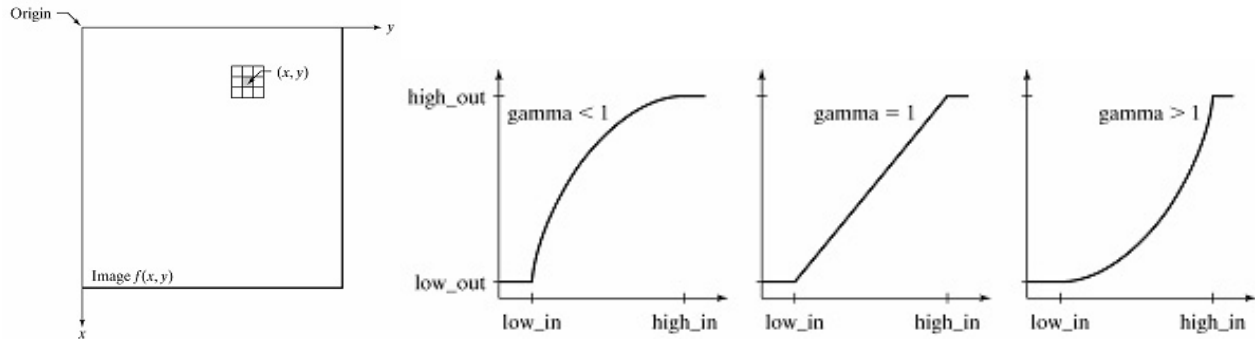


## Chapter 5 Image Processing in the Spatial-Frequency Domains

### Processing in Spatial-Domain



Operations in the spatial domain follows a generic formula:

$$g(x, y) = T[f(x, y)]$$

If the support region is only a single pixel then they are known as **point operations**. If it is over a region then called **mask operations**.

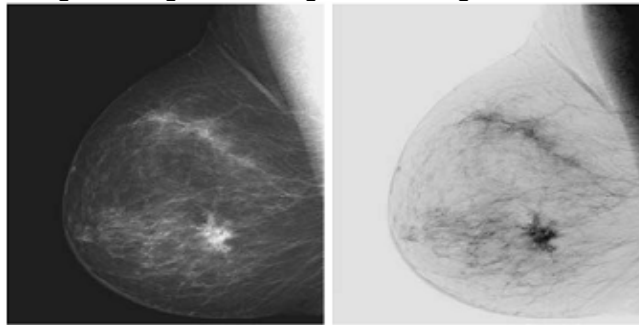
Intensity transformation function:

$$s = T(r)$$

where  $s, r$  are input and output variables, respectively. One such arrangement is shown above.

#### Examples:

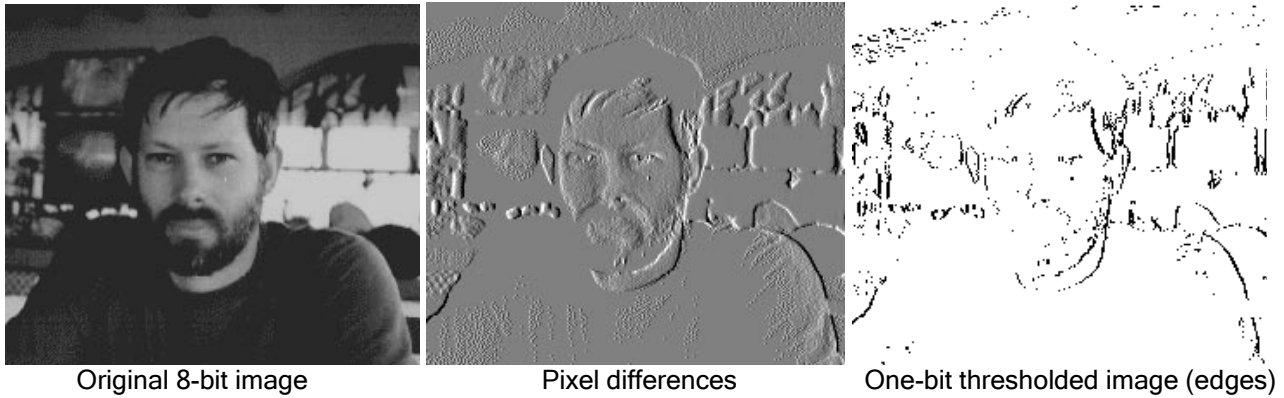
**Negation:** Below we have the original image from a digital mammogram and its negated version.



**Pixel differencing: Finding vertical edges:** An edge is a junction between two pixels where there is a significant change in gray-level. A simple operation to find vertical edges is just to subtract each pixel's gray-level from that on its right. If changes bigger than, say, 10 gray levels are regarded as significant, then we can produce a binary image indicating where there are edges. Consider a 3x4 segment of image:

	4	5	6	7		5	6	7		5	6	7
4	14	17	20	21	→	3	3	1		0	0	0
5	19	30	35	36		11	5	1	→	1	0	0
6	18	45	40	38		27	-5	-2		1	0	0

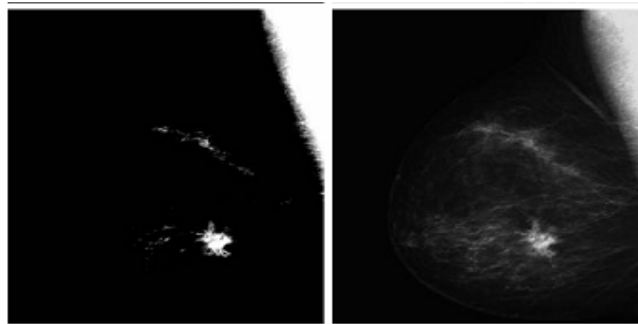
Note that we have to make an arbitrary choice to put the result of subtracting (4,4) from (5,4) into (5,4). The second operation is thresholding. Choosing the threshold (in this case 10) is a problem in its own right. If we want to detect all vertical edges, then we need to threshold so that we record a "1" wherever the difference is greater than 10 or less than -10. These two operations, though simple, in fact conform to the structure used by a very wide range of visual systems.



**Log and Power Law Transformations:** They are defined by the following expression<sup>1</sup>:

$$s = c \cdot \log(1 + r) \quad \text{for log} \qquad s = c \cdot r^\gamma$$

where  $r$  is the input and  $s$  is the output of the transformation,  $r$  is constant (usually 1.0) and  $r \geq 0$ . In GW figure 3.6 they present the frequently employed family of log transformations in a graphical form. Below we display two pixel level enhancements: Pixel intensity expansion in the range [0.25,0.75] and the power transformation with the  $\gamma = 2.0$  from figure 3.6 of GW.



**3. Histogram Processing:** Probability approximations are found for each pixel by computing first the frequency of occurrences of each gray level (histogram):  $h(r_k) = n_k / n$  for  $k = 0, 1, \dots, L-1$  and then they are normalized to obtain probabilities. They could be histogram points, bar graphs, discrete impulses (stem) or continuous plots.

Transformation based on histogram equalization: Given that we know the pdf (or histogram) for the pixel intensity in an image, the histogram equalized output is the cdf (area under the pdf curve (accumulated histogram values)):

Continuous case:  $s = T(r) = \int_0^r p_r(u) \cdot du$

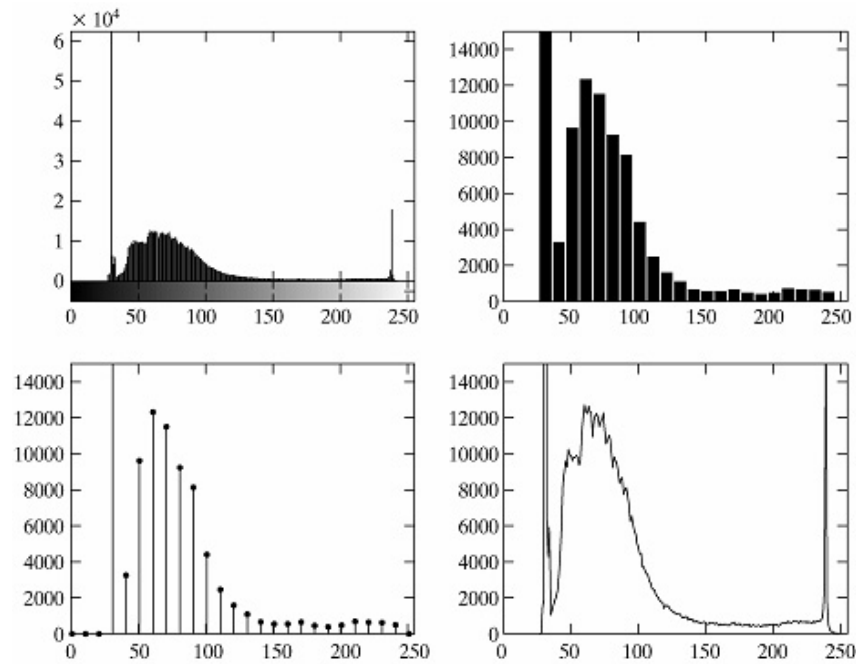
Discrete case:  $s_k = T(r_k) = \sum_{j=0}^k p_r(r_j) = \sum_{j=0}^k n_j / n$  for  $k = 0, 1, \dots, L-1$

Inverse transformation can be used to come back to the original case, i.e., invertible operation:

$$r_k = T^{-1}(s_k)$$

There is a neat 5-point algorithm for implementing histogram equalization including examples in GW p:99-102:

<sup>1</sup> It is worth noting that GW text starts with  $r$  being input and  $s$  as the output in the first description, but immediately, swp the definitions of these two variables starting with the log and power law transformations. This is a well-known inconsistency and we need to be careful.



Transformation based on histogram equalization: Given that we know the pdf (or histogram) for the pixel intensity in an image, the histogram equalized output is the cdf (area under the pdf curve (accumulated histogram values)):

$$\text{Continuous case: } s = T(r) = \int_0^r p_r(u) du$$

$$\text{Discrete case: } s_k = T(r_k) = \sum_{j=0}^k p_r(r_j) = \sum_{j=0}^k n_j / n \quad \text{for } k = 0, 1, \dots, L-1$$

Inverse transformation can be used to come back to the original case, i.e., invertible operation:

$$r_k = T^{-1}(s_k)$$

There is a neat 5-point algorithm for implementing histogram equalization including examples in GW p:99-102:

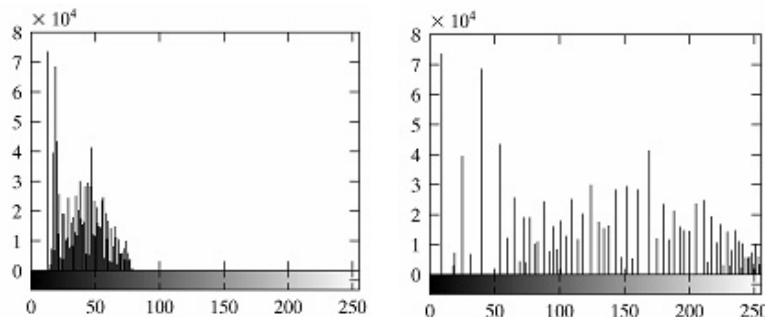
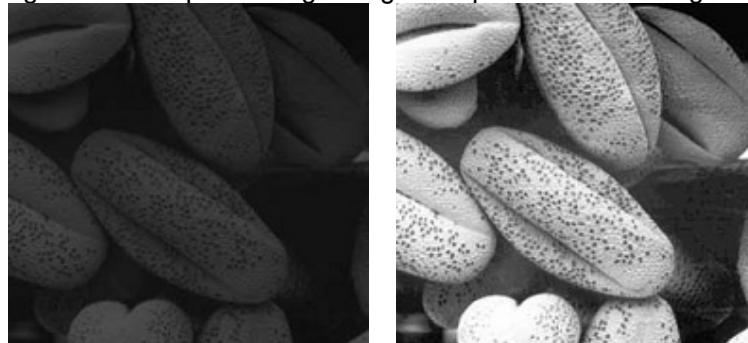
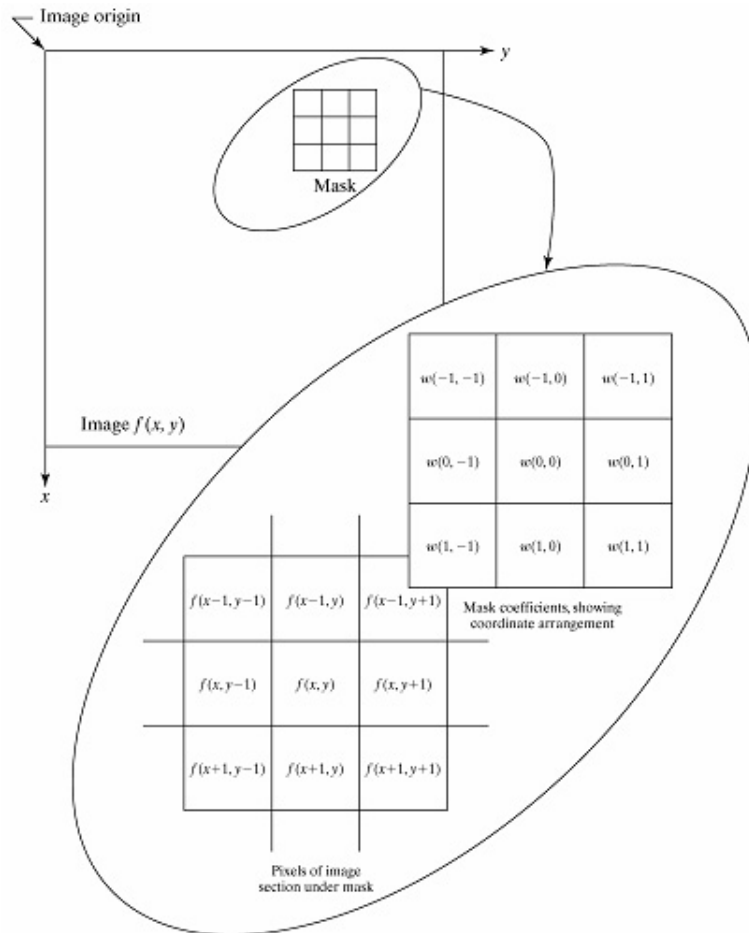
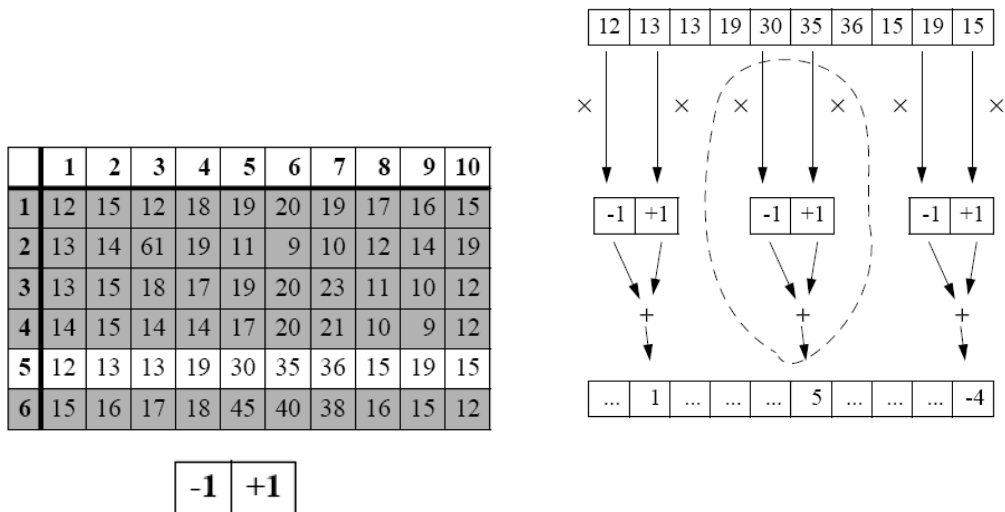


Image enhancement using convolution masks with  $m$  rows and  $n$  columns (convolution kernels):

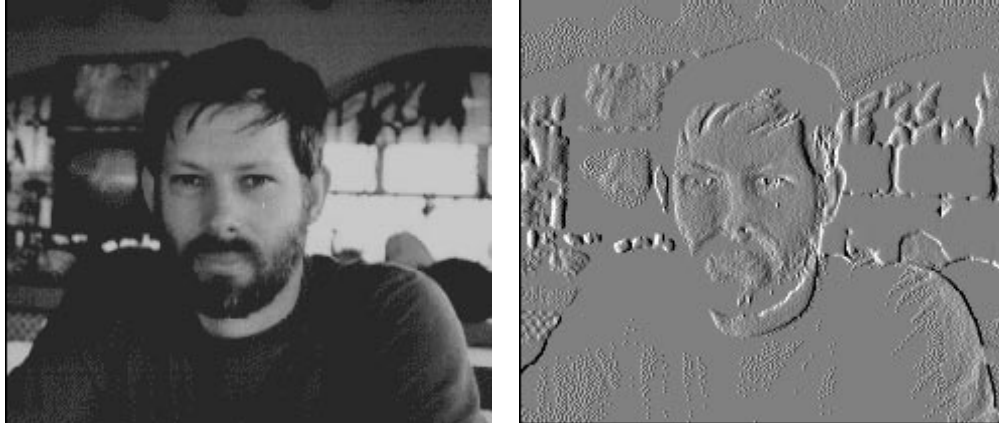
$$g(x, y) = \sum_{s=-a}^a \sum_{u=-b}^b w(s, u) \cdot f(x + s, y + u) \quad \text{with } a = \frac{m-1}{2}; b = \frac{n-1}{2}$$



Consider one row of the image and a horizontal mask, above equation is operated only over a single row:



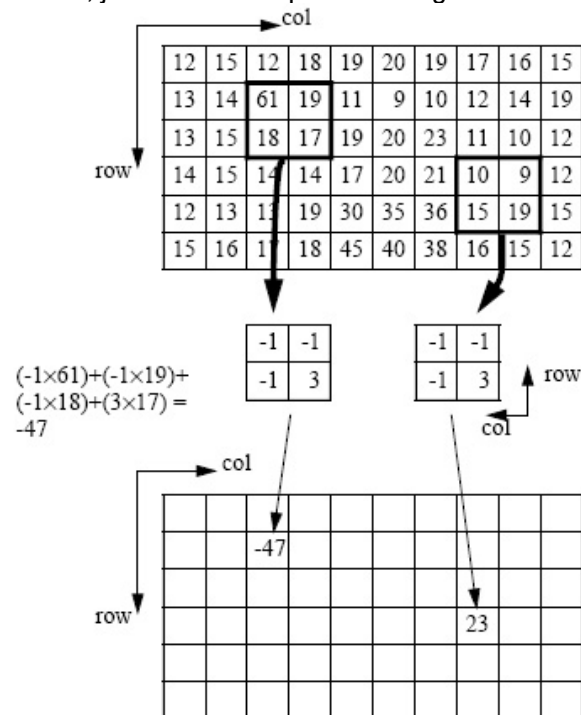
The impact on the image is smoothing (averaging) resulting in emphasizing vertical edges.



- Masks could be vertical, which results in vertical edge enhancement.
- Masks could be more than 2 columns or rows, combining a larger range of pixel values. For example the example below combines some smoothing and horizontal differencing:

-1	-2	0	+2	+1
----	----	---	----	----

- Mask can be 2-dimensional—in fact, just like a small piece of image.



- Diagonal Differences:

-1	0	0	-1
0	+1	+1	0

- Center surround mask (averaging, smoothing);

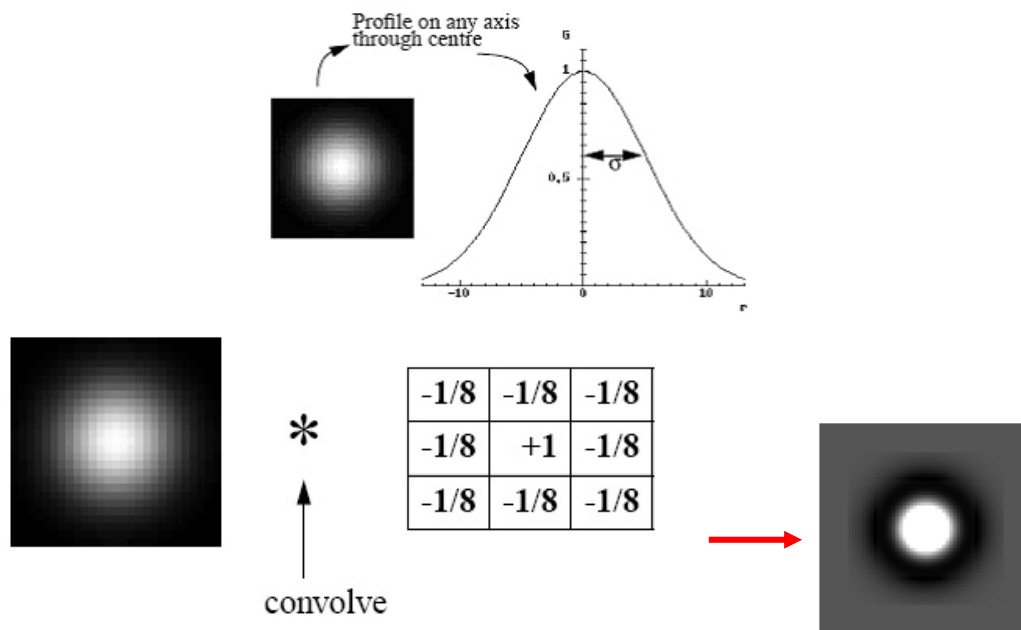
-1/8	-1/8	-1/8
-1/8	+1	-1/8
-1/8	-1/8	-1/8



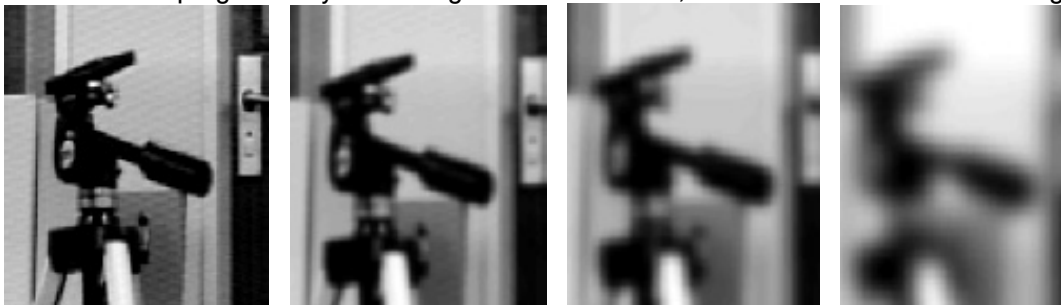
- The example below also combines smoothing and differencing, and is known as the **Sobel operator**.

-1	0	+1
-2	0	+2
-1	0	+1

**Gaussian smoothing mask:** A more effective smoothing mask falls off gently at the edges, where its width is described by the standard deviation  $\sigma$ .



They are effectively used for removing any texture with a scale smaller than the mask dimensions. Small-scale texture is said to have a high *spatial frequency*. Smoothing removes this, leaving low spatial frequencies. Here is the effect of progressively increasing  $s$ . Its values are 1, 2 and 4 in the 3 smoothed images.

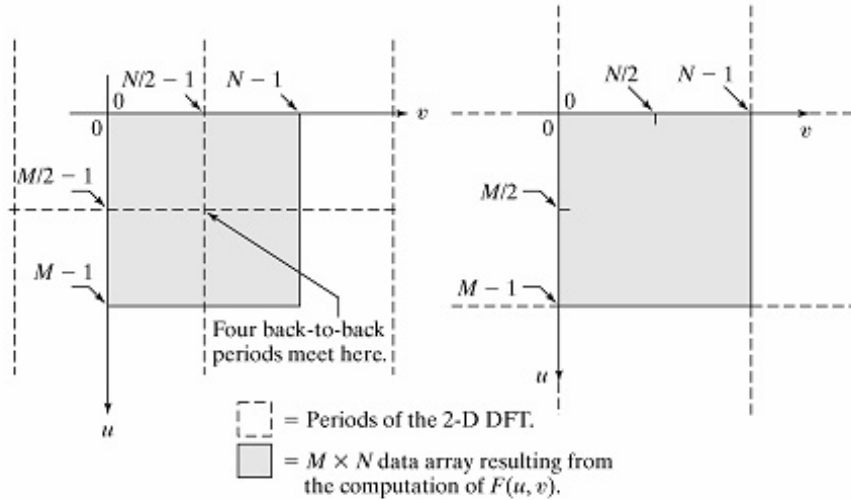


- In addition, there are median filtering masks and Laplacian masks used in spatial-domain image enhancement.
- Finally, various combined masks are used in image sharpening. These are discussed in detail in GW 3.6-3.7.

## Processing in Frequency-Domain

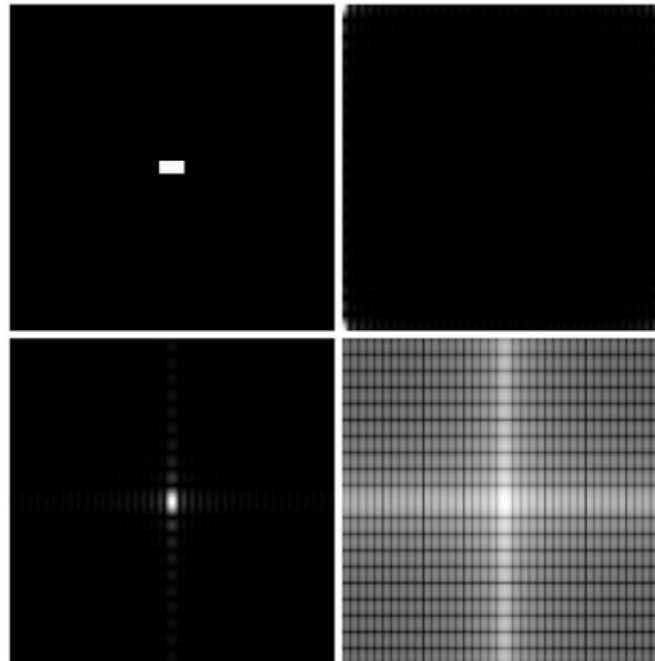
Based on the notation used in GW, the FT for an  $M \times N$  image is given by:

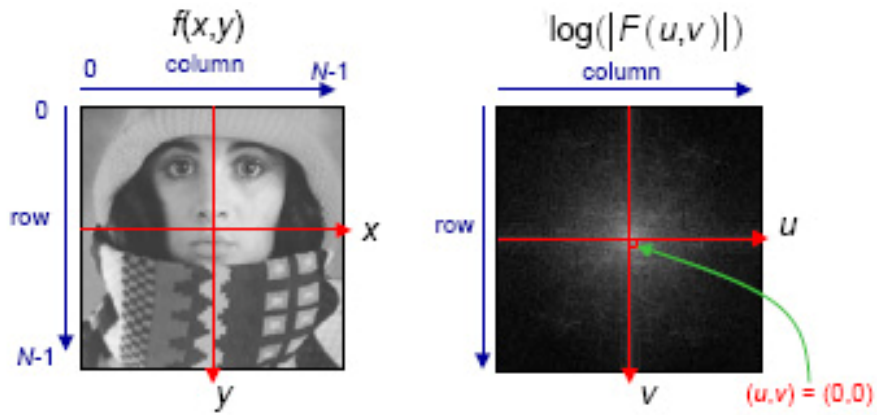
$$F(u, v) = \frac{1}{M \cdot N} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(x, y) e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$



a b

**FIGURE 4.2** (a)  $M \times N$  Fourier spectrum (shaded), showing four back-to-back quarter periods contained in the spectrum data. (b) Spectrum obtained by multiplying  $f(x, y)$  by  $(-1)^{x+y}$  prior to computing the Fourier transform. Only one period is shown shaded because this is the data that would be obtained by an implementation of the equation for  $F(u, v)$ .



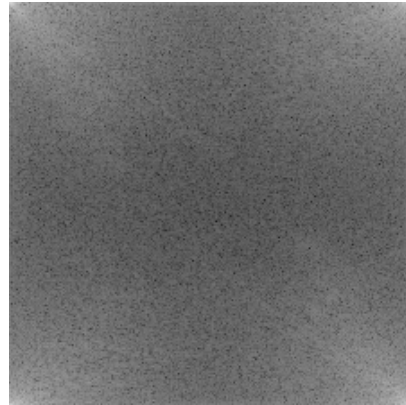


Lena (512x512) 8-bits/pixel original. FFT log magnitude without placing (0,0) in the center, shifted version (commonly known as the DFT) and the corresponding phase spectrum. (using 2-D fft algorithm in Image Processing toolbox in Matlab.)

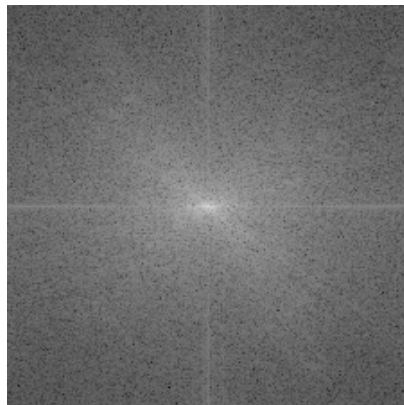
Image



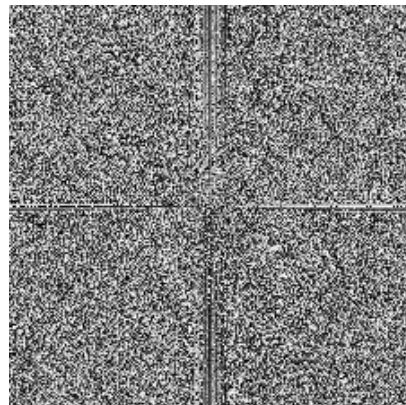
FFT log Magnitude (not shifted)



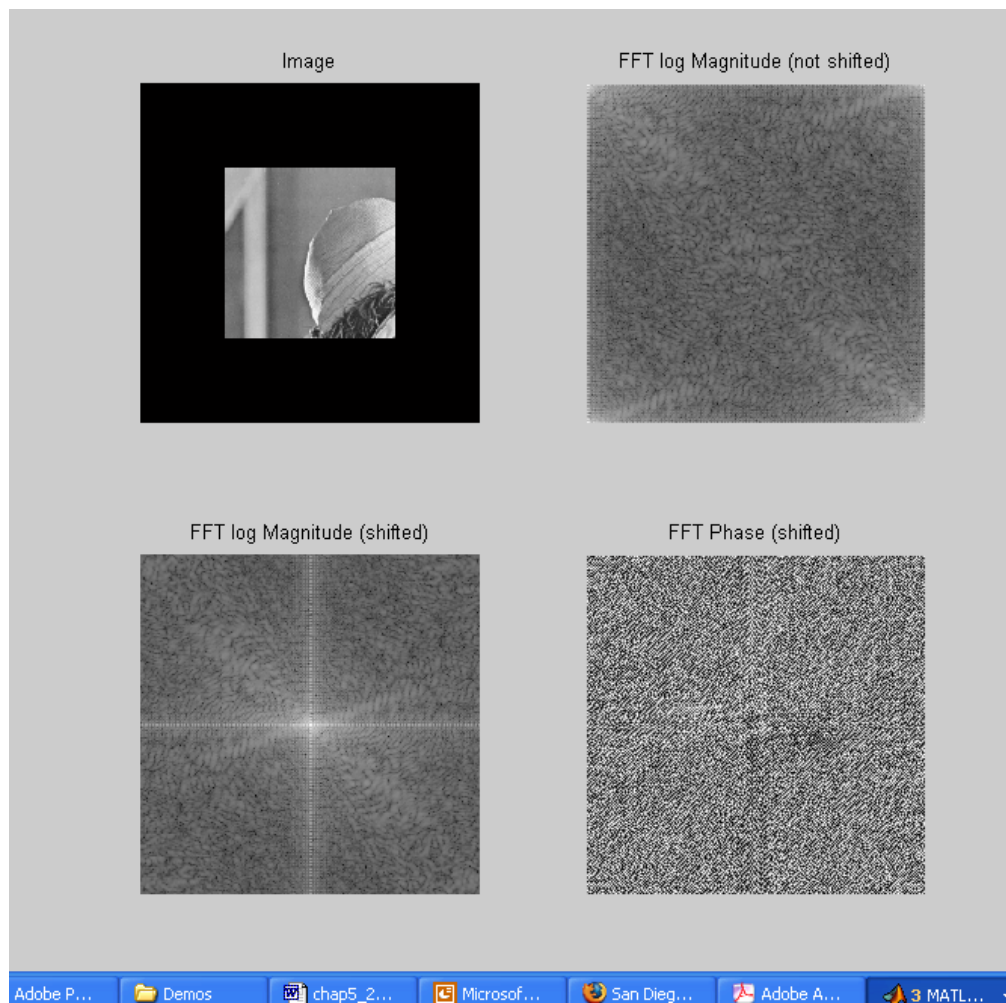
FFT log Magnitude (shifted)



FFT Phase (shifted)

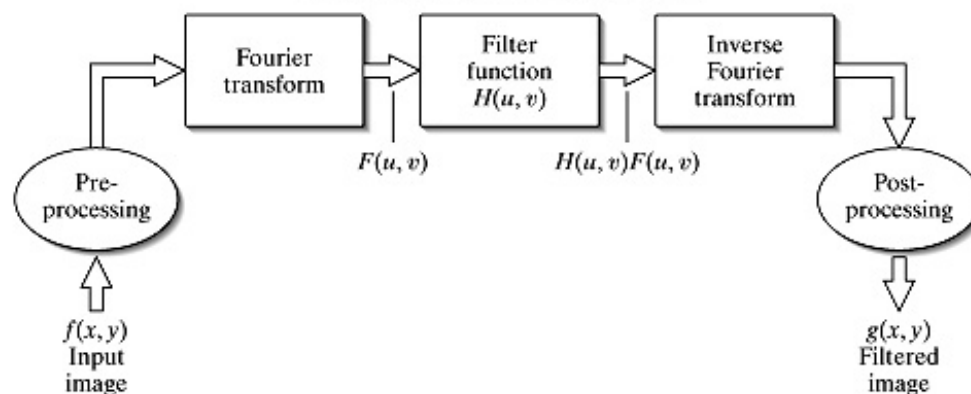







## Image Enhancement via Filtering

Frequency domain filtering operations




Even more critical than the case in 1-D signal filtering, fast convolution is ubiquitously used to implement filters in image processing applications. Recall that 2 forward FFTs, 2 inverse FFTs and a point-by-point image multiply in the frequency-domain will be at a significantly smaller fraction of the time required to perform a 2-D convolution.

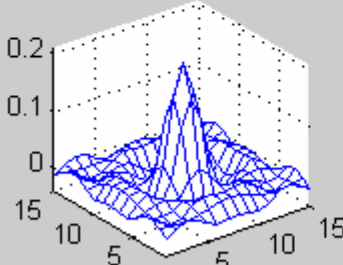
Original Vertigo Image



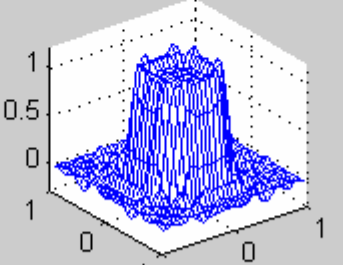
Filtered Image



Filter Coefficients



Frequency Response



Apply Filter

Select an Image: Vertigo

Type of filter: lowpass

Window method: Bartlett

Cutoff: 0.5

Order: 15

Design Method:

fsamp2

fwind1

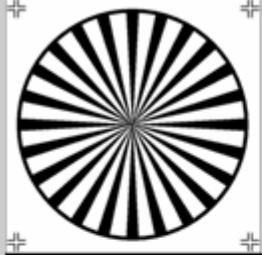
fwind2

ftrans2

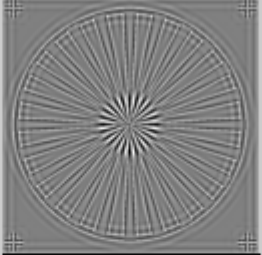
Info

Close

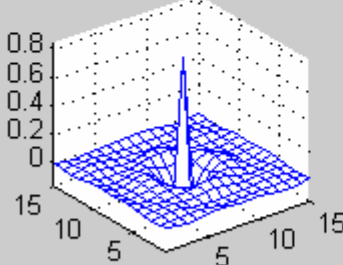
Original Vertigo Image



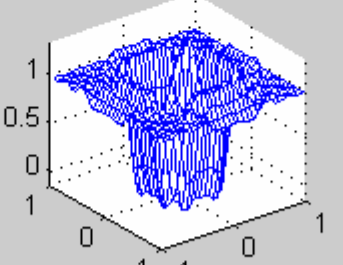
Filtered Image



Filter Coefficients



Frequency Response



Apply Filter

Select an Image: Vertigo

Type of filter: highpass

Window method: Bartlett

Cutoff: 0.5

Order: 15

Design Method:

fsamp2

fwind1

fwind2


ftrans2

Info


Close

© Hüseyin Abut, August 2005

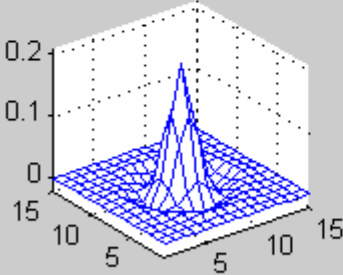
Original Pepper Image



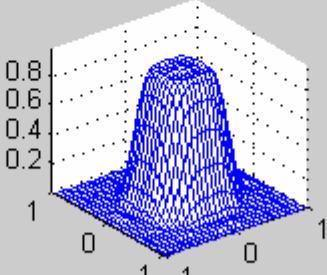
Filtered Image



Filter Coefficients



Frequency Response



Select an Image: Pepper ▾

Type of filter: lowpass ▾

Window method: Bartlett ▾

Cutoff:

Order:

Design Method:


fsamp2

fwind1

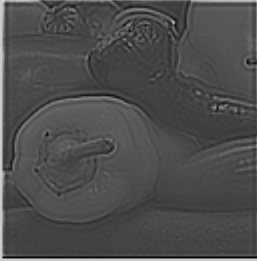
fwind2

ftrans2

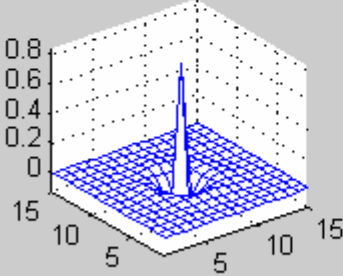
Original Pepper Image



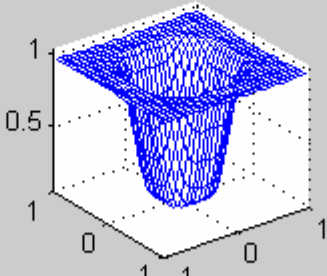
Filtered Image



Filter Coefficients



Frequency Response



Select an Image: Pepper ▾

Type of filter: highpass ▾

Window method: Bartlett ▾

Cutoff:

Order:

Design Method:

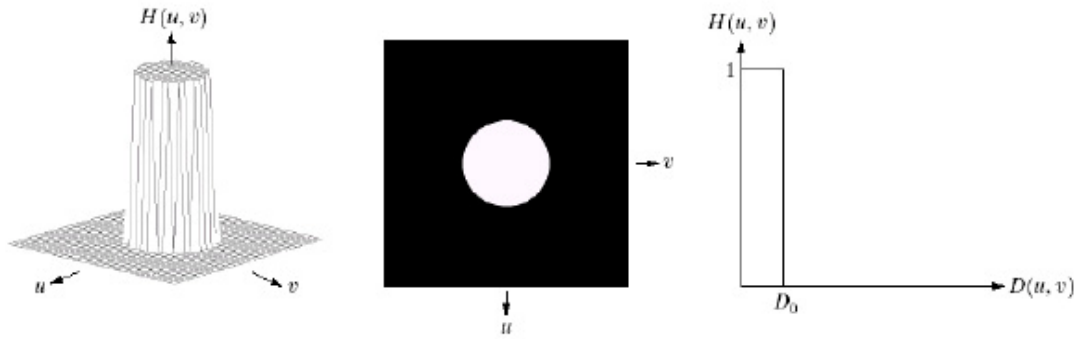
fsamp2

fwind1

fwind2

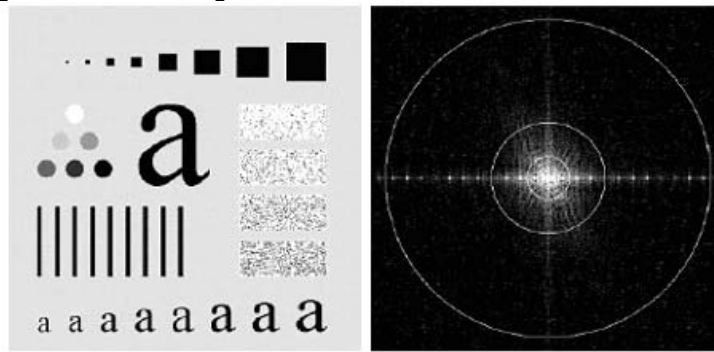
ftrans2

Ideal low-pass using rectangular spatial window:



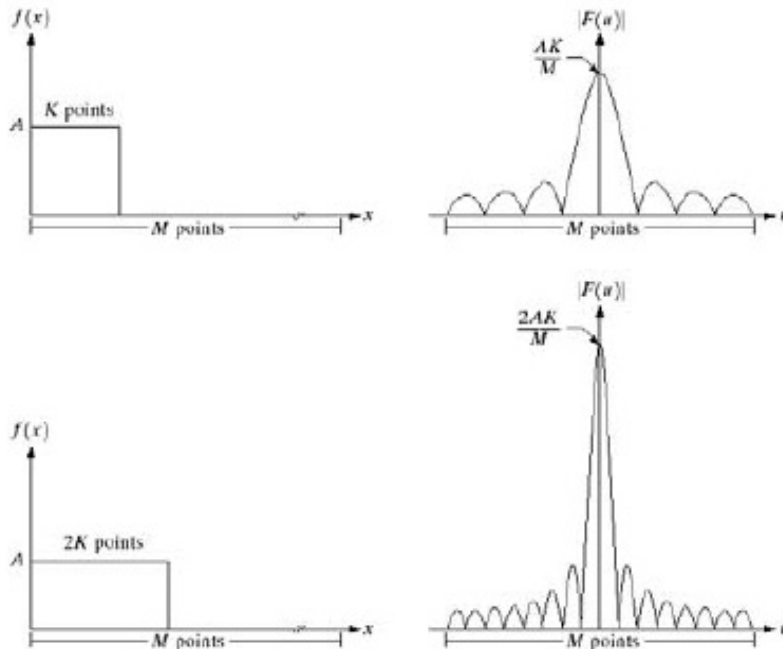
**FIGURE 4.10** (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Power Distribution through 2-D DFT in images:

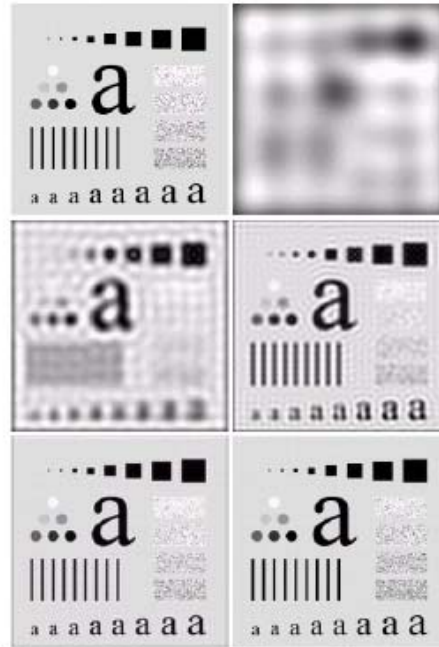


**FIGURE 4.11** (a) An image of size  $500 \times 500$  pixels and (b) its Fourier spectrum. The superimposed circles have radii values of 5, 15, 30, 80, and 230, which enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power, respectively.

Ideal low-pass filtering results in blurring due to smoothing (averaging) and ringing due to Gibbs phenomenon of representing using a finite number of terms in representing an ideal rectangular pulse.

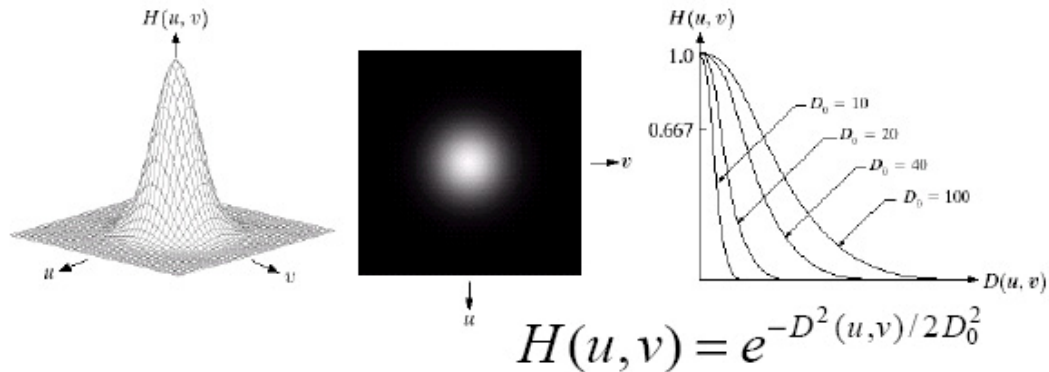


**FIGURE 4.2** (a) A discrete function of  $M$  points, and (b) its Fourier spectrum. (c) A discrete function with twice the number of nonzero points, and (d) its Fourier spectrum.



**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.

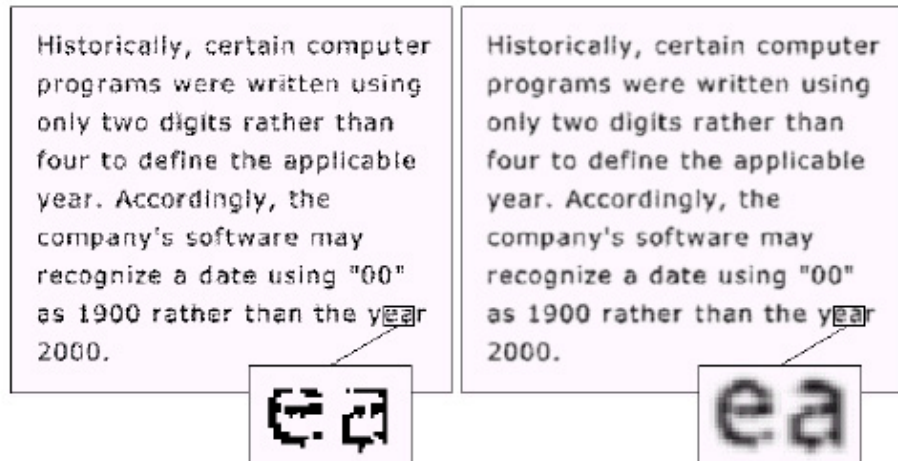
Low-pass filtering using Gaussian Low-pass filter characteristics.



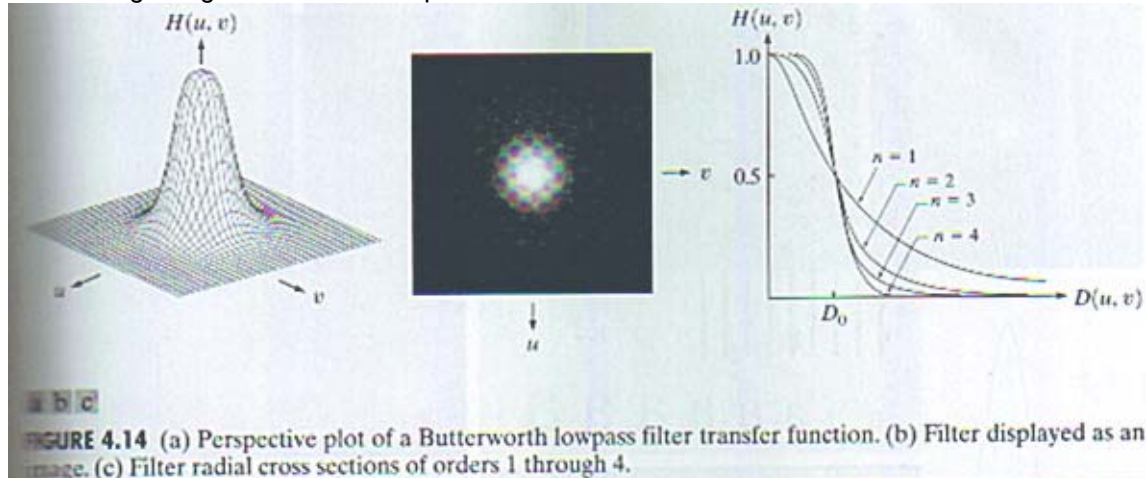
**FIGURE 4.17** (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of  $D_0$ .

Application to Text Smoothing:

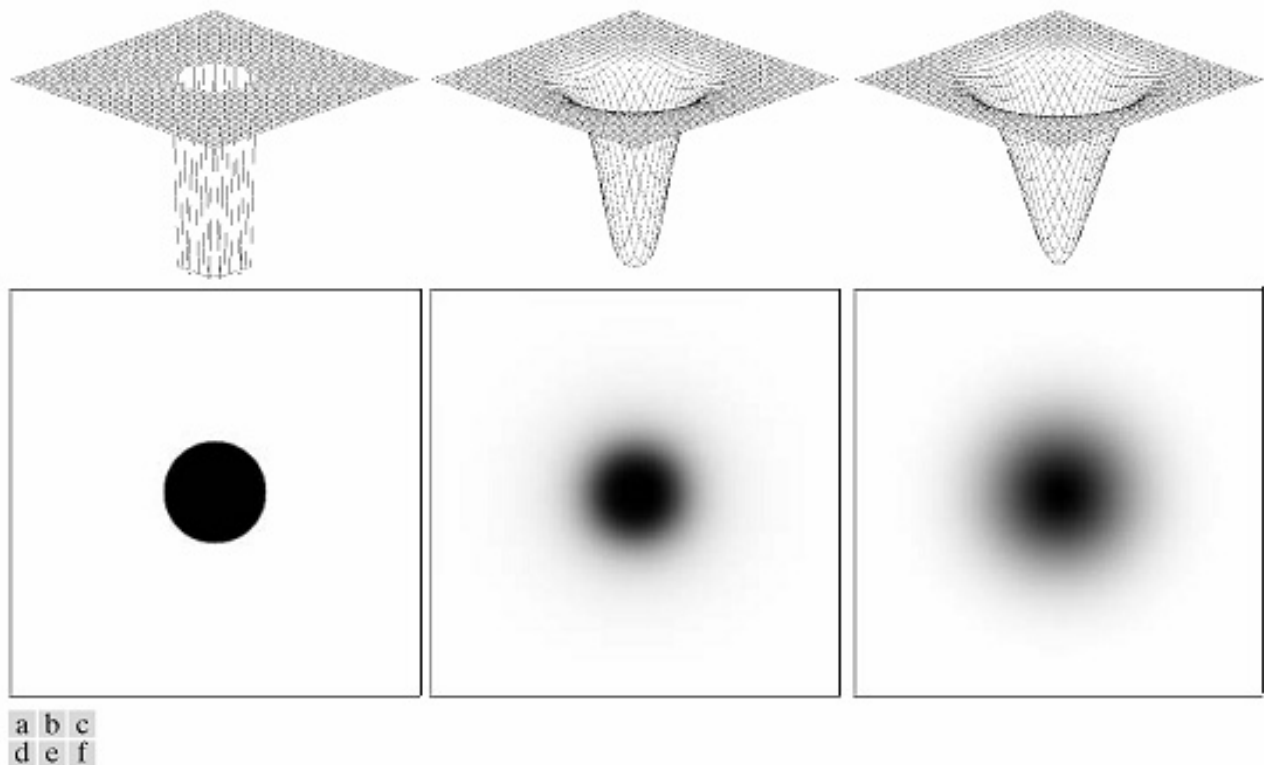
**FIGURE 4.19** (a) Sample text of poor resolution (note broken characters in magnified view). (b) Result of filtering with a GLPF (broken character segments were joined).



Low-pass filtering using Butterworth Low-pass filter characteristics.



High-pass filtering using ideal, Butterworth and Bartlett windows.



Padding Images with zeros is very frequently used for computational efficiency gains. For instance an image of size  $480 \times 640$  will be processed significantly faster via fast convolution using 2-D FFT by padding both rows and columns with zeros to bring to a  $512 \times 1024$ , i.e.,  $(2^9 \times 2^{10})$  and then doing 2-D IFFT. Otherwise, we would need  $480 \times 640$  double convolution or IDFT, which are both equally costly by a couple of orders of power!

Below we have an example of padding with zeros and reconstruction. The effects in the original image (NW corner) is not significant but the impact on the remaining three quadrants are there. But this does not play any role since we reformat the image back to the original  $480 \times 640$  to generate a replica.



**FIGURE 4.39** Padded lowpass filter in the spatial domain (only the real part is shown).

- Note origin shifting



**FIGURE 4.40** Result of filtering with padding. The image is usually cropped to its original size since there is little valuable information past the image boundaries.

In Sections GW 4.5-4.7, there are several other filtering techniques, such as Laplacian and homomorphic filtering. In addition, subjects dealing with using 2-D FFT for image processing applications are discussed including zero-padding for avoiding DFT usage.