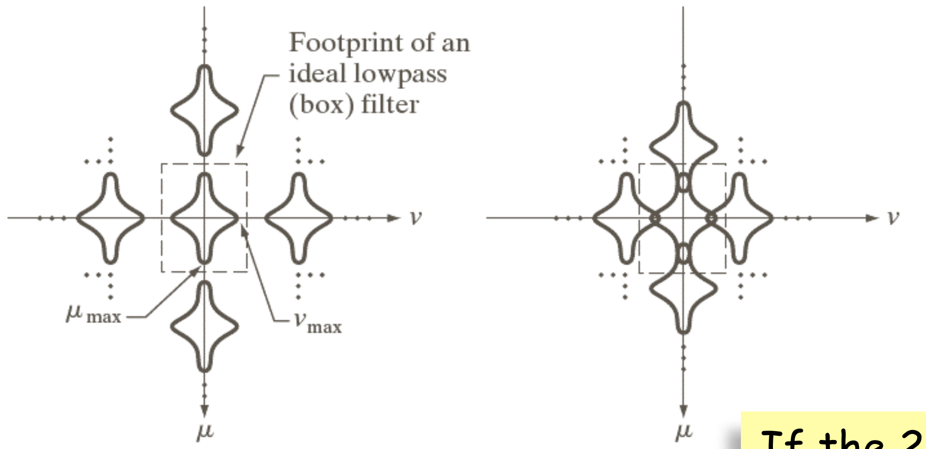
Lecture #9

- Sampling and band-limited functions
  - Aliasing & Jaggies
  - Moire
- Power Spectrum
- MATLAB
  - Power Spectrum
  - Edges
  - Geometric Transforms
- Properties of the 2-D DFT
  - Centering, rotation
  - Magnitude and phase

Undersampling in 2-D

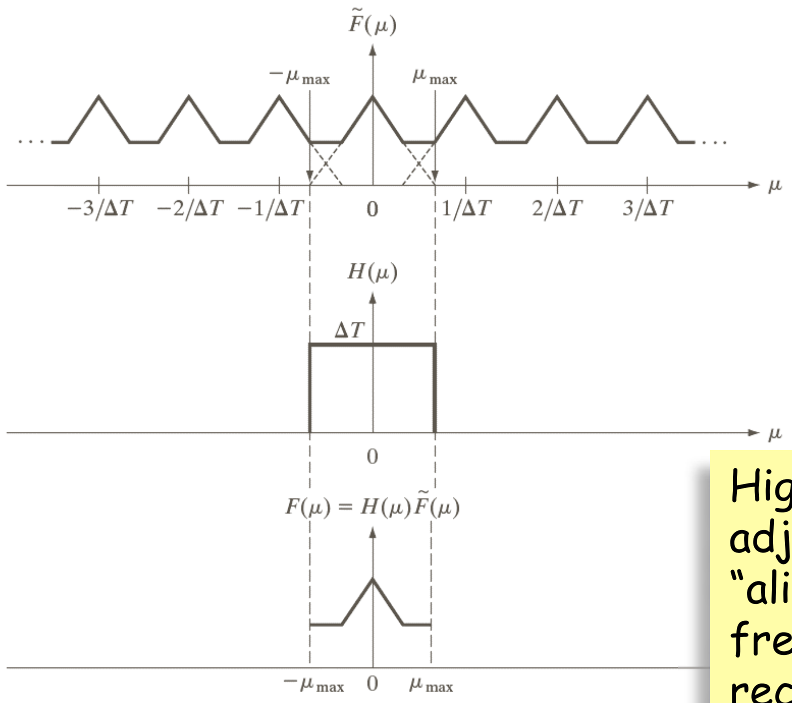


a b

FIGURE 4.15 Two-dimensional Fourier transforms of (a) an over-sampled, and (b) under-sampled band-limited function.

If the 2-D samples impulses are too far apart (undersampling) their 2-D Fourier transforms will overlap.

Undersampling in 2-D



High frequencies from adjacent spectra will "alias" as lower frequencies in the reconstructed image

FIGURE 4.9 (a) Fourier transform of an under-sampled, band-limited function. (Interference from adjacent periods is shown dashed in this figure). (b) The same ideal lowpass filter used in Fig. 4.8(b). (c) The product of (a) and (b). The interference from adjacent periods results in aliasing that prevents perfect recovery of $F(\mu)$ and, therefore, of the original, band-limited continuous function. Compare with Fig. 4.8.

Undersampling in 2-D

16 pixel squares

6 pixel squares

0.9174 pixel squares

0.4798 pixel squares

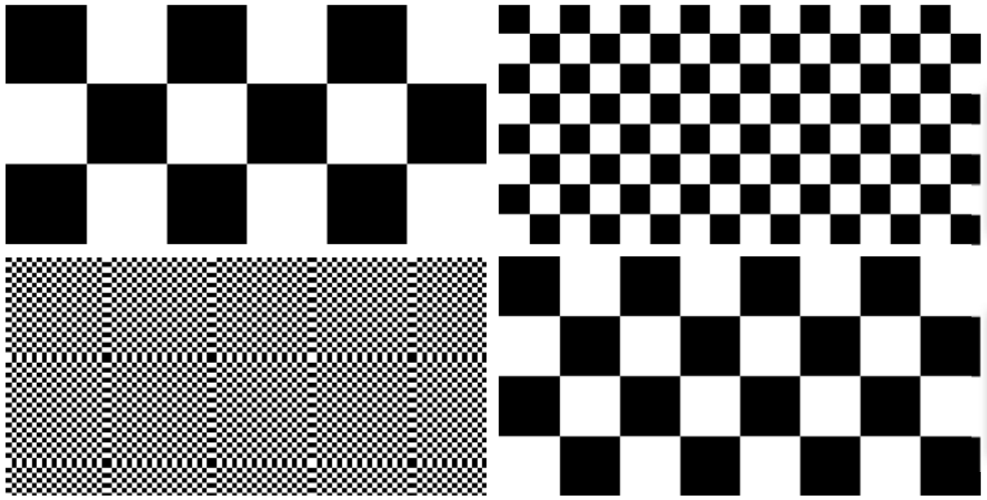
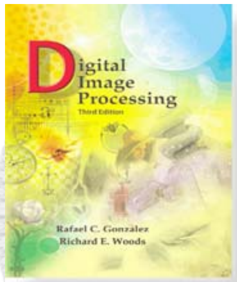


FIGURE 4.16 Aliasing in images. In (a) and (b), the lengths of the sides of the squares are 16 and 6 pixels, respectively, and aliasing is visually negligible. In (c) and (d), the sides of the squares are 0.9174 and 0.4798 pixels, respectively, and the results show significant aliasing. Note that (d) masquerades as a "normal" image.

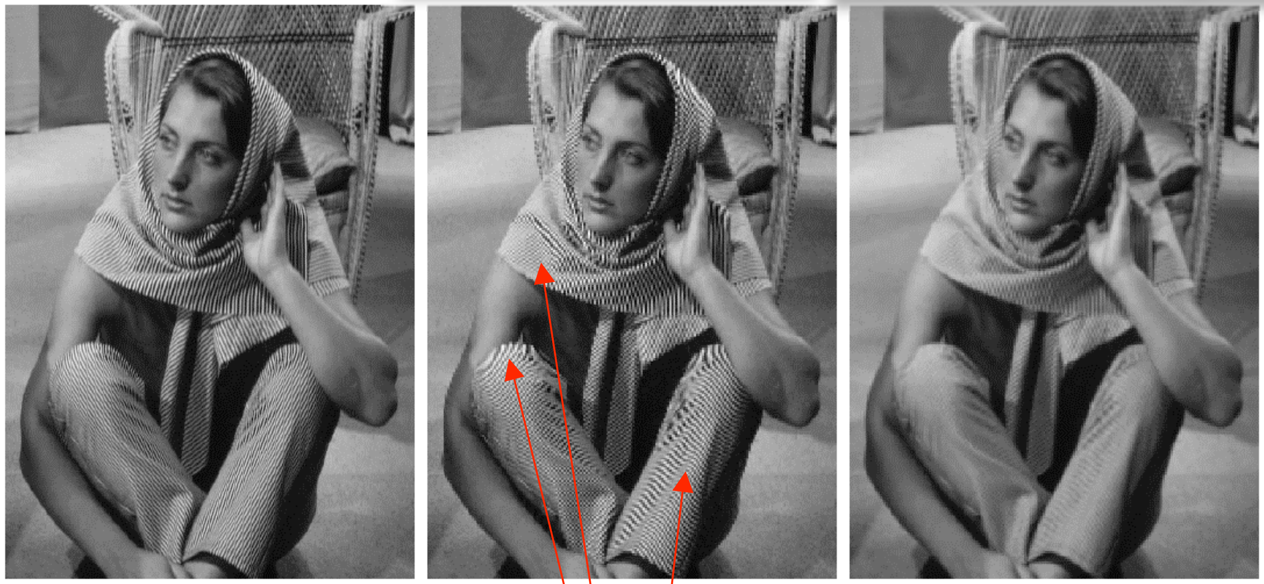
Results of checkboard images sampled) with a 96x96 pixel resolution sensor. Sensor max resolution=1 pixel.



Aliasing

50% pixel deletion

3x3 averaging prior to sampling to band-limit image



a b c

FIGURE 4.17 Illustration of aliasing on resampled images. (a) A digital image with negligible visual aliasing. (b) Result of resizing the image to 50% of its original size by pixel deletion. Aliasing is clearly visible. (c) Result of blurring the image in (a) with a 3 × 3 averaging filter prior to resizing. The image is slightly more blurred than (b), but aliasing is not longer objectionable. (Original image courtesy of the Signal Compression Laboratory, University of California, Santa Barbara.)

Note the aliasing

“Jaggies”

1024x1024-> 256x256
Bilinear interpolation
followed by pixel
replication back to
1024x1024

Use 5x5 smoothing
prior to processing to
band-limit function

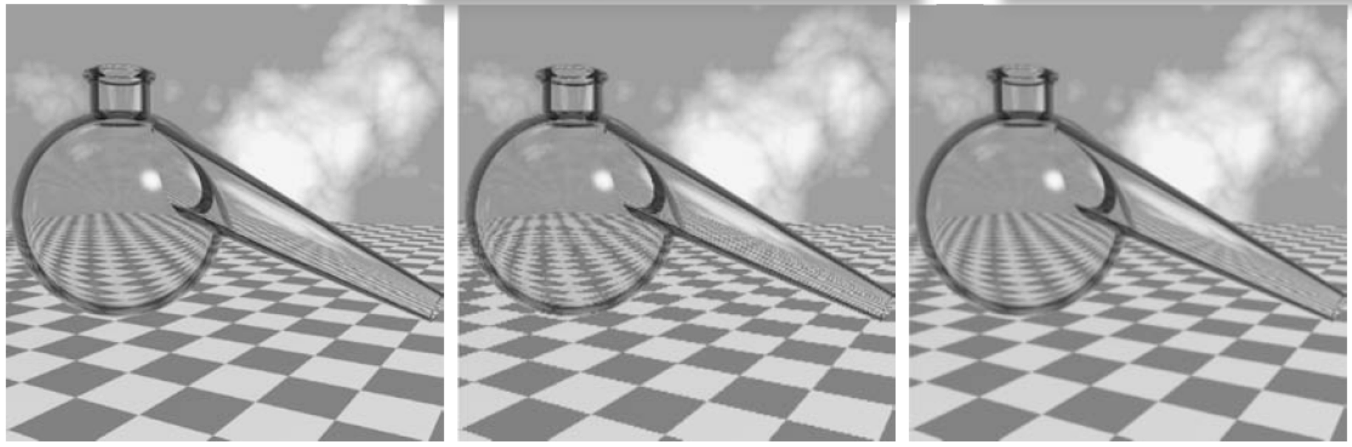
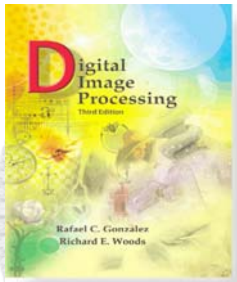


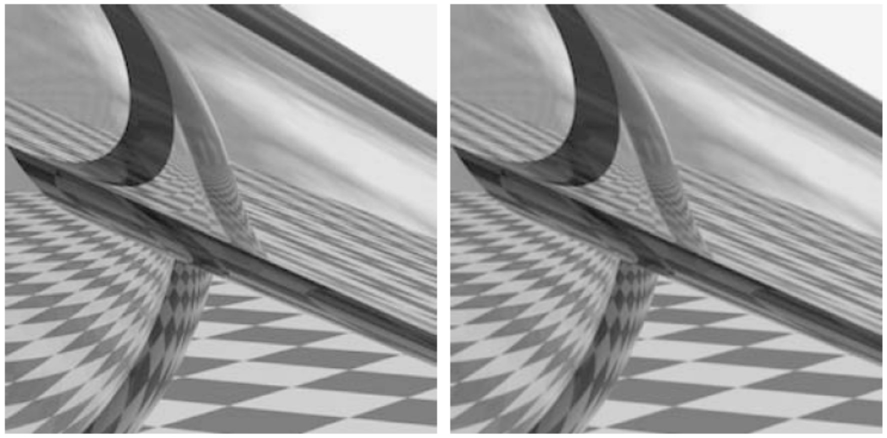
FIGURE 4.18 Illustration of jaggies. (a) A 1024 × 1024 digital image of a computer-generated scene with negligible visible aliasing. (b) Result of reducing (a) to 25% of its original size using bilinear interpolation. (c) Result of blurring the image in (a) with a 5 × 5 averaging filter prior to resizing it to 25% using bilinear interpolation. (Original image courtesy of D. P. Mitchell, Mental Landscape, LLC.)



"Jaggies"

256x256-> 1024x1024 using pixel replication

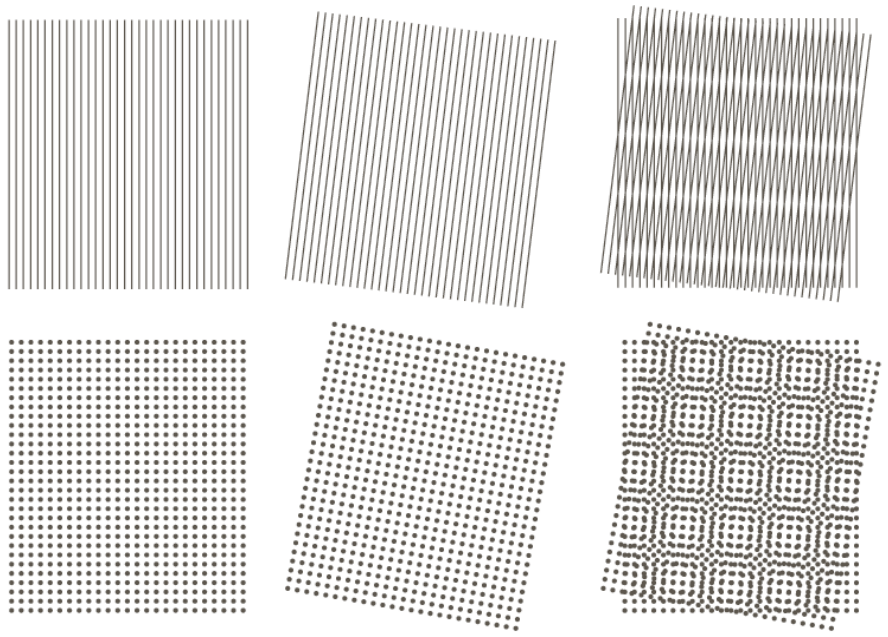
256x256-> 1024x1024 using bilinear interpolation



a b

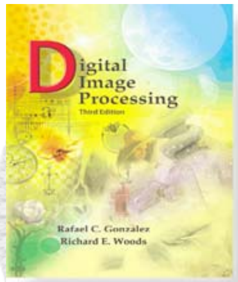
FIGURE 4.19 Image zooming. (a) A 1024 × 1024 digital image generated by pixel replication from a 256 × 256 image extracted from the middle of Fig. 4.18(a). (b) Image generated using bi-linear interpolation, showing a significant reduction in jaggies.

Moiré effect



a b c
d e f

FIGURE 4.20 Examples of the moiré effect. These are ink drawings, not digitized patterns. Superimposing one pattern on the other is equivalent mathematically to multiplying the patterns.



Moiré effect

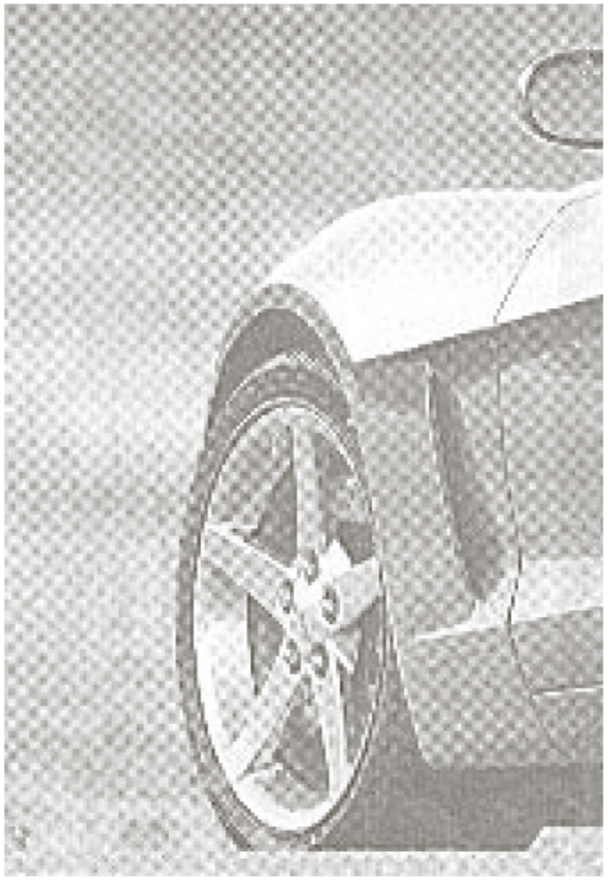
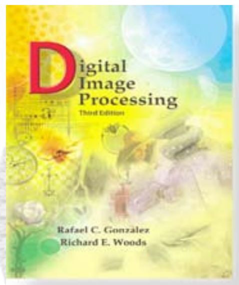


FIGURE 4.21 A newspaper image of size 246 × 168 pixels sampled at 75 dpi showing a moiré pattern. The moiré pattern in this image is the interference pattern created between the ±45° orientation of the halftone dots and the north–south orientation of the sampling grid used to digitize the image.



Moiré effect

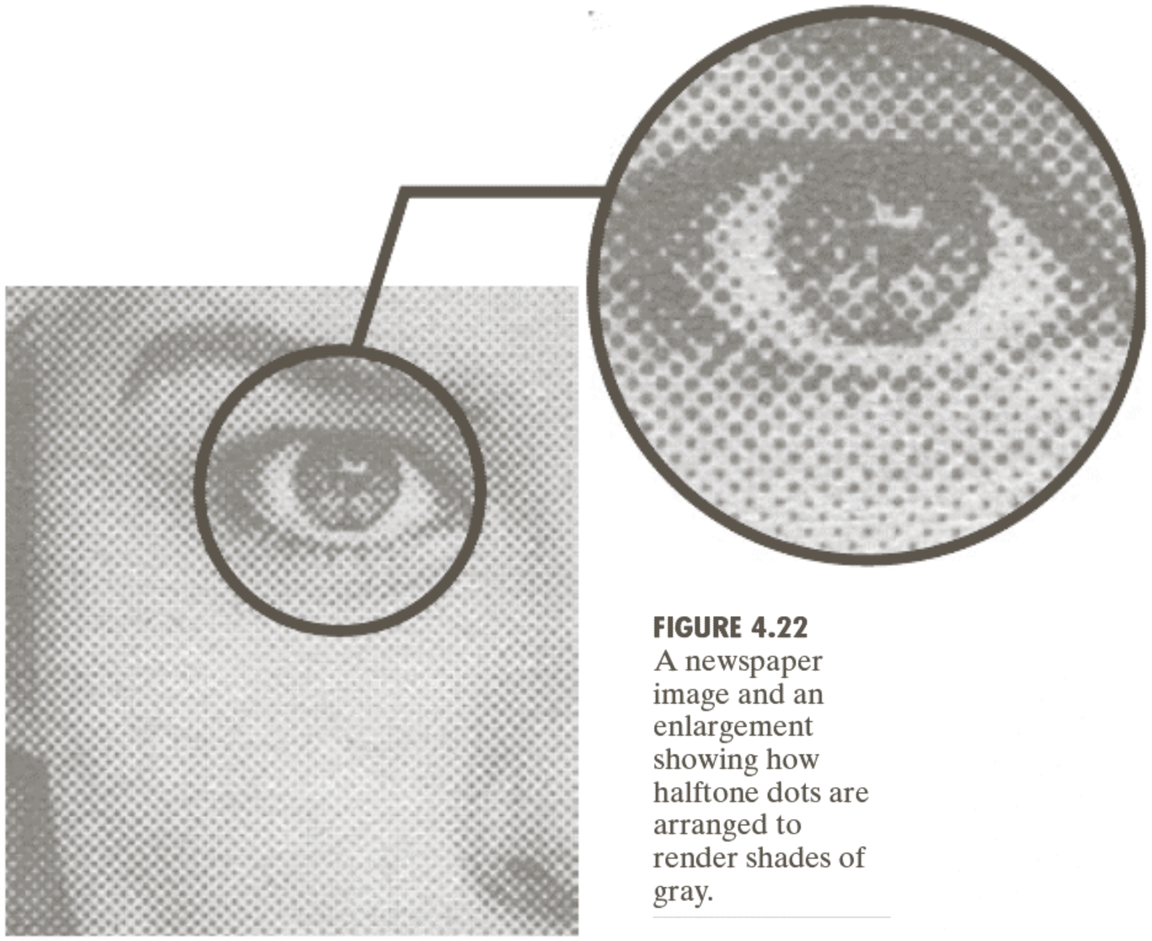
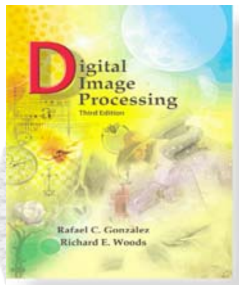


FIGURE 4.22 A newspaper image and an enlargement showing how halftone dots are arranged to render shades of gray.

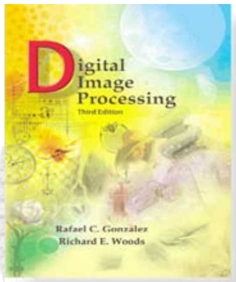


The Power Spectrum

The power spectrum of a signal is the square of the magnitude of its Fourier Transform.

$$\begin{aligned}\left|\mathrm{I}(u,v)\right|^2 &= \mathrm{I}(u,v)\,\mathrm{I}^*(u,v)\\ &= \left[\operatorname{Re}\mathrm{I}(u,v) + j\operatorname{Im}\mathrm{I}(u,v)\right]\left[\operatorname{Re}\mathrm{I}(u,v) - j\operatorname{Im}\mathrm{I}(u,v)\right]\\ &= \left[\operatorname{Re}\mathrm{I}(u,v)\right]^2 + \left[\operatorname{Im}\mathrm{I}(u,v)\right]^2.\end{aligned}$$

At each location (u,v) it indicates the squared intensity of the frequency component with period $\lambda = 1/\sqrt{u^2+v^2}$ and orientation $\theta = \tan^{-1}(v/u)$.



MATLAB/The Power Spectrum

The power spectrum (PS) is defined by $\mathrm{PS}(\mathrm{I}) = \left| \mathcal{F}\{\mathrm{I}(u,v)\} \right|^2$.

We take the base-e logarithm of the PS in order to view it. Otherwise its dynamic range could be too large to see everything at once. We add 1 to it first so that the minimum value of the result is 0 rather than –infinity, which it would be if there were any zeros in the PS. Recall that

$\log(f^2) = 2\log(f)$.

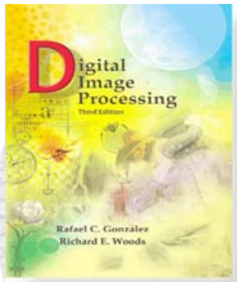
For display, the log of the power spectrum is usually used.

Multiplying by 2 is not necessary if you are generating a PS for viewing, since you'll probably have to scale it into the range 0-255 anyway. It is much easier to see the structures in a Fourier plane if the origin is in the center. Therefore we usually perform an fftshift on the PS before it is displayed.

```
>> PS = fftshift(2*log(abs(fft2(I))+1));
>> M  = max(PS(:));
>> image(uint8(255*(PS/M)));
```

If the PS is being calculated for later computational use -- for example the autocorrelation of a function is the inverse FT of the PS of the function -- it should be calculated by

```
>> PS = abs(fft2(I)).^2;
```

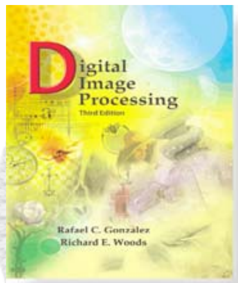


MATLAB/Calculating Edges

MATLAB’s edge detection routines

```
% [g,t] = edge (f, ‘method’, parameters)
% f is input image, g is output image
% t is an optional threshold for the output image
% ‘method’ can be sobel, prewitt, roberts, laplacian of a gaussian,
%  zero crossings, or Canny

>> f=imread(‘fig10.10(a).jpg’);                          % load in building figure
>> [g_sobel_default,0.074]=edge(f,’sobel’);              % figure 10.7(a)
>> [g_log_default, 0.0025]=edge(f,’ log’);               % figure 10.7(c)
% log is short for laplacian of a Gaussian
>> [g_canny_default, [0.019,0.047]]=edge(f,’canny’);  % figure 10.7(e)

% hand optimized functions
>> g_sobel_best=edge(f,’sobel’, 0.05);                   % figure 10.7(b)
%0.05 is a threshold for the output
>> g_log_best=edge(f,’ log’,0.003, 2.25);                % figure 10.7(d)
%0.003 is the output threshold and 2.25 is the standard deviation of the Gaussian
>> g_canny_best=edge(f,’canny’, [0.04,0.10],1.5);        % figure 10.7(f)
%0.04 and 0.10 are the output thresholds and 1.5 is the standard deviation of the Gaussian
```

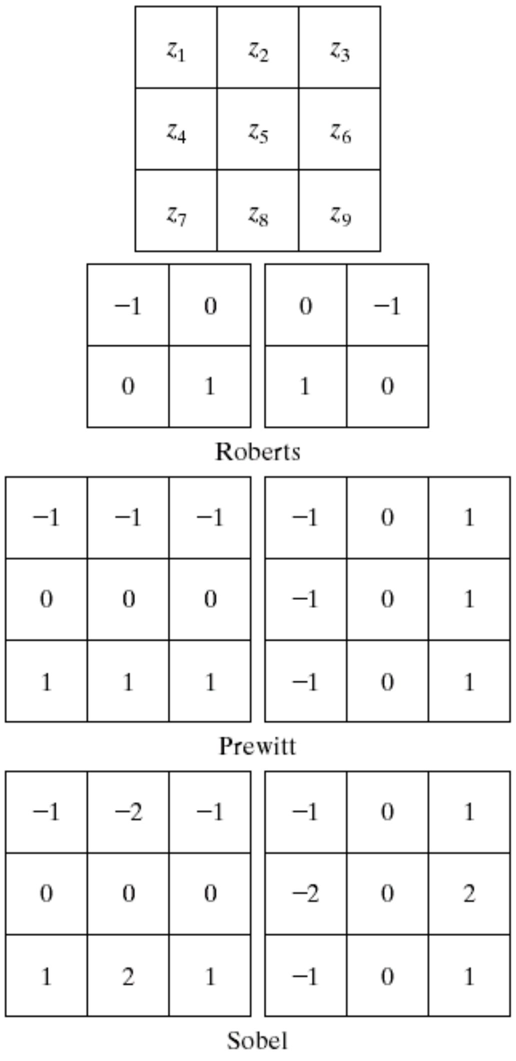
SEE GWE, Section 10.1.3 Edge Detection Using Function **edge**



Common Gradient Masks

a
b c
d e
f g

FIGURE 10.8 A 3 × 3 region of an image (the z's are gray-level values) and various masks used to compute the gradient at point labeled z_5.



z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0		0	−1
0	1		1	0

Roberts

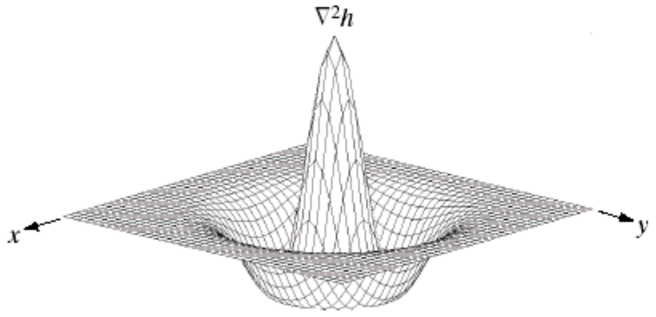
−1	−1	−1		−1	0	1
0	0	0		−1	0	1
1	1	1		−1	0	1

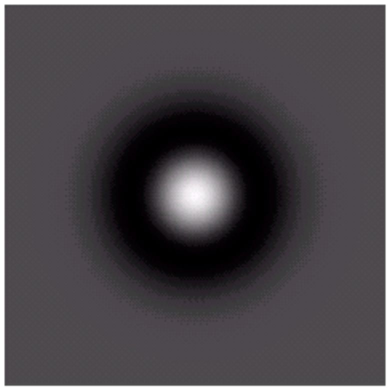
Prewitt

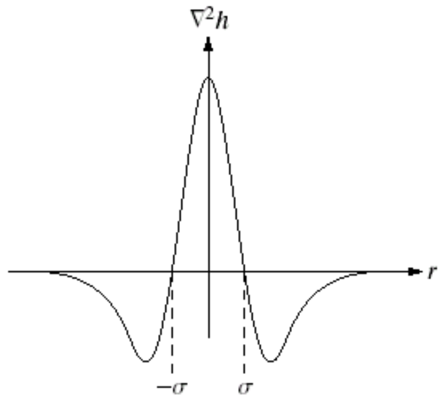
−1	−2	−1		−1	0	1
0	0	0		−2	0	2
1	2	1		−1	0	1

Sobel

Second Derivatives: the LoG







0	0	–1	0	0
0	–1	–2	–1	0
–1	–2	16	–2	–1
0	–1	–2	–1	0
0	0	–1	0	0

a b
c d

FIGURE 10.14 Laplacian of a Gaussian (LoG). (a) 3-D plot. (b) Image (black is negative, gray is the zero plane, and white is positive). (c) Cross section showing zero crossings. (d) 5×5 mask approximation to the shape of (a).

Derivative Examples

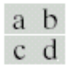
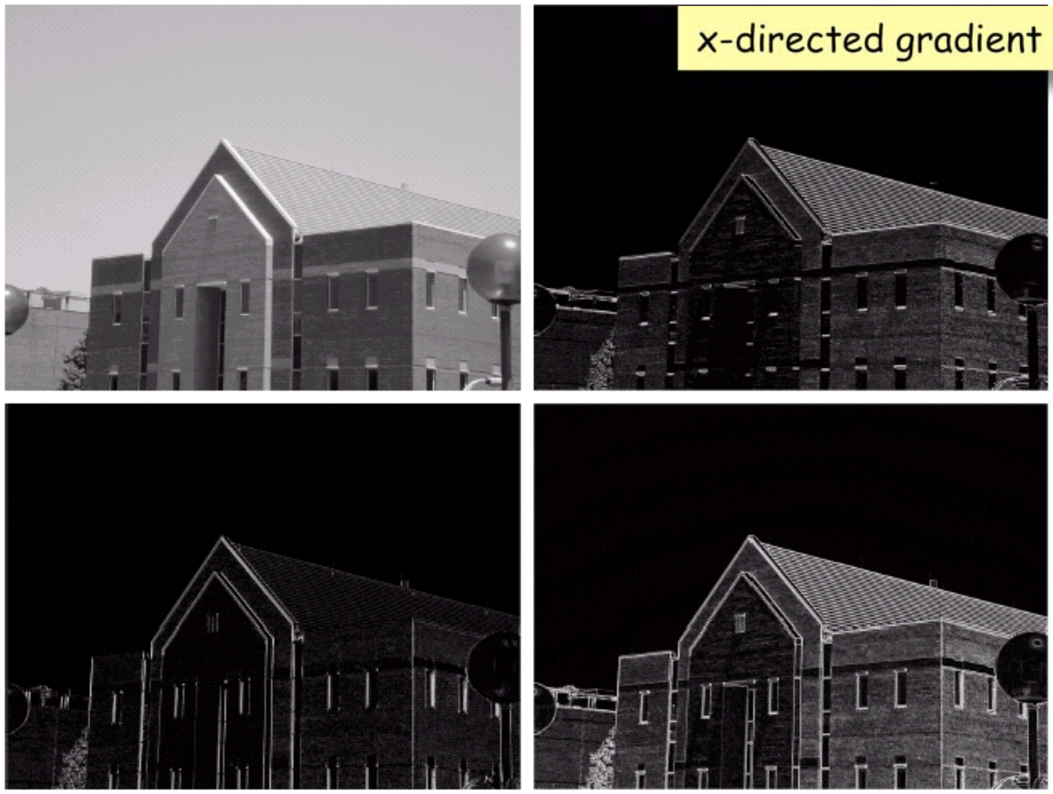


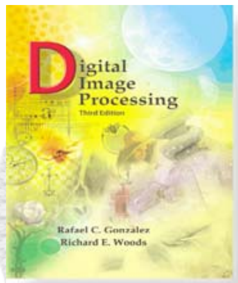
FIGURE 10.10 (a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.



x-directed gradient magnitude $|G_x|$

y-directed gradient magnitude $|G_y|$

$|G_x|+|G_y|$



Derivative Examples

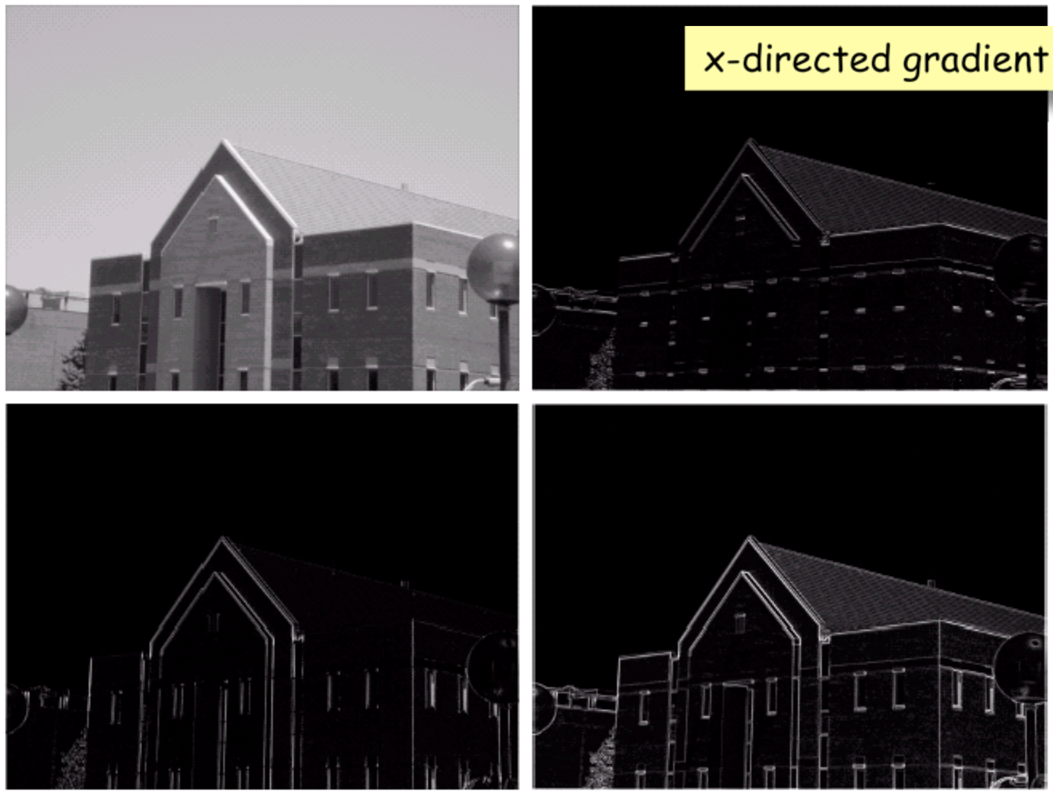
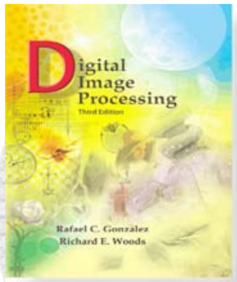

Smoothed with a 5x5 averaging filter
x-directed gradient magnitude $|G_x|$
y-directed gradient magnitude $|G_y|$
$|G_x|+|G_y|$
a b

FIGURE 10.11 Same sequence as in Fig. 10.10, but with the original image smoothed with a 5 × 5 averaging filter.



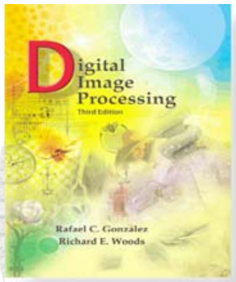
MATLAB/Calculating Edges

MATLAB's edge detection routines

```
% [g,t] = edge (f, 'method', parameters)
% f is input image, g is output image
% t is an optional threshold for the output image
% 'method' can be sobel, prewitt, roberts, laplacian of a gaussian,
%  zero crossings, or Canny

>> f=imread('fig10.10(a).jpg');                          % load in building figure
>> [g_sobel_default,0.074]=edge(f,'sobel');              % figure 10.7(a)
>> [g_log_default, 0.0025]=edge(f,' log');               % figure 10.7(c)
% log is short for laplacian of a Gaussian
>> [g_canny_default, [0.019,0.047]]=edge(f,'canny');  % figure 10.7(e)

% hand optimized functions
>> g_sobel_best=edge(f,'sobel', 0.05);                   % figure 10.7(b)
%0.05 is a threshold for the output
>> g_log_best=edge(f,' log',0.003, 2.25);                % figure 10.7(d)
%0.003 is the output threshold and 2.25 is the standard deviation of the Gaussian
>> g_canny_best=edge(f,'canny', [0.04,0.10],1.5);       % figure 10.7(f)
%0.04 and 0.10 are the output thresholds and 1.5 is the standard deviation of the Gaussian
```

SEE GWE, Section 10.1.3 Edge Detection Using Function **edge**

Canny algorithm:
1. Smooth the image with a Gaussian
2. Compute the gradient magnitude
3. Apply non-maximal suppression to the gradient magnitude image (edge thinning)
4. Use double thresholding and connectivity to detect and link edges. Basically the two thresholds create strong and weak edges and connectivity analysis is used to link edges

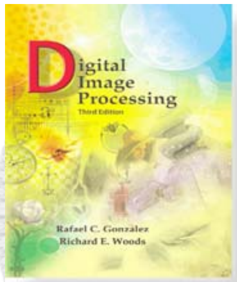


MATLAB/Calculating Edges

MATLAB's image registration routines

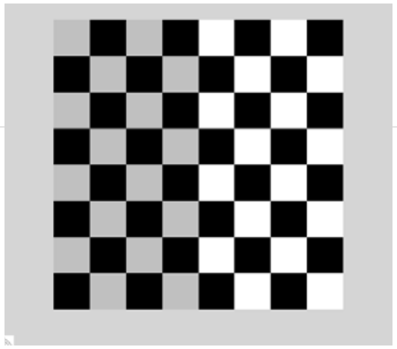
```
>> C=checkerboard(NP, M, N);

% construct a checkboard
% NP is number of pixels on each side of square
% M is number of rows; N is number of columns

% MATLAB uses transforms in a t_form structure
% this is an example of entering an affine transformation

>> T=[2 0 0; 0 3 0; 0 0 1];
>> tform=maketform('affine',T);

% The MATLAB function imtransform implements inverse (reverse) mapping

>> g=imtransform(f, tform, 'interp');

% f is the input image, g is the output image
% 'interp' can be nearest, bilinear, or bicubic
% if 'interp' is omitted it defaults to bilinear
```

SEE GWE, Section 5.11 Geometric Transformations and Image Registration

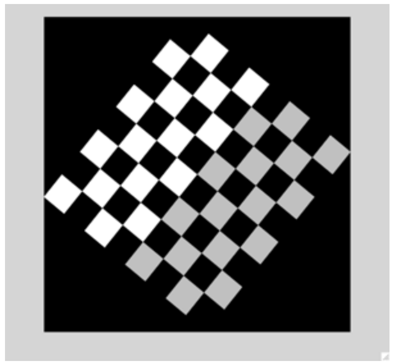


MATLAB/Image Transformation

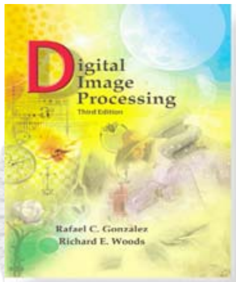
MATLAB's image transformation routines using affine transforms



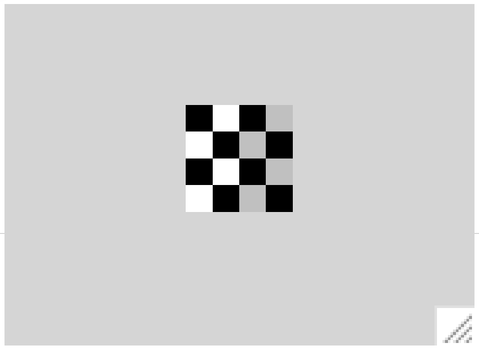
```
>> f=checkerboard(50);
>> imshow(f);

>> Tscale=[1.5 0 0; 0 2 0; 0 0 1];
>> Trotation=[cos(pi/4) sin(pi/4) 0;-sin(pi/4) cos(pi/4) 0; .1 -.1 1];
>> Tshear=[1 0 0; .2 1 0; 0 0 1];
>> T3=Tscale*Trotation*Tshear;

% combine affine transformations

>> tform3=maketform('affine',T3);

>> Trans_f=imtransform(f,tform3);
>> imshow(Trans_f)
Warning: Image is too big to fit on screen; displaying at 67%
> In imuitools/private/initSize at 86
  In imshow at 201
```



SEE GWE, Section 5.11 Geometric Transformations and Image Registration
Also see: **http://www.mathworks.com/products/demos/image/create_gallery/tform.html**

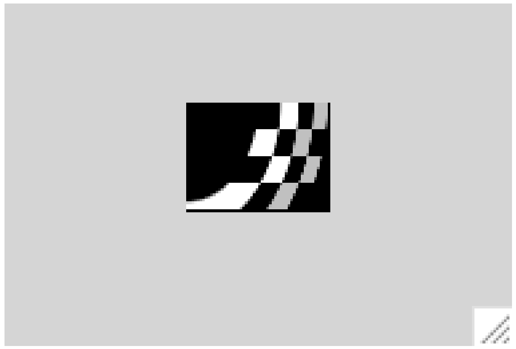


MATLAB/Image Transformation

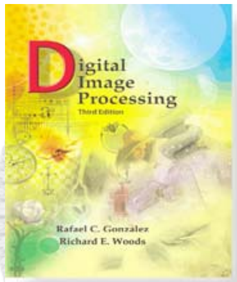


MATLAB’s image transformation can also use polynomials

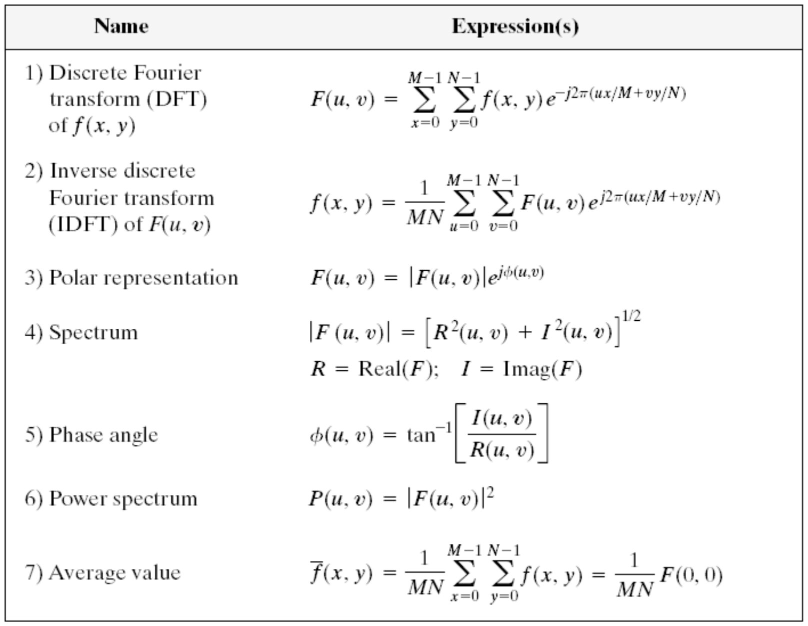
```
>> pairs1 = [1 1; 5 21; 17 40; 28 1; 32 20; 45 40; 72 1; 77 20; 90 40];
>> pairs2 = [1 1; 1 21; 1 40; 20 1; 20 20; 20 40; 40 1; 40 20; 40 40];

% pairs1 and pair2 are corresponding pairs of registration points
% (1,1) corresponds to (1,1)
% (5,21) corresponds to (1,21)
%  etc.

>> t_poly = cp2tform(pairs1, pairs2, 'polynomial',2);

% cp2tform constructs the polynomial transform t_poly of the type t_form
% t=cp2tform(pairs1, pairs2, ‘type’ );

% cp2tform will calculate the transformation t
% from pairs1 to pairs2 of corresponding registration points
% ‘type” includes 'linear conformal’, 'affine’,'projective’, 'polynomial' (Order 2, 3, or 4)

>> I = checkerboard(10,2);
>> J = imtransform(I,t_poly);
```



SEE GWE, Table 5.4



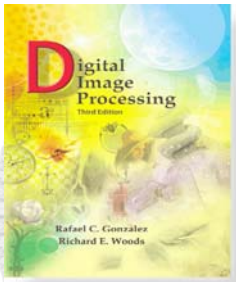
Summary of DFT Expressions



Name	Expression(s)
1) Discrete Fourier transform (DFT) of $f(x, y)$	$F(u, v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi(ux/M+vy/N)}$
2) Inverse discrete Fourier transform (IDFT) of $F(u, v)$	$f(x, y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u, v)\, e^{j2\pi(ux/M+vy/N)}$
3) Polar representation	$F(u, v) = \lvert F(u, v)\rvert e^{j\phi(u,v)}$
4) Spectrum	$\lvert F(u, v)\rvert = \left[R^2(u, v) + I^2(u, v)\right]^{1/2}$ $R = \text{Real}(F); \quad I = \text{Imag}(F)$
5) Phase angle	$\phi(u, v) = \tan^{-1}\left[\frac{I(u, v)}{R(u, v)}\right]$
6) Power spectrum	$P(u, v) = \lvert F(u, v)\rvert^2$
7) Average value	$\overline{f}(x, y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y) = \frac{1}{MN}F(0, 0)$

(Continued)

TABLE 4.2 Summary of DFT definitions and corresponding expressions.



DFT Properties

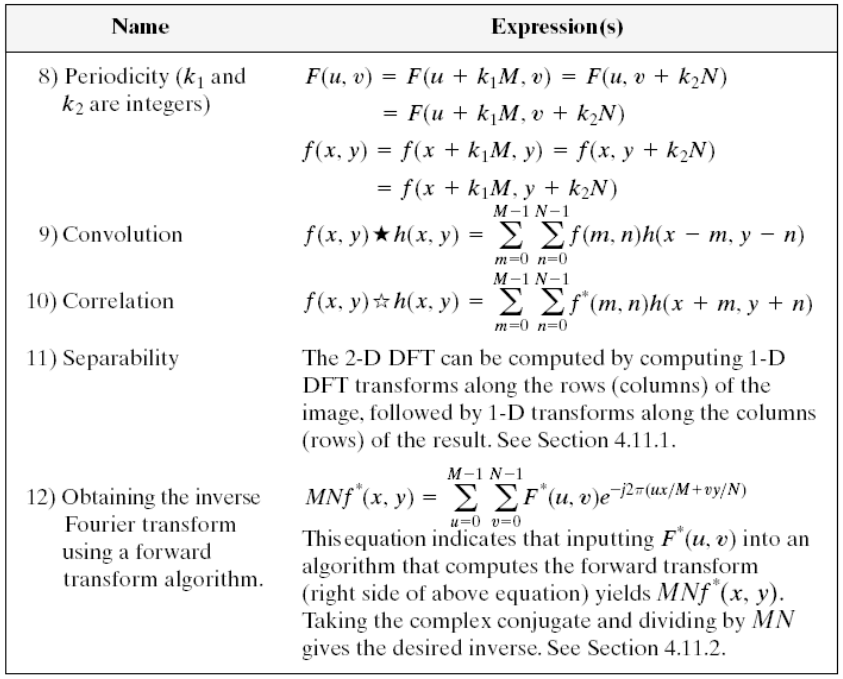
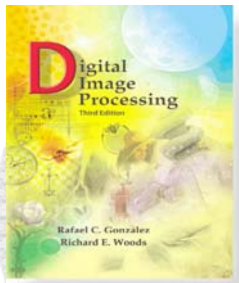


TABLE 4.2 (*Continued*)

Name	Expression(s)
8) Periodicity (k_1 and k_2 are integers)	$F(u,v) = F(u + k_1M, v) = F(u, v + k_2N) = F(u + k_1M, v + k_2N)$ $f(x,y) = f(x + k_1M, y) = f(x, y + k_2N) = f(x + k_1M, y + k_2N)$
9) Convolution	$f(x,y) \bigstar h(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m, y-n)$
10) Correlation	$f(x,y) \star h(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f^*(m,n)h(x+m, y+n)$
11) Separability	The 2-D DFT can be computed by computing 1-D DFT transforms along the rows (columns) of the image, followed by 1-D transforms along the columns (rows) of the result. See Section 4.11.1.
12) Obtaining the inverse Fourier transform using a forward transform algorithm.	$MNf^*(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F^*(u,v)e^{-j2\pi(ux/M+vy/N)}$ This equation indicates that inputting $F^*(u,v)$ into an algorithm that computes the forward transform (right side of above equation) yields $MNf^*(x,y)$. Taking the complex conjugate and dividing by MN gives the desired inverse. See Section 4.11.2.

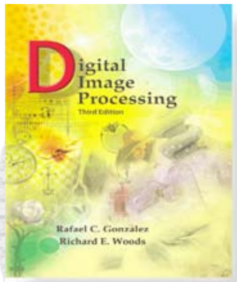


DFT Properties

Name	DFT Pairs
1) Symmetry properties	See Table 4.1
2) Linearity	$af_1(x,y) + bf_2(x,y) \Leftrightarrow aF_1(u,v) + bF_2(u,v)$
3) Translation (general)	$f(x,y)e^{j2\pi(u_0x/M + v_0y/N)} \Leftrightarrow F(u-u_0, v-v_0)$ $f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0/M + vy_0/N)}$
4) Translation to center of the frequency rectangle, $(M/2, N/2)$	$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-M/2, v-N/2)$ $f(x-M/2, y-N/2) \Leftrightarrow F(u,v)(-1)^{u+v}$
5) Rotation	$f(r, \theta+\theta_0) \Leftrightarrow F(\omega, \varphi+\theta_0)$ $x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$
6) Convolution theorem[†]	$f(x,y) \star h(x,y) \Leftrightarrow F(u,v)H(u,v)$ $f(x,y)h(x,y) \Leftrightarrow F(u,v) \star H(u,v)$

(*Continued*)

TABLE 4.3
Summary of DFT pairs. The closed-form expressions in 12 and 13 are valid only for continuous variables. They can be used with discrete variables by sampling the closed-form, continuous expressions.



DFT Pairs

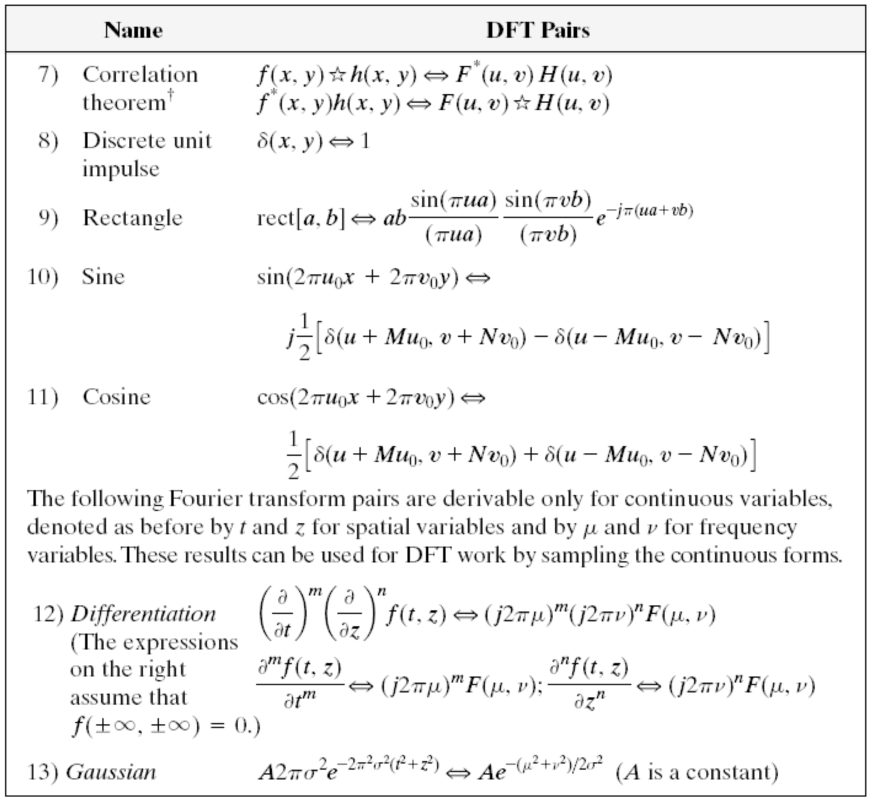
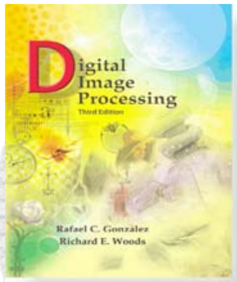


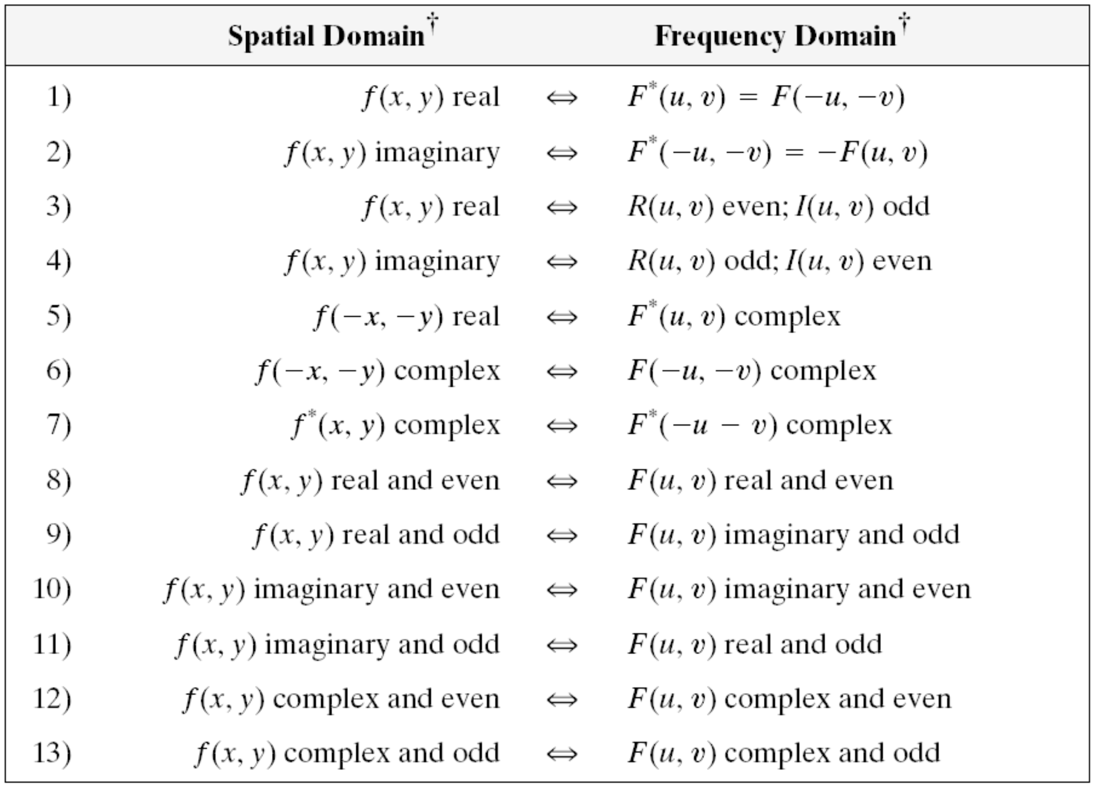
TABLE 4.3
(Continued)

Name	DFT Pairs
7) Correlation theorem[†]	$f(x, y) \star h(x, y) \Leftrightarrow F^*(u, v)\, H(u, v)$ $f^*(x, y)h(x, y) \Leftrightarrow F(u, v) \star H(u, v)$
8) Discrete unit impulse	$\delta(x, y) \Leftrightarrow 1$
9) Rectangle	$\text{rect}[a, b] \Leftrightarrow ab\dfrac{\sin(\pi ua)}{(\pi ua)}\dfrac{\sin(\pi vb)}{(\pi vb)}e^{-j\pi(ua+vb)}$
10) Sine	$\sin(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow$ $j\dfrac{1}{2}\left[\delta(u + Mu_0, v + Nv_0) - \delta(u - Mu_0, v - Nv_0)\right]$
11) Cosine	$\cos(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow$ $\dfrac{1}{2}\left[\delta(u + Mu_0, v + Nv_0) + \delta(u - Mu_0, v - Nv_0)\right]$
The following Fourier transform pairs are derivable only for continuous variables, denoted as before by t and z for spatial variables and by μ and ν for frequency variables. These results can be used for DFT work by sampling the continuous forms.	
12) *Differentiation* (The expressions on the right assume that $f(\pm\infty, \pm\infty) = 0$.)	$\left(\dfrac{\partial}{\partial t}\right)^m\left(\dfrac{\partial}{\partial z}\right)^n f(t, z) \Leftrightarrow (j2\pi\mu)^m(j2\pi\nu)^n F(\mu, \nu)$ $\dfrac{\partial^m f(t, z)}{\partial t^m} \Leftrightarrow (j2\pi\mu)^m F(\mu, \nu);\ \dfrac{\partial^n f(t, z)}{\partial z^n} \Leftrightarrow (j2\pi\nu)^n F(\mu, \nu)$
13) *Gaussian*	$A2\pi\sigma^2 e^{-2\pi^2\sigma^2(t^2+z^2)} \Leftrightarrow Ae^{-(\mu^2+\nu^2)/2\sigma^2}$ (A is a constant)

[†]Assumes that the functions have been extended by zero padding. Convolution and correlation are associative, commutative, and distributive.



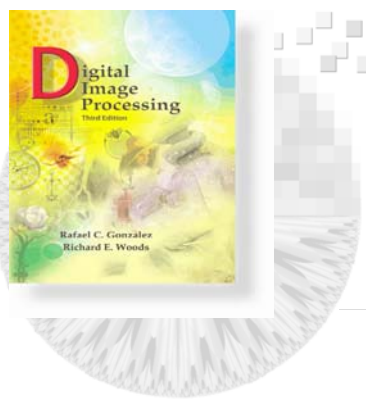
2-D DFT Symmetry Properties



	Spatial Domain[†]		Frequency Domain[†]
1)	$f(x, y)$ real	$\Leftrightarrow$	$F^*(u, v) = F(-u, -v)$
2)	$f(x, y)$ imaginary	$\Leftrightarrow$	$F^*(-u, -v) = -F(u, v)$
3)	$f(x, y)$ real	$\Leftrightarrow$	$R(u, v)$ even; $I(u, v)$ odd
4)	$f(x, y)$ imaginary	$\Leftrightarrow$	$R(u, v)$ odd; $I(u, v)$ even
5)	$f(-x, -y)$ real	$\Leftrightarrow$	$F^*(u, v)$ complex
6)	$f(-x, -y)$ complex	$\Leftrightarrow$	$F(-u, -v)$ complex
7)	$f^*(x, y)$ complex	$\Leftrightarrow$	$F^*(-u - v)$ complex
8)	$f(x, y)$ real and even	$\Leftrightarrow$	$F(u, v)$ real and even
9)	$f(x, y)$ real and odd	$\Leftrightarrow$	$F(u, v)$ imaginary and odd
10)	$f(x, y)$ imaginary and even	$\Leftrightarrow$	$F(u, v)$ imaginary and even
11)	$f(x, y)$ imaginary and odd	$\Leftrightarrow$	$F(u, v)$ real and odd
12)	$f(x, y)$ complex and even	$\Leftrightarrow$	$F(u, v)$ complex and even
13)	$f(x, y)$ complex and odd	$\Leftrightarrow$	$F(u, v)$ complex and odd

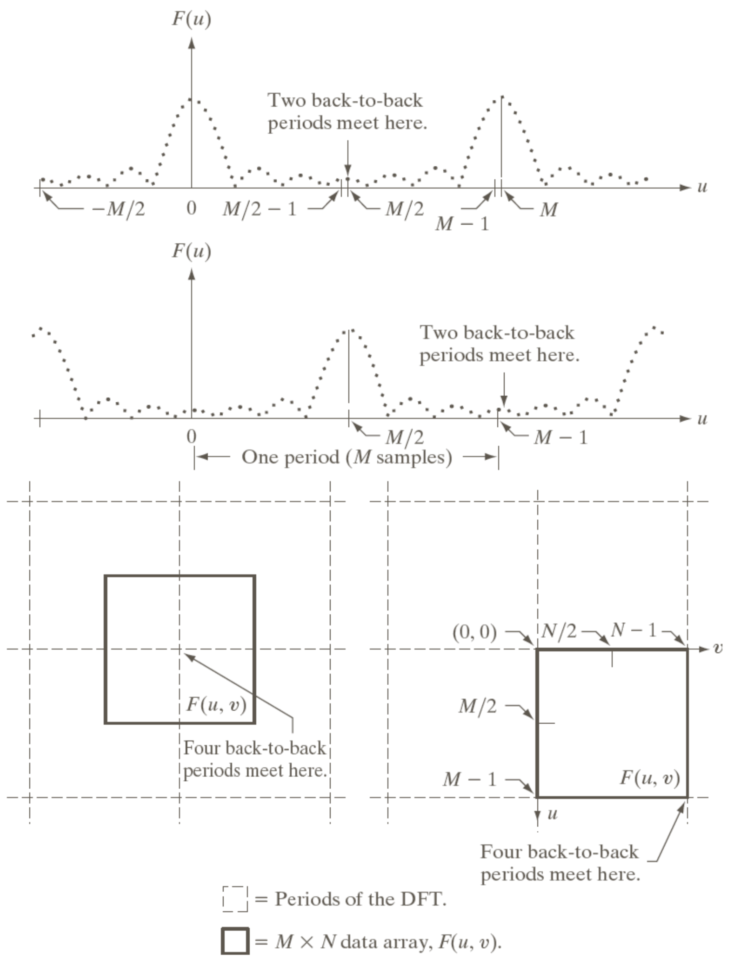
[†]Recall that x, y, u, and v are *discrete* (integer) variables, with x and u in the range $[0, M - 1]$, and y, and v in the range $[0, N - 1]$. To say that a complex function is *even* means that its real *and* imaginary parts are even, and similarly for an odd complex function.

TABLE 4.1 Some symmetry properties of the 2-D DFT and its inverse. $R(u, v)$ and $I(u, v)$ are the real and imaginary parts of $F(u, v)$, respectively. The term *complex* indicates that a function has nonzero real and imaginary parts.



DFT Centering

$$f(x,y)e^{j2\pi\left(u\frac{x_0}{M}+v\frac{y_0}{N}\right)} \Leftrightarrow F\left(u-\frac{M}{2},v-\frac{N}{2}\right)$$


$F(u)$
Two back-to-back periods meet here.
$-M/2$
0
$M/2-1$
$M/2$
$M-1$
M
u
$F(u)$
Two back-to-back periods meet here.
0
$M/2$
$M-1$
u
One period (M samples)
$F(u,v)$
Four back-to-back periods meet here.
(0, 0)
$N/2$
$N-1$
v
$M/2$
$M-1$
$F(u,v)$
u
Four back-to-back periods meet here.
= Periods of the DFT.
= $M \times N$ data array, $F(u,v)$.

a
b
c d

FIGURE 4.23 Centering the Fourier transform. (a) A 1-D DFT showing an infinite number of periods. (b) Shifted DFT obtained by multiplying $f(x)$ by $(-1)^x$ before computing $F(u)$. (c) A 2-D DFT showing an infinite number of periods. The solid area is the $M \times N$ data array, $F(u,v)$, obtained with Eq. (4.5-15). This array consists of four quarter periods. (d) A Shifted DFT obtained by multiplying $f(x,y)$ by $(-1)^{x+y}$ before computing $F(u,v)$. The data now contains one complete, centered period, as in (b).

Properties of 2-D DFT

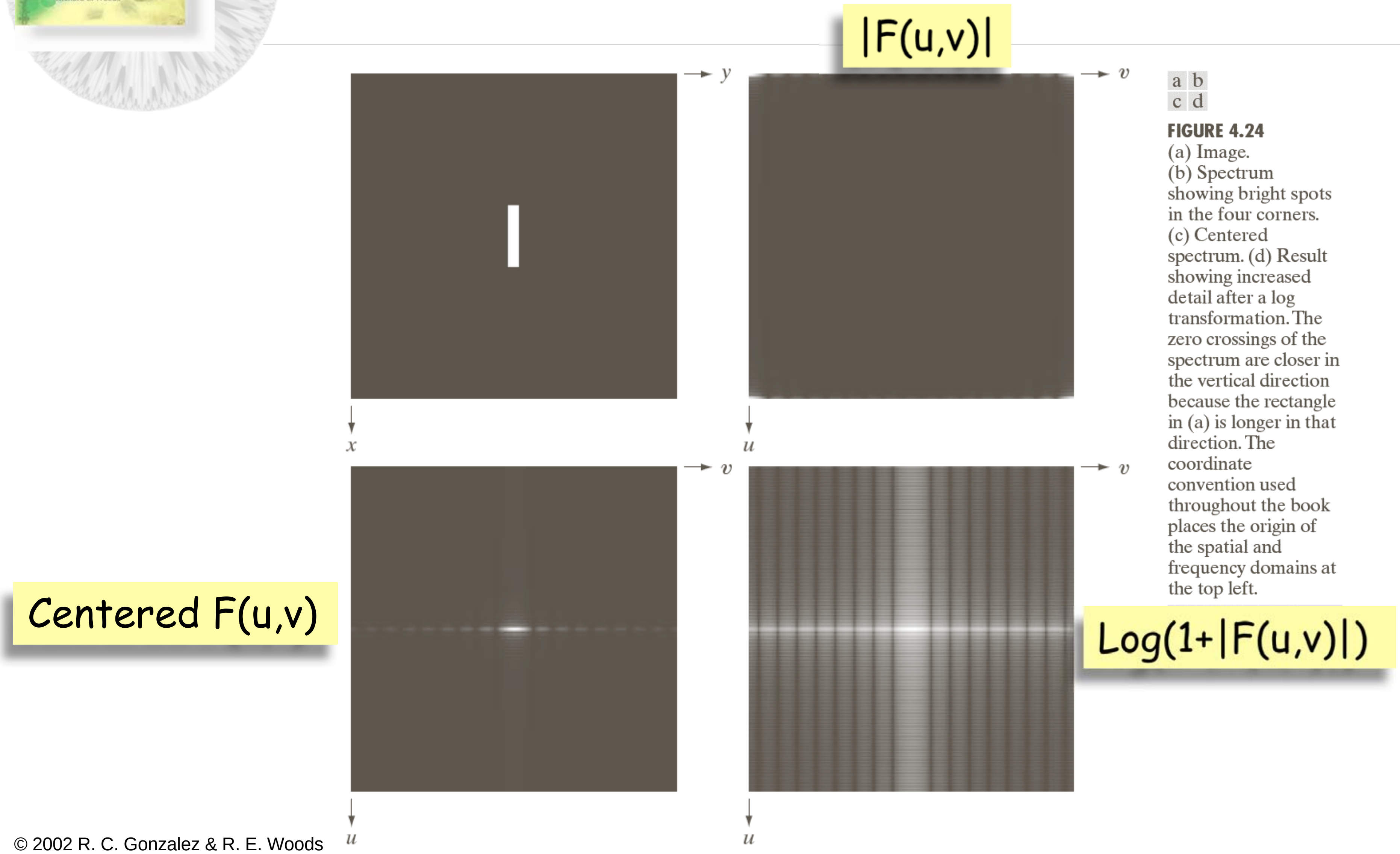
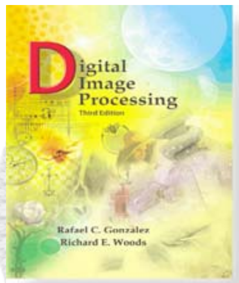


FIGURE 4.24 (a) Image. (b) Spectrum showing bright spots in the four corners. (c) Centered spectrum. (d) Result showing increased detail after a log transformation. The zero crossings of the spectrum are closer in the vertical direction because the rectangle in (a) is longer in that direction. The coordinate convention used throughout the book places the origin of the spatial and frequency domains at the top left.



Properties of 2-D DFT

$$f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi\left(u\frac{x_0}{M}+v\frac{y_0}{N}\right)}$$

$$f(r,\theta+\theta_0) \Leftrightarrow F(\omega,\varphi+\theta_0)$$

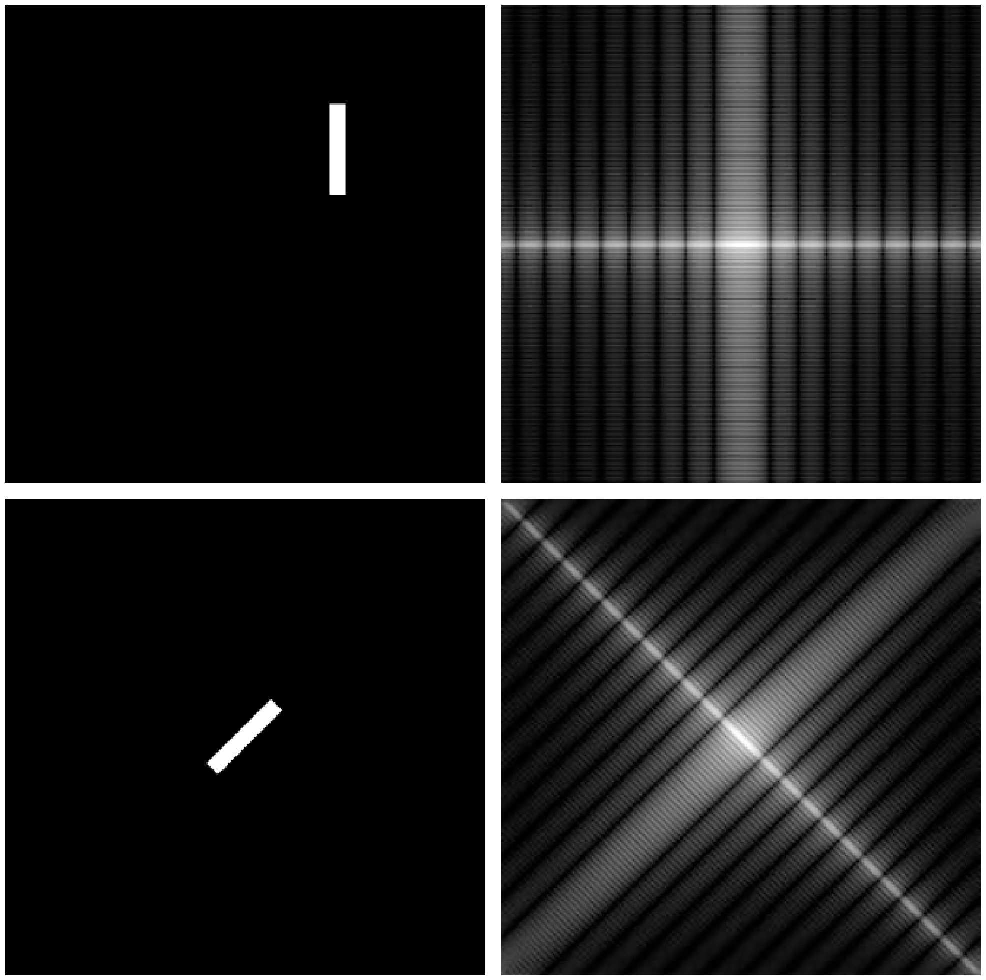
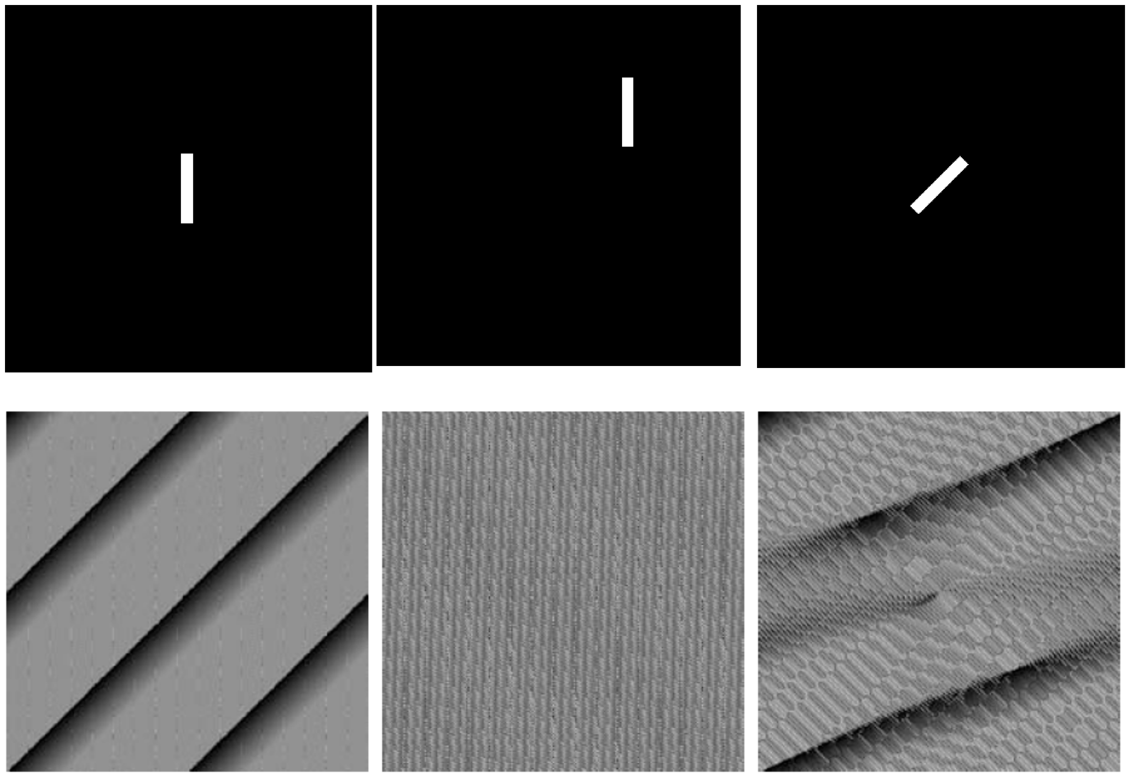


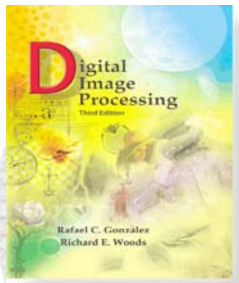
FIGURE 4.25 (a) The rectangle in Fig. 4.24(a) translated, and (b) the corresponding spectrum. (c) Rotated rectangle, and (d) the corresponding spectrum. The spectrum corresponding to the translated rectangle is identical to the spectrum corresponding to the original image in Fig. 4.24(a).

Properties of ∠2-D DFT

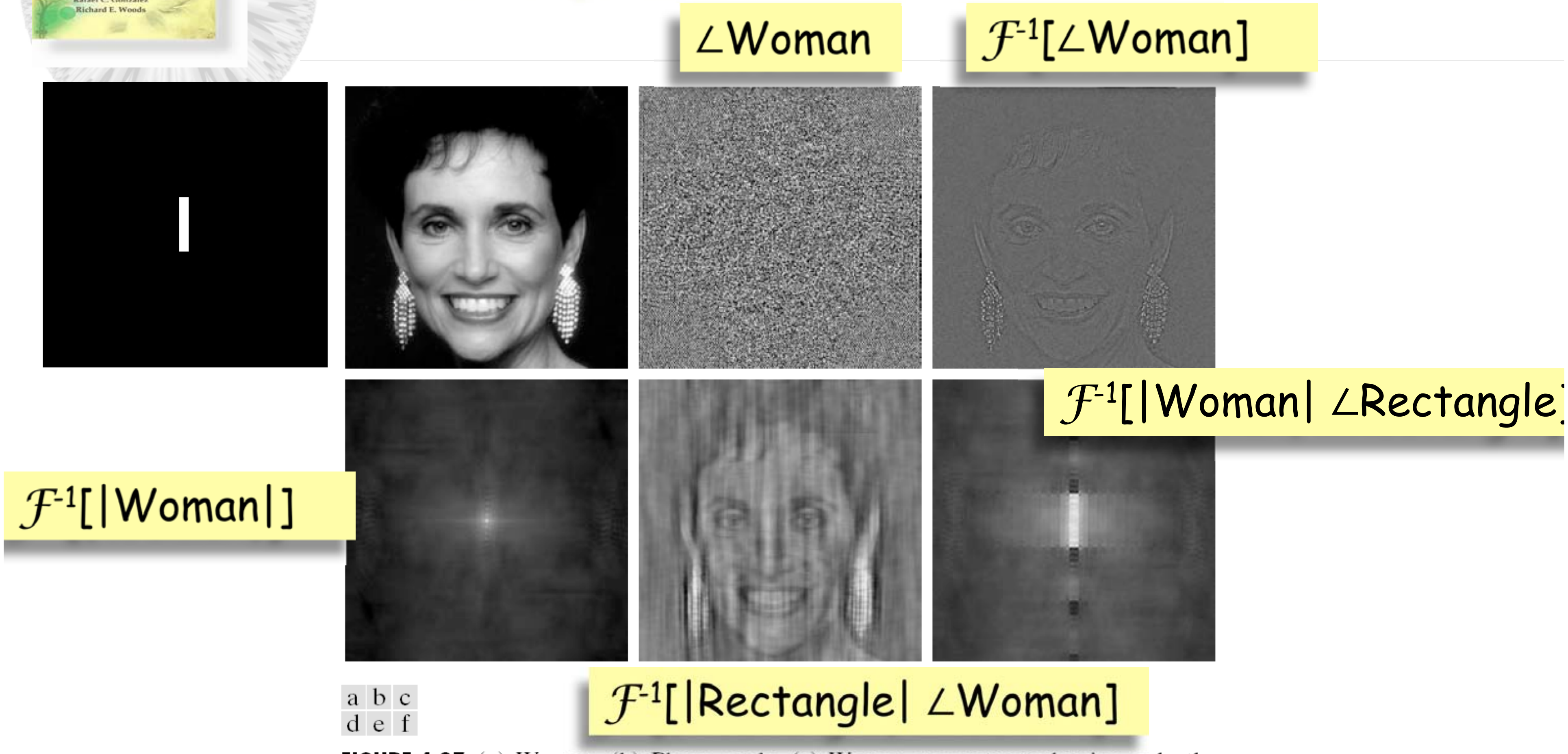


a b c

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).



Properties of 2-D DFT



a b c
d e f

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.