u, v)
```

ii) Pseudo Inverse Filter:

- Modified and stabilized version of the inverse filter giving better results
- both inverse and pseudo inverse are sensitive to noise

```
H'(u, v) = 1/H(u, v), H(u, v)!=0
H'(u, v) = 0, otherwise
```

iii) Wiener Filter: (Minimum Mean Square Error Filter)

- optimal trade off between filtering and noise smoothing (real and even)
- minimizes the overall mean square error by:

```
e^2 = F{(f-f')^2}
where, f -> original image
f' -> restored image
E{.} -> mean value of arguments

H(u, v) = H'(u, v)/(|H(u, v)|^2 + (Sn(u, v)/Sf(u, v))
where H(u, v) -> Transform of degradation function
Sn(u, v) -> Power spectrum of the noise
Sf(u, v) -> Power spectrum of the undergraded original image
```

iii) Wiener Filter: removes noise and inputs in the blurring simultaneously

```
H(u, v)=1
W(u, v) = 1 / (1 + Sn(u, v)/Sf(u, v))
W(u, v) = SNR/(1 + SNR)
where, SNR = Sf(u, v)/Sn(u, v)
```

No blur only additive Noise

```
Sn(u, v)=0
W(u, v) = 1/H(u, v)
```

No noise only blur

Limitations of Restoration Filters:

- Not effective when images are restored for the human eye
- Cannot handle the common cause of non-stationary signals and noise
- Cannot handle spatially variant blurring point spread function