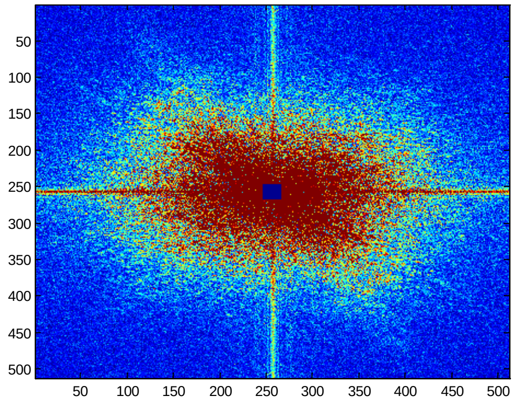
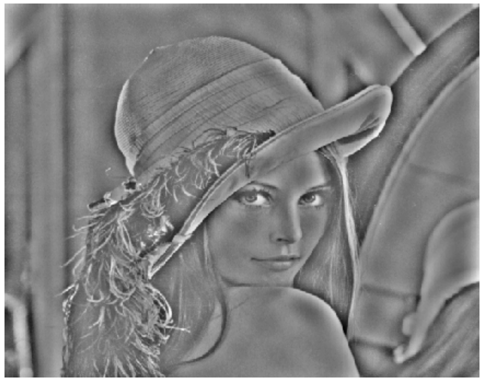
COMP 4687

Frequency Domain Filtering

Frequency Domain Filtering





Frequency domain filtering can be used for

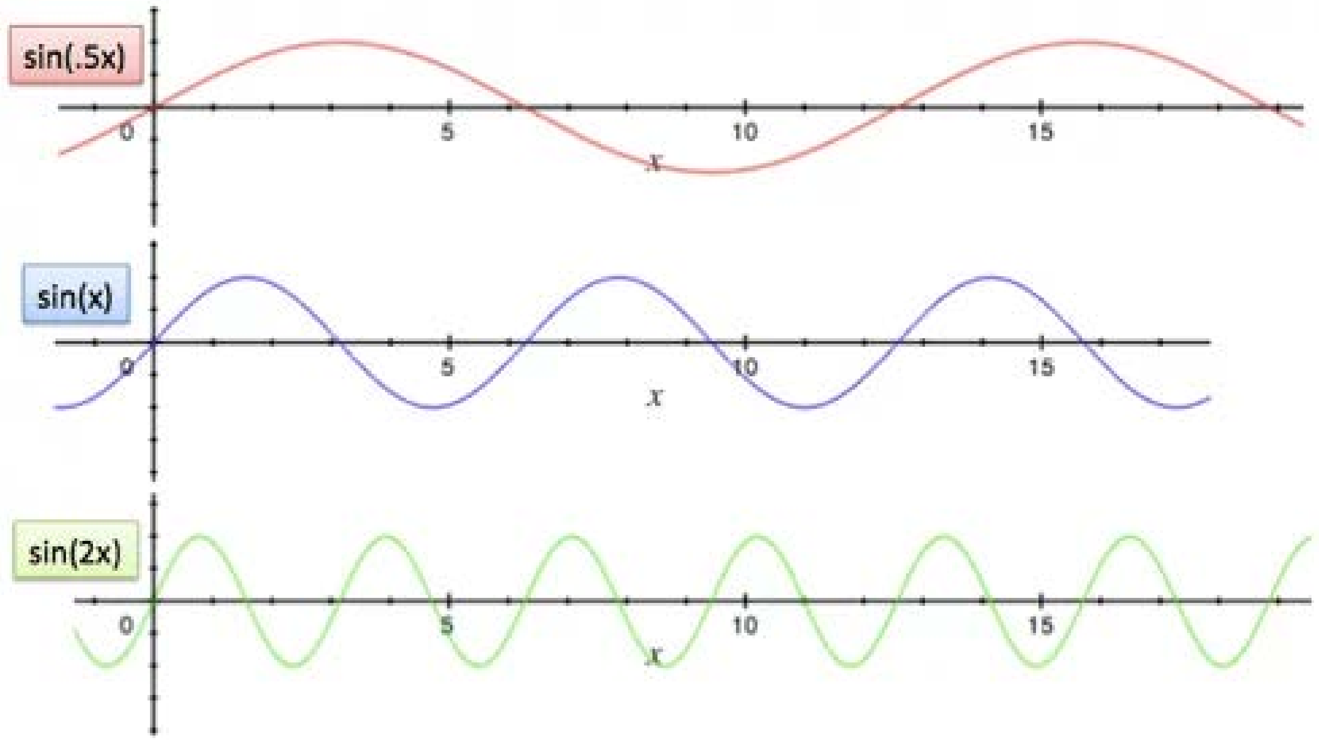
Noise removal

Recovering certain details in the image

Analysis of spatial frequency using sinusoids

Fourier analysis is very important in signal processing; its theory and practice fills many books.

The mathematician Fourier imagined the surface of the sea as a sum of sine waves. Large waves caused by the tide or ships had long wavelengths (low frequency), while smaller waves caused by the wind or dropped objects, etc. had short wavelengths (high frequency).



Sinusoids can form a good basis vector set

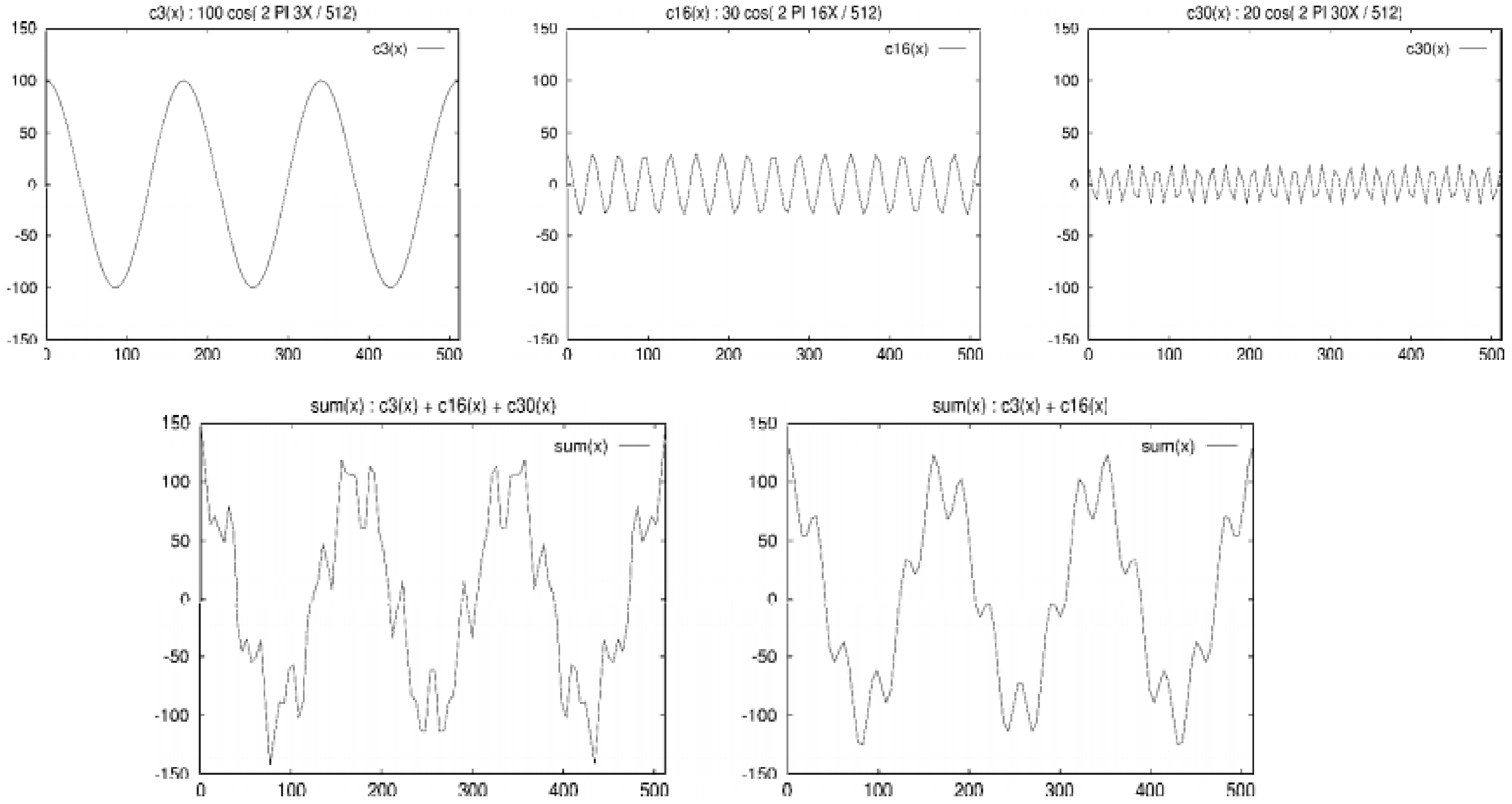
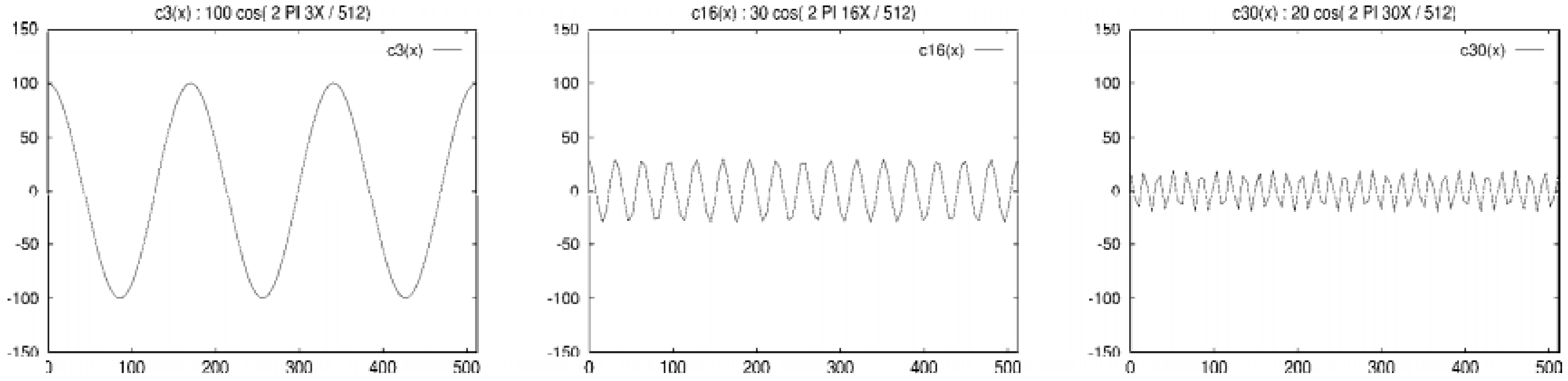
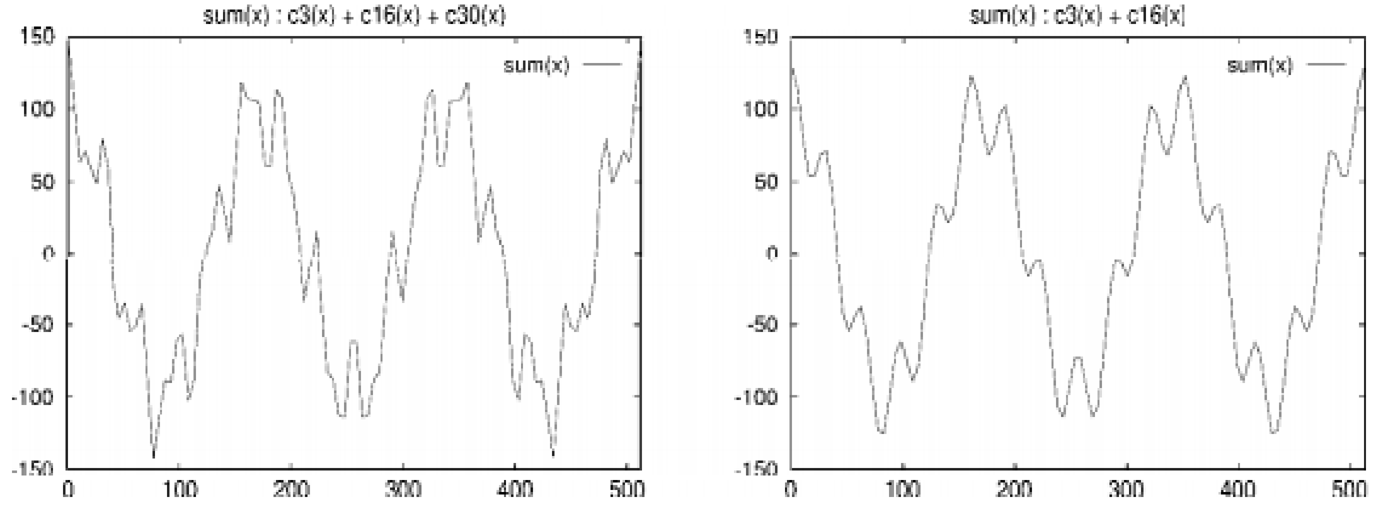


Figure 5.37

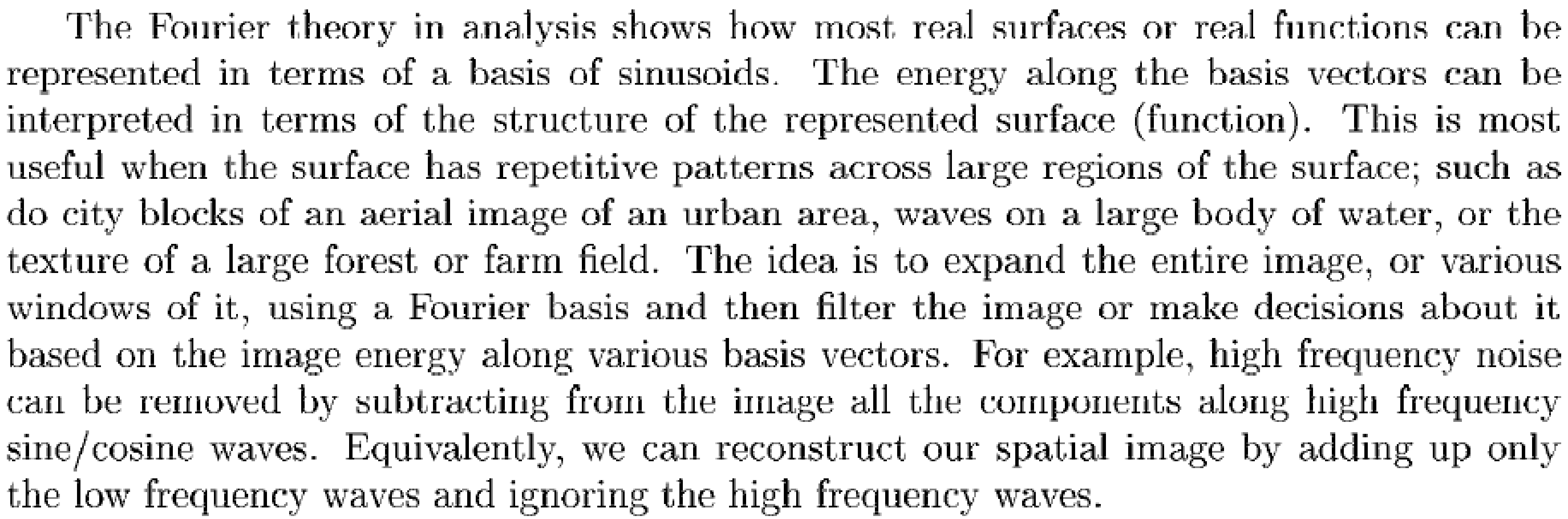
- (Top row) Three sinusoids, $100cos(2\pi\frac{3x}{512})$, $30cos(2\pi\frac{16x}{512})$ and $20cos(2\pi\frac{30x}{512})$;
- (bottom row left) sum of all three and (bottom row right) sum of first two.



The top row of Figure 5.37 shows three pure waves of 3, 16, and 30 cycles across a 1D space of $x \in [0, 512]$: in the bottom row are two functions, one which sums all three waves and one which sums only the first two. Similar sets of waves can be used to create 2D picture functions or even 3D density functions.



Sinusoids can form a good basis vector set



The Fourier theory in analysis shows how most real surfaces or real functions can be represented in terms of a basis of sinusoids. The energy along the basis vectors can be interpreted in terms of the structure of the represented surface (function). This is most useful when the surface has repetitive patterns across large regions of the surface; such as do city blocks of an aerial image of an urban area, waves on a large body of water, or the texture of a large forest or farm field. The idea is to expand the entire image, or various windows of it, using a Fourier basis and then filter the image or make decisions about it based on the image energy along various basis vectors. For example, high frequency noise can be removed by subtracting from the image all the components along high frequency sine/cosine waves. Equivalently, we can reconstruct our spatial image by adding up only the low frequency waves and ignoring the high frequency waves.

Fourier theory

- Any **periodic function** can be written as a **weighted sum** of **sines** and **cosines** of different frequencies (Fourier, 1807).

- Even functions that are not periodic (but whose area under the curve is finite) can be expressed as the **integral** of **sines** and **cosines** multiplied by a weighing function.

  → **Fourier transform**

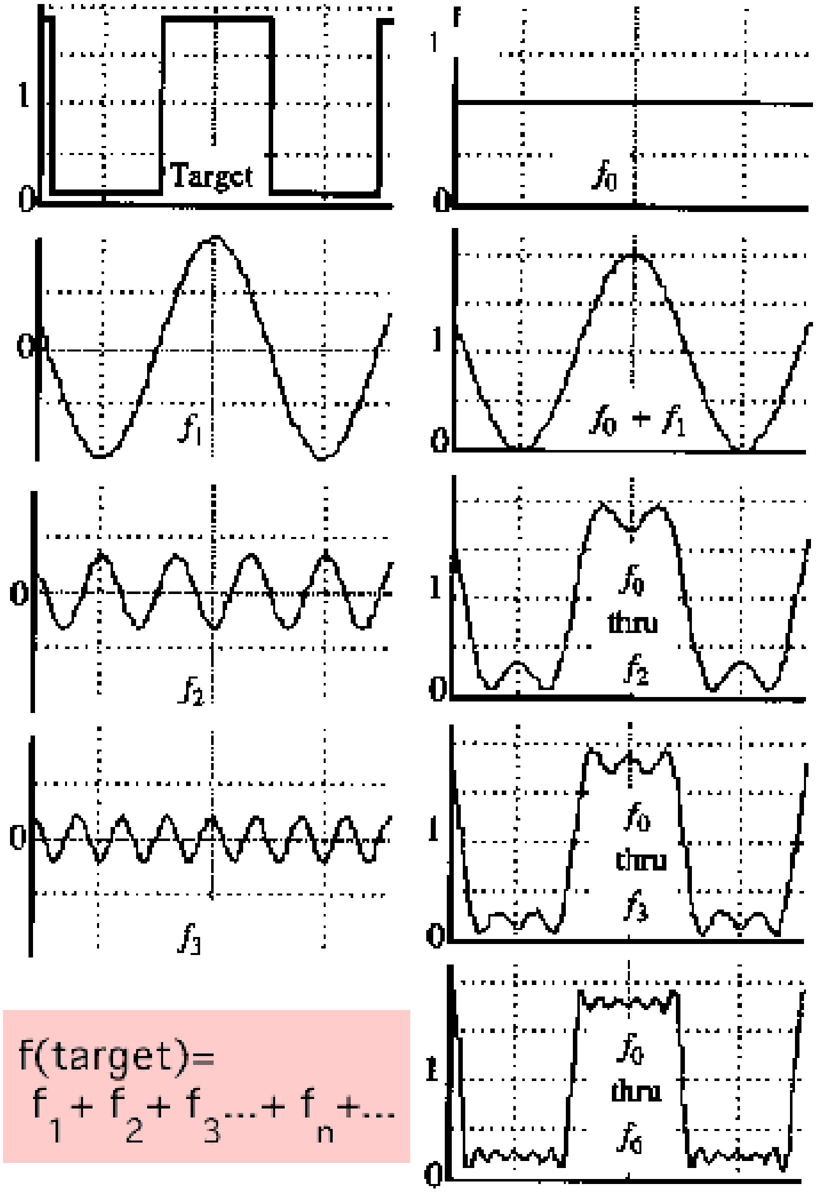
- Has widespread applications in engineering

Fourier theory

- **The Fourier theory** shows how most real functions can be represented in terms of a basis of sinusoids.

- The building block:
  - a sin(ωx + Φ)
- Add enough of them to get any signal you want.

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$



f(target)=
$f_1 + f_2 + f_3 ... + f_n + ...$

Operations using the Fourier basis

For an intuitive introduction, let us assume an orthonormal set of sinusoidal basis images (or picture functions) $E_k \approx E_{u,v}(x, y)$. Presently, k and u, v are integers which define a finite set of basis vectors. It will soon become clear how parameters u, v determine the basis vectors, but right now, we'll use the single index k to keep our focus on the fundamental concept.

The bottom row of Figure 5.37 shows two signals formed by summing three or two of the pure cosine waves shown in the top row of the figure.

More complex functions can be created by using more pure cosine waves. Figure 5.39 shows the result of summing scaled versions of the three "pure" waves in Figure 5.38 to create a new picture function.

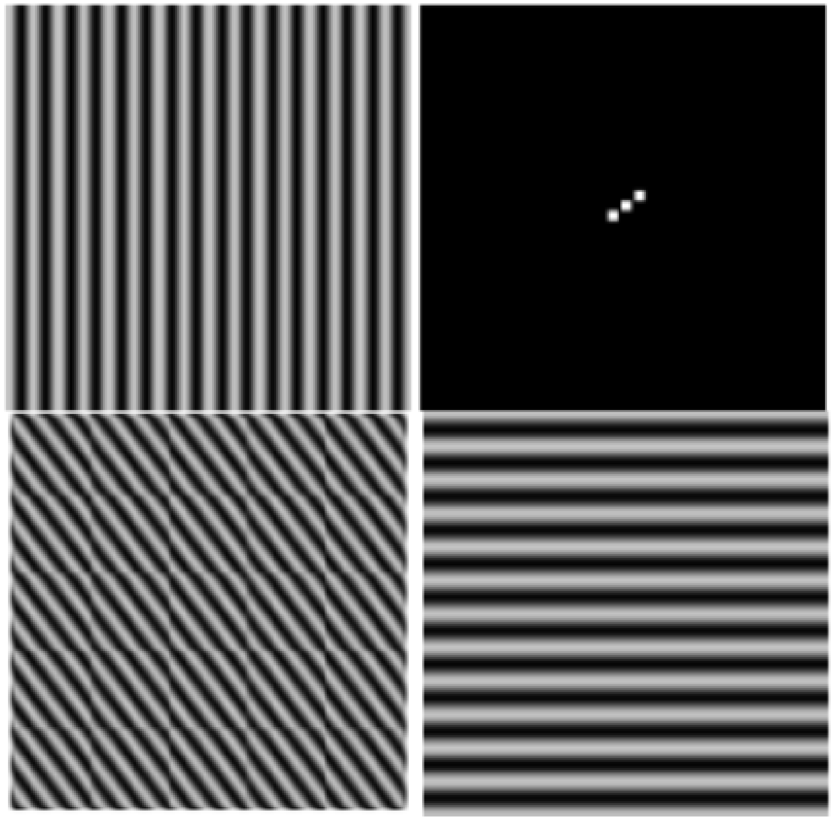


Figure 5.38: Different sinusoidal picture functions in the spatial domain $[x, y]$: The top left image was generated by the formula $100cos(2\pi(16x/512)) + 100$ and has 16 cycles along the x-axis.

The bottom right image was generated by the formula $100cos(2\pi(12y/512)) + 100$ and has 12 cycles along the y-axis.

The bottom left image was generated by the formula $100cos(2\pi(16x/512 + 12y/512)) + 100$: note how the waves of the bottom left image align with those of the top left and bottom right.

The Fourier power spectrum is shown at the top right.

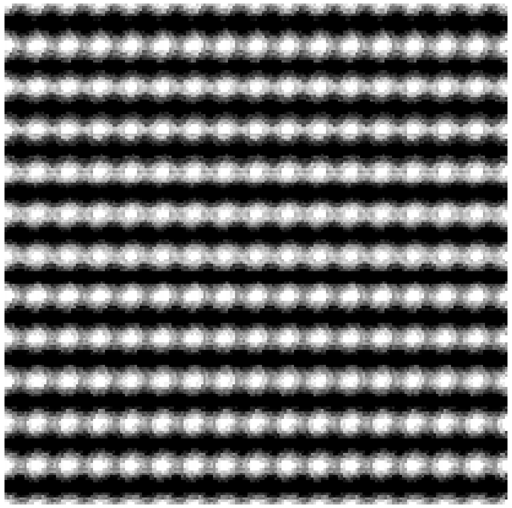


Figure 5.39: A picture function formed by the sum $I[x, y] = 100E_i + 30E_j + 10E_k$ where E_i is as in the bottom right of Figure 5.38, E_j is as in the top left, and E_k is as in the bottom left. (Perhaps it models an orchard or foam padding?)

Operations using the Fourier basis

$$E_k \approx E_{u,v}(x,y)$$

By using the Fourier basis functions E_k, any picture function can be represented as

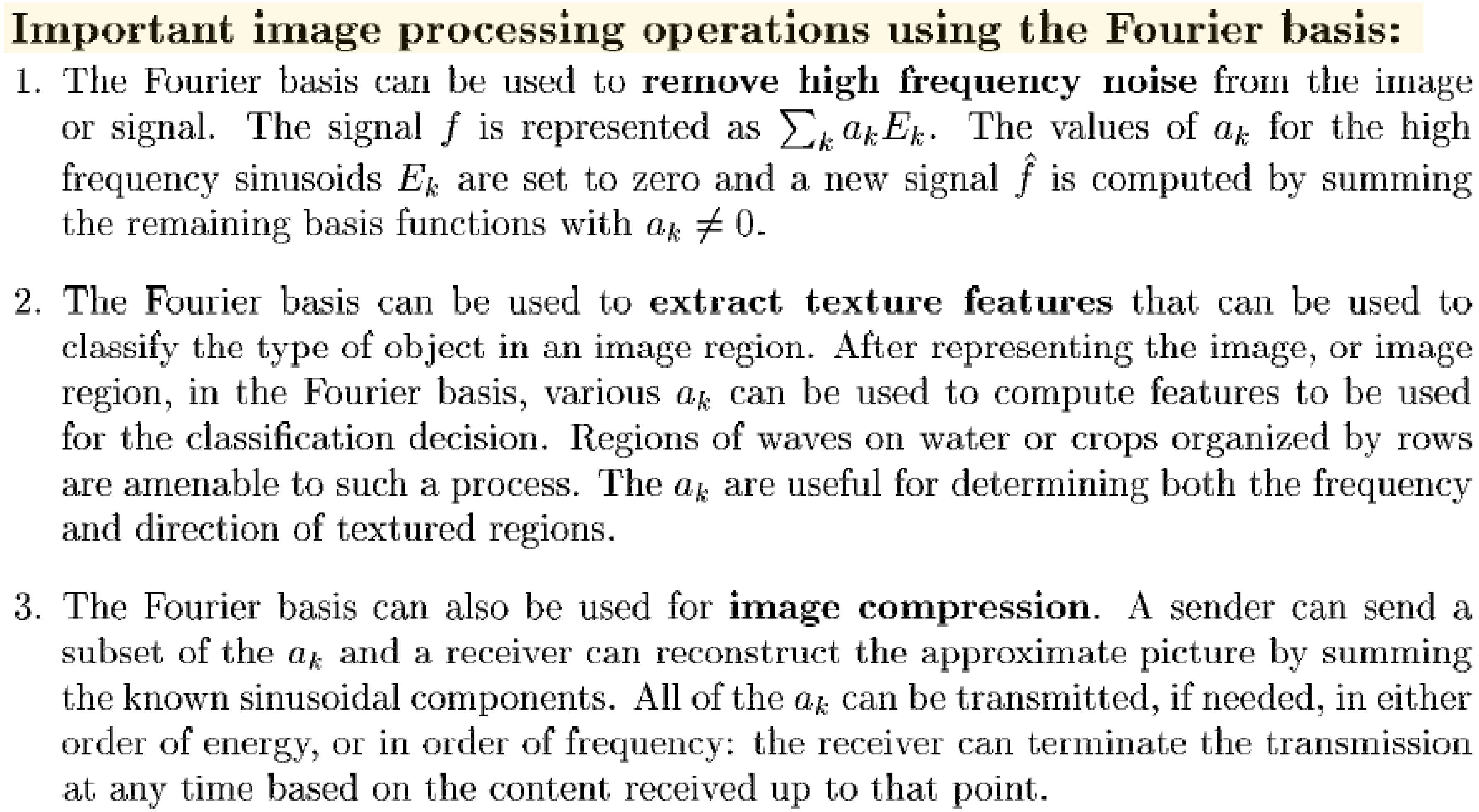
$$I[x,y] = \sum_{k=0}^{N-1} a_k E_k[x,y]$$

As in previous sections, a_k is a measure of the similarity between $I[x,y]$ and $E_k[x,y]$ and the energy of image $I[x,y]$ along that particular component wave. Useful image processing operations can be done by operating on the values of a_k rather than on the individual intensities $I[x,y]$.

Three major operations are described below.

Fourier Basis

$I[x,y] = \sum_{k=0}^{N-1} a_k E_k[x,y]$ where $E_k \approx E_{u,v}(x,y)$



Important image processing operations using the Fourier basis:

1. The Fourier basis can be used to **remove high frequency noise** from the image or signal. The signal f is represented as $\sum_k a_k E_k$. The values of a_k for the high frequency sinusoids E_k are set to zero and a new signal $\hat{f}$ is computed by summing the remaining basis functions with $a_k \neq 0$.

2. The Fourier basis can be used to **extract texture features** that can be used to classify the type of object in an image region. After representing the image, or image region, in the Fourier basis, various a_k can be used to compute features to be used for the classification decision. Regions of waves on water or crops organized by rows are amenable to such a process. The a_k are useful for determining both the frequency and direction of textured regions.

3. The Fourier basis can also be used for **image compression**. A sender can send a subset of the a_k and a receiver can reconstruct the approximate picture by summing the known sinusoidal components. All of the a_k can be transmitted, if needed, in either order of energy, or in order of frequency: the receiver can terminate the transmission at any time based on the content received up to that point.

Fourier Basis

$I[x,y] \;=\; \sum_{k=0}^{N-1} a_k E_k[x,y]$ where $E_k \;\approx\; E_{u,v}(x,y)$

Our goal here is to produce a useful basis of picture functions and an understanding of how to use it in practice. Some important mathematical background using continuous functions is needed. For this development, lets assume that the origin of our coordinate system is at the center of our picture function, which is defined for a square region of the xy-plane. When we proceed to a digital version, the picture will be represented by digital image $I[x,y]$ with N^2 samples. First, we establish a set of sinusoids of different frequency as an orthogonal basis for continuous signals f. If m, n are any two different integers, then the two cosine waves with these frequency parameters are orthogonal over the interval $[-\pi, \pi]$.

2D Picture Functions

$I[x,y] = \sum_{k=0}^{N-1} a_k \boxed{E_k[x,y]}$ where $E_k \approx E_{u,v}(x,y)$

Use the definition $e^{j\omega} = cos\ \omega + j\ sin\ \omega$.

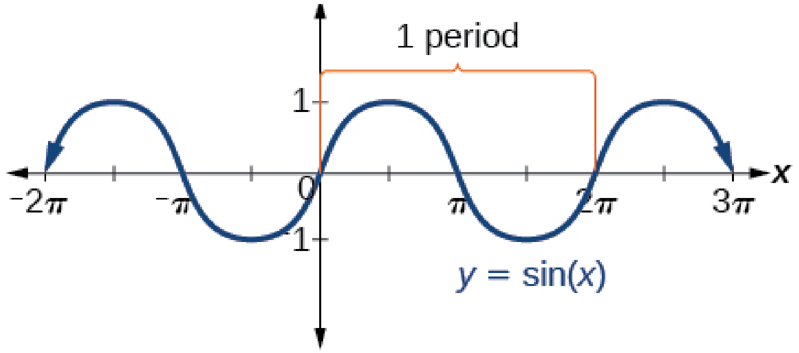
59 Definition *The complex valued picture function*

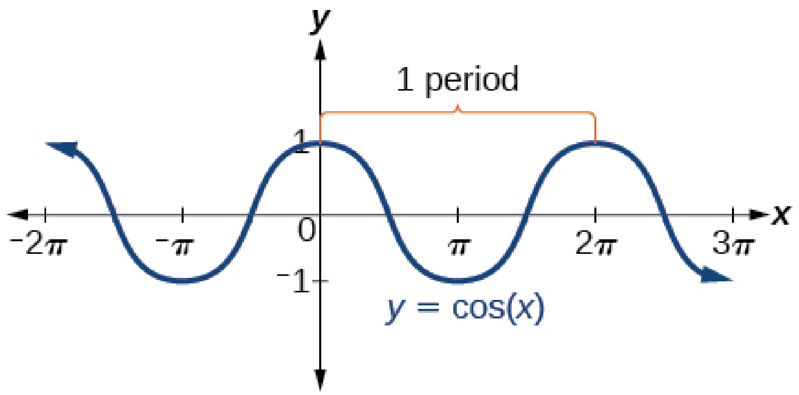
$$\boxed{E_{u,v}(x,y)} \equiv e^{-j\ 2\pi(ux+vy)}$$
$$= cos(2\pi(ux+vy)) - jsin(2\pi(ux+vy)) \qquad (5.30)$$

where u and v are spatial frequency parameters as shown in Figure 5.38 and $j = \sqrt{-1}$.

2D Picture Functions

Using a complex value is a convenience that allows us to separately account for both a cosine and sine wave of the same frequency: the sine wave has the same structure as the cosine wave but is 1/4 wavelength out of phase with the cosine wave. When one of these basis functions correlates highly with a picture function, it means that the picture function has high energy in frequency u, v. The *Fourier transform*, converts a picture function into an array of these correlations. We start with the integral form and then give the discrete summation for digital images below.





2D Picture Functions

$$\begin{aligned} E_{u,v}(x,y) &\equiv e^{-j\ 2\pi(ux+vy)} \\ &= cos(2\pi(ux+vy)) - j sin(2\pi(ux+vy)) \end{aligned}$$

60 Definition *The 2D* **Fourier Transform** *transforms a spatial function $f(x,y)$ into the u,v frequency domain.*

$$\begin{aligned} F(u,v) &\equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\boxed{E_{u,v}(x,y)}dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\boxed{e^{-j\ 2\pi(ux+vy)}}dxdy \end{aligned} \tag{5.31}$$

Often, we want to work with the *power spectrum*, which combines energy from sine and cosine waves of the same frequency components u,v. A power spectrum is shown at the top right of Figure 5.38.

61 Definition *The* **Fourier power spectrum** *is computed as*

$$P(u,v) \equiv (\ Real(F(u,v))^2 + Imaginary(F(u,v))^2\ )^{1/2} \tag{5.32}$$

1-D Fourier transform (Continuous)

- The *Fourier transform*, $F(u)$, of a single variable, continuous function, $f(x)$, is defined by

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

- Given $F(u)$, we can obtain $f(x)$ using the *inverse Fourier transform*

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

1-D Fourier transform (Discrete)

- The *discrete Fourier transform (DFT)*, $F(u)$, of a discrete function of one variable, $f(x)$, $x = 0, 1, 2, \ldots, M-1$, is defined by

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

for $u = 0, 1, 2, \ldots, M-1$.

- Given $F(u)$, we can obtain the original function back using the *inverse DFT*

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

for $x = 0, 1, 2, \ldots, M-1$.

Discrete Fourier Transform

DEFINITION *The* **Discrete Fourier Transform (DFT)** *transforms an image of $N \times N$ spatial samples $I[x, y]$ into an $N \times N$ array $F[u, v]$ of coefficents used in its frequency representation*

$$F[u,v] \equiv \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I[x,y] E_{u,v}[x,y]$$

$$F[u,v] \equiv \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I[x,y] e^{\frac{-2\pi}{N} j (xu+yv)}$$

To compute the single frequency domain element ("pixel") $F[u, v]$, we just compute the dot product between the entire image $I[x, y]$ and a "mask" $E_{u,v}[x, y]$, which is usually not actually created, but computed implicitly in terms of u, v and the *cos* and *sin* functions as needed. We also define an inverse transform to transform a frequency domain representation $F[u, v]$ into a spatial image $I[x, y]$. Although it is useful to display the transform F as a 2D image, it might be less confusing to insist that it is NOT really an image. We have used the formal term *frequency representation* instead.

Discrete Fourier Transform

Definition *The* **Inverse Discrete Fourier Transform(IDFT)** *transforms an* $N \times N$ *frequency representation* $F[u,v]$ *into an* $N \times N$ *image* $I[x,y]$ *of spatial samples.*

$$I[x,y] \equiv \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F[u,v] e^{\frac{+2\pi \; j}{N}(ux+vy)}$$

If $F[u,v]$ was computed by "forward" transforming $I[x,y]$ in the first place, we want the inverse transform to return to that original image.

We first focus the discussion on the practical use of the **DFT & IDFT**. For storage or communication of an image, it might be useful to transform it into its frequency representation; the input image can be recovered by using the inverse transform. In image processing, it is common to perform some enhancement operation on the frequency representation before transforming back to recover the spatial image. For example, high frequencies can be reduced or even removed by reducing, or zeroing, the elements of $F[u,v]$ that represent high frequency waves.

Fourier transform 2D

- Discrete Fourier transform:

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

for $u = 0, 1, 2, \ldots, M-1,\ v = 0, 1, 2, \ldots, N-1.$

- Inverse discrete Fourier transform:

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

for $x = 0, 1, 2, \ldots, M-1,\ y = 0, 1, 2, \ldots, N-1.$

Fourier transform

- $F(u,v)$ can also be expressed in polar coordinates as

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

where

$$|F(u,v)| = \left(\Re^2\{F(u,v)\} + \Im^2\{F(u,v)\}\right)^{1/2}$$

is called the *magnitude* or *spectrum* of the Fourier transform, and

$$\phi(u,v) = \tan^{-1}\left(\frac{\Im\{F(u,v)\}}{\Re\{F(u,v)\}}\right)$$

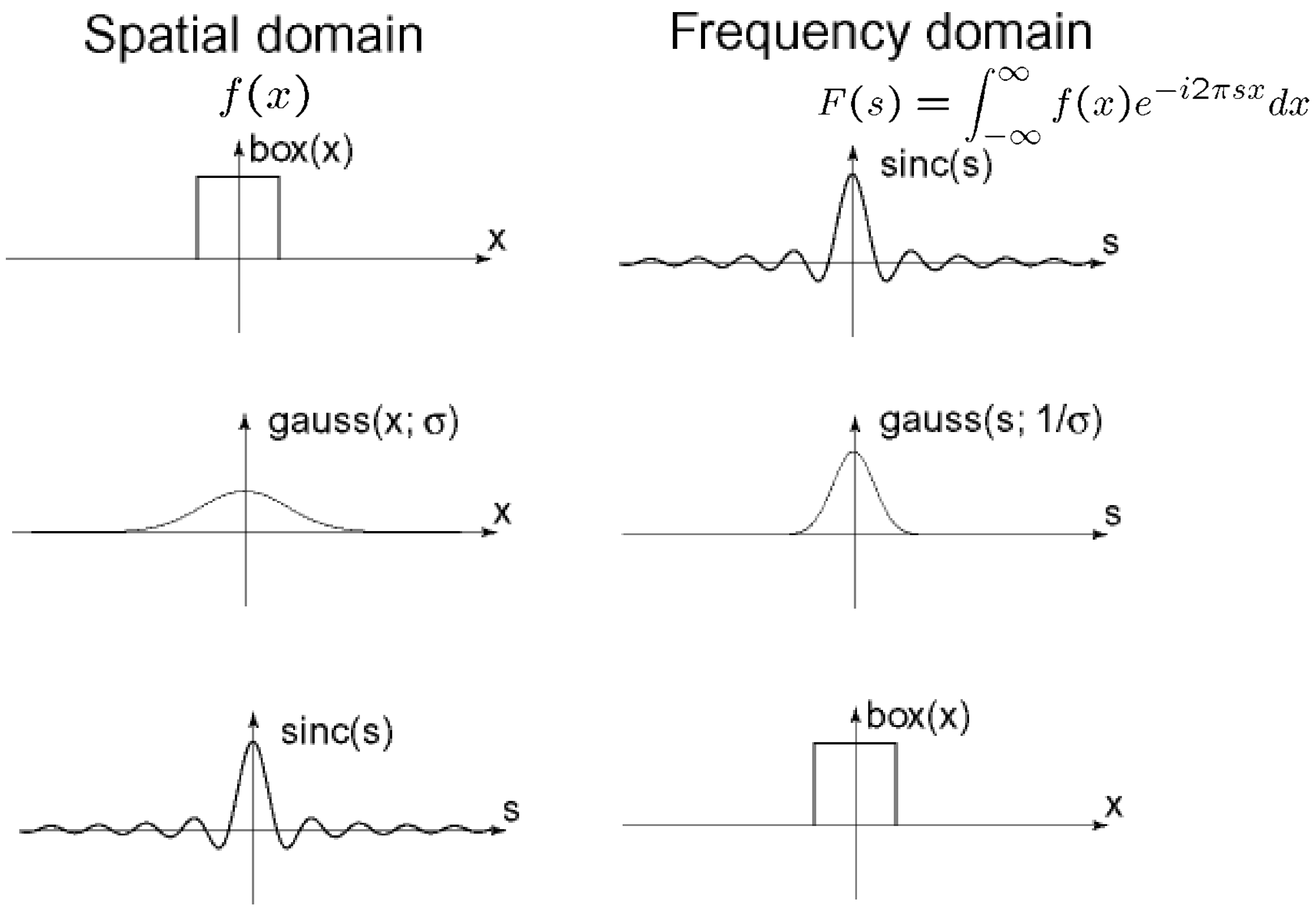
is called the *phase angle* or *phase spectrum*.

- $\Re\{F(u,v)\}$ and $\Im\{F(u,v)\}$ are the real and imaginary parts of $F(u,v)$, respectively.

Fourier transform

- The spectrum need not be interpreted as an image, but rather as a 2D display of the power in the original image versus the frequency components u and v.
- The value $F(0,0)$ is the average of $f(x,y)$.
- Fourier transform is conjugate symmetric ($F(u,v) = F^*(-u,-v)$) and its spectrum is symmetric about the origin ($|F(u,v)| = |F(-u,-v)|$) (when $f(x,y)$ is real).

Fourier transform



From Selim Aksoy@Bilkent

Adapted from Alexei Efros, CMU

Fourier transform

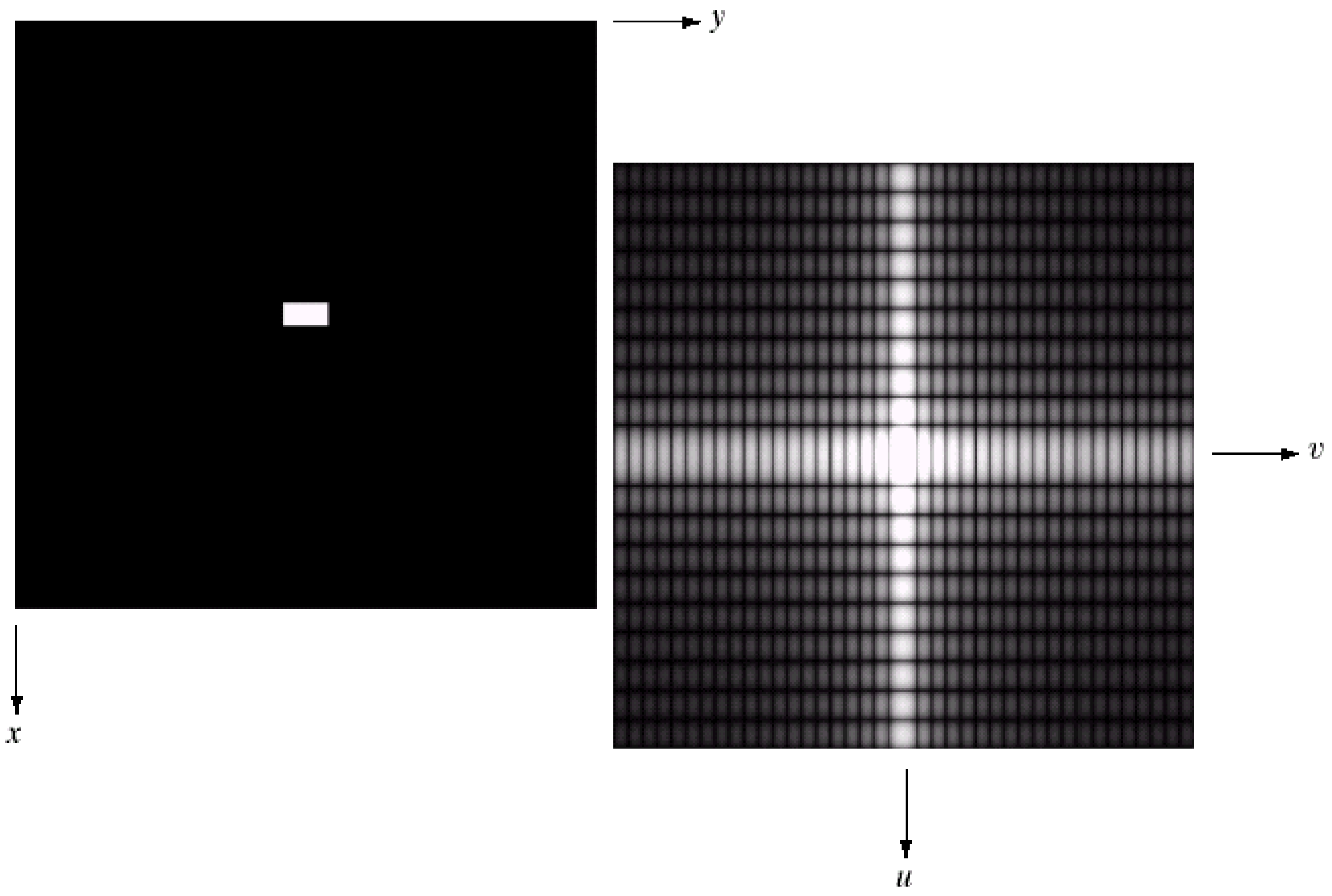


FIGURE 4.3 (a) Image of a 20 × 40 white rectangle on a black background of size 512 × 512 pixels. (b) Centered Fourier spectrum shown after application of the log transformation given in Eq. (3.2-2). Compare with Fig. 4.2.

Adapted from Gonzales and Woods

Fourier transform

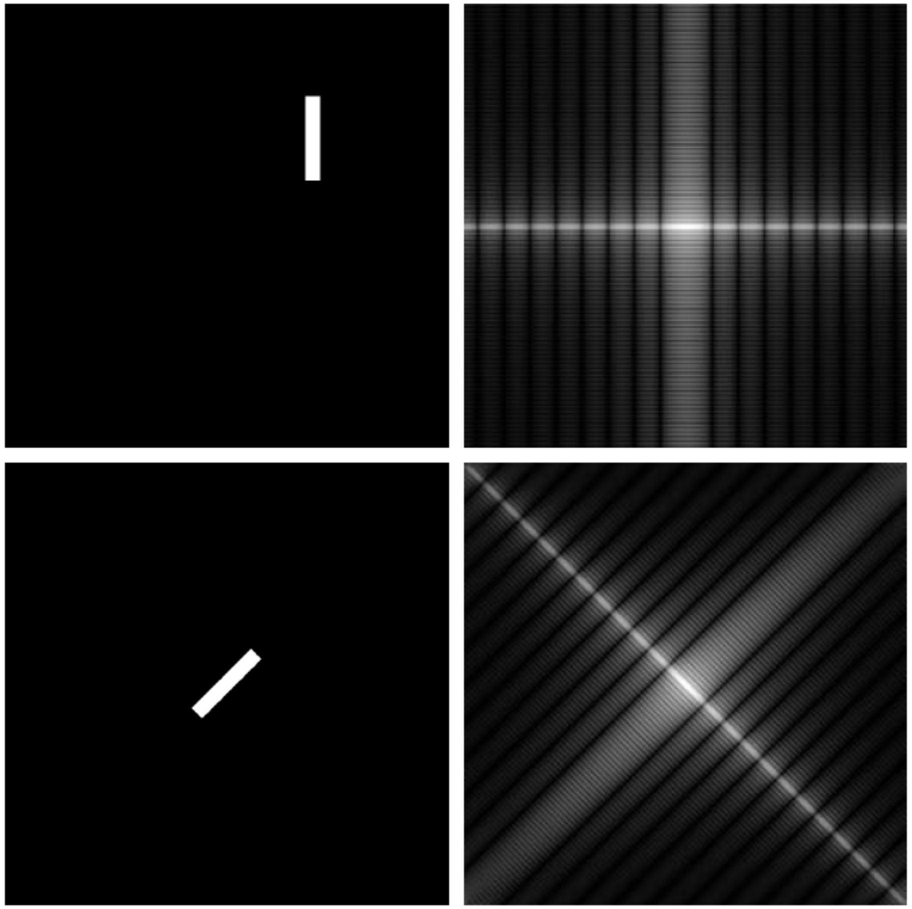


FIGURE 4.25 (a) The rectangle in Fig. 4.24(a) translated, and (b) the corresponding spectrum. (c) Rotated rectangle, and (d) the corresponding spectrum. The spectrum corresponding to the translated rectangle is identical to the spectrum corresponding to the original image in Fig. 4.24(a).

Fourier transform

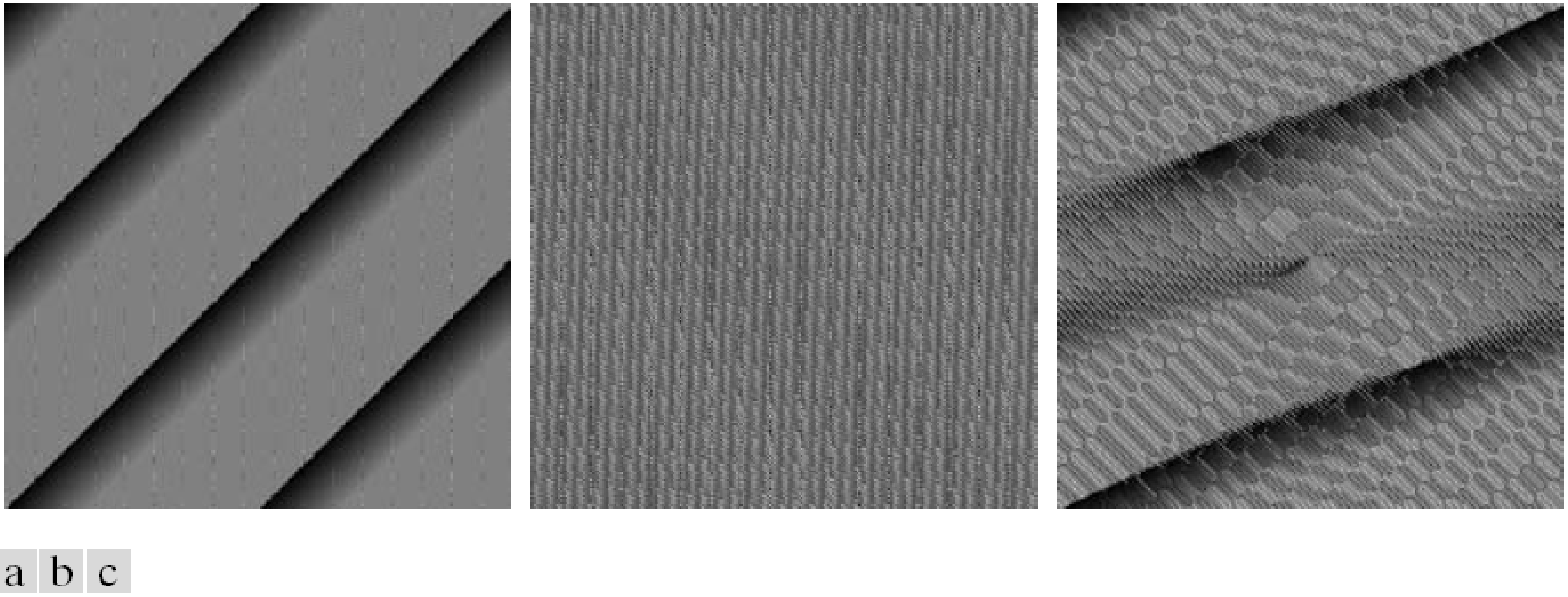


FIGURE 4.26 Phase angle array corresponding (a) to the image of the centered rectangle in Fig. 4.24(a), (b) to the translated image in Fig. 4.25(a), and (c) to the rotated image in Fig. 4.25(c).

Fourier transform -Phase

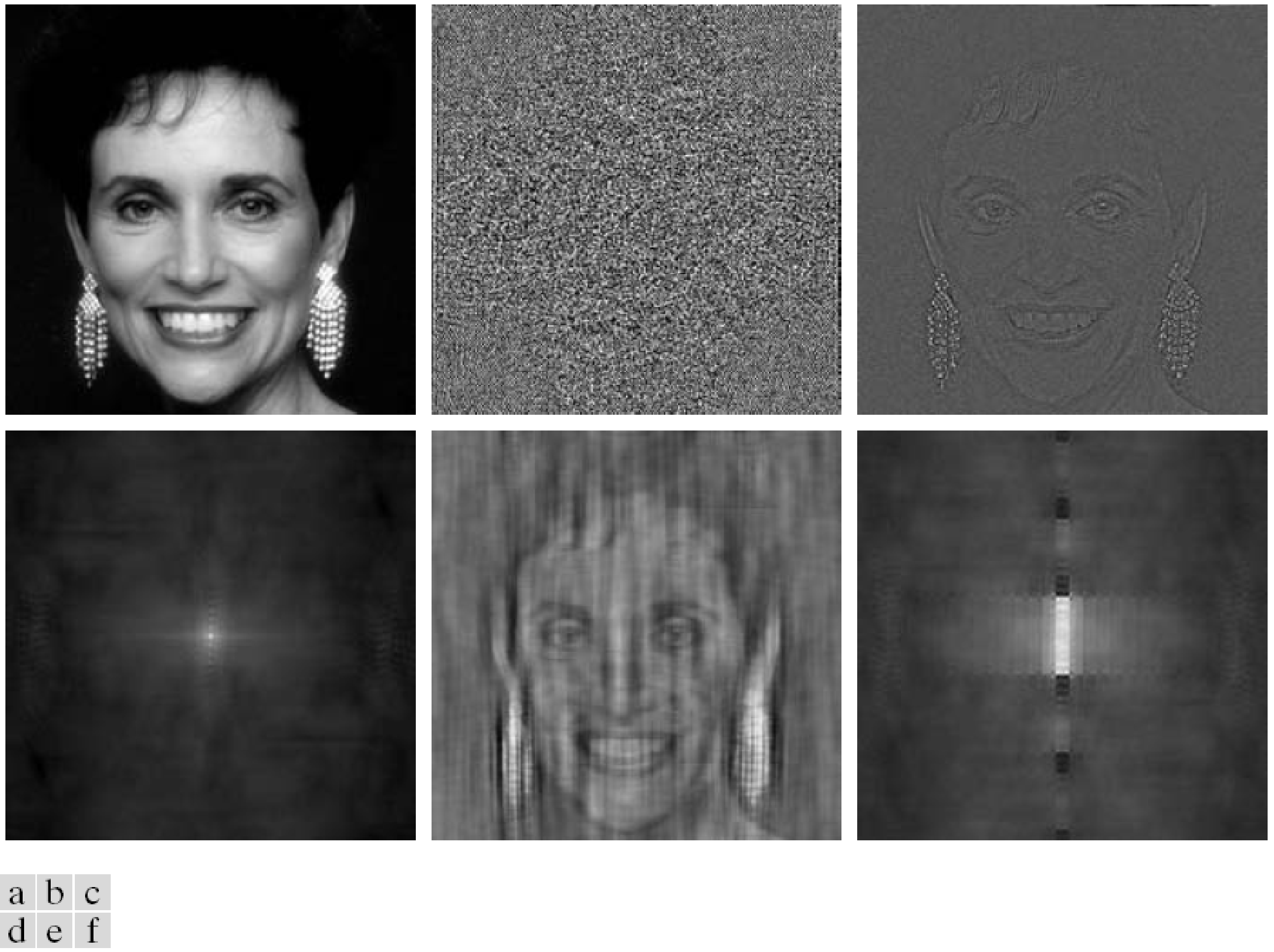


FIGURE 4.27 (a) Woman. (b) Phase angle. (c) Woman reconstructed using only the phase angle. (d) Woman reconstructed using only the spectrum. (e) Reconstruction using the phase angle corresponding to the woman and the spectrum corresponding to the rectangle in Fig. 4.24(a). (f) Reconstruction using the phase of the rectangle and the spectrum of the woman.

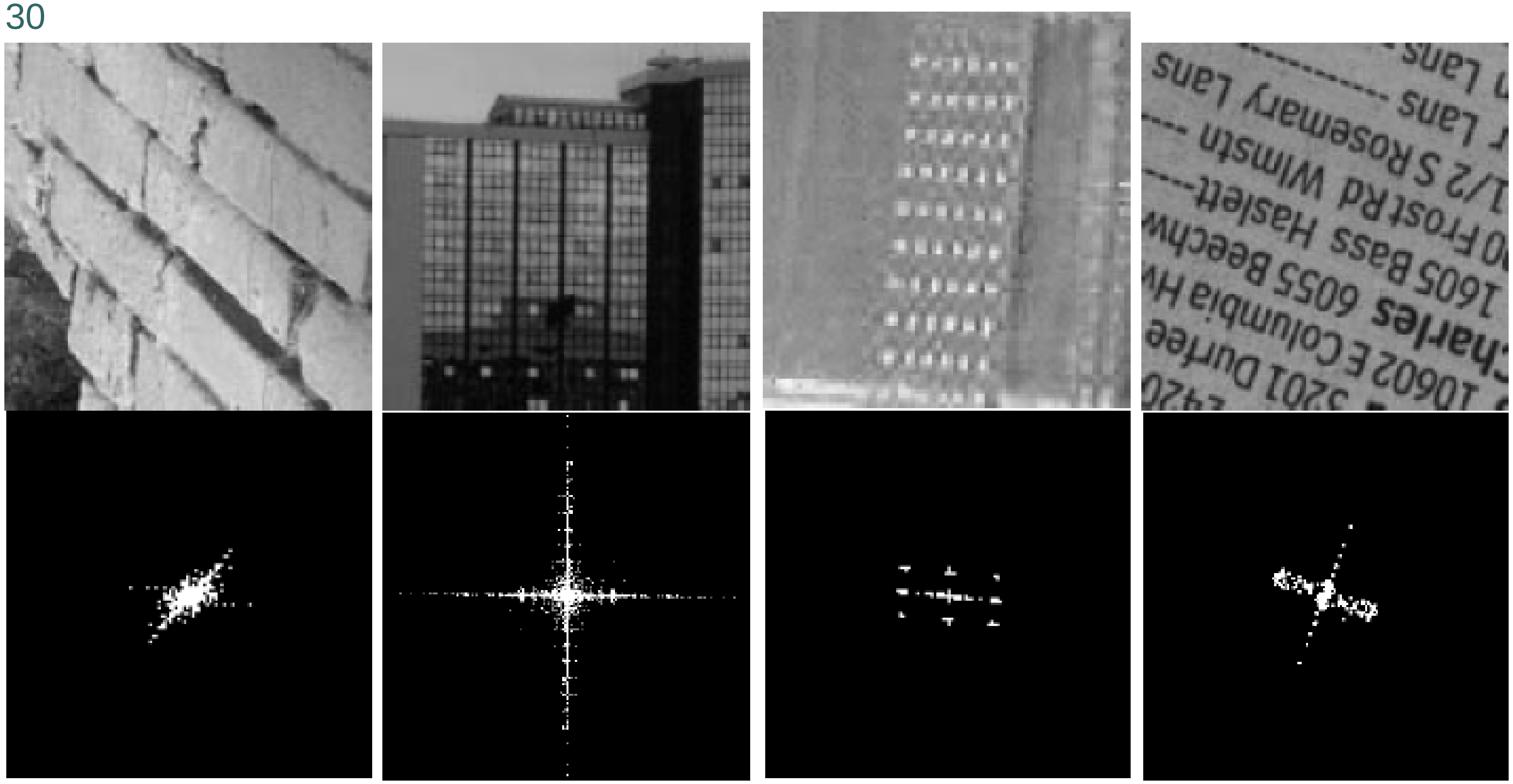
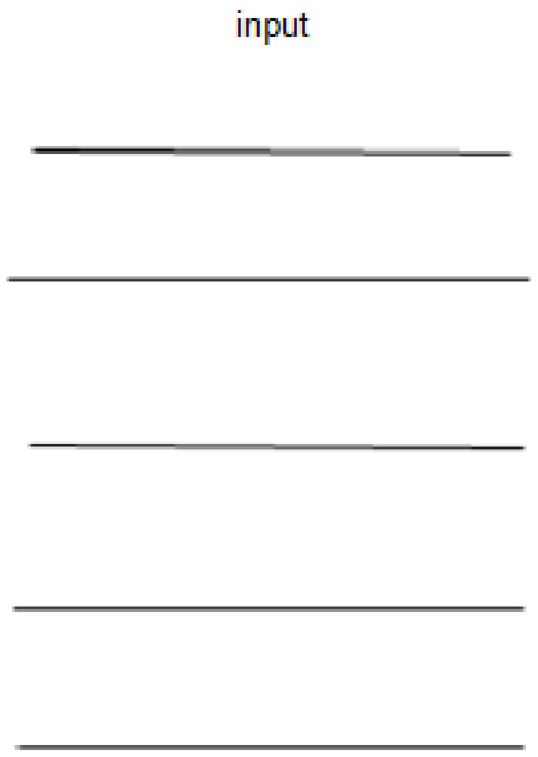
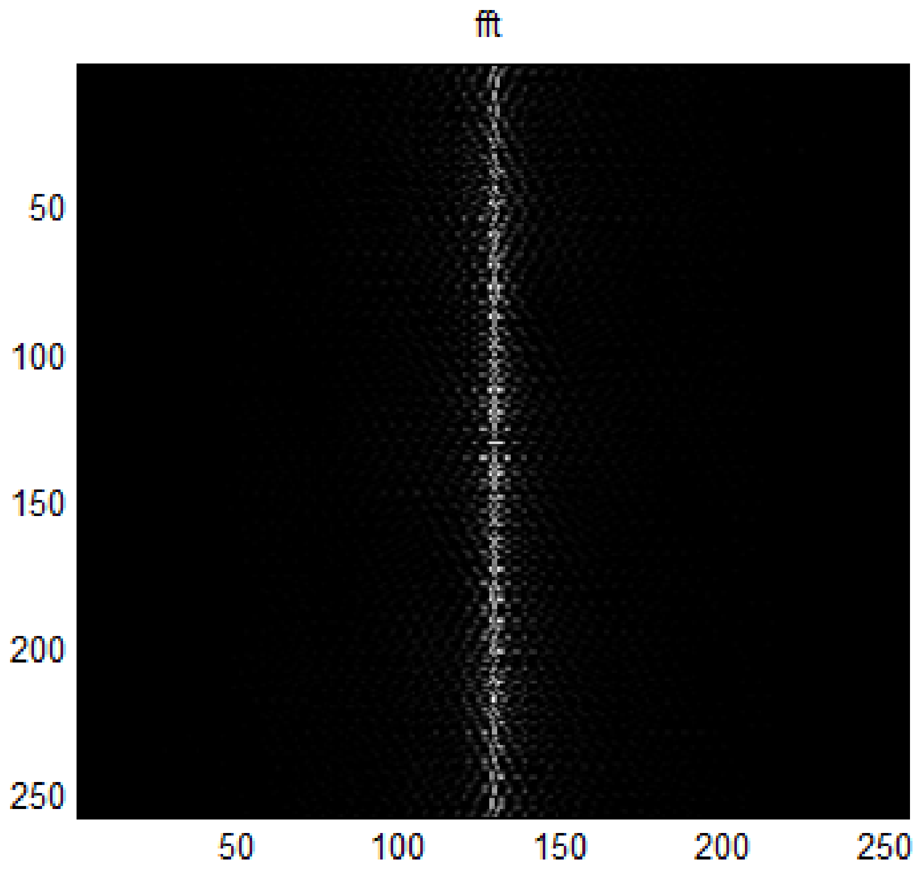


Figure 5.42: Four images (above) and their power spectrums (below). The power spectrum of the brick texture shows energy in many sinusoids of many frequencies, but the dominant direction is perpendicular to the 6 dark seams running about 45 degrees with the X-axis. There is noticable energy at 0 degrees with the X axis, due to the several short vertical seams. The power spectrum of the building shows high frequency energy in waves along the X-direction and the Y-direction. The third image is an aerial image of an orchard: the power spectrum shows the rows and columns of the orchard and also the "diagnonal rows". The far right image, taken from a phone book, shows high frequency power at about 60^o with the X-axis, which represents the texture in the lines of text. Energy is spread more broadly in the perpendicular direction also in order to model the characters and their spacing.

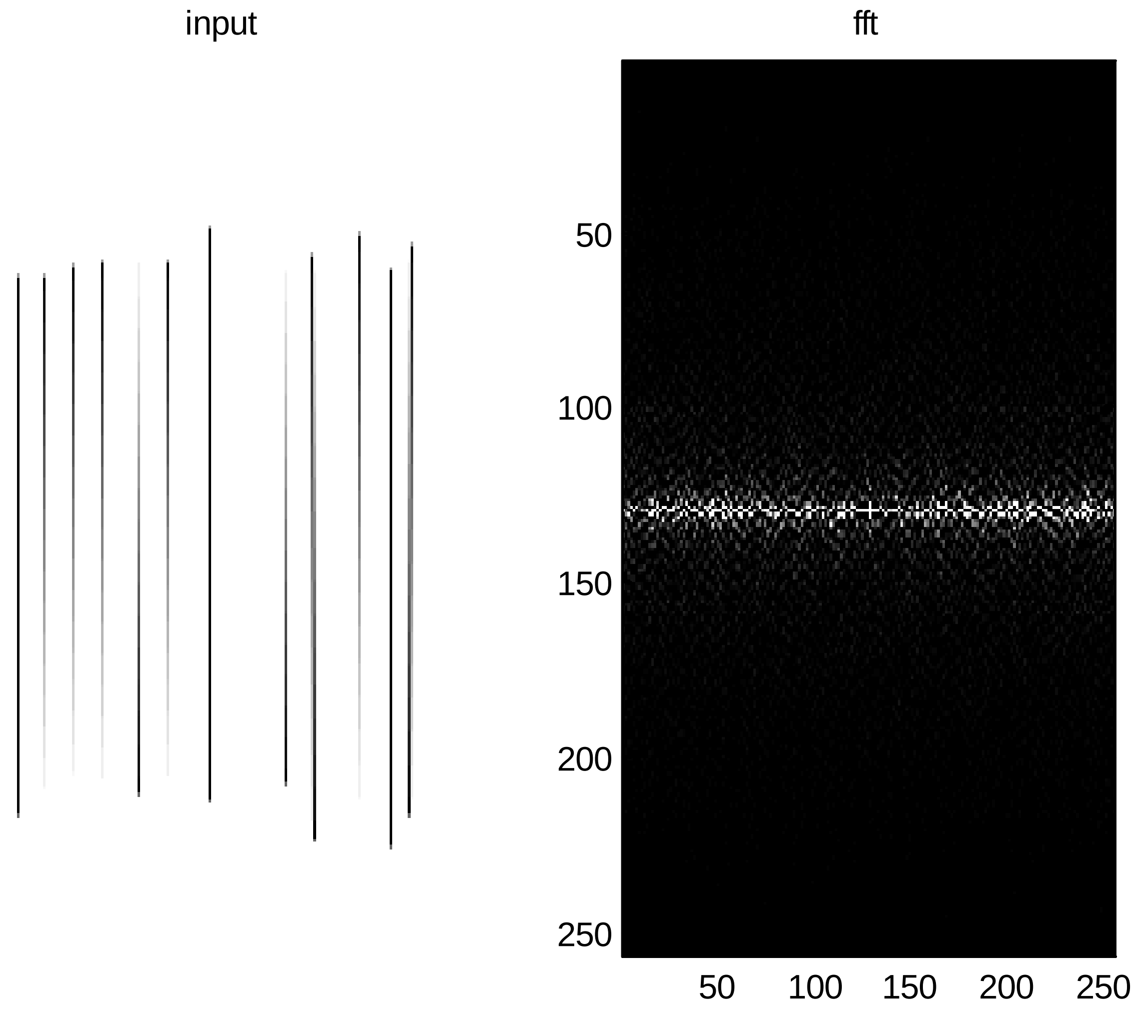
Adapted from Shapiro and Stockman

Fourier Transform

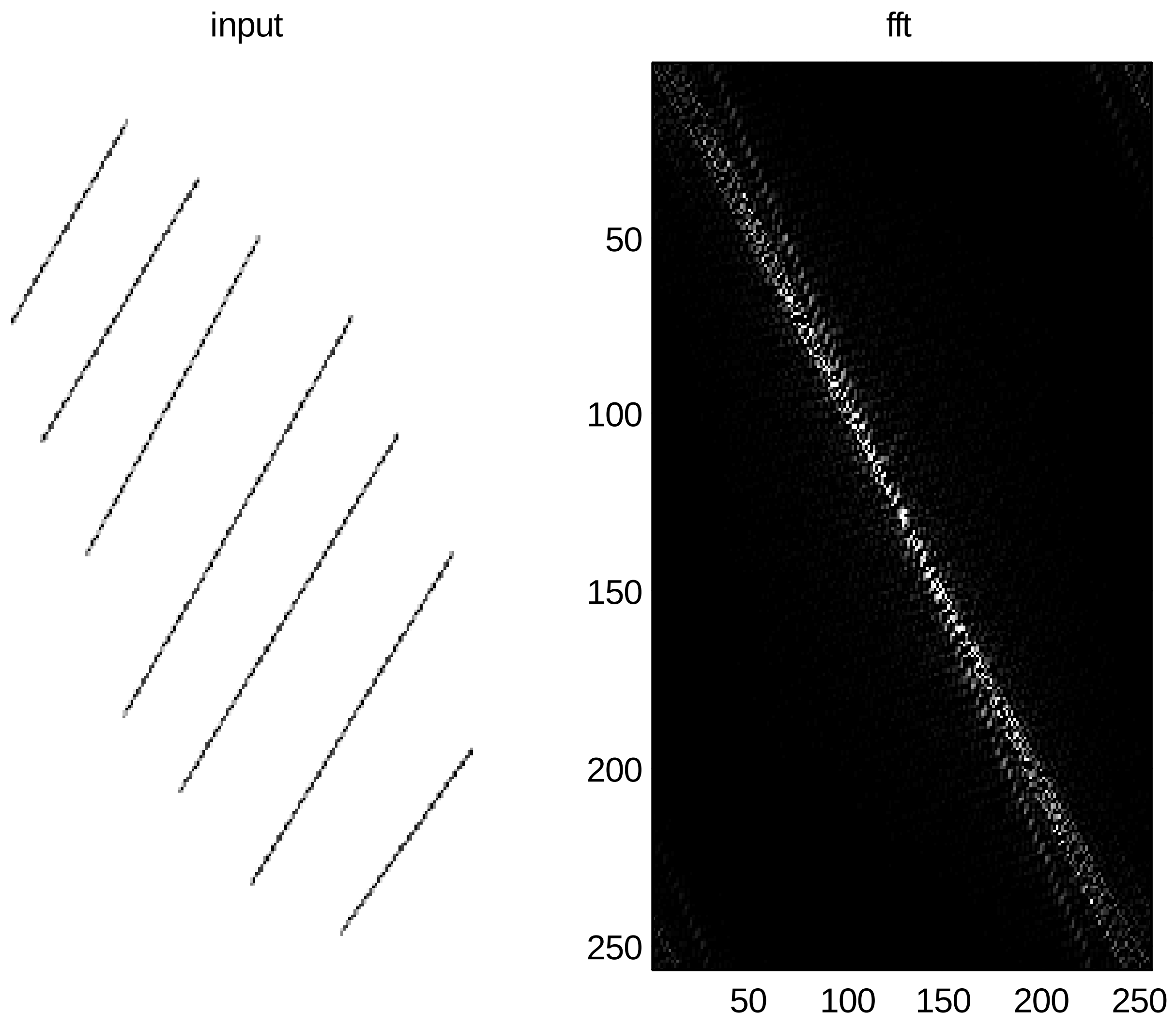




Fourier Transform



Fourier Transform



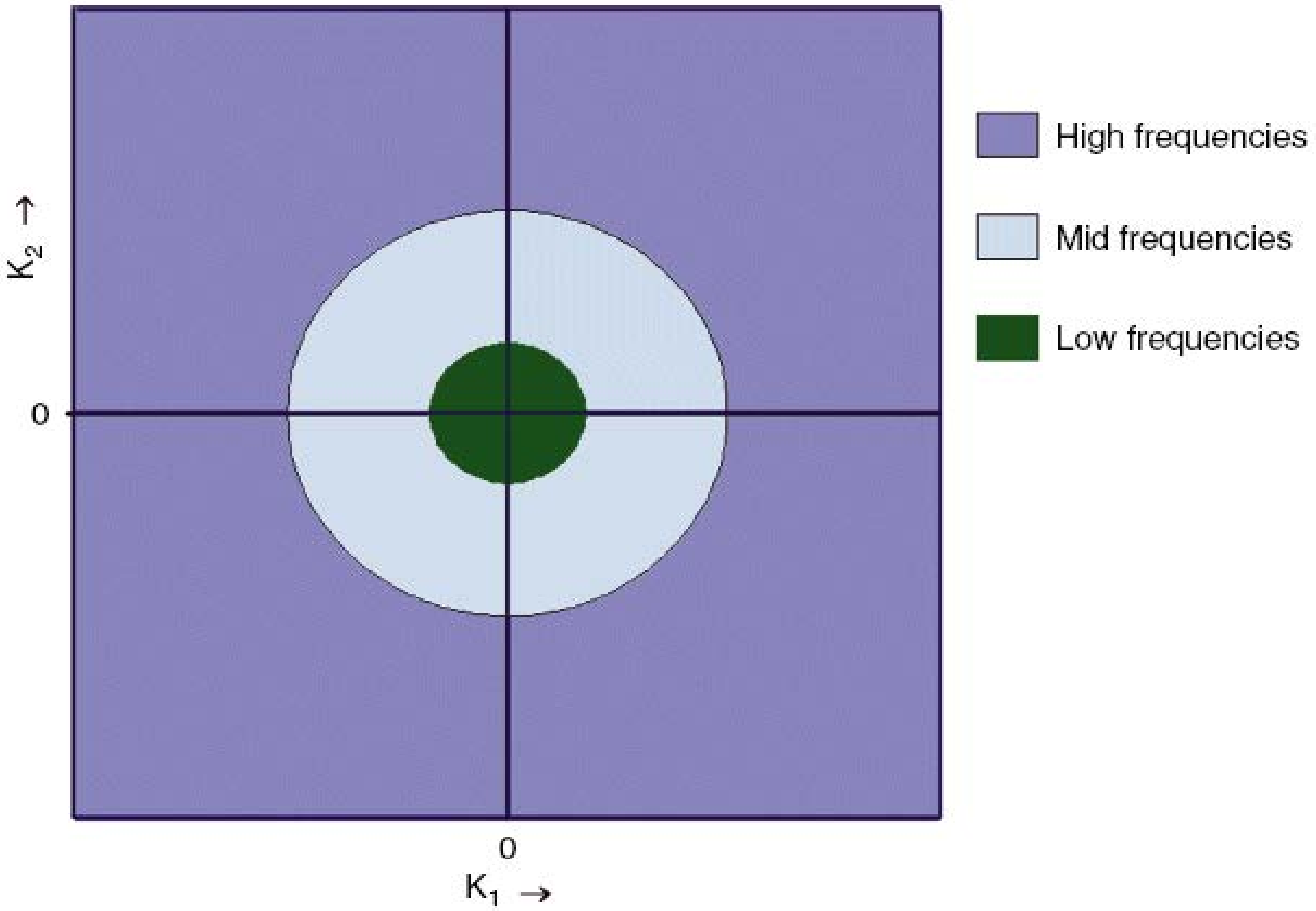


Figure 7.14 *Location of low, mid and high frequency components of the two-dimensional amplitude spectrum.*

Frequency Domain Filtering

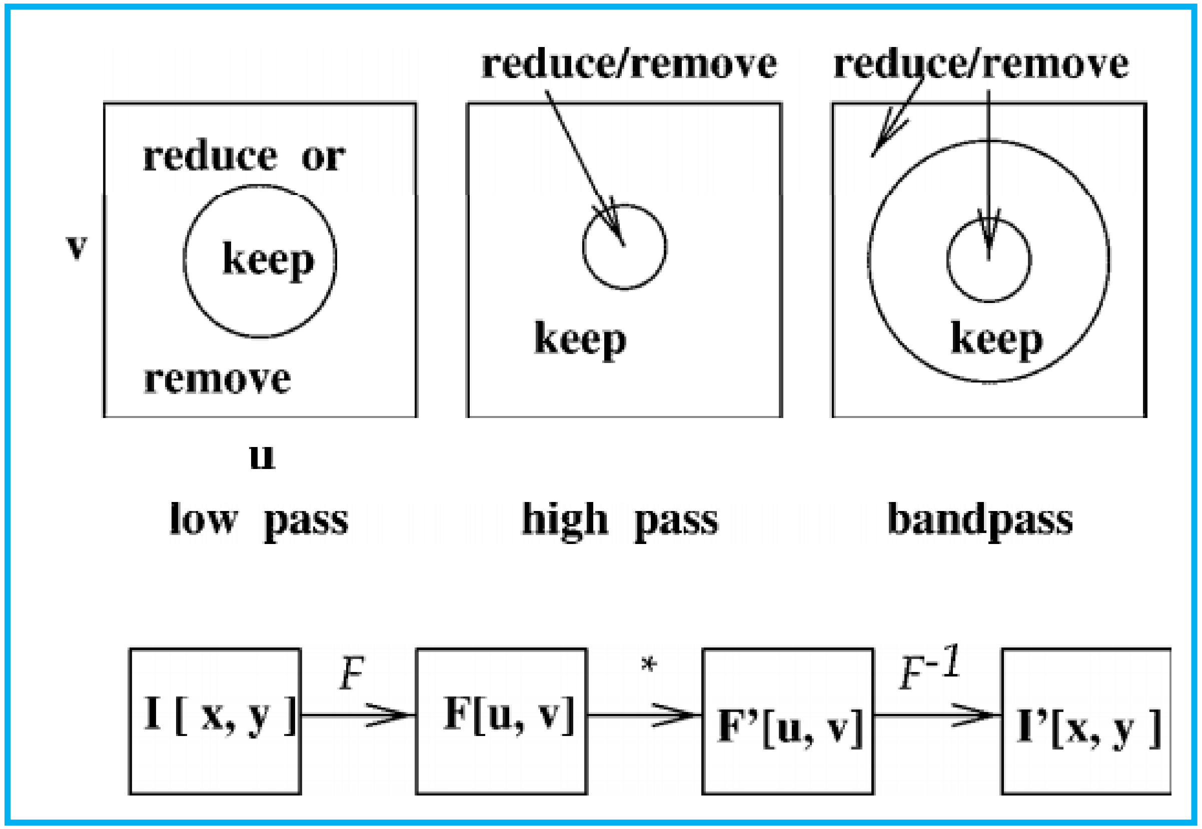
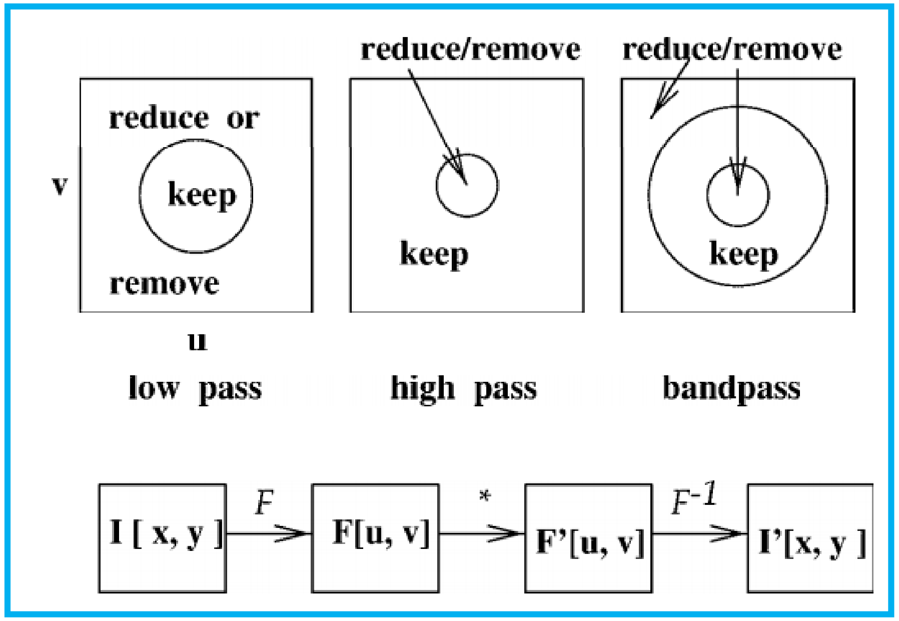


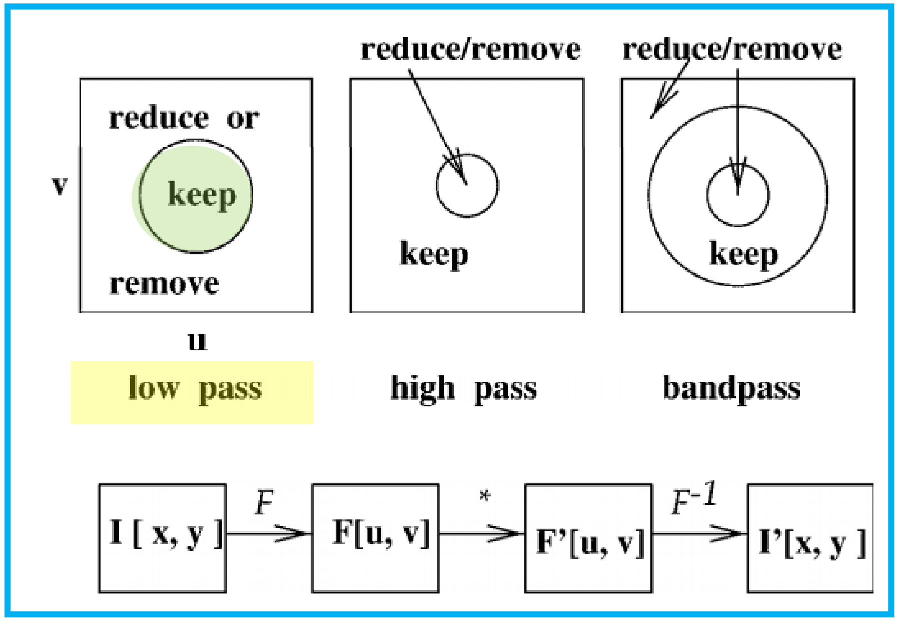
Figure 5.43: Bandpass filtering can be performed by a Fourier transformation into the frequency domain (**F**[**u**, **v**]) followed by multiplication (*) by a bandpass filter. The multiplication can set the "coefficients" u, v of various frequencies to zero, as shown in the top row. This modified frequency representation is then inverse transformed to obtain the modified spatial image **I**′[**x**, **y**].

Frequency Domain Filtering



Bandpass filtering is a common image processing operation performed in the frequency domain and is sketched in Figure 5.43. The **DFT** is used to transform the image into its frequency representation where the coefficients for some frequencies are reduced, perhaps to zero, while others are preserved.

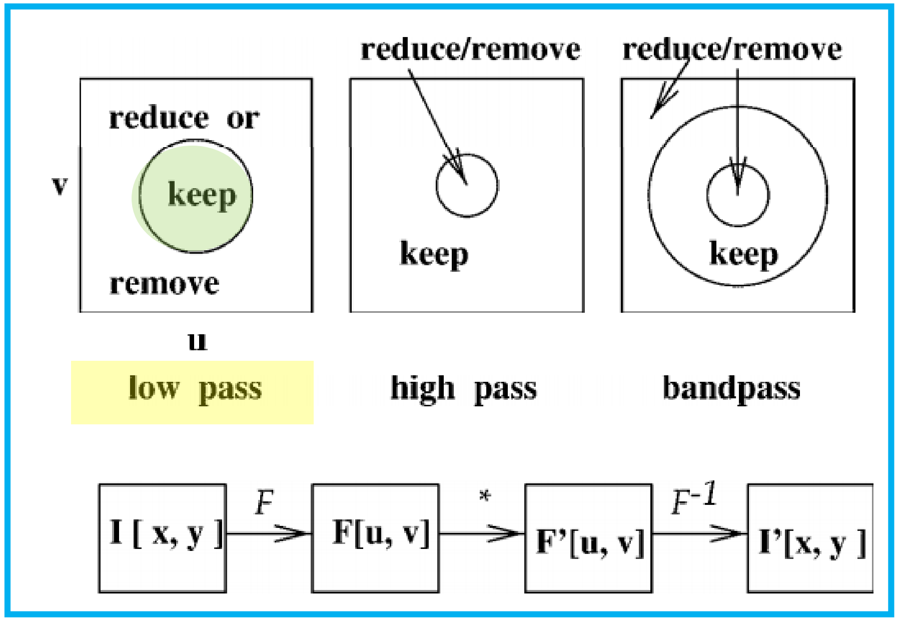
Frequency Domain Filtering



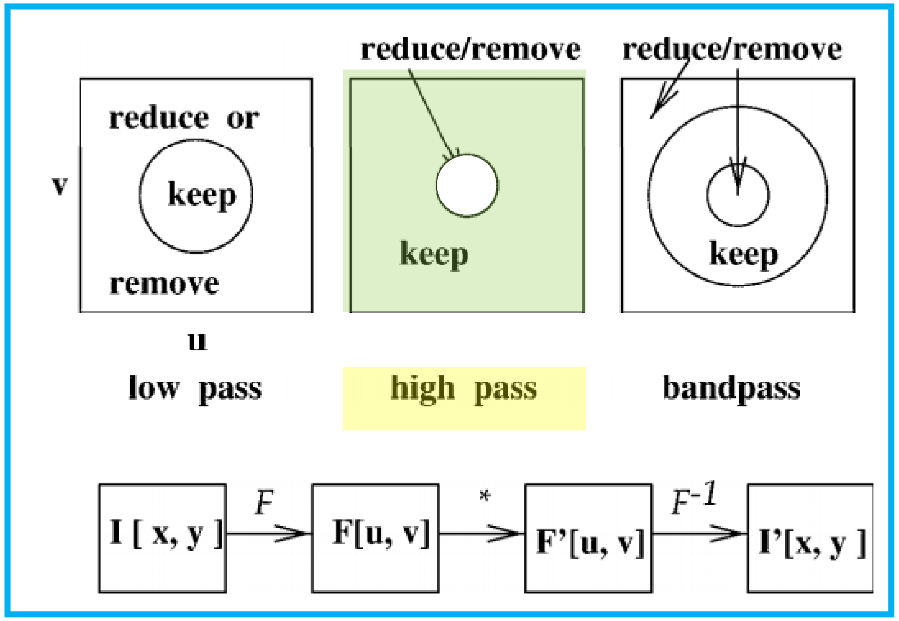
A sketch of the *low pass filter* is given at the left of Figure 5.43; intuitively, the high frequencies are erased and then the altered frequency representation is inverse transformed via Equation 5.35 to obtain a smoothed version of the original image. Instead of *erasing* elements of the frequency representation, we can take the dot product of $F[u, v]$ with a 2D Gaussian, which will weight the low frequency components high and weight the high frequency components low. Figure 5.43 also shows how the frequency representation would be altered to accomplish high pass and bandpass filtering.

The theory of convolution provides more insight to these operations.

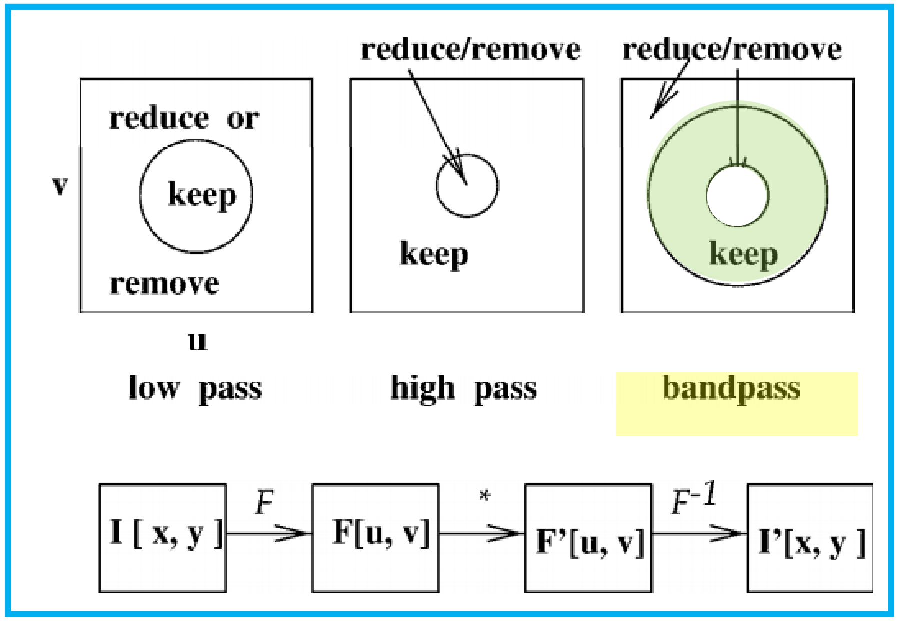
Low pass Filtering

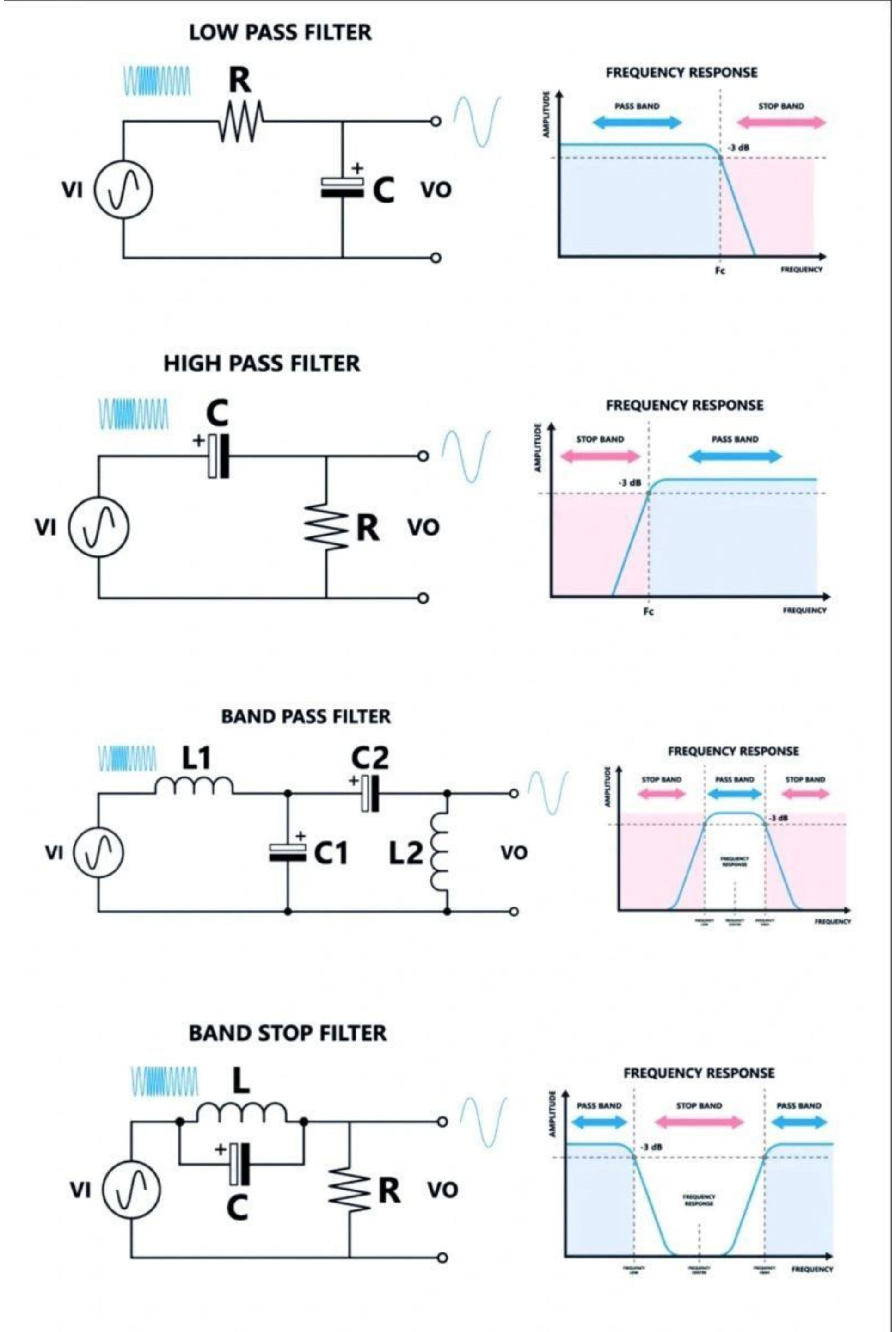


High pass Filtering

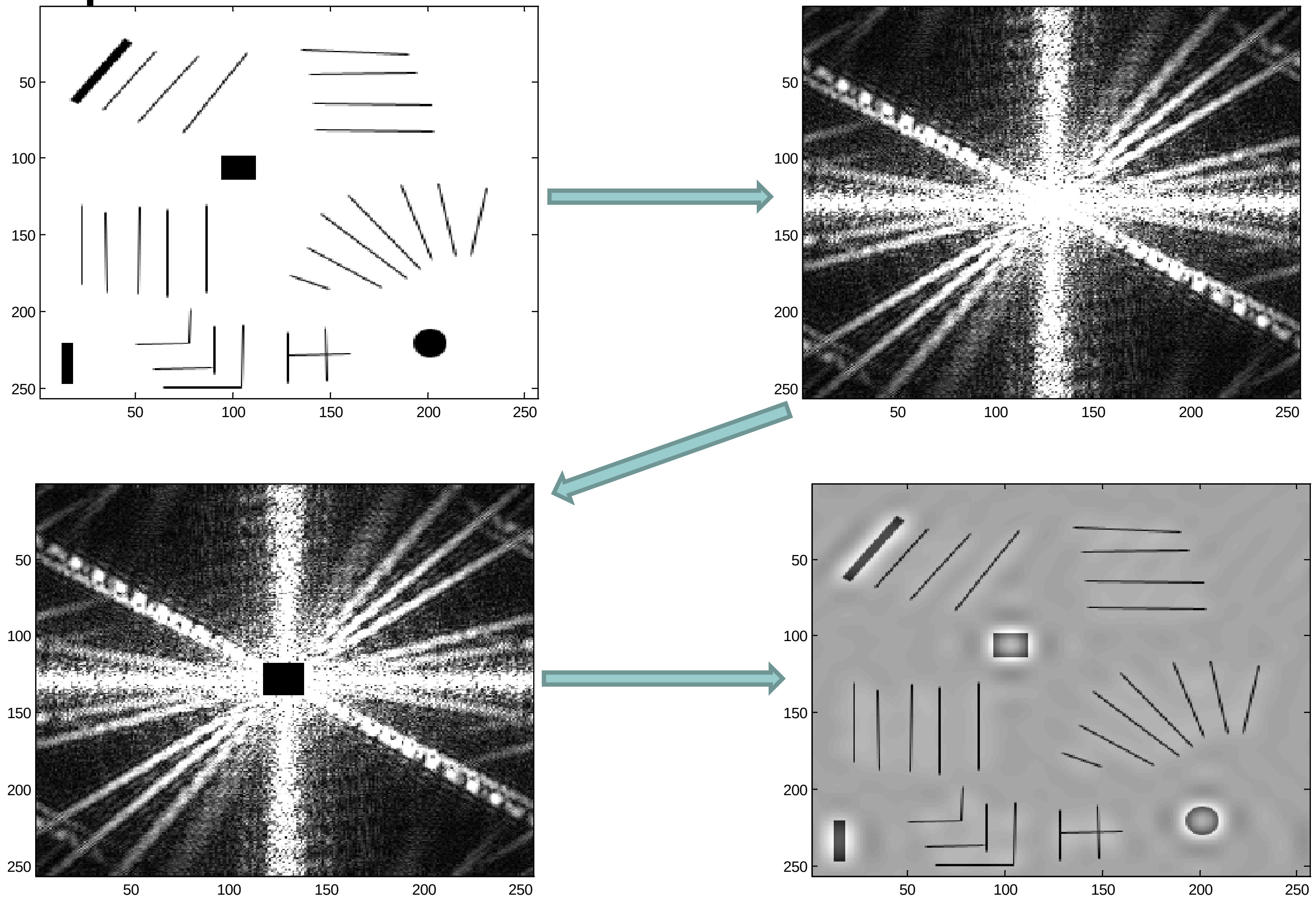


Bandpass Filtering



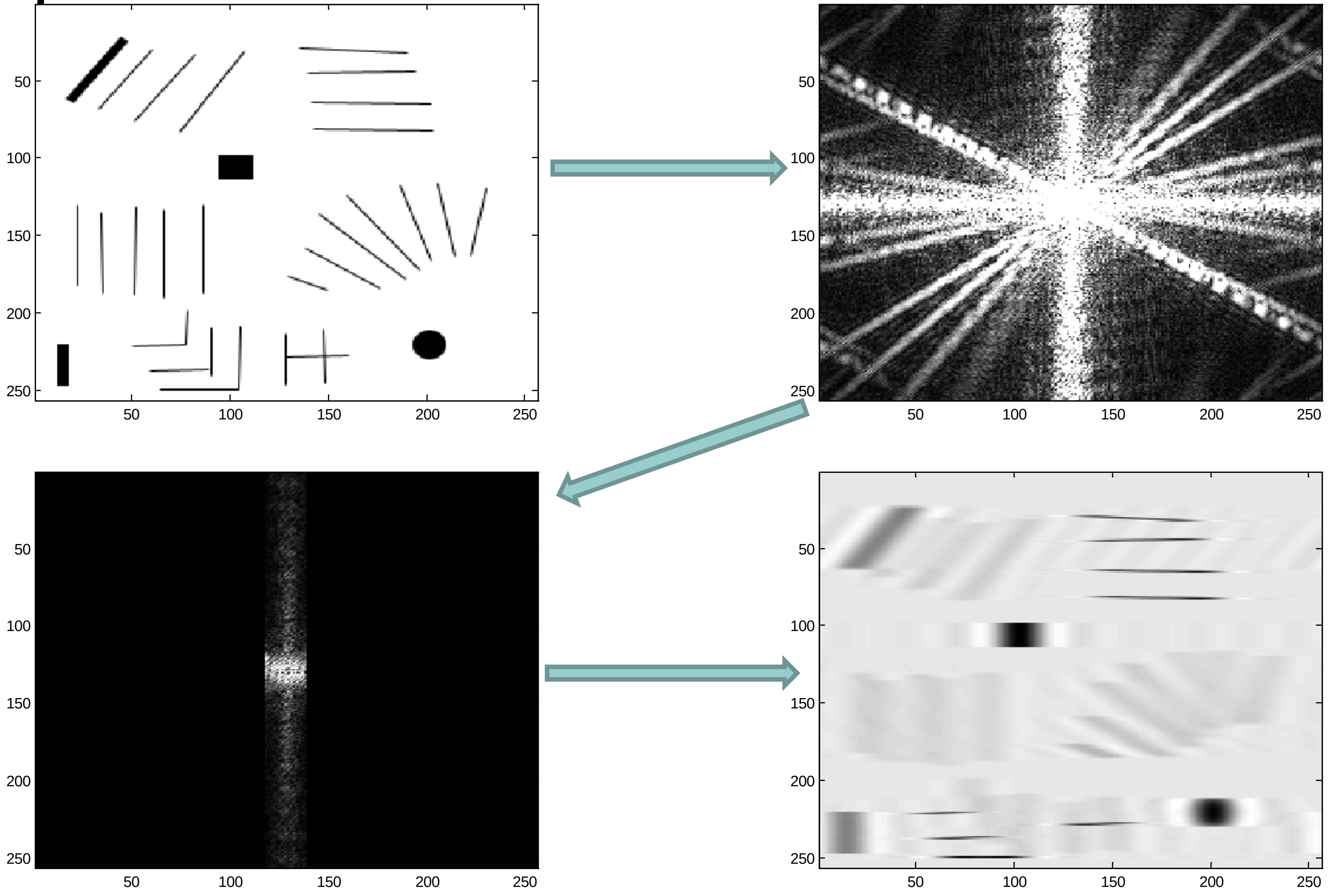

LOW PASS FILTER
R
C
VI
VO
FREQUENCY RESPONSE
AMPLITUDE
PASS BAND
STOP BAND
-3 dB
Fc
FREQUENCY
HIGH PASS FILTER
C
R
VI
VO
FREQUENCY RESPONSE
AMPLITUDE
STOP BAND
PASS BAND
-3 dB
Fc
FREQUENCY
BAND PASS FILTER
L1
C2
C1
L2
VI
VO
FREQUENCY RESPONSE
AMPLITUDE
STOP BAND
PASS BAND
STOP BAND
-3 dB
FREQUENCY RESPONSE
FREQUENCY
BAND STOP FILTER
L
C
R
VI
VO
FREQUENCY RESPONSE
AMPLITUDE
PASS BAND
STOP BAND
PASS BAND
-3 dB
FREQUENCY RESPONSE
FREQUENCY

Frequency domain filtering

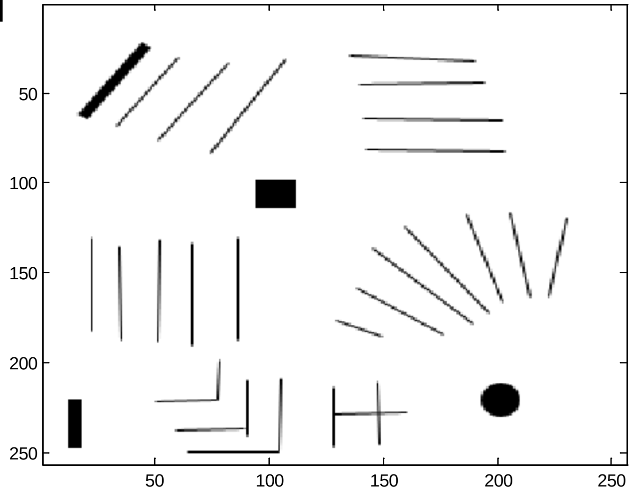


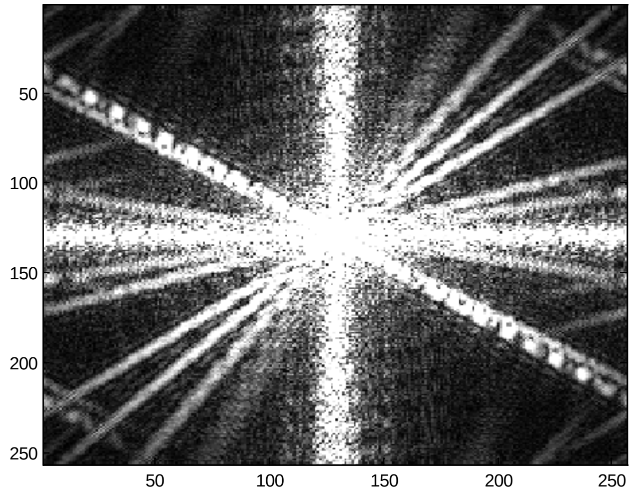
High pass

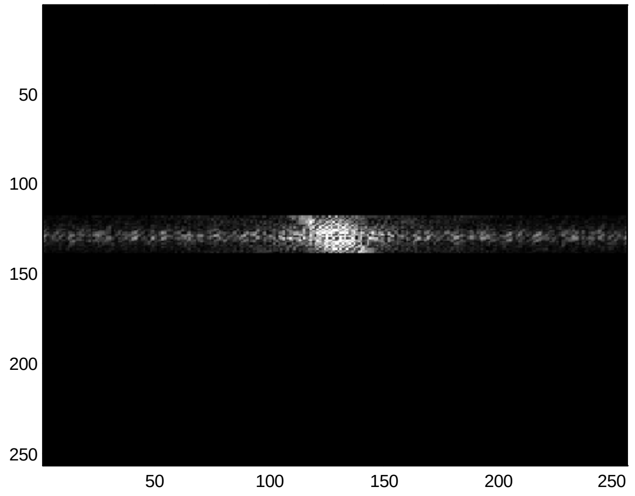
Frequency domain filtering

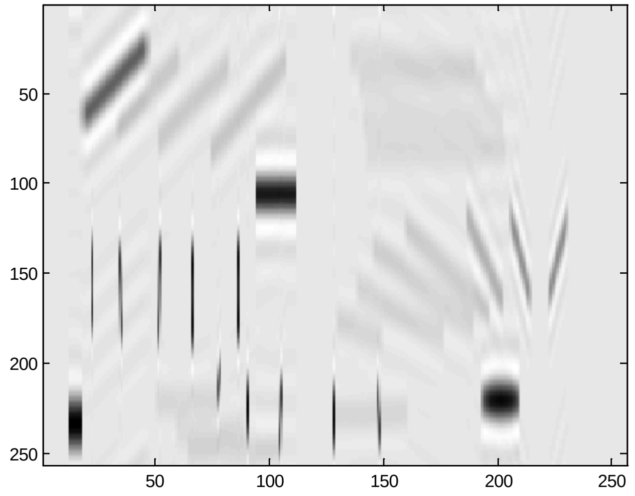


Frequency domain filtering









Convolution theorem

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

The Fourier transform of a convolution is the pointwise product of Fourier transforms

$$f * g = \mathcal{F}^{-1}\big\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\big\}$$

Convolution theorem

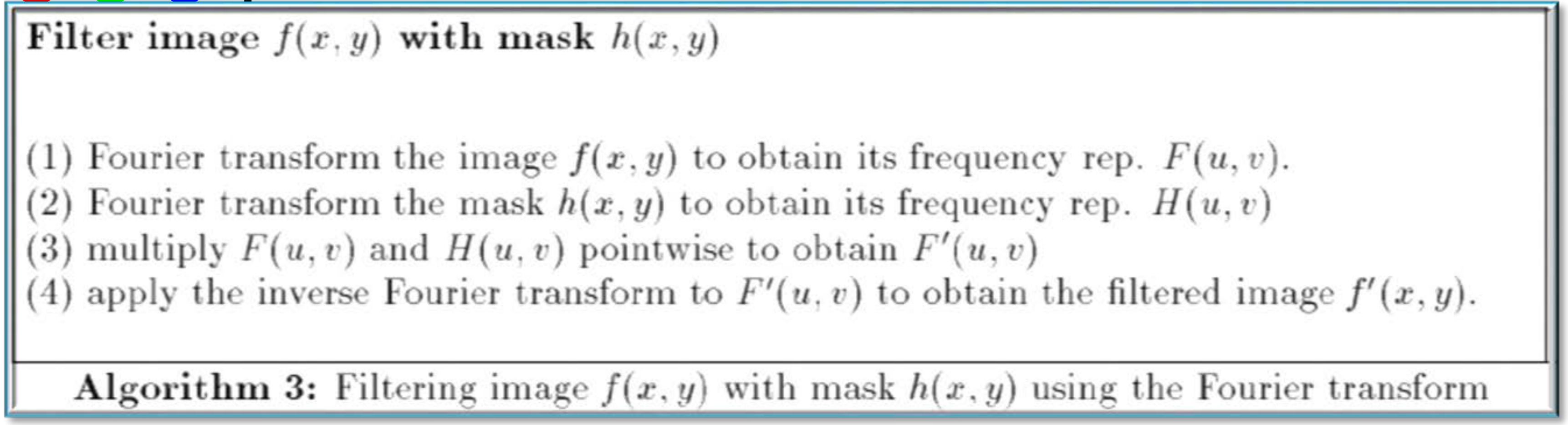
The *convolution theorem* tells us that
convolution of two functions in the spatial domain is equivalent to pointwise multiplication of their frequency representations.

If $f(x, y)$ and $h(x, y)$ are well-behaved functions of spatial parameters x, y, then

$$\mathbf{F}(f(x,y)\star h(x,y)) \equiv \mathbf{F}((f\star h)(x,y)) \ = \ \mathbf{F}(f(x,y))\mathbf{F}(h(x,y)) = \mathbf{F}(u,v)\mathbf{H}(u,v),$$

where $\mathbf{F}$ is the Fourier transform operator and $\star$ is the convolution operator.

Frequency domain filtering



Filter image $f(x,y)$ with mask $h(x,y)$

(1) Fourier transform the image $f(x,y)$ to obtain its frequency rep. $F(u,v)$.
(2) Fourier transform the mask $h(x,y)$ to obtain its frequency rep. $H(u,v)$
(3) multiply $F(u,v)$ and $H(u,v)$ pointwise to obtain $F'(u,v)$
(4) apply the inverse Fourier transform to $F'(u,v)$ to obtain the filtered image $f'(x,y)$.

Algorithm 3: Filtering image $f(x,y)$ with mask $h(x,y)$ using the Fourier transform

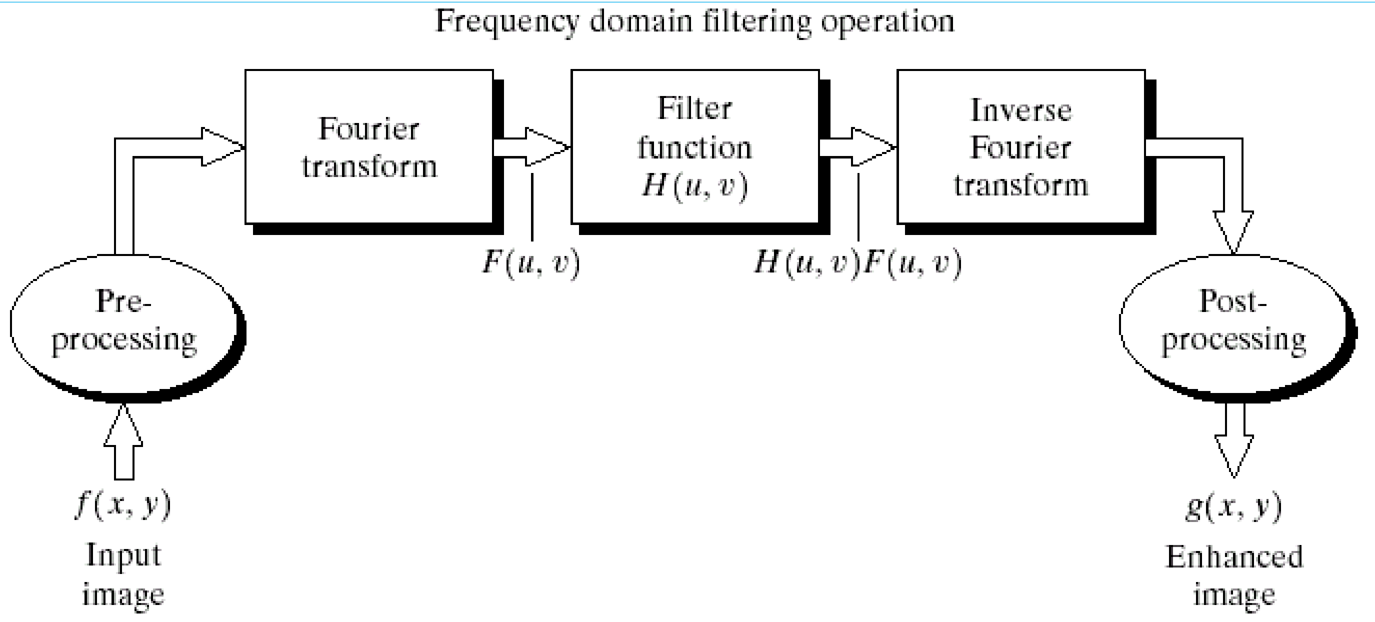
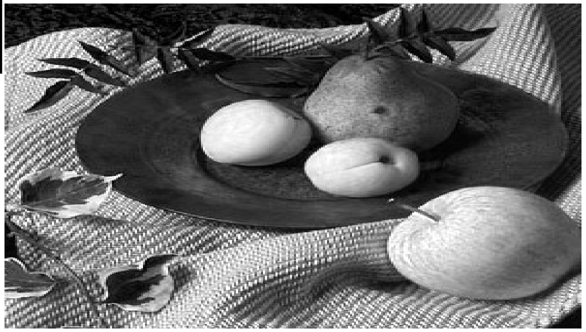
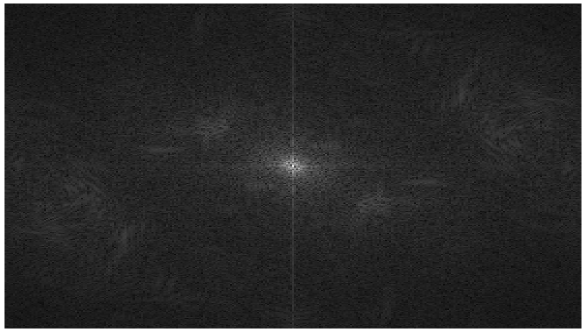


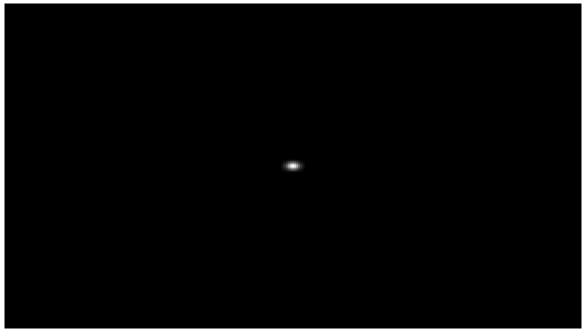
FIGURE 4.5 Basic steps for filtering in the frequency domain.

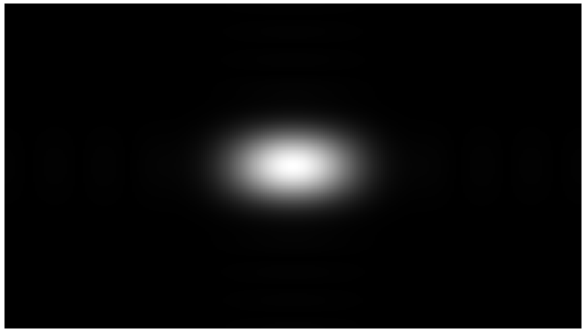
Frequency domain filtering

$f(x,y)$


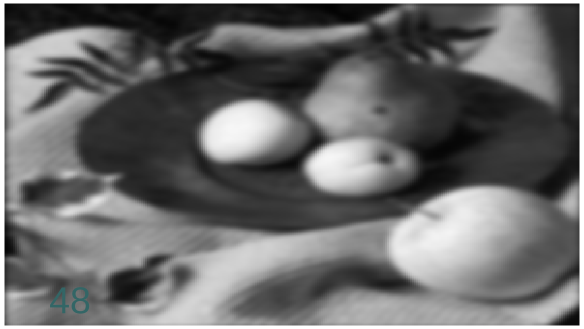

$|F(u,v)|$

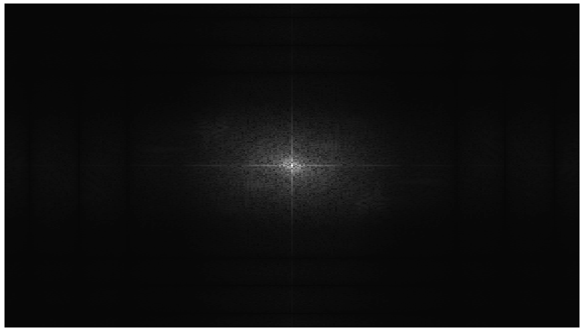
$\star$ $\times$

$h(x,y)$



$|H(u,v)|$

$\Downarrow$ $\Downarrow$

$g(x,y)$


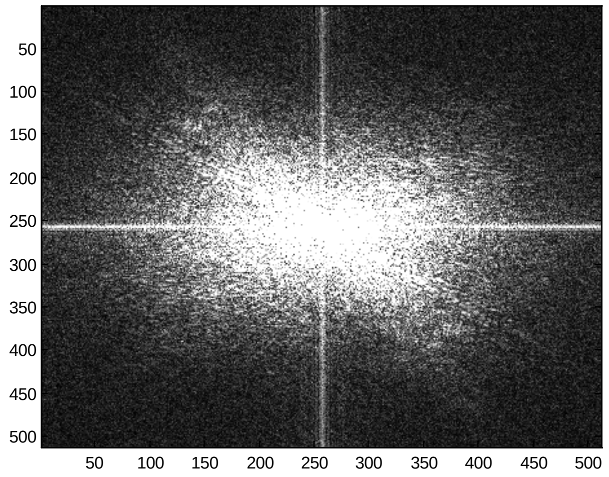

$|G(u,v)|$

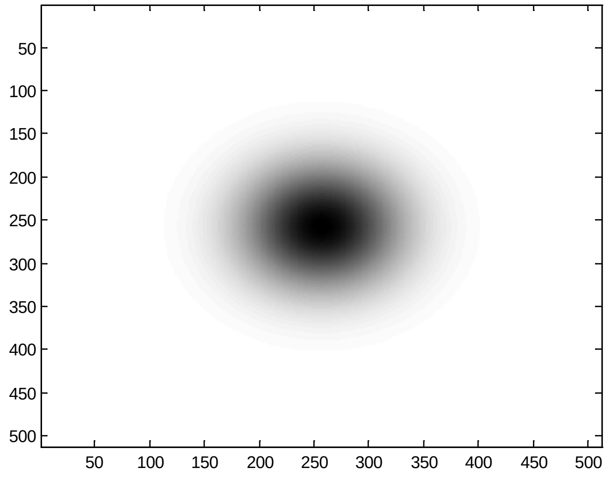
Adapted from Alexei Efros, CMU

High pass filtering

f(x,y)

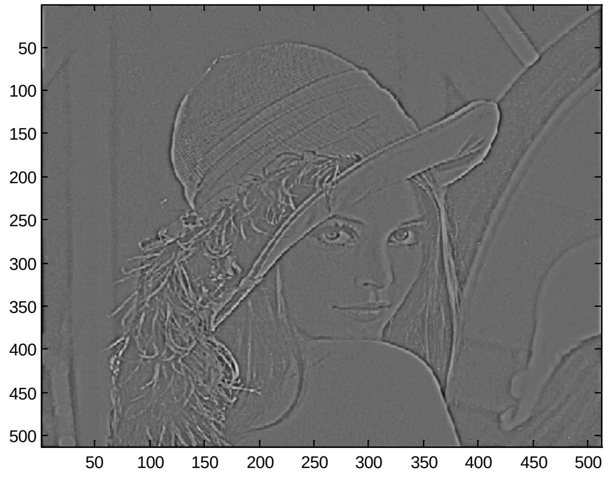
F(u,v)





H(u,v) Highpass filter

Inverted gaussian(u,v)



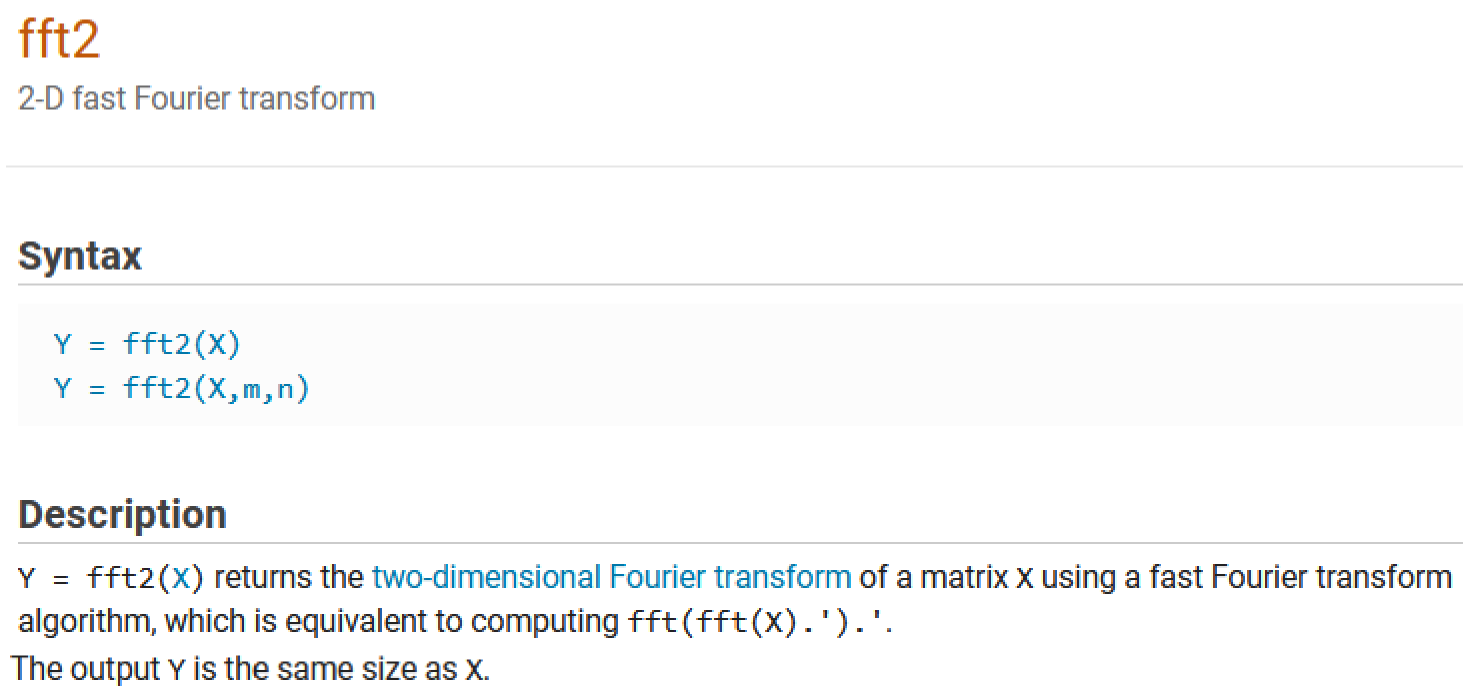
Inverse fft(F(u,v)H(u,v))

MATLAB FUNCTIONS

• **fft2**	2-D fast Fourier transform
• fftn	N-D fast Fourier transform
• **ifft2**	2-D inverse fast Fourier transform
• **fftshift**, **ifftshift**	Shift zero-frequency component to center of spectrum
• Conv2	2-D convolution
• normxcorr2	Normalized 2-D cross-correlation

• And others type help images, help fft2

MATLAB FUNCTIONS



fft2

2-D fast Fourier transform

Syntax

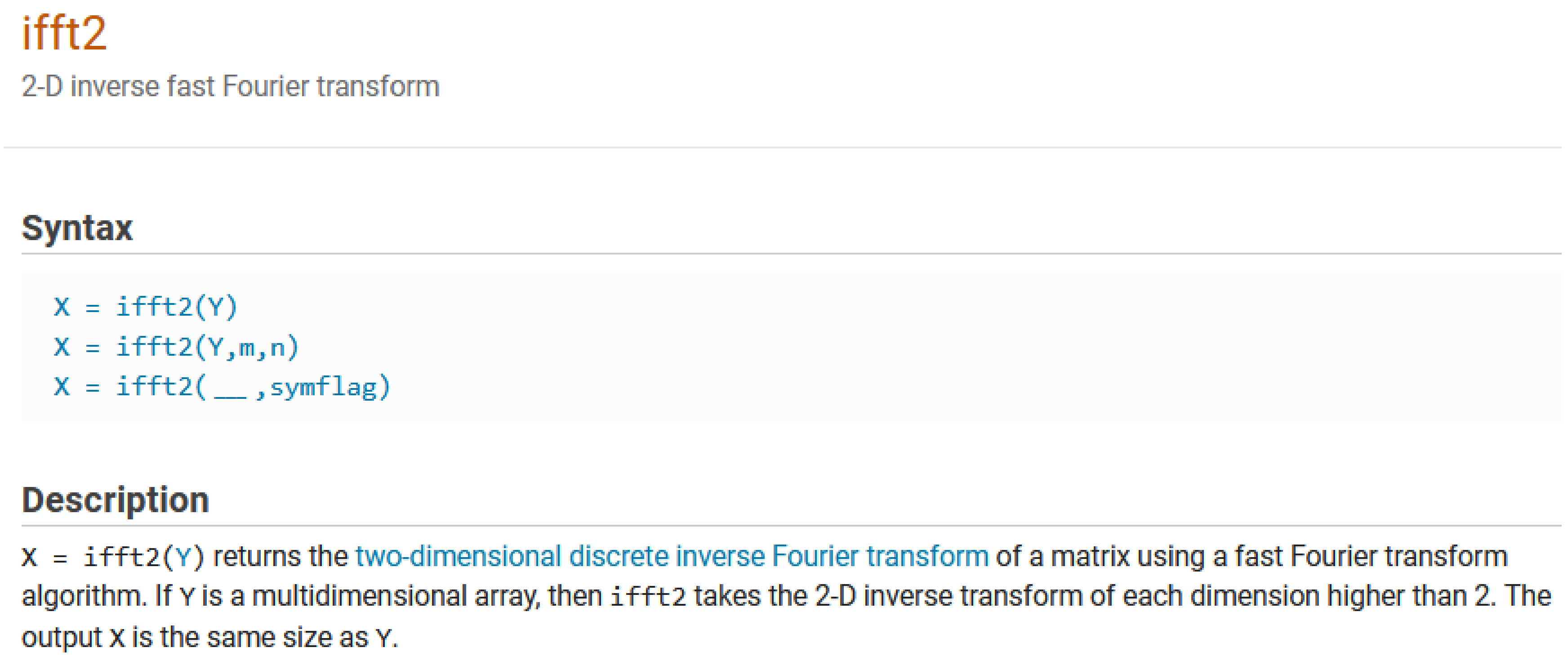
```
Y = fft2(X)
Y = fft2(X,m,n)
```

Description

`Y = fft2(X)` returns the two-dimensional Fourier transform of a matrix `X` using a fast Fourier transform algorithm, which is equivalent to computing `fft(fft(X).').'`.

The output `Y` is the same size as `X`.

MATLAB FUNCTIONS



ifft2

2-D inverse fast Fourier transform

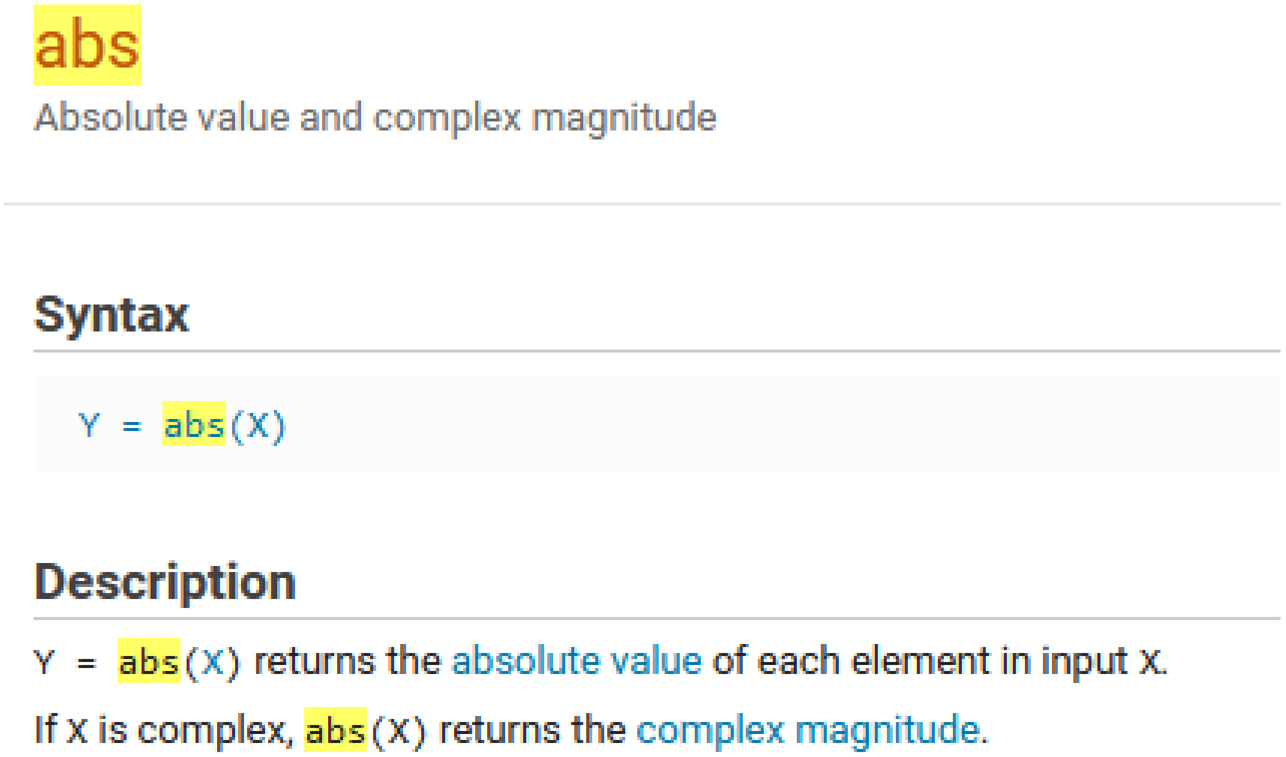
Syntax

```
X = ifft2(Y)
X = ifft2(Y,m,n)
X = ifft2(___,symflag)
```

Description

`X = ifft2(Y)` returns the two-dimensional discrete inverse Fourier transform of a matrix using a fast Fourier transform algorithm. If `Y` is a multidimensional array, then `ifft2` takes the 2-D inverse transform of each dimension higher than 2. The output `X` is the same size as `Y`.

MATLAB FUNCTIONS



abs

Absolute value and complex magnitude

Syntax

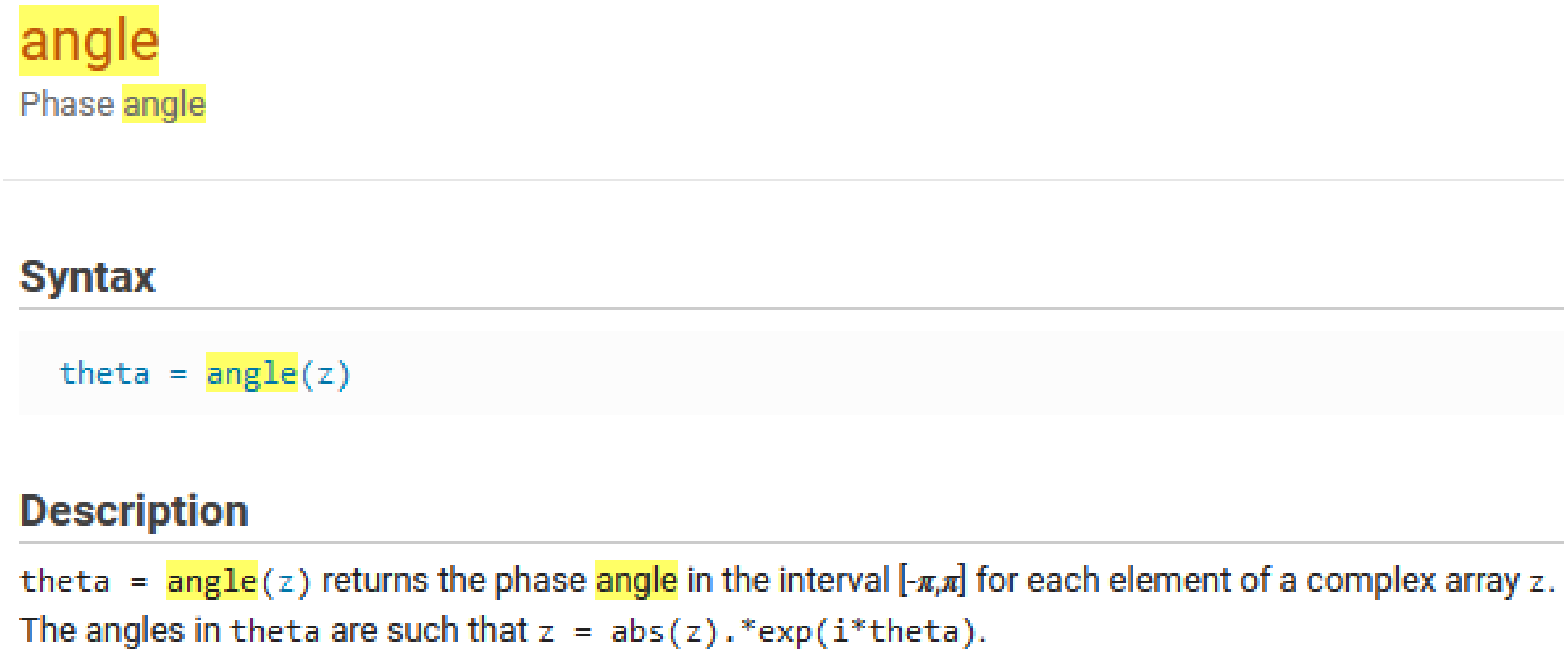
```
Y = abs(X)
```

Description

`Y = abs(X)` returns the absolute value of each element in input `X`.

If `X` is complex, `abs(X)` returns the complex magnitude.

MATLAB FUNCTIONS



angle

Phase angle

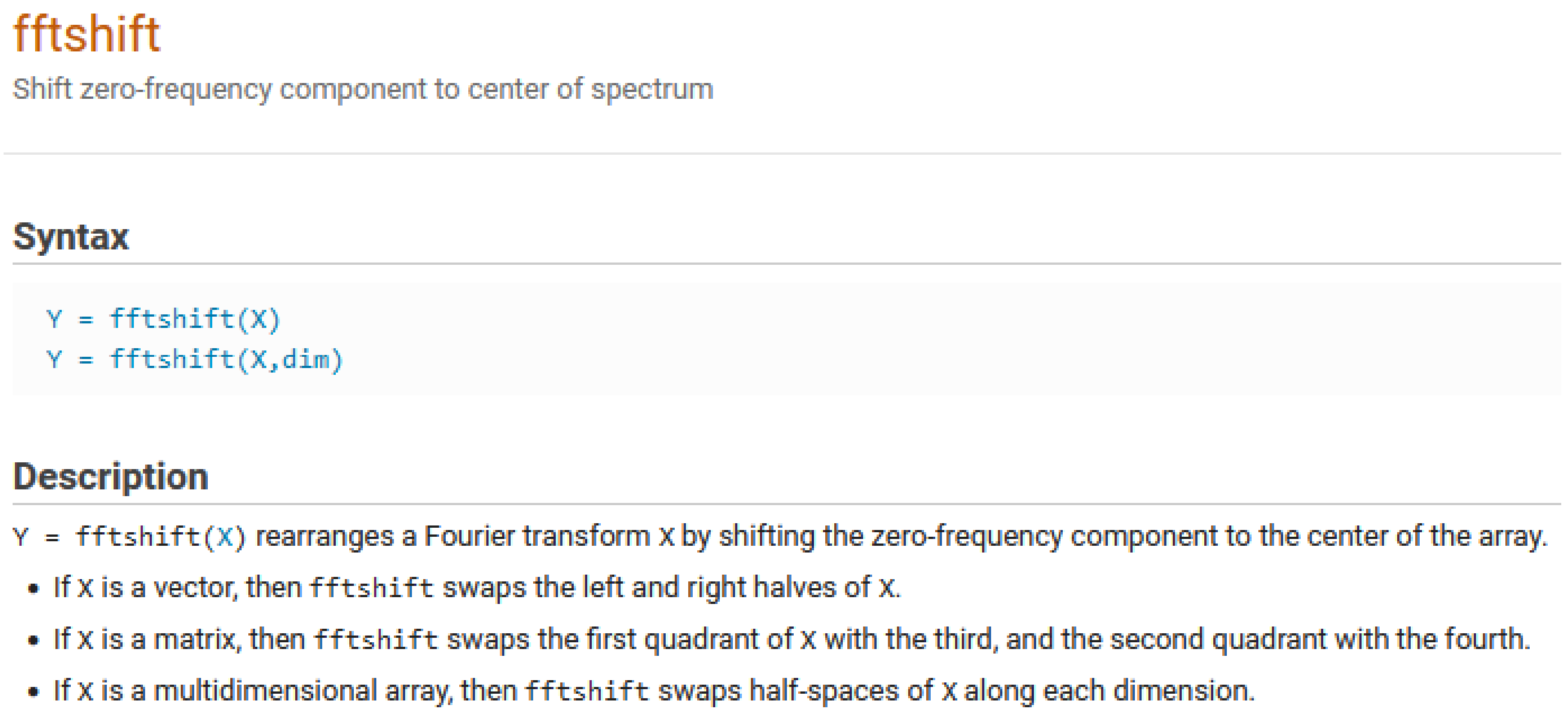
Syntax

```
theta = angle(z)
```

Description

`theta = angle(z)` returns the phase angle in the interval [-π,π] for each element of a complex array `z`. The angles in `theta` are such that `z = abs(z).*exp(i*theta)`.

MATLAB FUNCTIONS



fftshift

Shift zero-frequency component to center of spectrum

Syntax

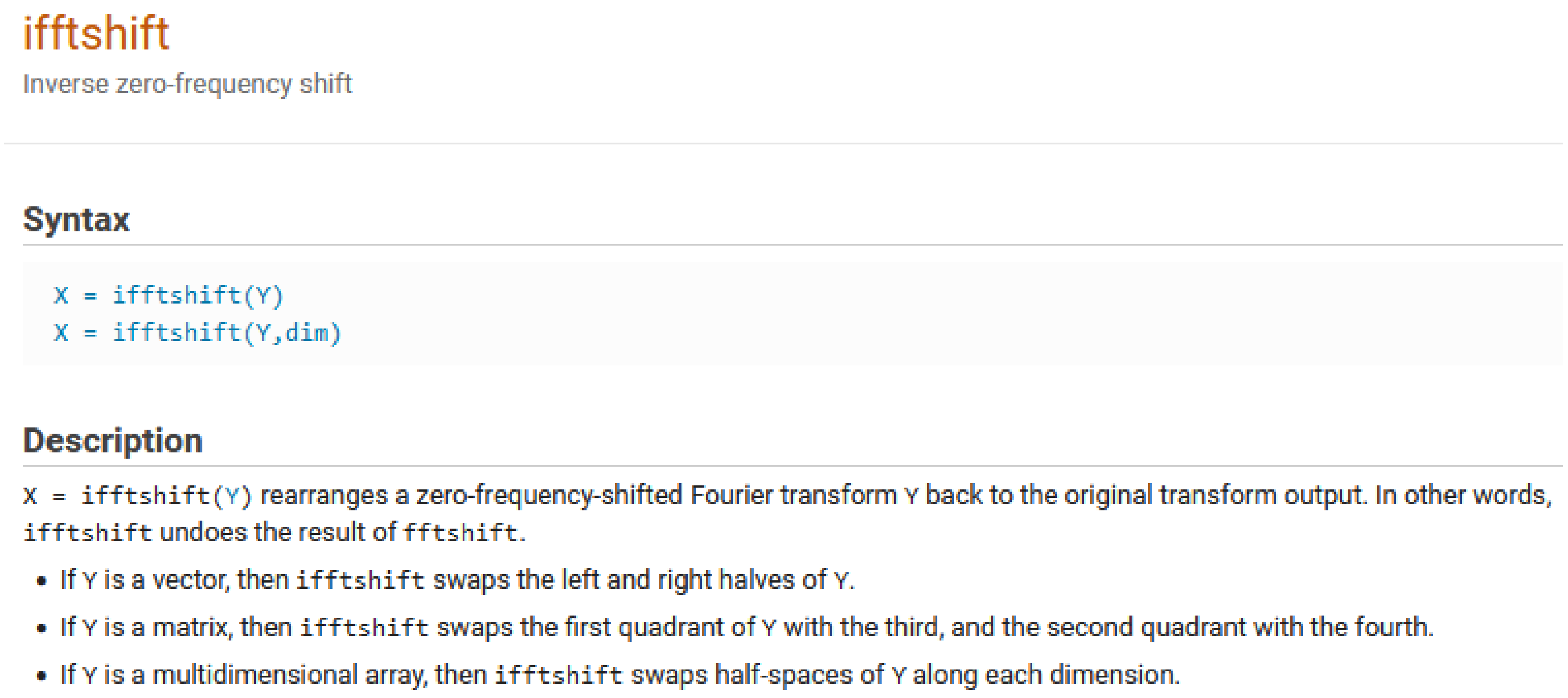
```
Y = fftshift(X)
Y = fftshift(X,dim)
```

Description

`Y = fftshift(X)` rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.

- If `X` is a vector, then `fftshift` swaps the left and right halves of `X`.
- If `X` is a matrix, then `fftshift` swaps the first quadrant of `X` with the third, and the second quadrant with the fourth.
- If `X` is a multidimensional array, then `fftshift` swaps half-spaces of `X` along each dimension.

MATLAB FUNCTIONS



ifftshift

Inverse zero-frequency shift

Syntax

```
X = ifftshift(Y)
X = ifftshift(Y,dim)
```

Description

`X = ifftshift(Y)` rearranges a zero-frequency-shifted Fourier transform `Y` back to the original transform output. In other words, `ifftshift` undoes the result of `fftshift`.

- If `Y` is a vector, then `ifftshift` swaps the left and right halves of `Y`.
- If `Y` is a matrix, then `ifftshift` swaps the first quadrant of `Y` with the third, and the second quadrant with the fourth.
- If `Y` is a multidimensional array, then `ifftshift` swaps half-spaces of `Y` along each dimension.