TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE
Faculty of Information Technology

Chapter 3. Image Enhancement in Frequency Domain
(Digital Image Processing)

PGS.TS. Hoàng Văn Dũng
Email: dunghv@hcmute.edu.vn

Outline

1. Background
2. Fourier transformation
3. Smoothing (lowpass) filter
4. Sharpening (highpass) filters
5. OpenCV in use

Background

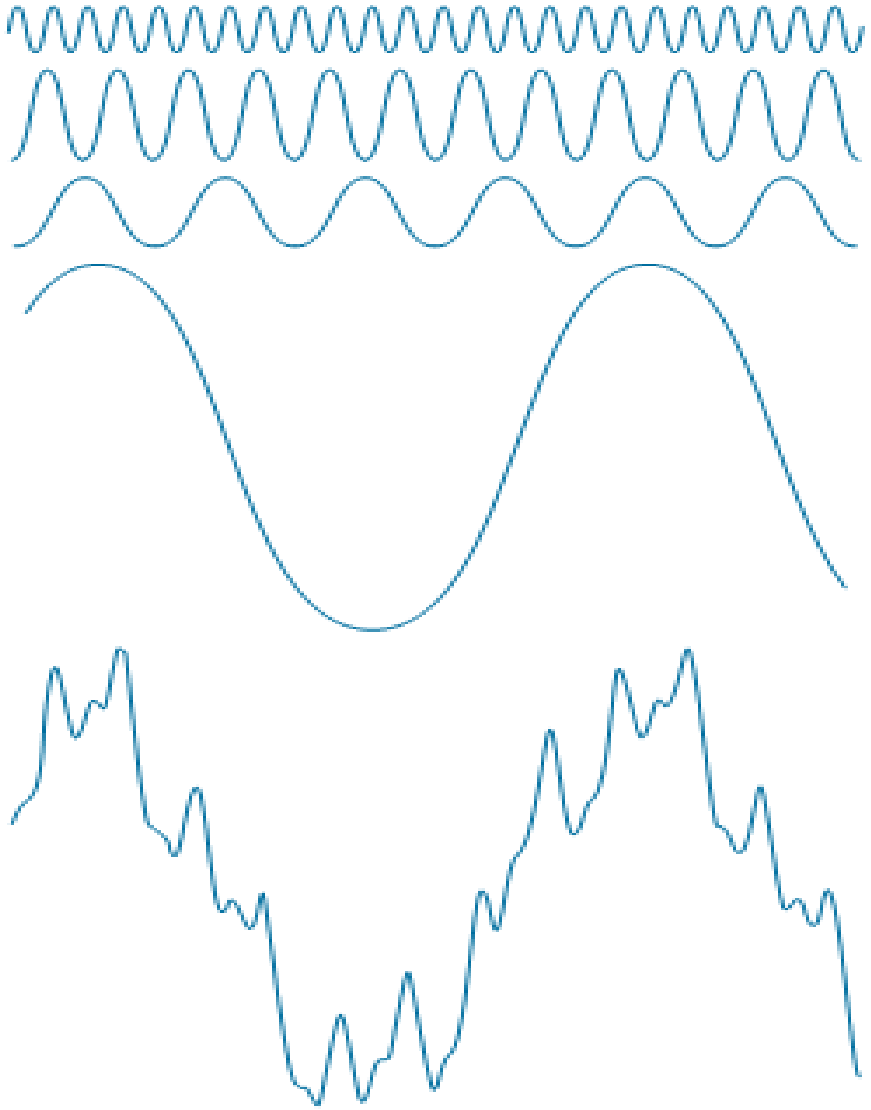
- Any function that periodically repeats itself can be expressed as the sum of sines and/or cosines of different frequencies, each multiplied by a different coefficient (Fourier series).
- Even functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and/or cosines multiplied by a weighting function (Fourier transform).

Fourier transformation

- The **frequency domain** refers to the plane of the two dimensional discrete Fourier transform of an image.

- The purpose of the Fourier transform is to represent a signal as a linear combination of sinusoidal signals of various frequencies.

Example:
The function at the bottom is the sum of the four functions above it.

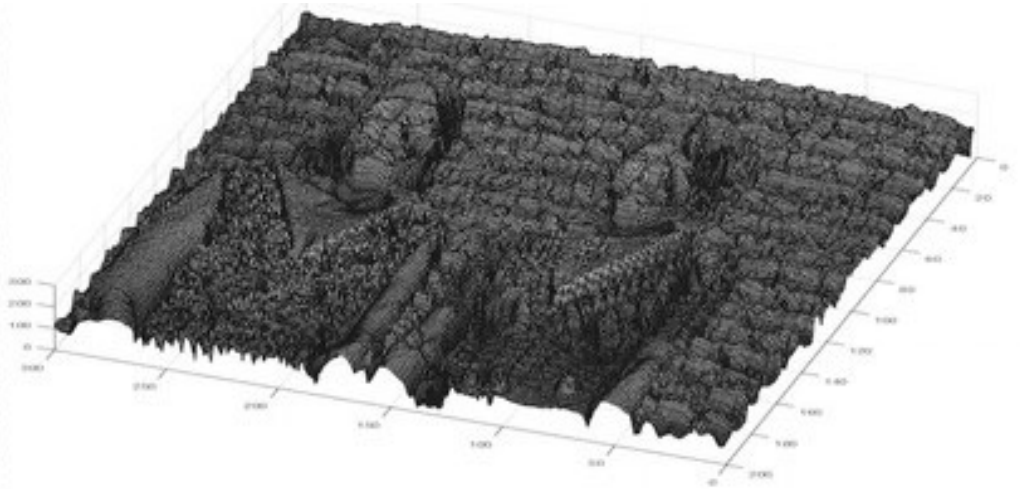


Outline

1. Background
2. Fourier transformation
3. Smoothing (lowpass) filter
4. Sharpening (highpass) filters
5. OpenCV in use

Fourier transformation

- The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent.
- The number of frequencies corresponds to the number of pixels in the spatial domain image, i.e. the image in the spatial and Fourier domain are of the same size.



Fourier transformation

- For a square image of size N×N, the two-dimensional DFT is given by:

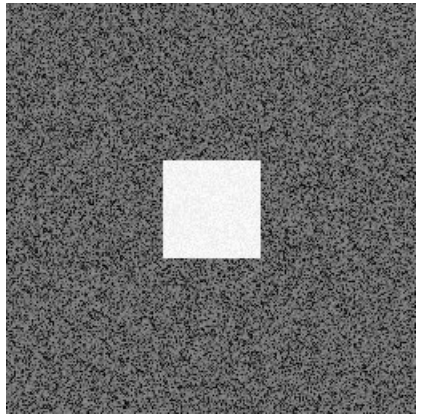
$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-\imath 2\pi \left(\frac{ki}{N} + \frac{lj}{N}\right)}$$

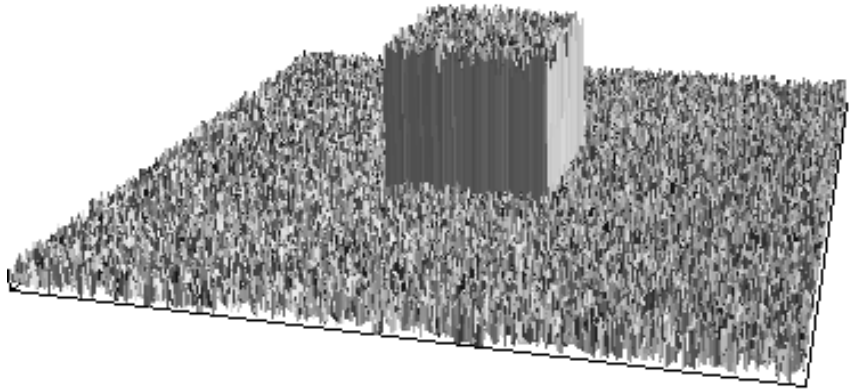
where *f(a,b)* is the image in the spatial domain and the exponential term is the basis function corresponding to each point *F(k,l)* in the Fourier space.

The equation can be interpreted as: the value of each point *F(k,l)* is obtained by multiplying the spatial image with the corresponding base function and summing the result.

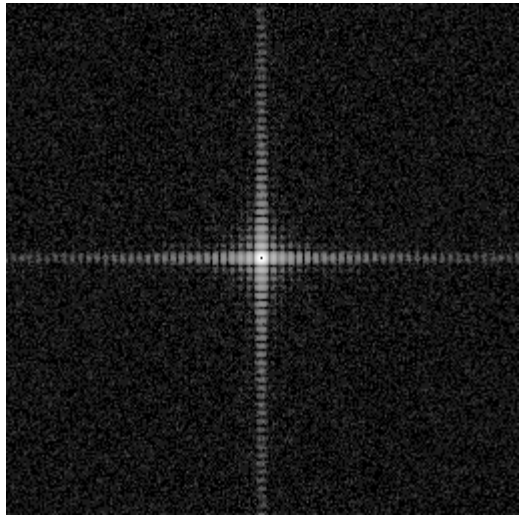
Fourier transformation

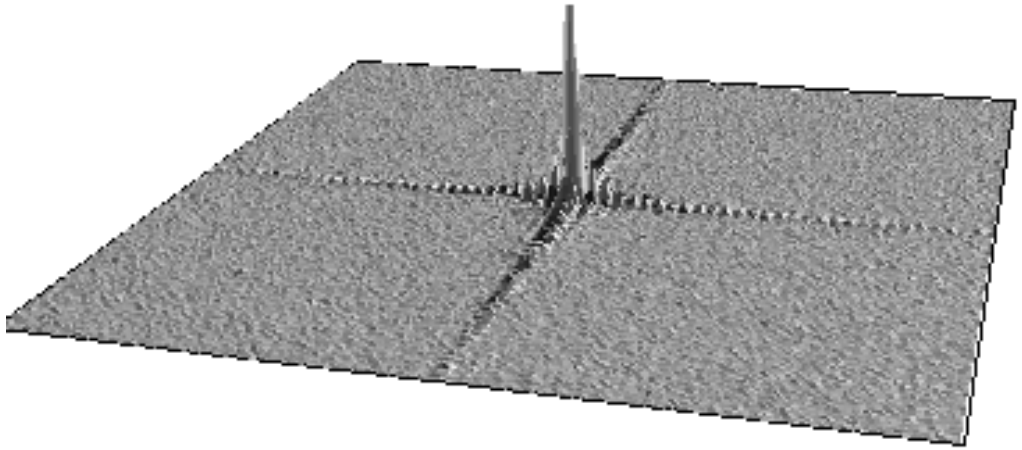
- The original signal is a rectangular pulse with added noise





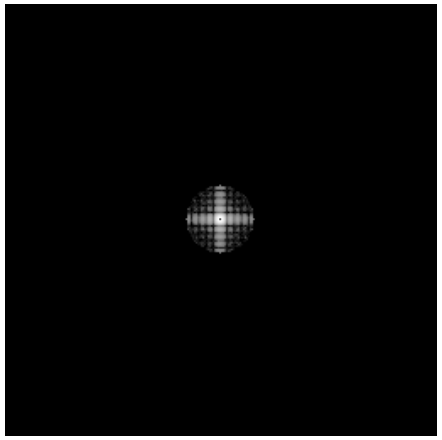
- Fourier transform of the rectangular pulse is the two dimensional equivalent of the sync function, the Fourier transform of white noise is a constant.

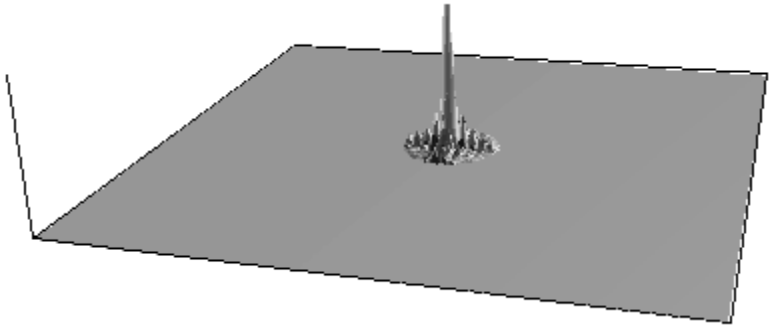




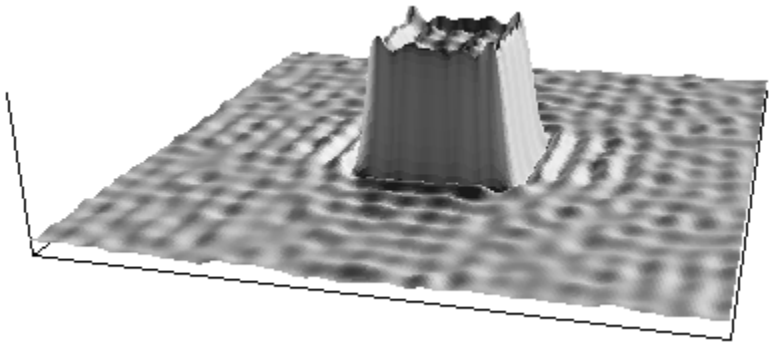
Fourier transformation

- Applying a low pass filter in the frequency domain means zeroing all frequency components above a cut-off frequency.



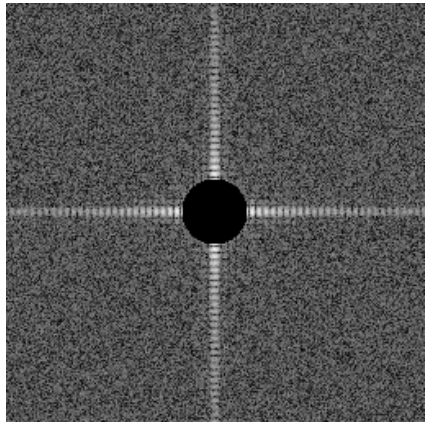


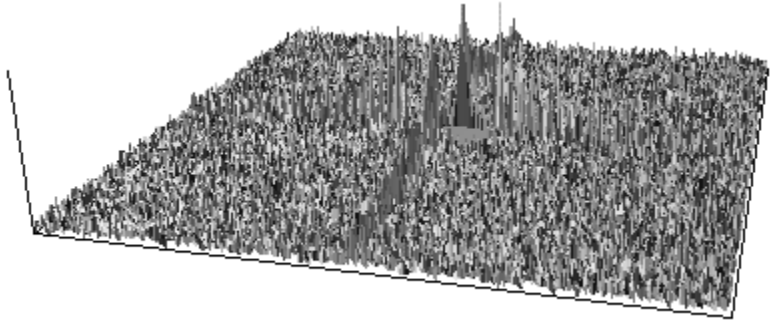
- The result transformed back into the spatial domain



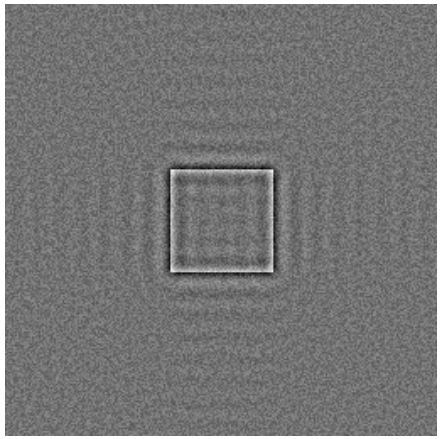
Fourier transformation

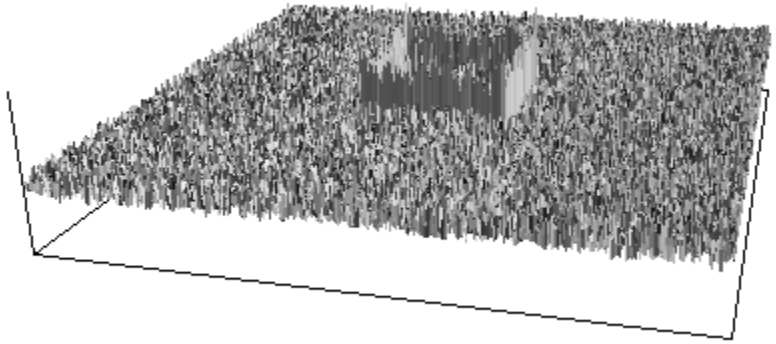
- Applying a high pass filter frequency domain is the opposite to the low pass filter





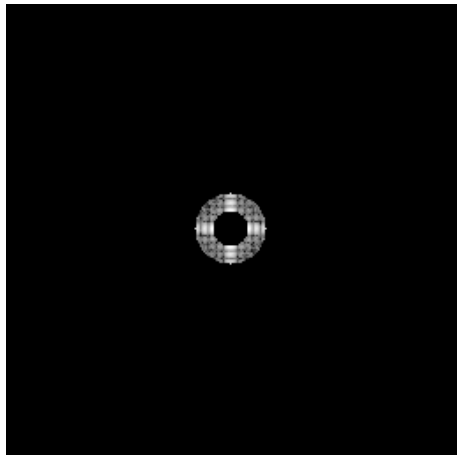
- Transformed back into the spatial domain

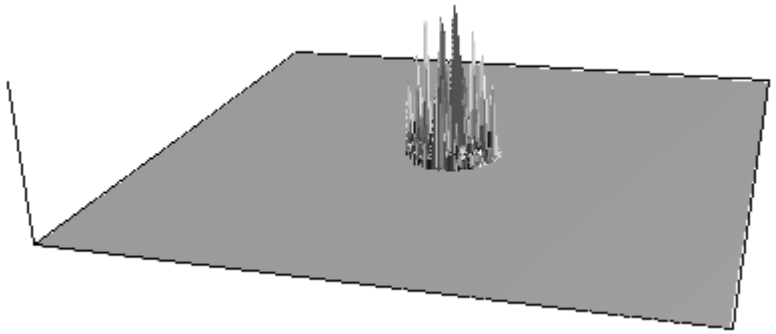




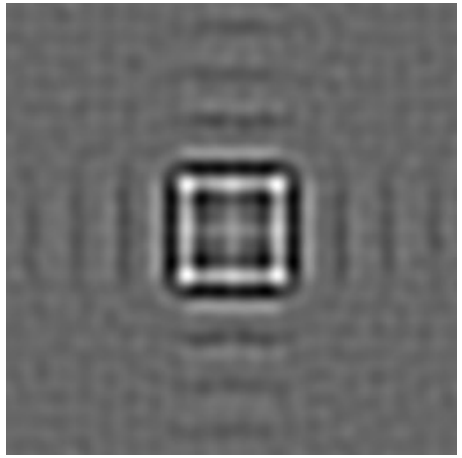
Fourier transformation

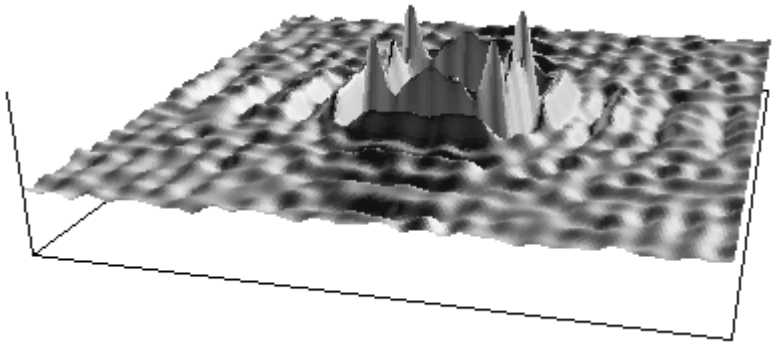
- Applying a band pass filter, frequency domain.





- Transformed back into the spatial domain

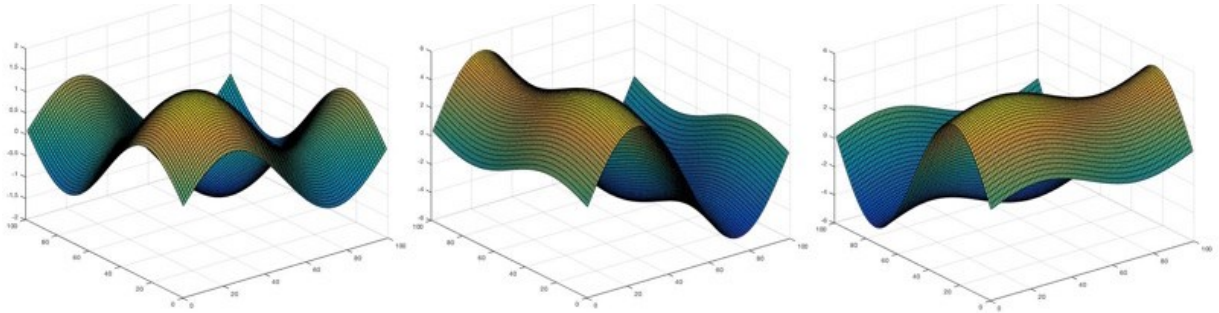




Fourier transformation

- Two-dimensional sine waves are written as $z = asin(hx + ky)$

where *x* and *y* give the coordinates for points on the "sheet", *z* is the height, or intensity, of the wave at that point, *a* gives the amplitude (the maximum height) of the wave, and *h* and *k* give the number of times the wave repeats in the *x* and *y* directions respectively (they are the *x* and *y* frequencies).



waves

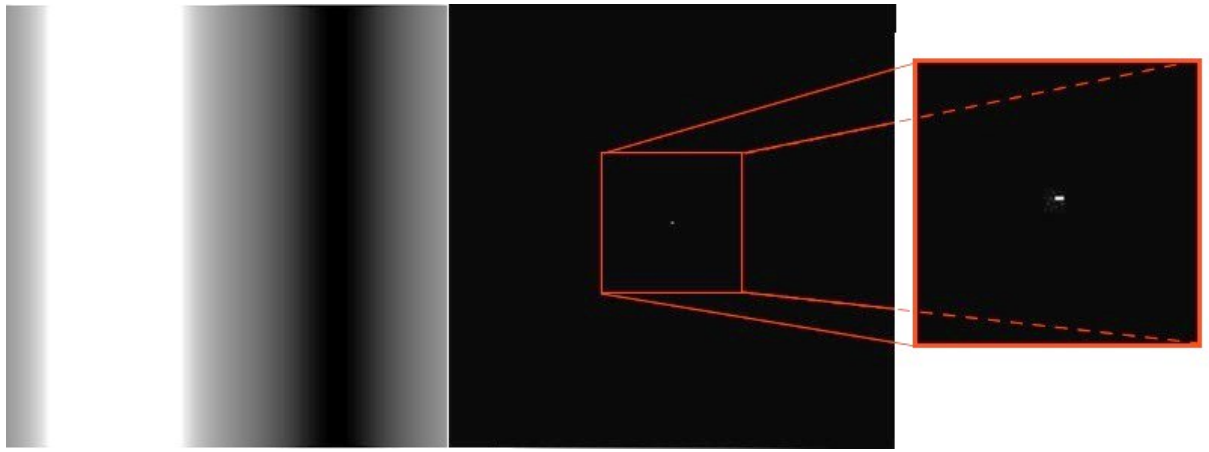
$sin(x)+sin(y)$,

$5sin(x)+sin(y)$,

$sin(x)+5sin(y)$

Fourier transformation

- Each pixel in the Fourier transform has a coordinate (h, k) representing the contribution of the sine wave with x-frequency h, and y-frequency k in the Fourier transform.
- The center point represents the (0,0) wave – a flat plane with no ripples – and its intensity (its brightness in color in the grey scale) is the average value of the pixels in the image.



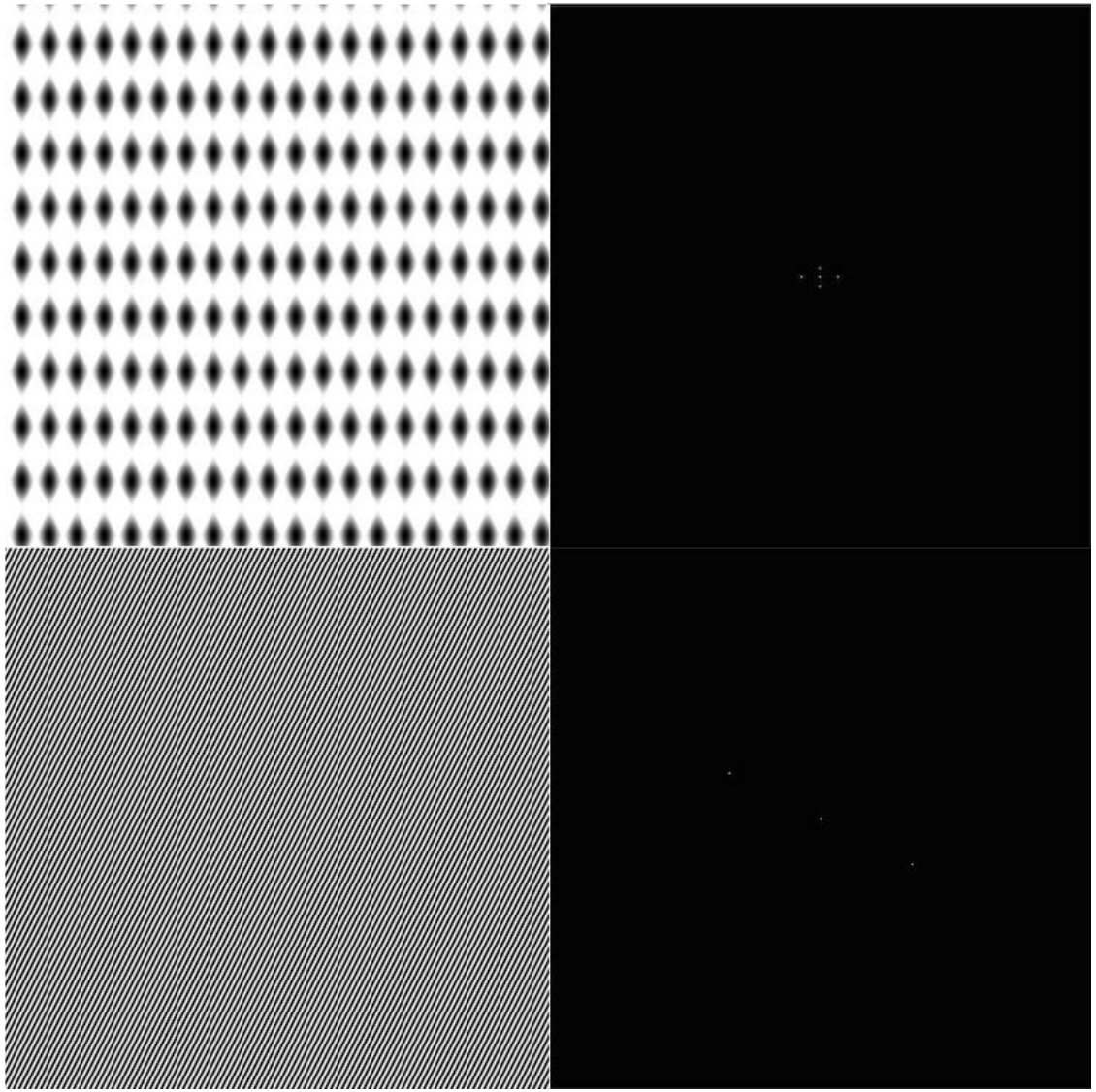
This is the two-dimensional wave *sin(x)* (which we saw earlier) viewed as a grayscale image

The wave *sin(x)* represented as a grayscale image, and the Fourier transform of that image.

Fourier transformation

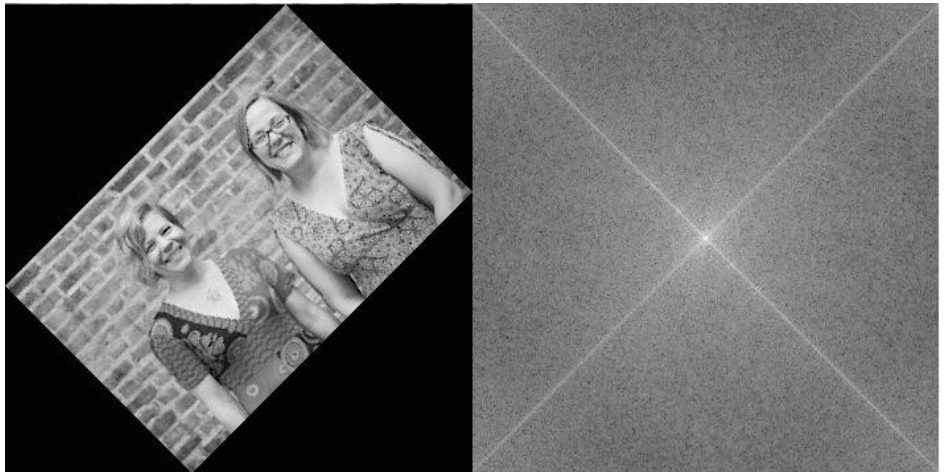
- Top: The wave sin(20x)+sin(10y) and its Fourier transform, showing two pairs of bright pixels (at the coordinates (20,0) and (0,10) and their reflections) representing these contributions of these two waves.

- Bottom: The wave sin(100x+50y) and its Fourier transform, showing just the pair of bright pixels at the coordinates (100,50) and its reflection.



Fourier transformation

- *Plus* editors and their Fourier transform, showing a series of contributions of vertical wave represented by bright points along the vertical axis.



The *Plus* editors rotated by 45 degrees, and their Fourier transform.

Fourier transformation

- The one-dimensional Fourier transform and its inverse
  - Fourier transform (continuous case)

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx \quad \text{where } j = \sqrt{-1}$$

  - Inverse Fourier transform:

$$e^{j\theta} = \cos\theta + j\sin\theta$$

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$

- The two-dimensional Fourier transform and its inverse
  - Fourier transform (continuous case)

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-j2\pi(ux+vy)} dxdy$$

  - Inverse Fourier transform:

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v) e^{j2\pi(ux+vy)} dudv$$

1D DFT and IDFT

- The one-dimensional Fourier transform and its inverse
  - Fourier transform (discrete case- DFT)

$$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{-j2\pi ux/M} \quad \text{for } u = 0,1,2,\ldots,M-1$$

  - Inverse Fourier transform (IDFT):

$$f(x) = \sum_{u=0}^{M-1} F(u)e^{j2\pi ux/M} \quad \text{for } x = 0,1,2,\ldots,M-1$$

Về bản chất, chuyển đổi Fourier sử dụng trong số phức, nhưng trong xử lý ảnh, ta xem phần ảo là 0

1D DFT and IDFT

- Since $e^{j\theta} = \cos\theta + j\sin\theta$ and the fact $\cos(-\theta) = \cos\theta$ then discrete Fourier transform can be redefined

$$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)[\cos 2\pi ux/M - j\sin 2\pi ux/M]$$

$$\text{for } u = 0,1,2,\ldots,M-1$$

  - Frequency (time) domain: the domain (values of u) over which the values of $F(u)$ range; because u determines the frequency of the components of the transform.
  - Frequency (time) component: each of the M terms of $F(u)$.

2D DFT and IDFT

- The two-dimensional Fourier transform and its inverse
  - Fourier transform (discrete case) DTC

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

$$\text{for } u = 0,1,2,\ldots,M-1, v = 0,1,2,\ldots,N-1$$

  - Inverse Fourier transform:

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$$

$$\text{for } x = 0,1,2,\ldots,M-1, y = 0,1,2,\ldots,N-1$$

    - u, v : the transform or frequency variables
    - x, y : the spatial or image variables

2D DFT and IDFT

- We define the Fourier spectrum, phase angle, and power spectrum as follows:

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{\frac{1}{2}} \quad \text{(spectrum)}$$

$$\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right] \quad \text{(phase angle)}$$

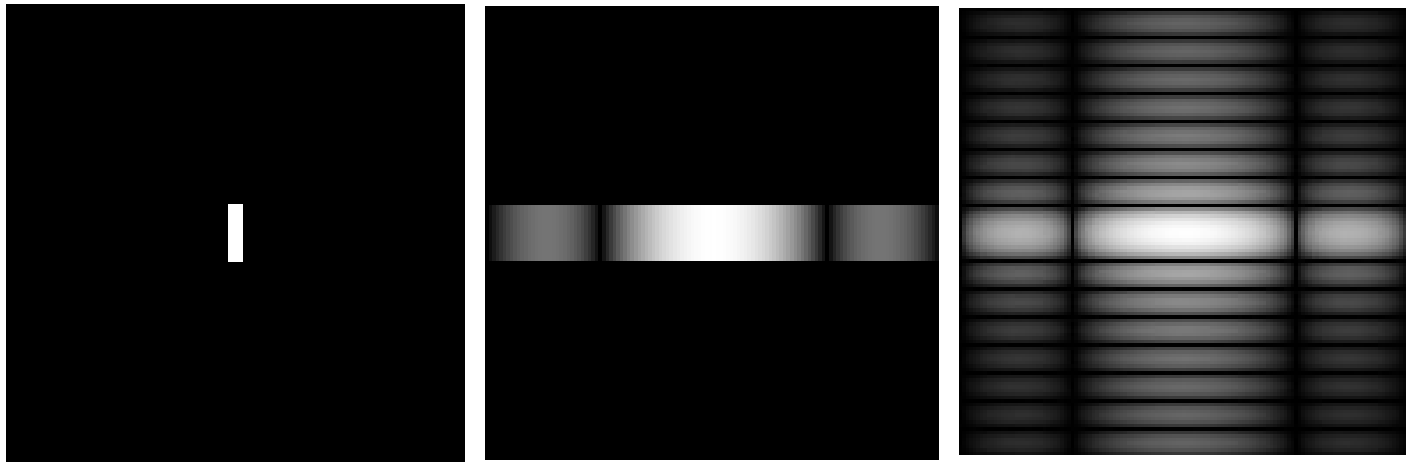
$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v) \quad \text{(power spectrum)}$$

  - $R(u,v)$: the real part of $F(u,v)$
  - $I(u,v)$: the imaginary part of $F(u,v)$

2D DFT and IDFT

The 2D DFT **F(u,v)** can be obtained by

1. taking the 1D DFT of every row of image **f(x,y)**, **F(u,y)**,
2. taking the 1D DFT of every column of **F(u,y)**

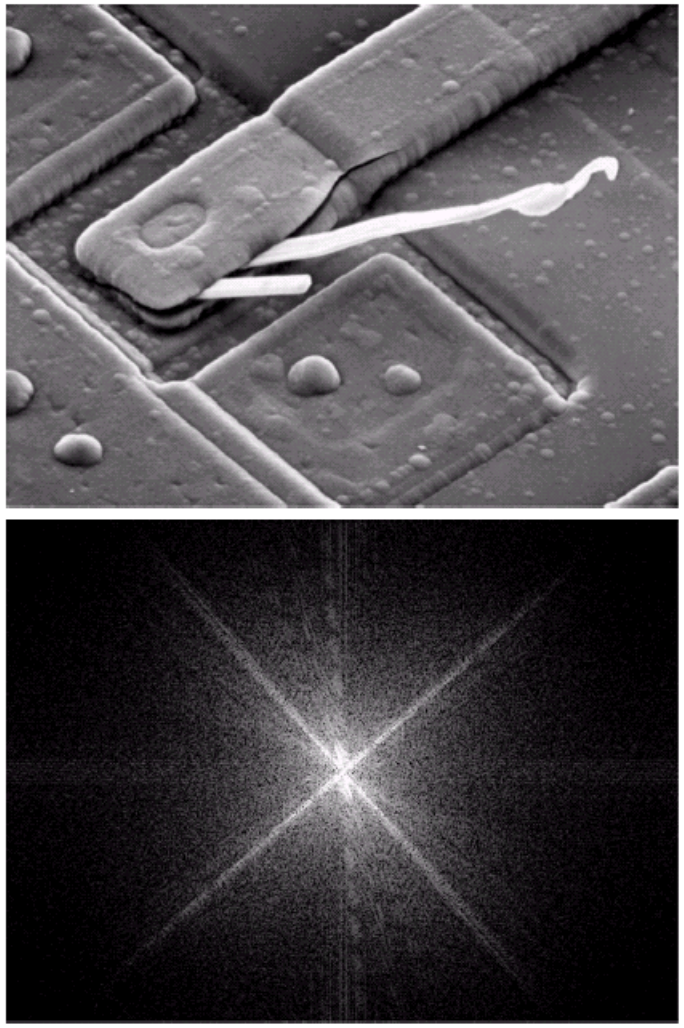


(a)f(x,y) (b)F(u,y) (c)F(u,v)

Outline

1. Background
2. Fourier transformation
3. Filtering in Frequency Domain
4. Smoothing (lowpass) filter
5. Sharpening (highpass) filters
6. OpenCV in use

Filtering in Frequency Domain



a
b

FIGURE 4.4
(a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

Filtering in Frequency Domain

- Basics of filtering in the frequency domain

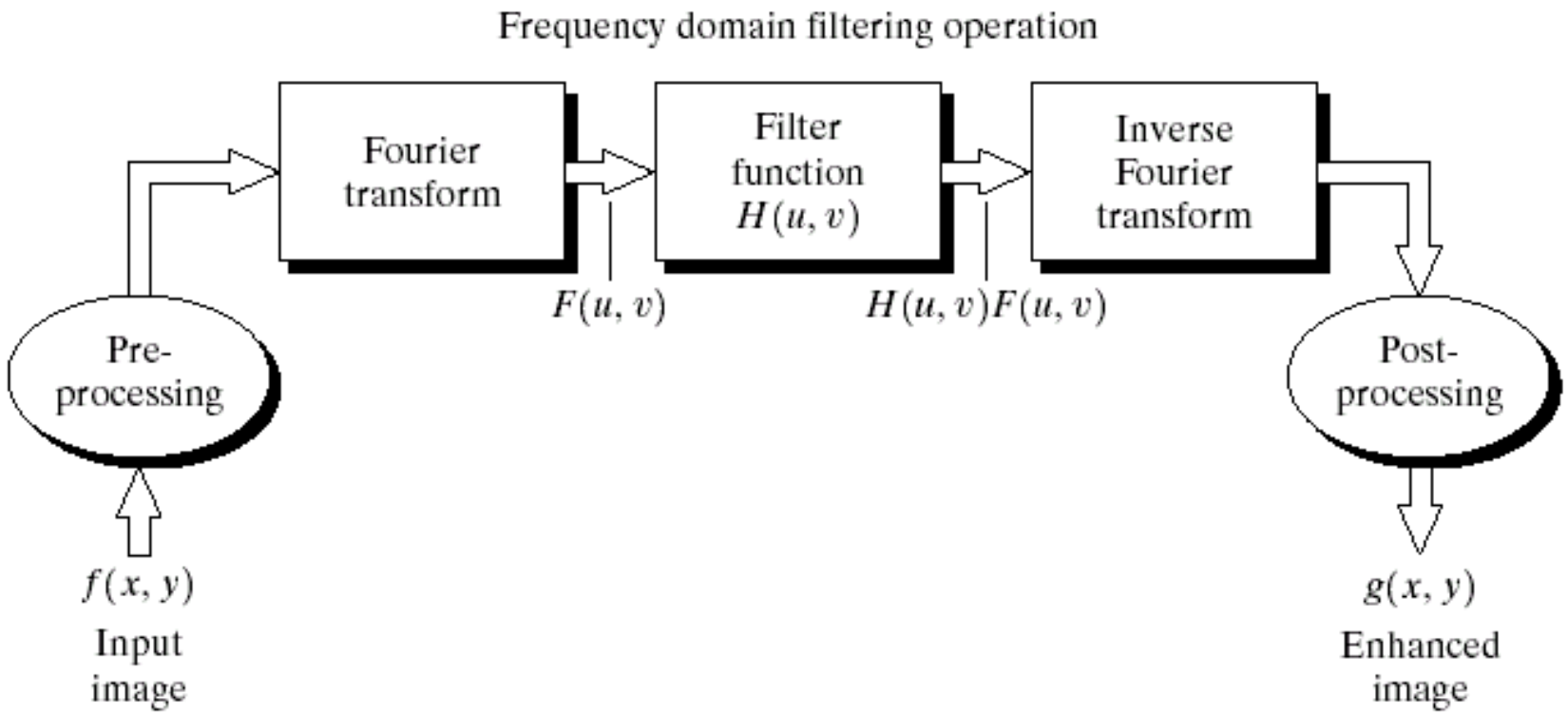


FIGURE 4.5 Basic steps for filtering in the frequency domain.

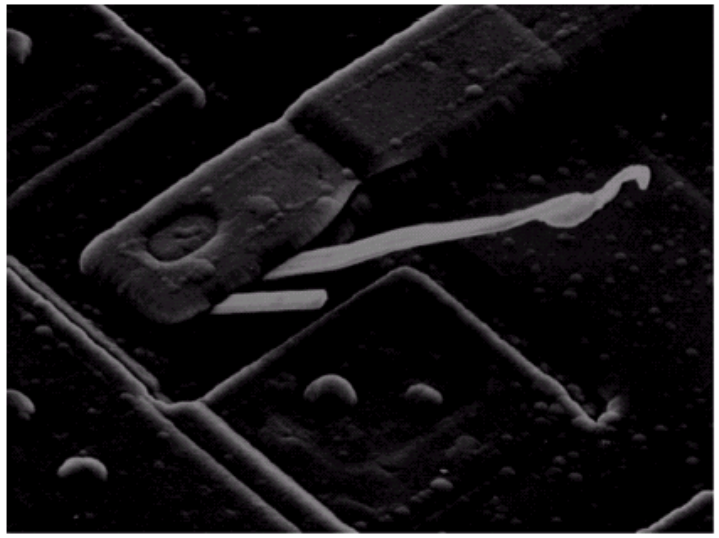
Filtering in Frequency Domain

- Some basic filters and their functions
  - Multiply all values of F(u,v) by the filter function (notch filter):

$$H(u,v) = \begin{cases} 0 & \text{if } (u,v) = (M/2, N/2) \\ 1 & \text{otherwise.} \end{cases}$$

  - All this filter would do is set F(0,0) to zero (force the average value of an image to zero) and leave all frequency components of the Fourier transform untouched.

FIGURE 4.6 Result of filtering the image in Fig. 4.4(a) with a notch filter that set to 0 the $F(0, 0)$ term in the Fourier transform.

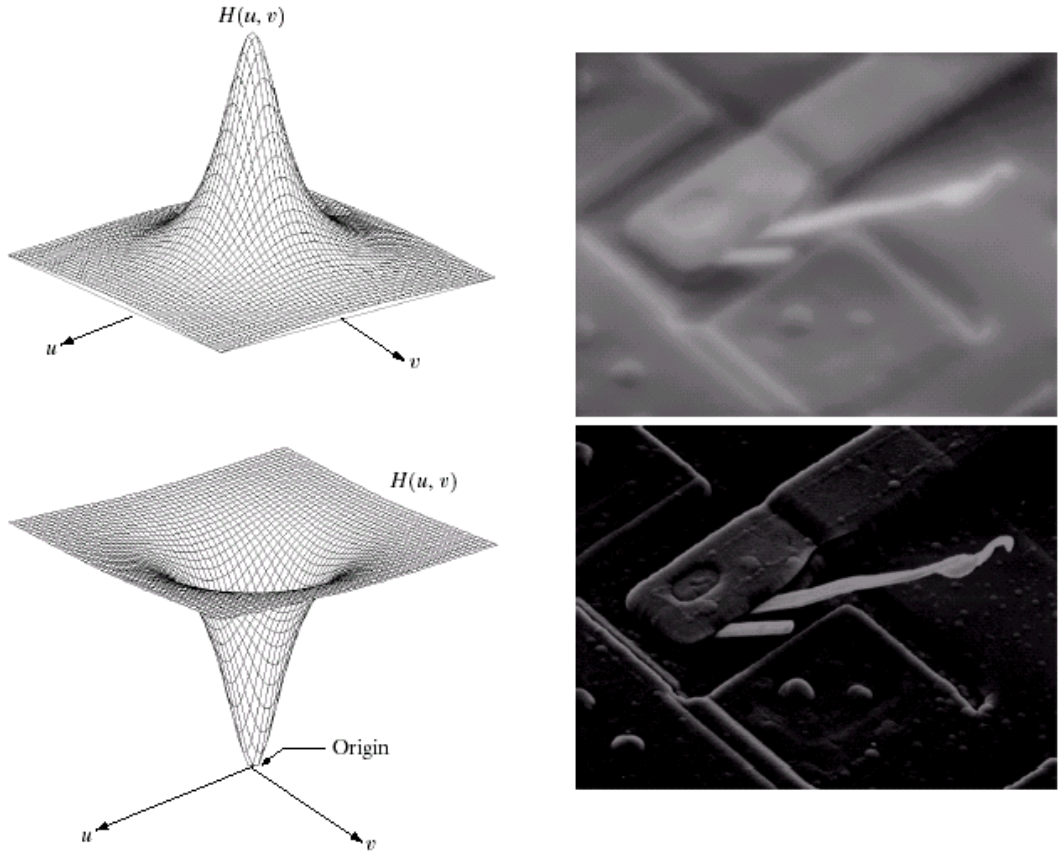


Filtering in Frequency Domain

- Some basic filters and their functions

Lowpass filter

Highpass filter



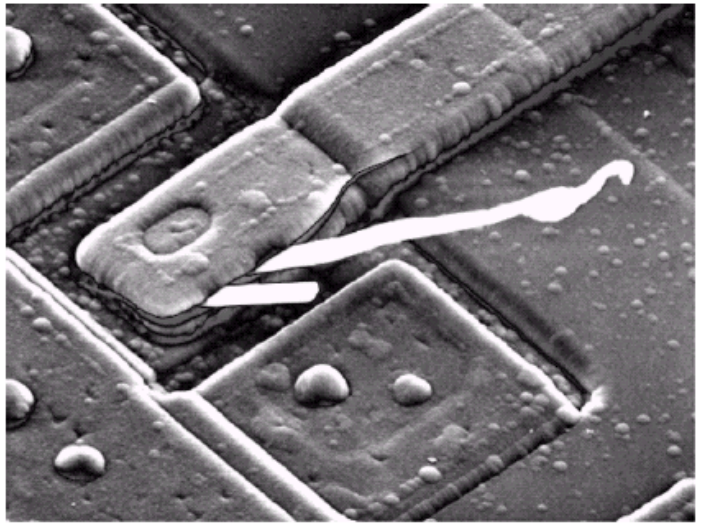
a b
c d

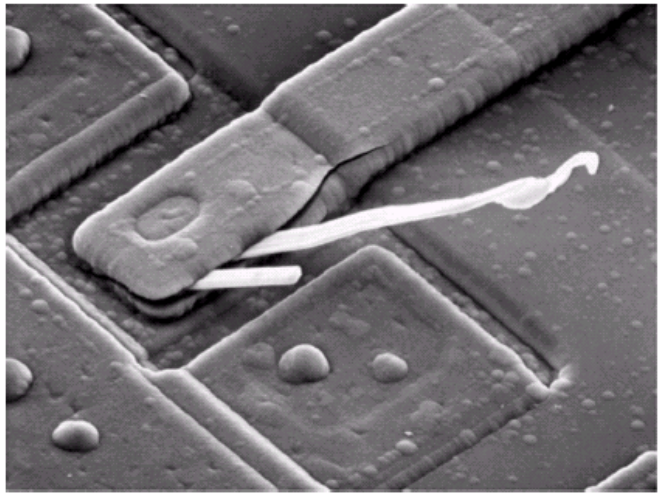
FIGURE 4.7 (a) A two-dimensional lowpass filter function. (b) Result of lowpass filtering the image in Fig. 4.4(a). (c) A two-dimensional highpass filter function. (d) Result of highpass filtering the image in Fig. 4.4(a).

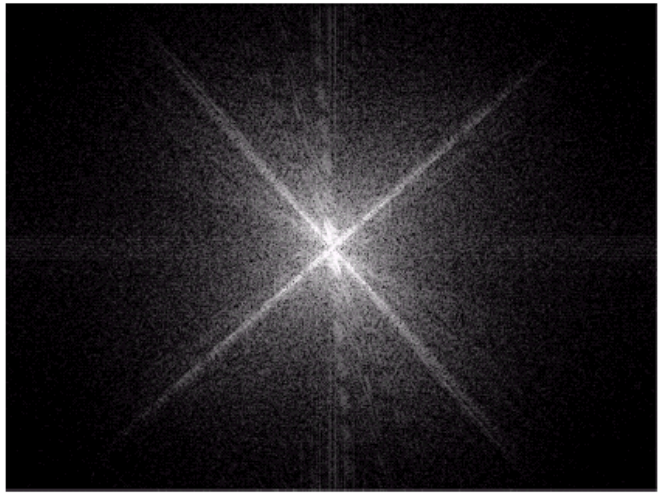
Filtering in Frequency Domain

- Some basic filters and their functions

FIGURE 4.8
Result of highpass filtering the image in Fig. 4.4(a) with the filter in Fig. 4.7(c), modified by adding a constant of one-half the filter height to the filter function. Compare with Fig. 4.4(a).







a
b

FIGURE 4.
(a) SEM i
a damage
integrated
(b) Fourie
spectrum
(Original
courtesy o
M. Hudak
Brockhou
Institute f
Materials
Research,
McMaster
University
Hamilton,
Ontario, C

Filtering in Frequency Domain

Correspondence between filtering in the spatial and frequency domain

- **Convolution theorem:**
  - The discrete convolution of two functions $f(x,y)$ and $h(x,y)$ of size MXN is defined as

$$f(x,y)*h(x,y)=\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n)$$

  - Let $F(u,v)$ and $H(u,v)$ denote the Fourier transforms of $f(x,y)$ and $h(x,y)$, then

$$f(x,y)*h(x,y) \Leftrightarrow F(u,v)H(u,v) \qquad \text{Eq. (4.2-31)}$$

$$f(x,y)\,h(x,y) \Leftrightarrow F(u,v)*H(u,v) \qquad \text{Eq. (4.2-32)}$$

Filtering in Frequency Domain

Correspondence between filtering in the spatial and frequency domain

- $A\delta(x - xo, y - y_0)$: an impulse function of strength A, located at coordinates (x_0,y_0)

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} s(x,y)A\delta(x - xo, y - y_0) = As(x_0, y_0)$$

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} s(x,y)\delta(x,y) = s(0,0)$$

where $\delta(x,y)$: a unit impulse located at the origin

- The Fourier transform of a unit impulse at the origin (Eq. (4.2-35))

$$F(u,v)\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\delta(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} = \frac{1}{MN}$$

Filtering in Frequency Domain

Correspondence between filtering in the spatial and frequency domain

- Let $f(x,y) = \delta(x,y)$, then the convolution (Eq. (4.2-36))

$$f(x,y)*h(x,y)=\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\delta(m,n)h(x-m,y-n) \;=\frac{1}{MN}h(x,y)$$

- Combine Eqs. (4.2-35) (4.2-36) with Eq. (4.2-31), we obtain

$$f(x,y)*h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

$$\delta(x,y)*h(x,y) \Leftrightarrow \Im[\delta(x,y)]H(u,v)$$

$$\frac{1}{MN}h(x,y) \qquad \frac{1}{MN}H(u,v)$$

$$h(x,y) \Leftrightarrow H(u,v)$$

Filtering in Frequency Domain

Correspondence between filtering in the spatial and frequency domain

- Let $H(u)$ denote a frequency domain, Gaussian filter function given the equation

$$H(u) = Ae^{-u^2/2\sigma^2}$$

where σ the standard deviation of the Gaussian curve.

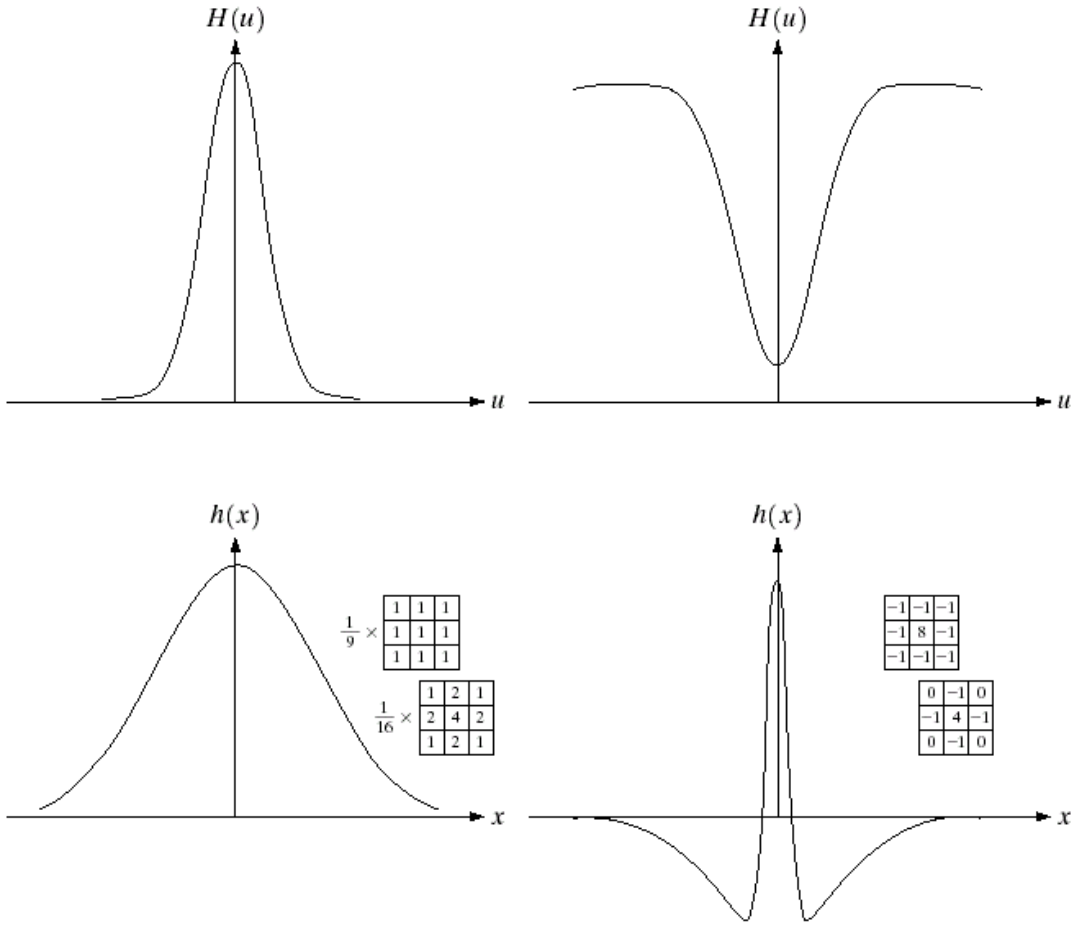
- The corresponding filter in the spatial domain is

$$h(x) = \sqrt{2\pi}\sigma A e^{-2\pi^2\sigma^2 x^2}$$

- Note: Both the forward and inverse Fourier transforms of a Gaussian function are real Gaussian functions.

Filtering in Frequency Domain

- Correspondence between filtering in the spatial and frequency domain



a b
c d

FIGURE 4.9
(a) Gaussian frequency domain lowpass filter.
(b) Gaussian frequency domain highpass filter.
(c) Corresponding lowpass spatial filter.
(d) Corresponding highpass spatial filter. The masks shown are used in Chapter 3 for lowpass and highpass filtering.

Smoothing Frequency Domain Filters

- The basic model for filtering in the frequency domain

$$G(u,v) = H(u,v)F(u,v)$$

where $F(u,v)$: the Fourier transform of an image to be smoothed

$H(u,v)$: a filter transfer function

- Smoothing is fundamentally a lowpass operation in the frequency domain.

- There are several standard forms of lowpass filters (LPF).
  - Ideal lowpass filter
  - Butterworth lowpass filter
  - Gaussian lowpass filter

Outline

1. Background
2. Fourier transformation
3. Filtering in Frequency Domain
4. Smoothing (lowpass) filter
5. Sharpening (highpass) filters
6. OpenCV in use

Smooth filtering

- Obtained images are low-frequency images (with little grayscale change) or high-frequency images (large grayscale change: subject boundary).
- => H (u, v) filter required which reduces high frequencies while passing low frequencies (low pass filter) => smooth images

Ideal Lowpass Filters

- The simplest lowpass filter is a filter that "cuts off" all high-frequency components of the Fourier transform that are at a distance greater than a specified distance D_0 from the origin of the transform.
- The transfer function of an ideal lowpass filter

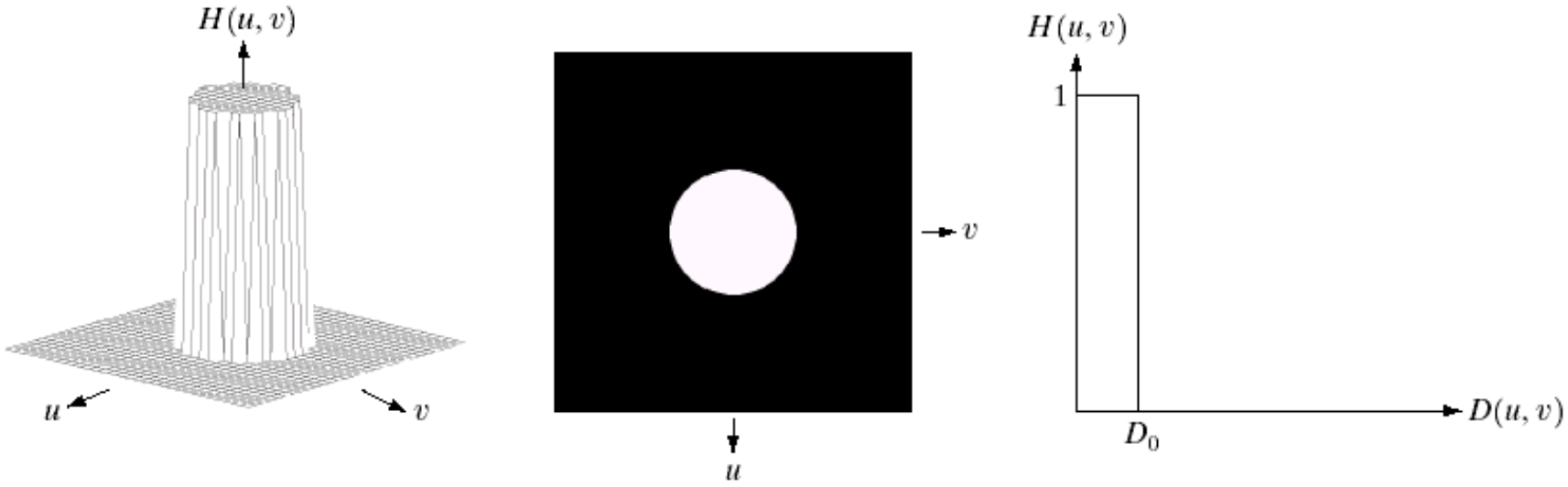
$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

where $D(u,v)$: the distance from point (u,v) to the center of ther frequency rectangle

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

Ideal Lowpass Filters

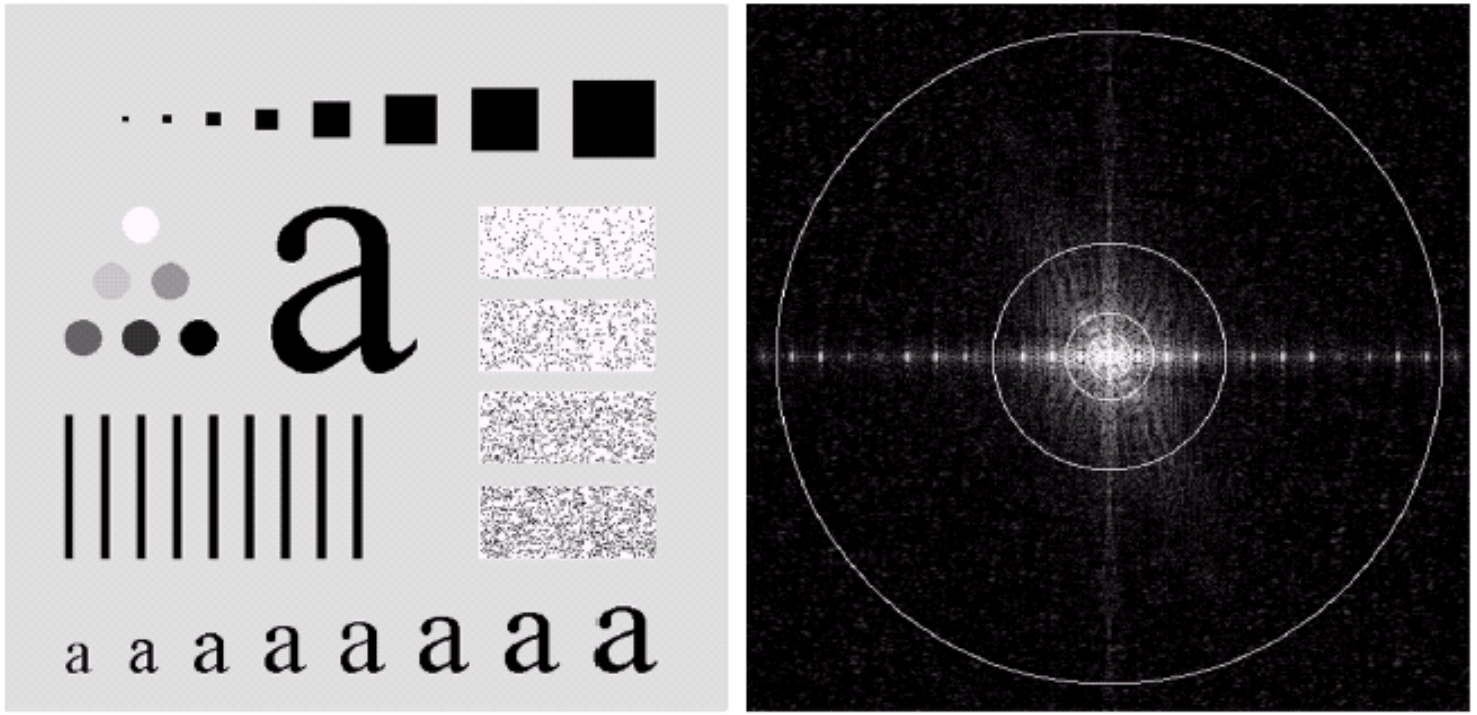
- Example:



a b c

FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Ideal Lowpass Filters

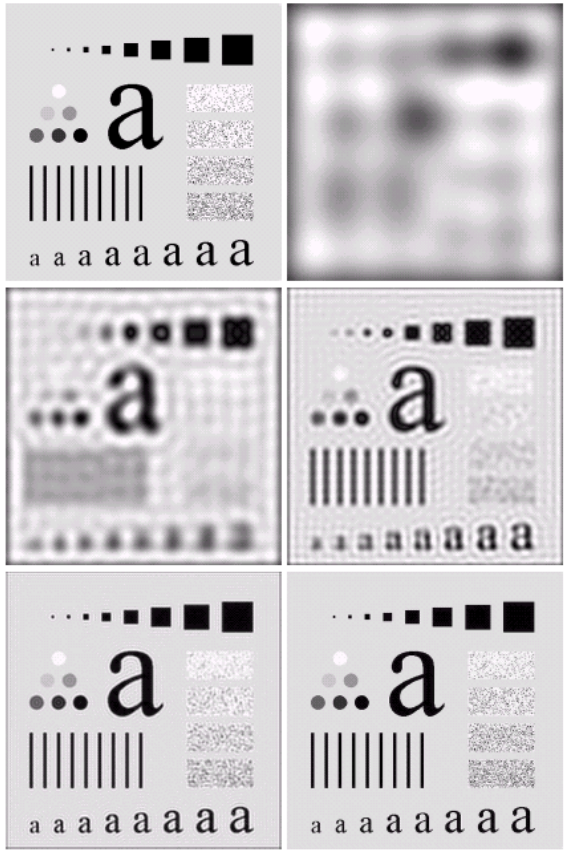


a b

FIGURE 4.11 (a) An image of size 500 × 500 pixels and (b) its Fourier spectrum. The superimposed circles have radii values of 5, 15, 30, 80, and 230, which enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power, respectively.

Ideal Lowpass Filters

- Example



a b
c d
e f

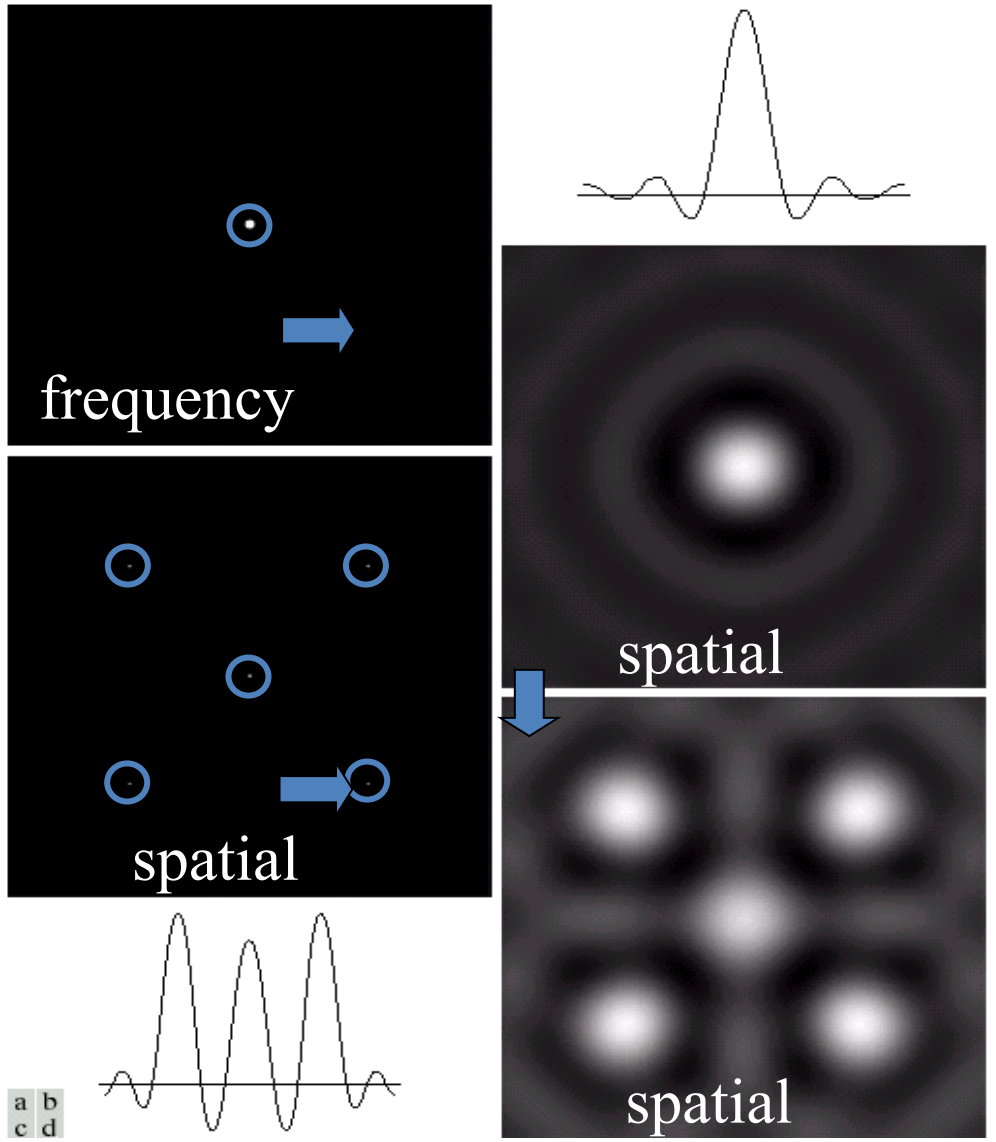
FIGURE 4.12 (a) Original image. (b)–(f) Results of ideal lowpass filtering with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). The power removed by these filters was 8, 5.4, 3.6, 2, and 0.5% of the total, respectively.

Ideal Lowpass Filters

- Example

Figure 4.13 (a) A frequency-domain ILPF of radius 5. (b) Corresponding spatial filter. (c) Five impulses in the spatial domain, simulating the values of five pixels. (d) Convolution of (b) and (c) in the spatial domain.

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

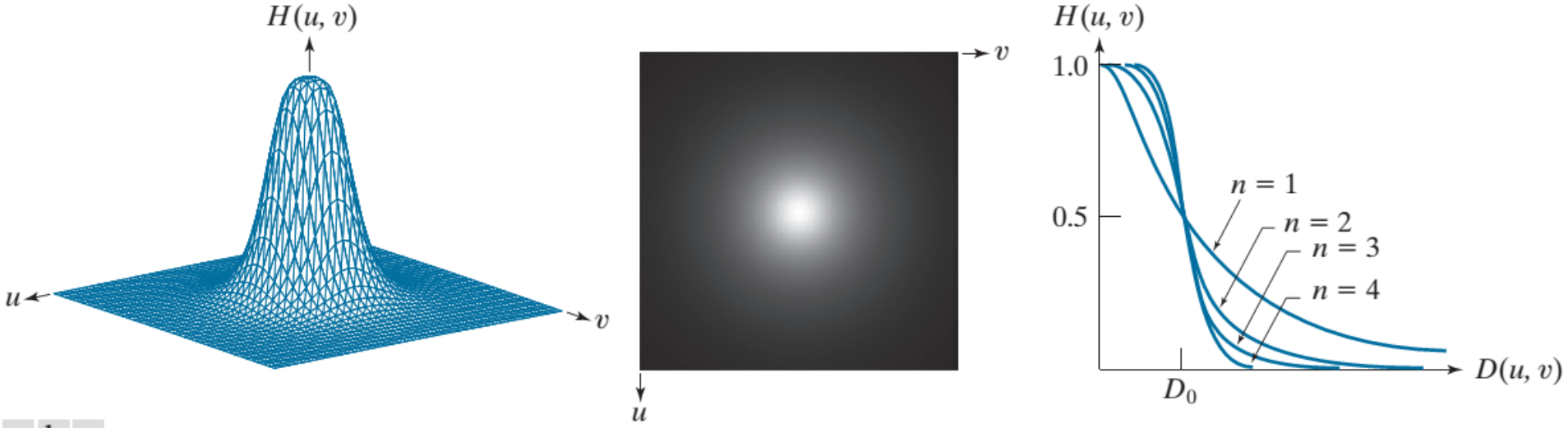


Butterworth Lowpass Filter

- Butterworth lowpass filter (BLPF) with order n

$$H(u,v) = \frac{1}{1+\left[D(u,v)/D_0\right]^{2n}}$$

in $G(u,v) = H(u,v)F(u,v)$



a b c

FIGURE 4.45 (a) Perspective plot of a Butterworth lowpass-filter transfer function. (b) Function displayed as an image. (c) Radial cross sections of BLPFs of orders 1 through 4.

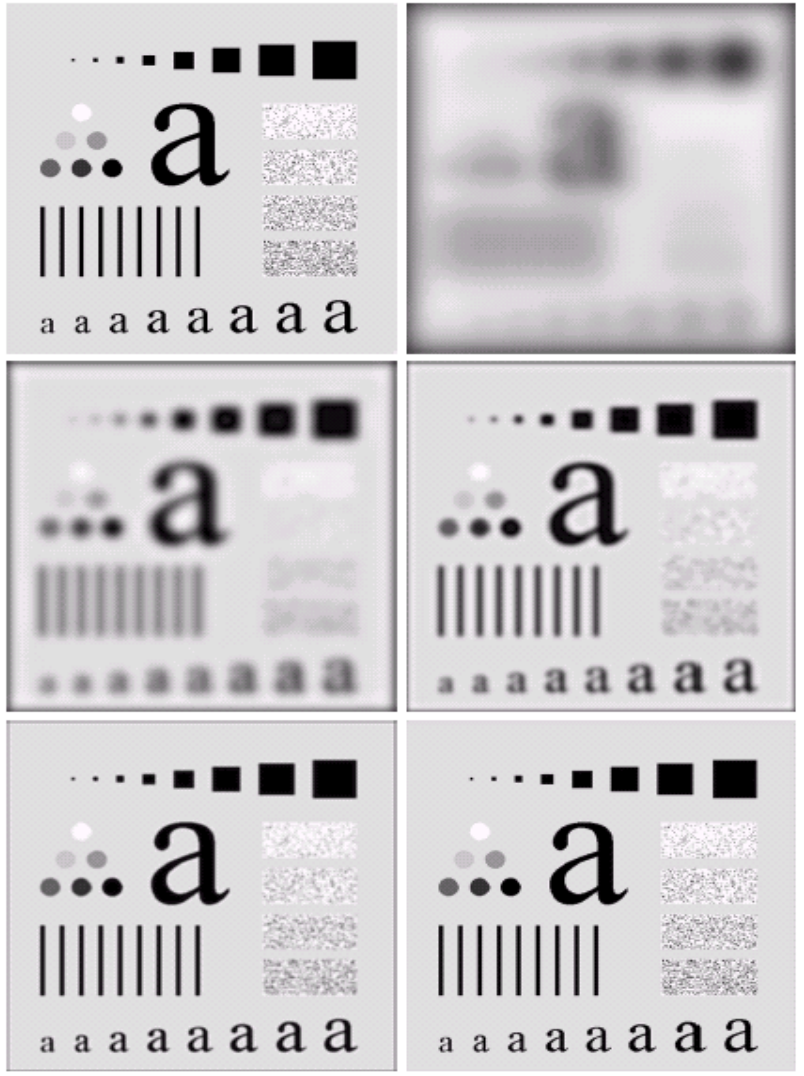
Butterworth Lowpass Filter

- Example

$$H(u,v) = \frac{1}{1+\left[D(u,v)/D_0\right]^{2n}}$$

n=2

D_0=5, 15, 30, 80, and 230

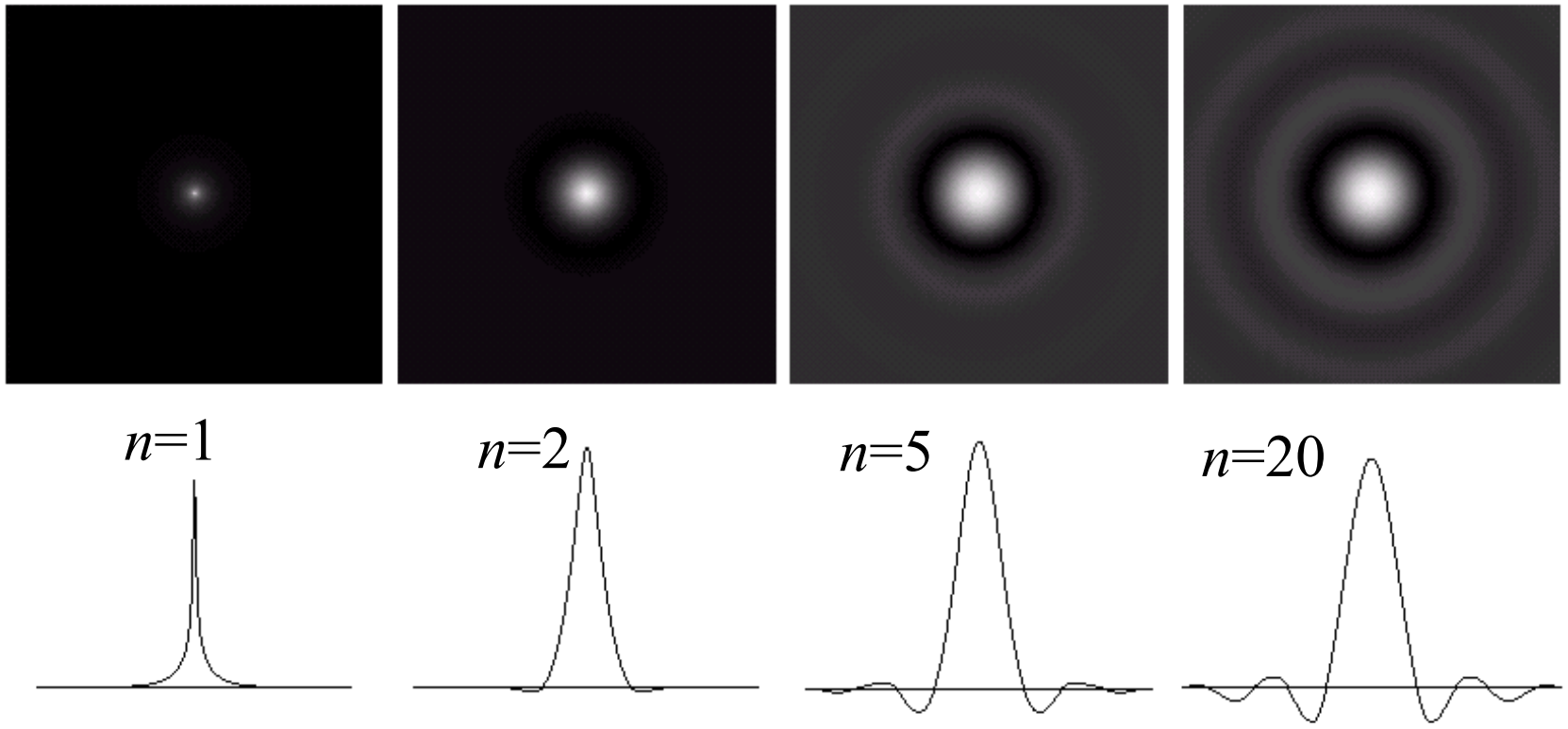


a b c d e f

FIGURE 4.15 (a) Original image. (b)–(f) Results of filtering with BLPFs of order 2, with cutoff frequencies at radii of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Fig. 4.12.

Butterworth Lowpass Filter

- Butterworth lowpass filters spatial representation



a b c d

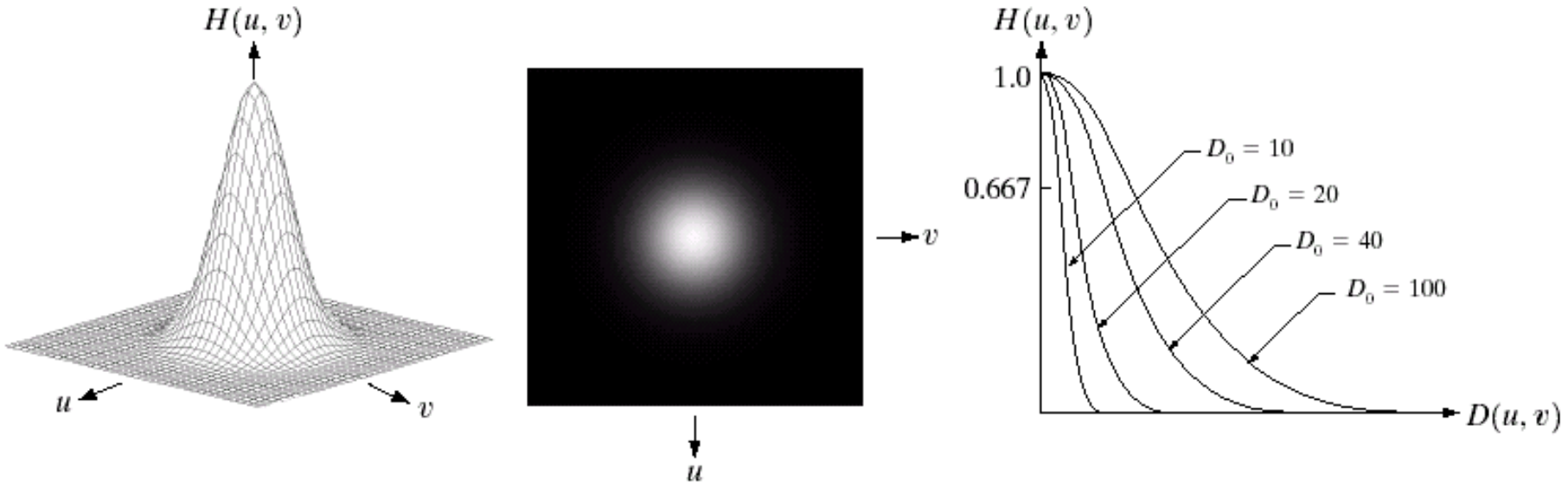
FIGURE 4.16 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding gray-level profiles through the center of the filters (all filters have a cutoff frequency of 5). Note that ringing increases as a function of filter order.

Gaussian Lowpass Filter

- Gaus kernel

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

in $G(u,v) = H(u,v)F(u,v)$



a b c

FIGURE 4.17 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

Gaussian Lowpass Filter

Example:

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

- Filter with

D_0=5, 15, 30, 80, and 230

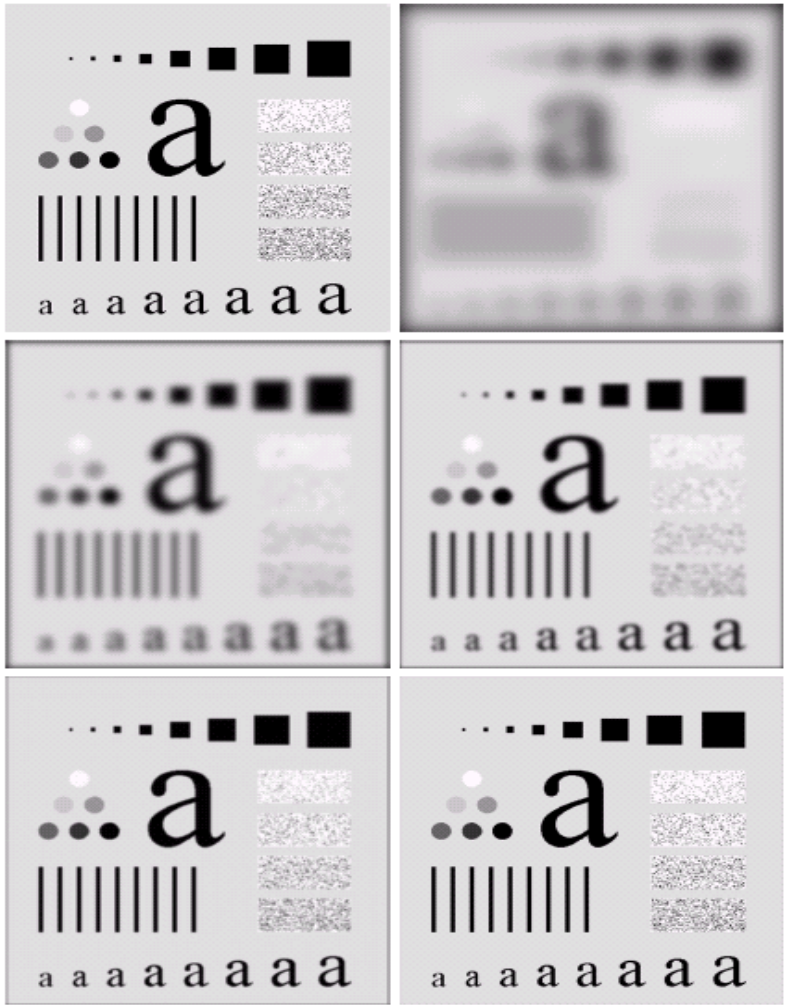


FIGURE 4.18 (a) Original image. (b)–(f) Results of filtering with Gaussian lowpass filters with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Figs. 4.12 and 4.15.

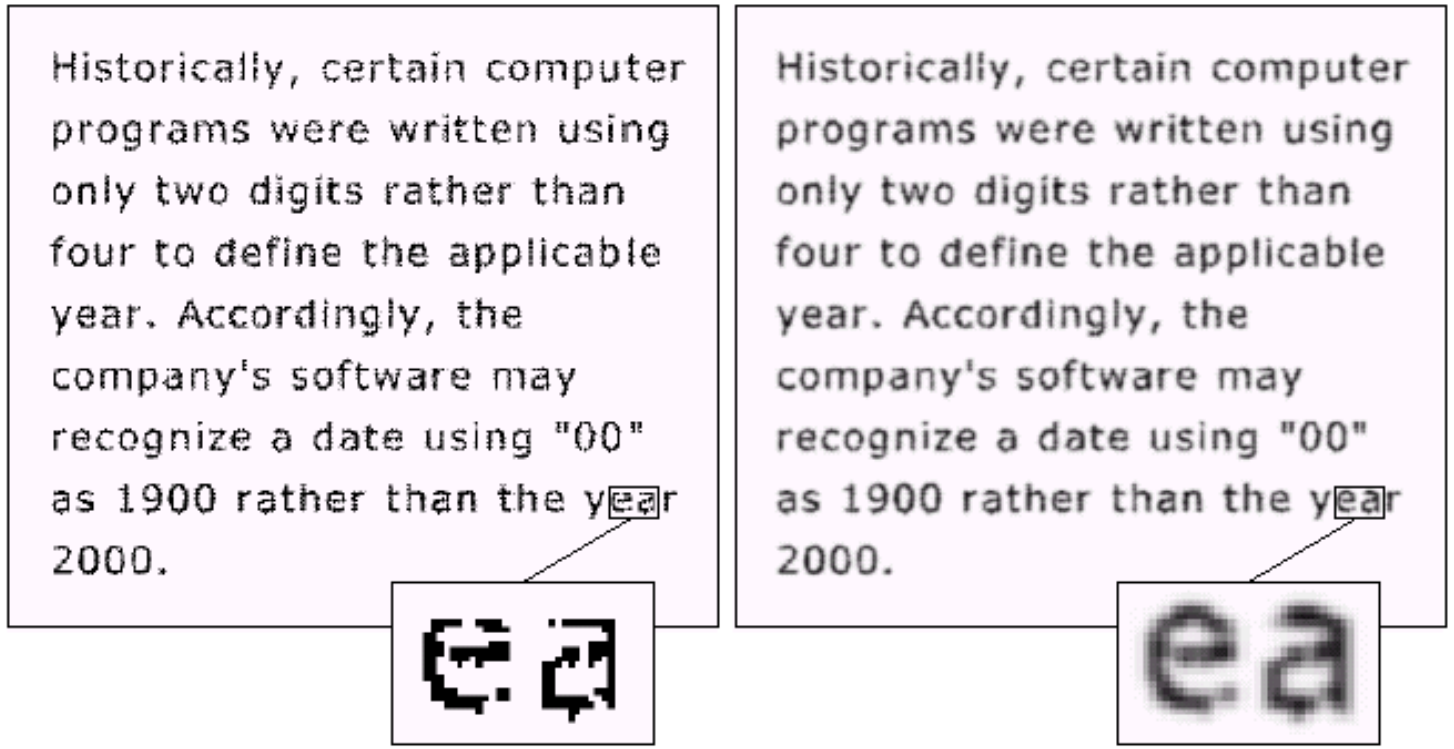
a b
c d
e f

Lowpass Filters

- Additional examples of lowpass filtering

a b

FIGURE 4.19
(a) Sample text of poor resolution (note broken characters in magnified view). (b) Result of filtering with a GLPF (broken character segments were joined).



Lowpass Filters

- Additional examples of lowpass filtering

a b c

FIGURE 4.20 (a) Original image (1028 × 732 pixels). (b) Result of filtering with a GLPF with $D_0 = 100$. (c) Result of filtering with a GLPF with $D_0 = 80$. Note reduction in skin fine lines in the magnified sections of (b) and (c).

Outline

1. Background
2. Fourier transformation
3. Filtering in Frequency Domain
4. Smoothing (lowpass) filter
5. Sharpening (highpass) filters
6. OpenCV in use

Sharpening Filter

- High passs filter

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

Ideal highpass filter

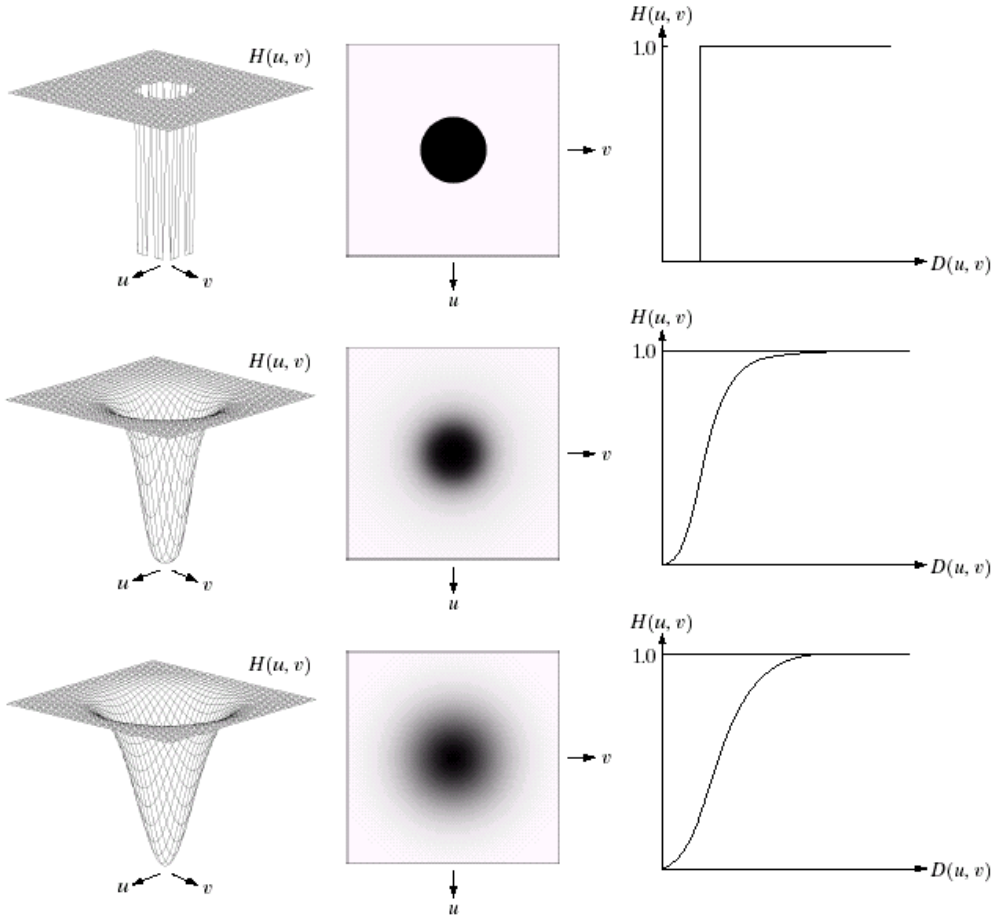
$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \le D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

Butterworth highpass filter

$$H(u,v) = \frac{1}{1+\left[D_0 / D(u,v)\right]^{2n}}$$

Gaussian highpass filter

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

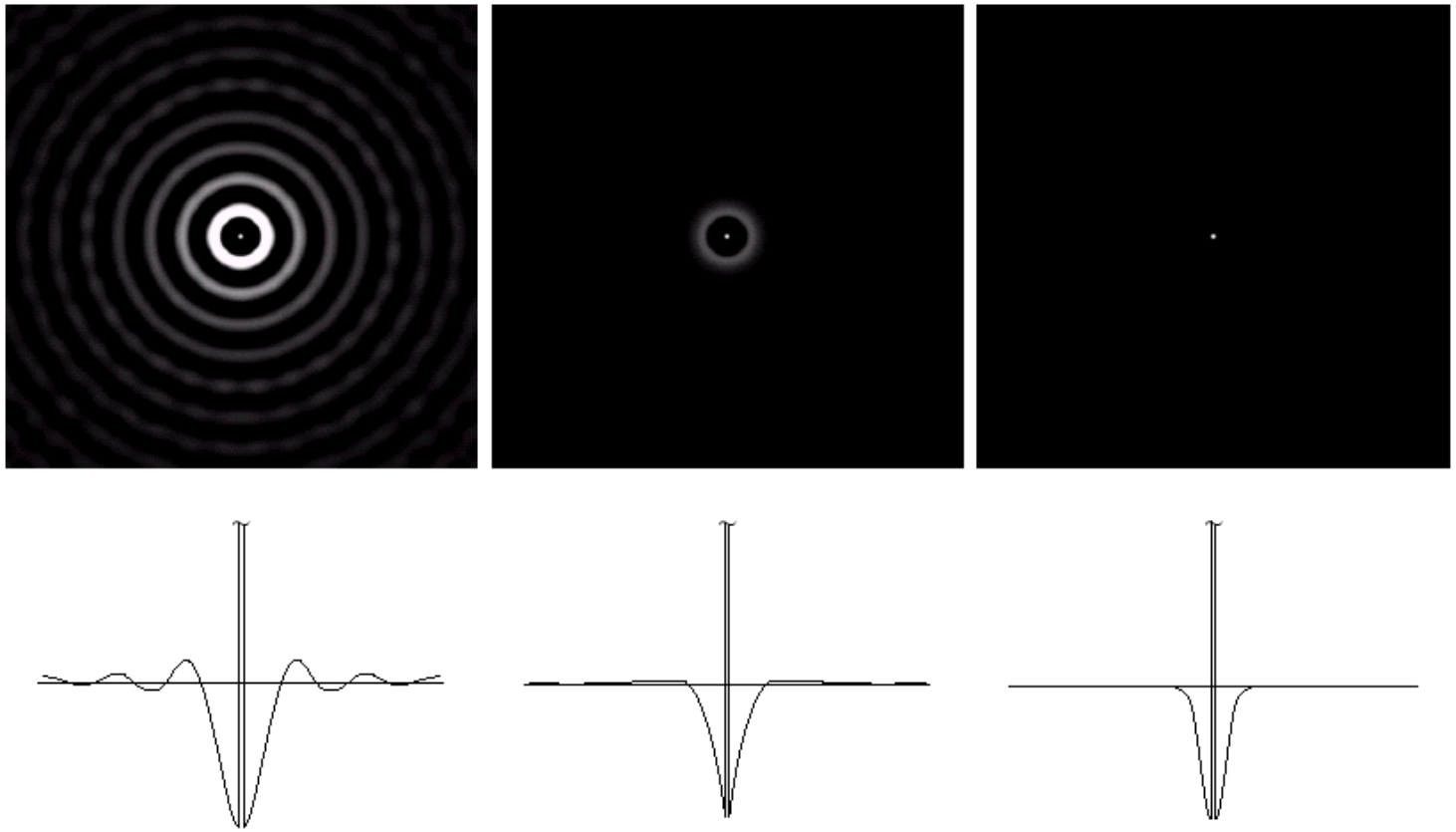


a b c
d e f
g h i

FIGURE 4.22 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Sharpening Filter

- Highpass filters spatial representations

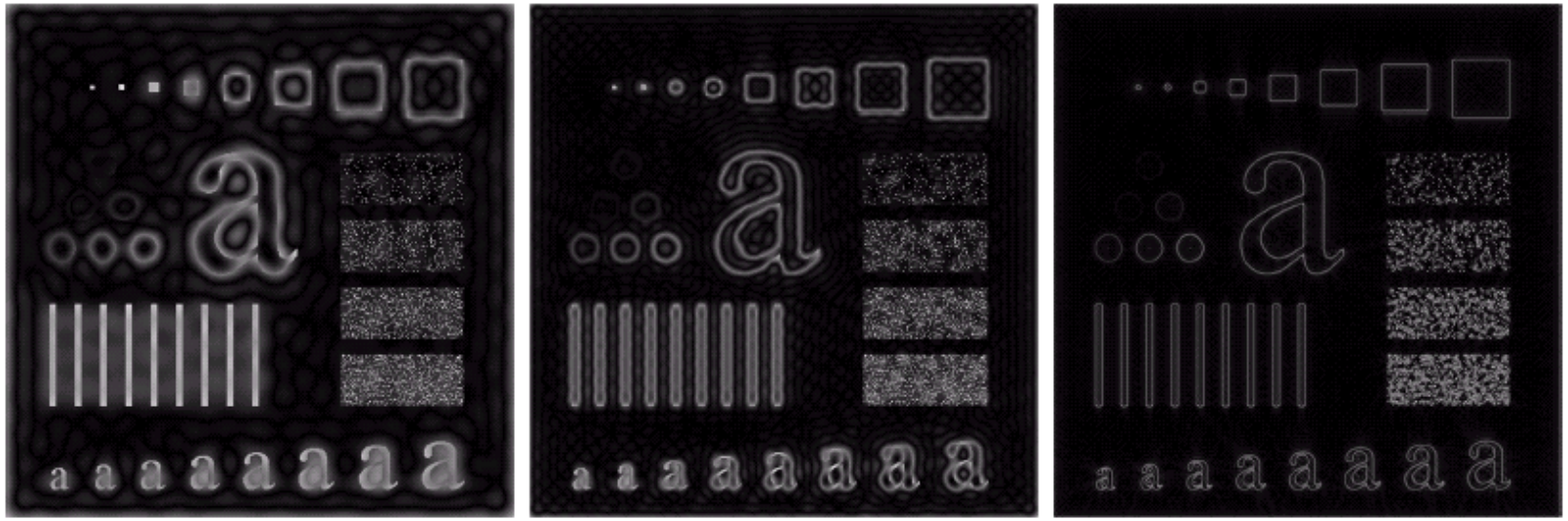


a b c

FIGURE 4.23 Spatial representations of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding gray-level profiles.

Ideal Highpass Filters

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

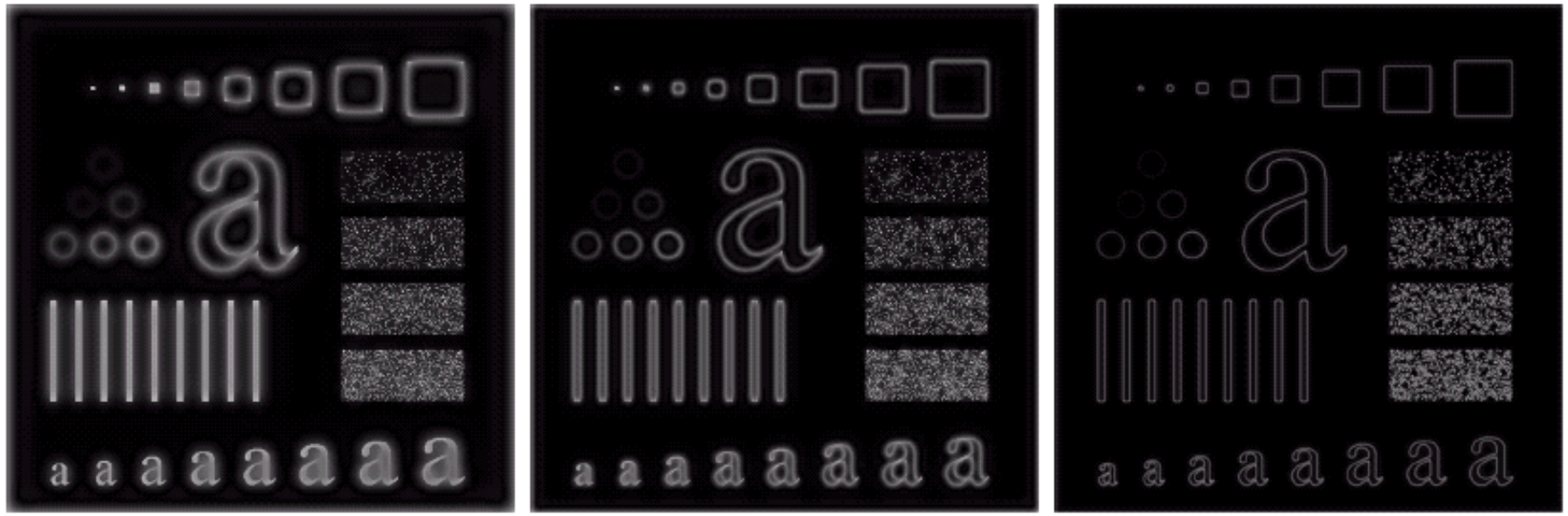


a b c

FIGURE 4.24 Results of ideal highpass filtering the image in Fig. 4.11(a) with $D_0 = 15$, 30, and 80, respectively. Problems with ringing are quite evident in (a) and (b).

Butterworth Highpass Filters

$$H(u,v) = \frac{1}{1+\left[D_0 / D(u,v)\right]^{2n}}$$



a b c

FIGURE 4.25 Results of highpass filtering the image in Fig. 4.11(a) using a BHPF of order 2 with $D_0 = 15$, 30, and 80, respectively. These results are much smoother than those obtained with an ILPF.

Gaussian Highpass Filters

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

a b c

FIGURE 4.26 Results of highpass filtering the image of Fig. 4.11(a) using a GHPF of order 2 with $D_0 = 15$, 30, and 80, respectively. Compare with Figs. 4.24 and 4.25.

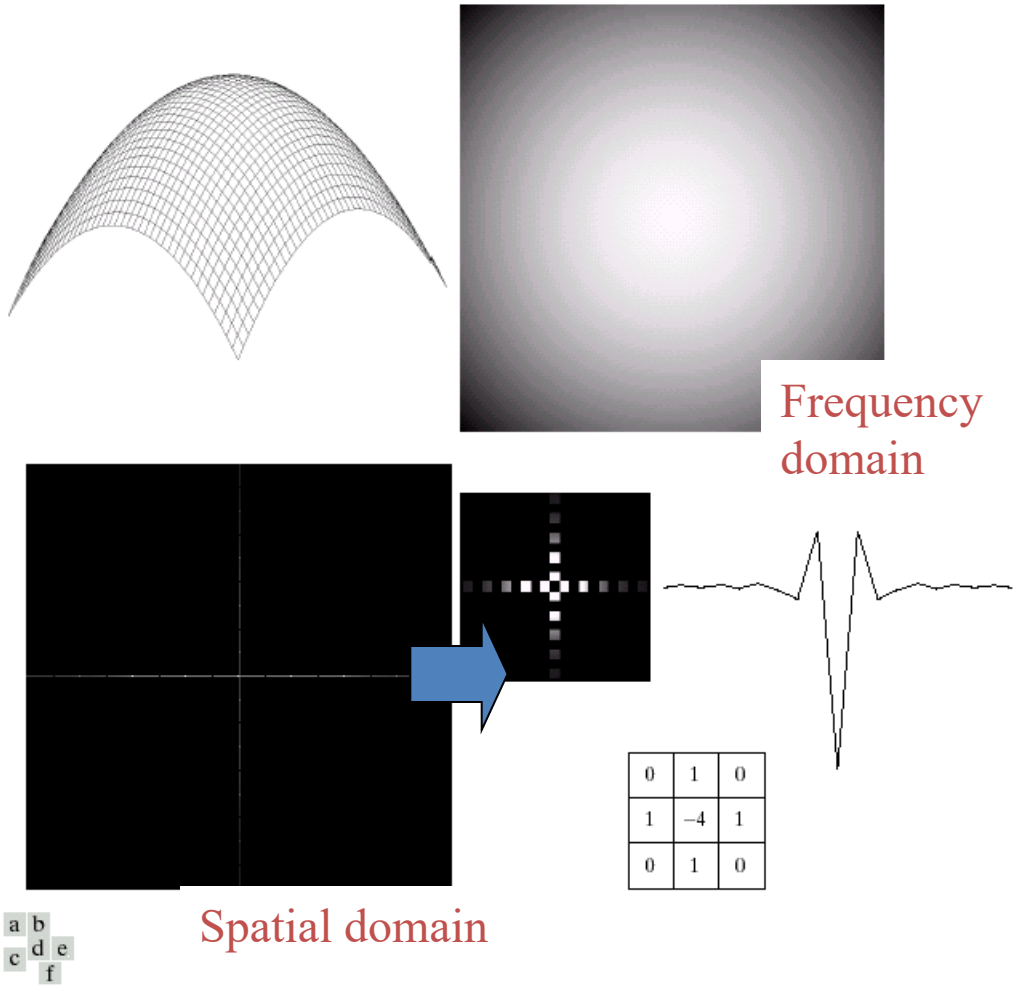
Laplacian in Frequency Domain

- The Laplacian filter

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

- Shift the center:

$$H(u,v) = -\left[(u - \frac{M}{2})^2 + (v - \frac{N}{2})^2\right]$$



a b
c d e
f

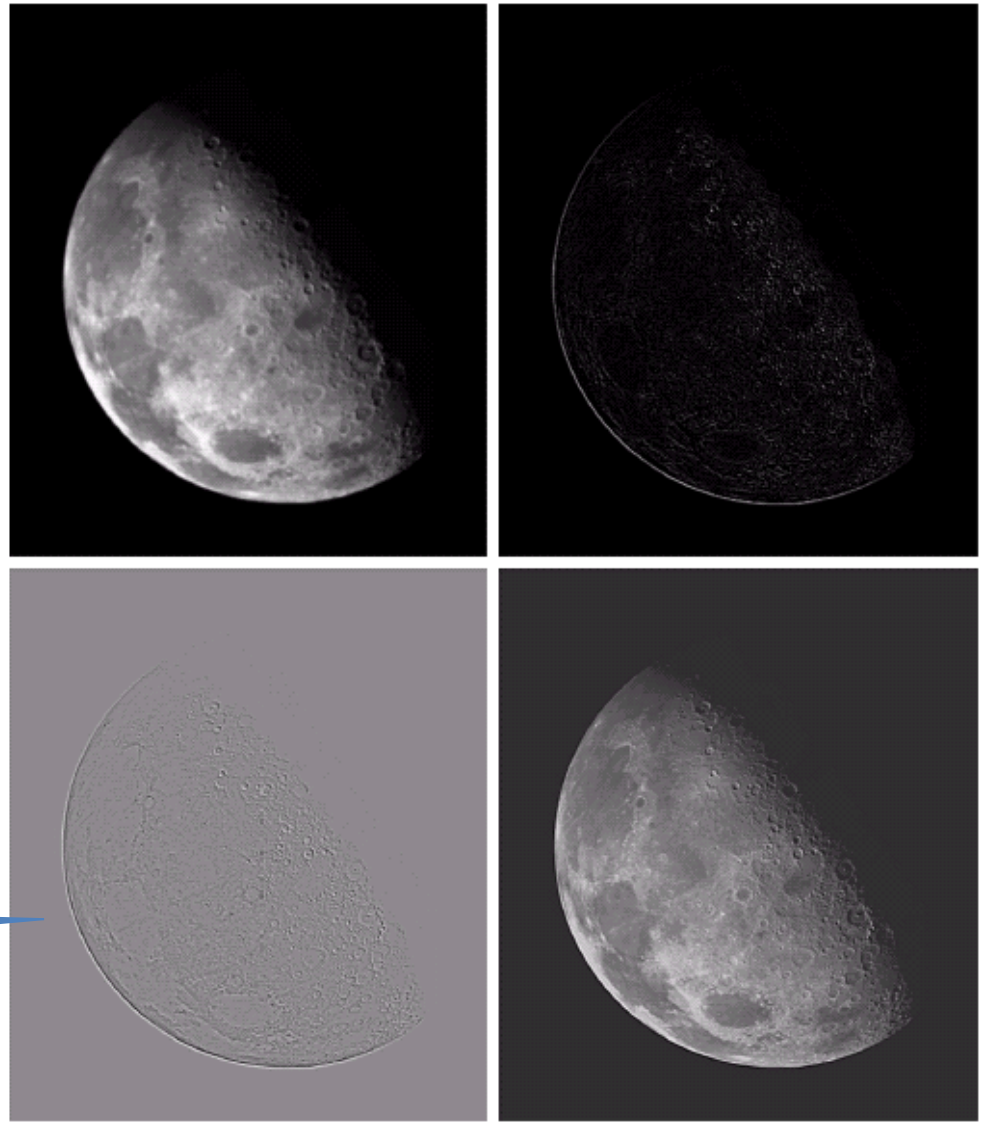
FIGURE 4.27 (a) 3-D plot of Laplacian in the frequency domain. (b) Image representation of (a). (c) Laplacian in the spatial domain obtained from the inverse DFT of (b). (d) Zoomed section of the origin of (c). (e) Gray-level profile through the center of (d). (f) Laplacian mask used in Section 3.7.

Laplacian in Frequency Domain

(a) Image of North Pole of the moon.
(b) Laplacian filtered image
(c) Laplacian image scaled
Image enhanced by using Eq. (4.4-12)

$g(x,y) = f(x,y) - \nabla^2 f(x,y)$
where
$\nabla^2 f(x,y)$: the Laplacian−filtered image in the spatial domain

For display purposes only



Implementation

- Implementation some additional properties of the 2d Fourier transform
- Separability row-column

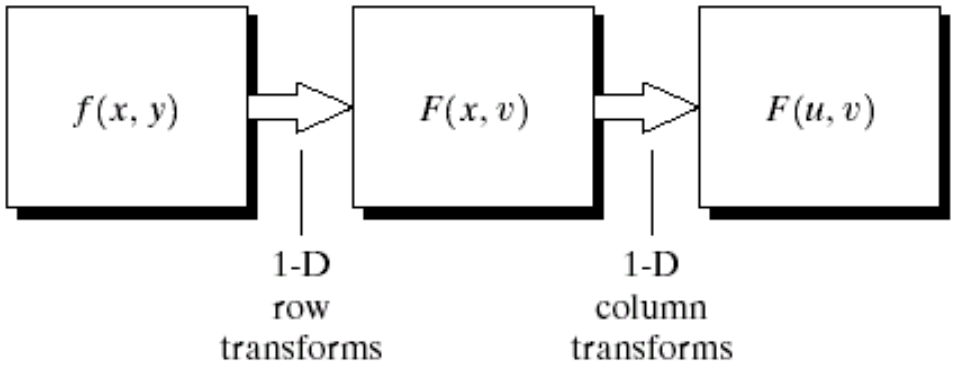


FIGURE 4.35
Computation of the 2-D Fourier transform as a series of 1-D transforms.

Implementation

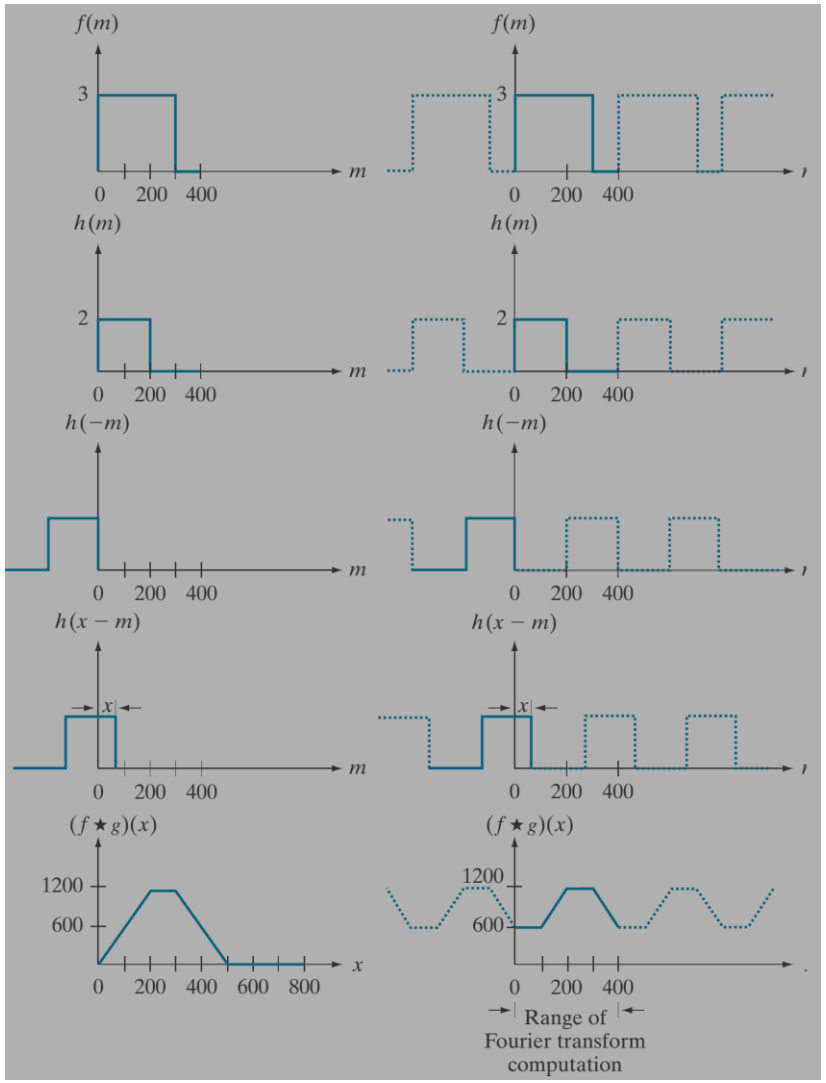
- Implementation more on periodicity

Left: Convolution of two discrete functions.
Right: Convolution of the same functions taking in to account the implied periodicity of DFT. The solid line in (j) is the result we would obtain using the DFT, equivalently
This erroneous result can be remedied by using zero padding.

Convolution

$$f(x) * h(x) = \frac{1}{M}\sum_{m=0}^{M-1} f(m)h(x-m)$$

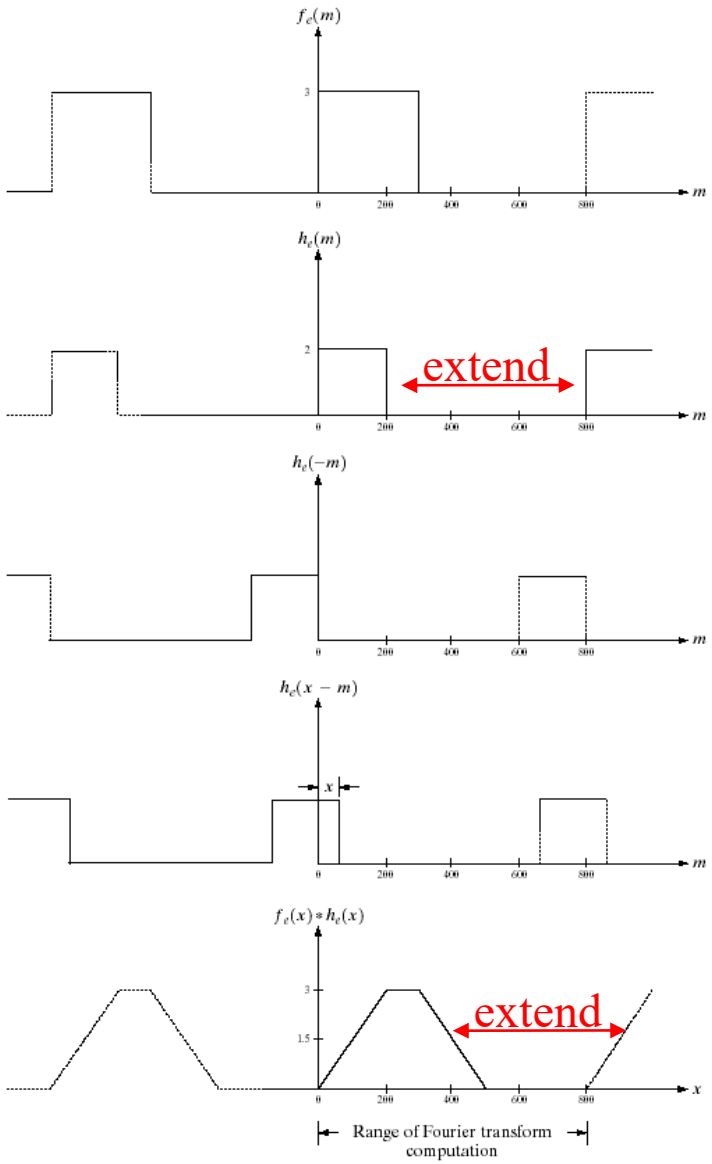
Fig. 4.36
a f
b g
c h
d i
e j



Implementation

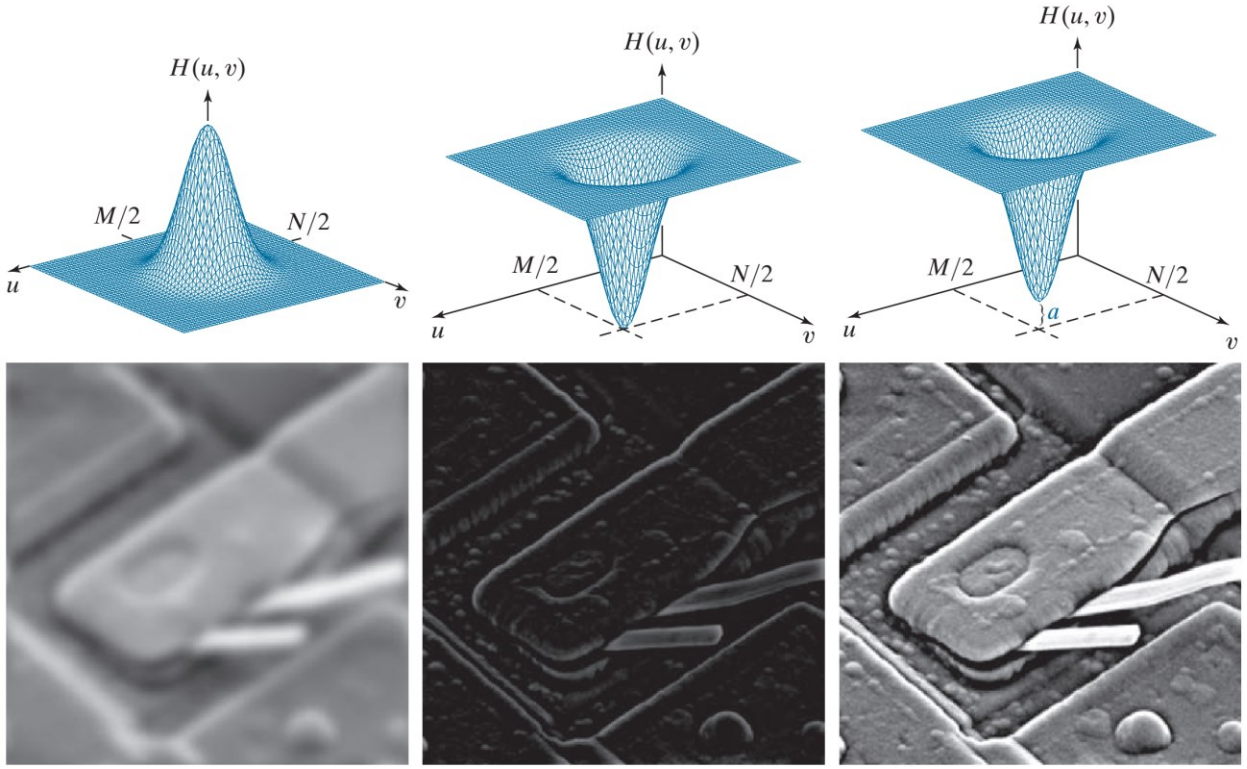
a
b
c
d
e

FIGURE 4.37 Result of performing convolution with extended functions. Compare Figs. 4.37(e) and 4.36(e).



Implementation

a b c
d e f



Example:

Top row: Frequency domain filter transfer functions of (a) a lowpass filter, (b) a highpass filter, and (c) an offset highpass filter.

Bottom row: Filtered images obtained by $g(x,y) = Real\{\Im^{-1}[H(u,v)F(u,v)]\}$, the offset in (c) is a= 0.85 and the height of H(u,v) is 1.

Summary of important properties of 2D FT

TABLE 4.1
Summary of some important properties of the 2-D Fourier transform.

Property	Expression(s)
Fourier transform	$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$
Inverse Fourier transform	$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)}$
Polar representation	$F(u, v) = \lvert F(u, v) \rvert e^{-j\phi(u,v)}$
Spectrum	$\lvert F(u, v) \rvert = [R^2(u, v) + I^2(u, v)]^{1/2}$, $R = \text{Real}(F)$ and $I = \text{Imag}(F)$
Phase angle	$\phi(u, v) = \tan^{-1}\left[\frac{I(u, v)}{R(u, v)}\right]$
Power spectrum	$P(u, v) = \lvert F(u, v) \rvert^2$
Average value	$\bar{f}(x, y) = F(0, 0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$
Translation	$f(x, y) e^{j2\pi(u_0 x/M + v_0 y/N)} \Leftrightarrow F(u - u_0, v - v_0)$ $f(x - x_0, y - y_0) \Leftrightarrow F(u, v) e^{-j2\pi(ux_0/M + vy_0/N)}$ When $x_0 = u_0 = M/2$ and $y_0 = v_0 = N/2$, then $f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$ $f(x - M/2, y - N/2) \Leftrightarrow F(u, v)(-1)^{u+v}$

Summary of important properties of 2D FT

TABLE 4.1
(continued)

Conjugate symmetry	$F(u, v) = F^*(-u, -v)$ $\lvert F(u, v)\rvert = \lvert F(-u, -v)\rvert$
Differentiation	$\dfrac{\partial^n f(x, y)}{\partial x^n} \Leftrightarrow (ju)^n F(u, v)$ $(-jx)^n f(x, y) \Leftrightarrow \dfrac{\partial^n F(u, v)}{\partial u^n}$
Laplacian	$\nabla^2 f(x, y) \Leftrightarrow -(u^2 + v^2)F(u, v)$
Distributivity	$\Im[f_1(x, y) + f_2(x, y)] = \Im[f_1(x, y)] + \Im[f_2(x, y)]$ $\Im[f_1(x, y) \cdot f_2(x, y)] \neq \Im[f_1(x, y)] \cdot \Im[f_2(x, y)]$
Scaling	$af(x, y) \Leftrightarrow aF(u, v), f(ax, by) \Leftrightarrow \dfrac{1}{\lvert ab\rvert} F(u/a, v/b)$
Rotation	$x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$ $f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$
Periodicity	$F(u, v) = F(u + M, v) = F(u, v + N) = F(u + M, v + N)$ $f(x, y) = f(x + M, y) = f(x, y + N) = f(x + M, y + N)$
Separability	See Eqs. (4.6-14) and (4.6-15). Separability implies that we can compute the 2-D transform of an image by first computing 1-D transforms along each row of the image, and then computing a 1-D transform along each column of this intermediate result. The reverse, columns and then rows, yields the same result.

Summary of important properties of 2D FT

Property	Expression(s)
Computation of the inverse Fourier transform using a forward transform algorithm	$\frac{1}{MN} f^*(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u, v) e^{-j2\pi(ux/M + vy/N)}$ This equation indicates that inputting the function $F^*(u, v)$ into an algorithm designed to compute the forward transform (right side of the preceding equation) yields $f^*(x, y)/MN$. Taking the complex conjugate and multiplying this result by MN gives the desired inverse.
Convolution†	$f(x, y) * h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) h(x - m, y - n)$
Correlation†	$f(x, y) \circ h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m, n) h(x + m, y + n)$
Convolution theorem†	$f(x, y) * h(x, y) \Leftrightarrow F(u, v) H(u, v)$; $f(x, y) h(x, y) \Leftrightarrow F(u, v) * H(u, v)$
Correlation theorem†	$f(x, y) \circ h(x, y) \Leftrightarrow F^*(u, v) H(u, v)$; $f^*(x, y) h(x, y) \Leftrightarrow F(u, v) \circ H(u, v)$

TABLE 4.1
(continued)

Summary of important properties of 2D FT

Some useful FT pairs:

Impulse	$\delta(x, y) \Leftrightarrow 1$
Gaussian	$A\sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2(x^2+y^2)} \Leftrightarrow Ae^{-(u^2+v^2)/2\sigma^2}$
Rectangle	$\text{rect}[a, b] \Leftrightarrow ab\dfrac{\sin(\pi ua)}{(\pi ua)}\dfrac{\sin(\pi vb)}{(\pi vb)}e^{-j\pi(ua+vb)}$
Cosine	$\cos(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow \dfrac{1}{2}[\delta(u + u_0, v + v_0) + \delta(u - u_0, v - v_0)]$
Sine	$\sin(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow j\dfrac{1}{2}[\delta(u + u_0, v + v_0) - \delta(u - u_0, v - v_0)]$

TABLE 4.1 *(continued)*

† Assumes that functions have been extended by zero padding.

Outline

1. Background
2. Fourier transformation
3. Smoothing (lowpass) filter
4. Sharpening (highpass) filters
5. OpenCV in use

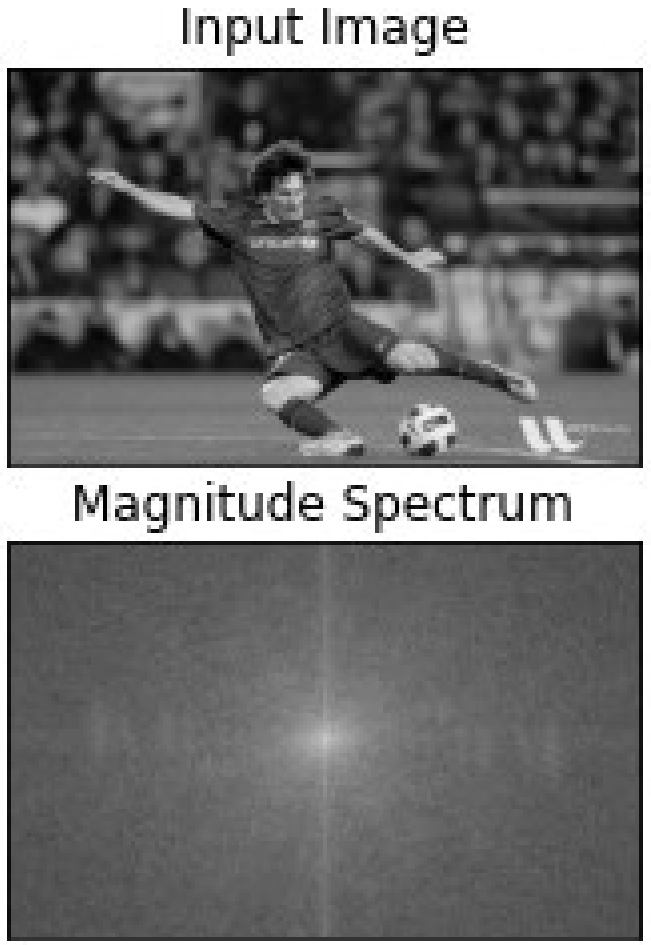
OpenCV

- To find the Fourier Transform of images using OpenCV
- To utilize the FFT functions available in Numpy
- Some applications of Fourier Transform
- We will see following functions : cv.dft(), cv.idft()

OpenCV

Find the frequency transform, the magnitude spectrum

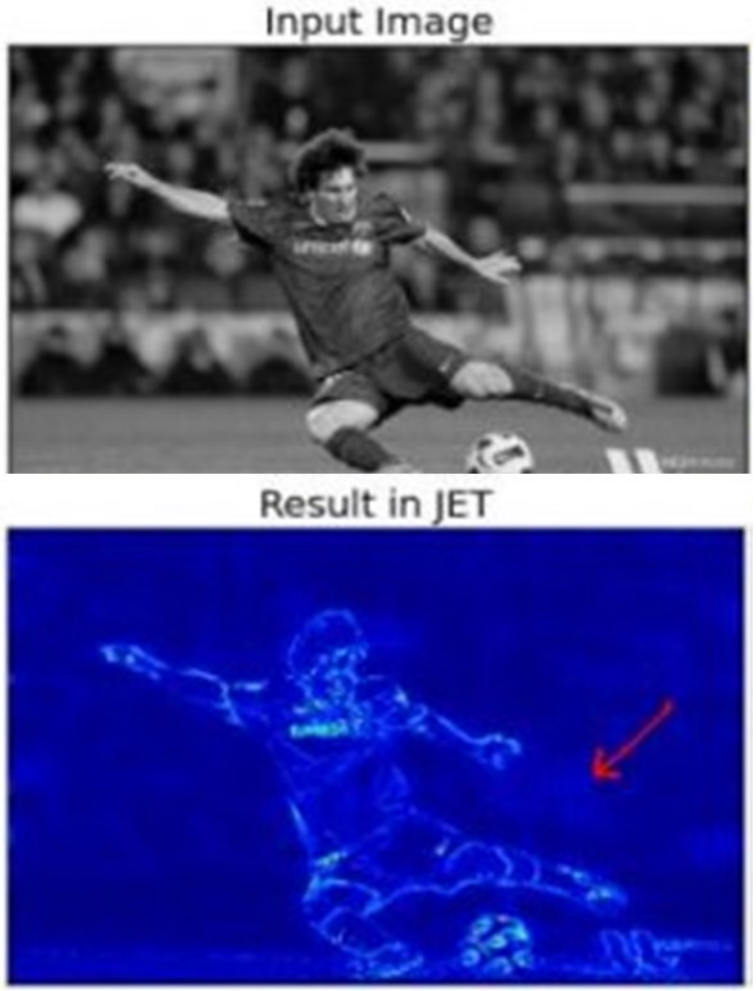
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread(‘/images/messiplayer.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum= 20*np.log(np.abs(fshift))
plt.subplot(121)
plt.imshow(img, cmap = 'gray')
plt.title('Input Image’),
plt.xticks([]), plt.yticks([])
plt.subplot(122),
plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum’),
plt.xticks([]), plt.yticks([])
plt.show()
```



OpenCV

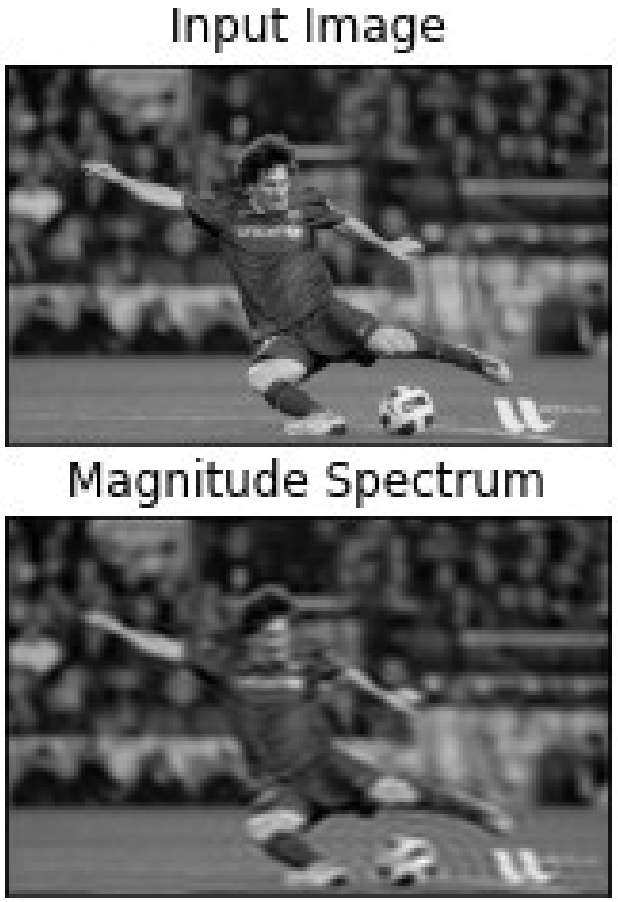
Find inverse FFT using np.ifft2() function

```
rows, cols = img.shape
crow,ccol = rows//2 , cols//2
fshift[crow-30:crow+31, ccol-30:ccol+31] = 0
f_ishift = np.fft.ifftshift(fshift)
img_back = np.fft.ifft2(f_ishift)
img_back = np.real(img_back)
plt.subplot(131)
plt.imshow(img, cmap = 'gray')
plt.title('Input Image’)
plt.xticks([]), plt.yticks([])
plt.subplot(132)
plt.imshow(img_back, cmap = 'gray')
plt.title('Image after HPF’)
plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(img_back)
plt.title('Result in JET’)
plt.xticks([]), plt.yticks([])
plt.show()
```



Fourier Transform

```
 rows, cols = img.shape
crow,ccol = rows//2 , cols//2
# create a mask remain all zeros, center is 1,
mask = np.zeros((rows,cols,2),np.uint8)
mask[crow-30:crow+30, ccol-30:ccol+30] = 1
# apply mask and inverse DFT
dft = cv.dft(np.float32(img),flags = cv.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv.idft(f_ishift)
img_back = cv.magnitude(img_back[:,:,0],img_back[:,:,1])
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Magnitude Spectrum’),
plt.xticks([]), plt.yticks([])
plt.show()
```



Thanks for your attention!

Q&A

19:31

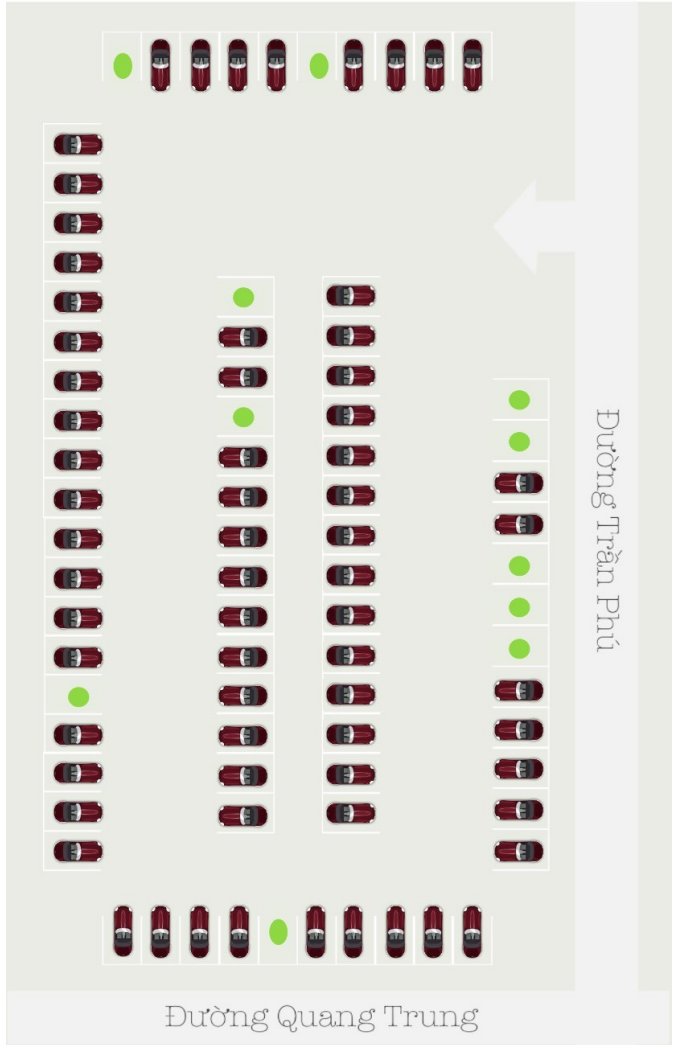
Bãi đỗ xe TTHC

MÔ HÌNH THỰC TẾ

16:36

Bãi đỗ xe TTHC

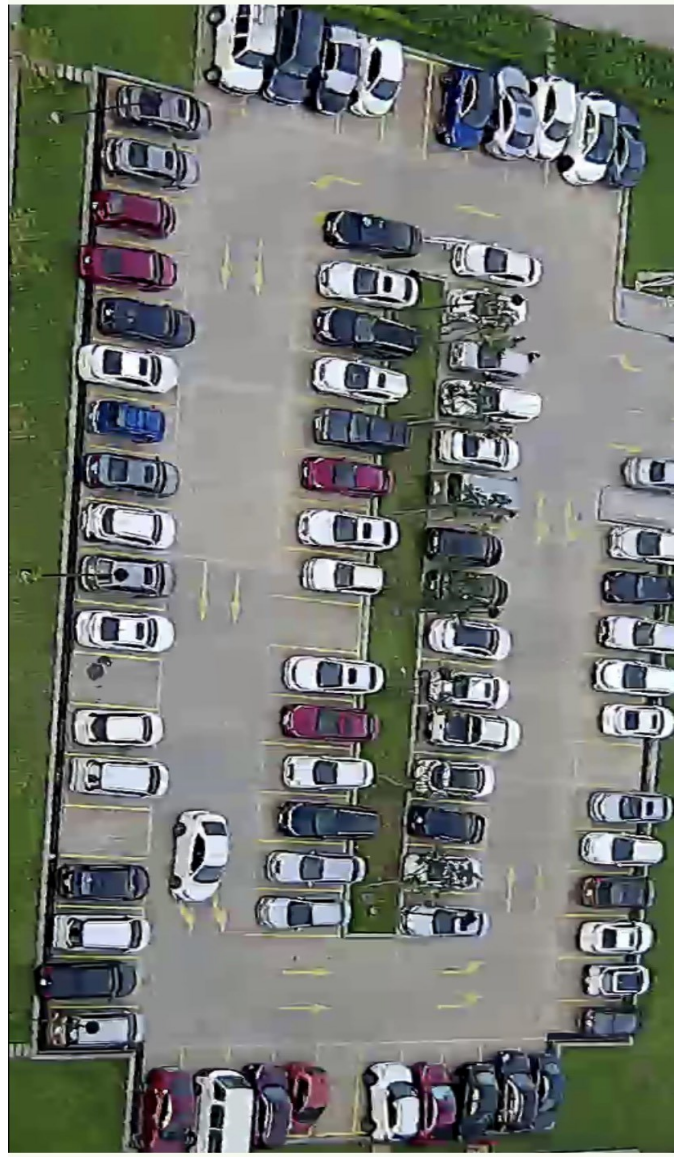
11 chỗ trống 4:36:14 PM



MÔ HÌNH THỰC TẾ

16:36

Bãi đỗ xe TTHC



MÔ HÌNH THỰC TẾ