①

"14"

An M-function for Filtering in the Frequency Domain

an M-function **dftfilt** that accepts as inputs an image and a filter function, handles all the filtering sequence steps, and outputs the filtered, cropped image.

Code :

```
function g = dftfilt(f, H)
% DFTFILL performs frequency domain filtering.
% G= DFTFILT(F, H) filters F in the frequency domain
% using the filter transfer function H. The output, G,
% is the filtered image, which has the same size as F.
% DFTFILT automatically pads F to be the same
% size as H. Function PADDEDSISE can be
% used to determine an appropriate size for H.
% DFTFILT assumes that F is real and that H is
% a real, Uncentered filter function.

% obtain the FFT of the padded input.
   F = fft2(f, size(H,1), size(H,2));
% perform filtering.
   g = real(ifft2(H.*F));
% Crop to original size.
   g = g(1: size(f,1), 1: size(f,2));
```

* using the function : (Calling from Matlab)

```
>> f= imread('moon.tif');
>> h= fspecial('sobel');
>> H = fft2(h, 1024, 1024);
>> g = dftfilt(f, H); % error (f: must be double)   (warning)
>> g = dftfilt( double(f), H); ✓
>> subplot(2,1,1);
>> imshow(f, [ ]);
>> subplot(2,1,2);
>> imshow(g, []);
```

Obtaining Frequency Domain Filters from Spatial Filters.

- In general spatial domain filtering is more effecient than frequency domain filtering when the filters are small.
- One way of obtaining a frequency domain filter H from a given spatial filter h is to use:

  H = fft2 (h, PQ(1), PQ(2)).

  which converts from a filter defined in the spatial domain to an equivalent filter in the frequency domain.
- another way to obtain a frequency domain filter is to use Matlab function freqz2 to compute the frequency response of a spatial filter as:

  H = freqz (h, R, c);

where: h: 2-D spatial filter. H: 2-D frequency filter.
R: number of rows C: number of Columns.

Example: (A comparison of filtering in the spatial and frequency domains)

Sobel Mask

```
f = imread('moon.tif');
h = fspecial('sobel') ; % apply a spatial filter Sobel
                        " Vertical Edge Detector"
freqz2(h) % Draw 3-D filter
          % transfer function
PQ = paddedsize(size(f)); padding the image
H = freqz2(h, PQ(1), PQ(2));
H1 = ifftshif(H) ; % Centering the frequency rectangle.
gs = imfilter(double(f), h); % filtering in spatial domain
gf = dftfilt(f, H1);  % filtering in frequency domain
subplot (3,1,1); imshow(f, [ ]);
sub plot (3,1,3); imshow(gs, [ ]);
Subplot (3,1,5) ; imshow(gf, [ ]);
% Note: if freqz2 is written without an output argument
% the absolute value of H is displayed on the Matlab
% desktop as a 3-D plot.
   Subplot(3,2,2); imshow(abs(H), [ ]); % absolute
   Subplot(3,2,4); imshow(abs(H1), [ ]); % Values of
*% imfilter pads the border with zeros.        % H and H1
```

```
subplot (3,2,6);
imshow ( abs(gf) > 0.2 * abs(max(gf(:))) );
% 0.2 multiplier → to show only the edge with strength
% greater than 20% of the maximum values of gf.
% d = abs( gs - gf);  to compare the two methods by
m1 = max(d(:))   % Computing their difference:
                 % = 5.4 e-012
m2 = min(d(:))   % = 0
```

Generating Filters Directly in the Frequency domain.

1) Low pass Frequency domain Filters (smoothing filters)

- One approch is to zero-pad images and then create filters in the frequency domain to be of the same size as the padded image (Remember: images and filters must be of the same size when using DFT)
- Smoothing (bluring) is achieved in the frequency domain by high-frequency attenuation using Lowpass filter

filters kinds:

1) Ideal low pass filter: ILPF: (very sharp filtering)
2) Butterworth: BLPF:

Butterworth filter has a parameter called the filter order, For high order values, the butterworth filter approaches the ideal filter, For lower order values, the butterworth filter is more like a Gaussian filter. "Butterworth provides a transition between two "extremes"

3) Gaussian: GLPF: very smooth filtering.

A) Ideal Low pass Filters:

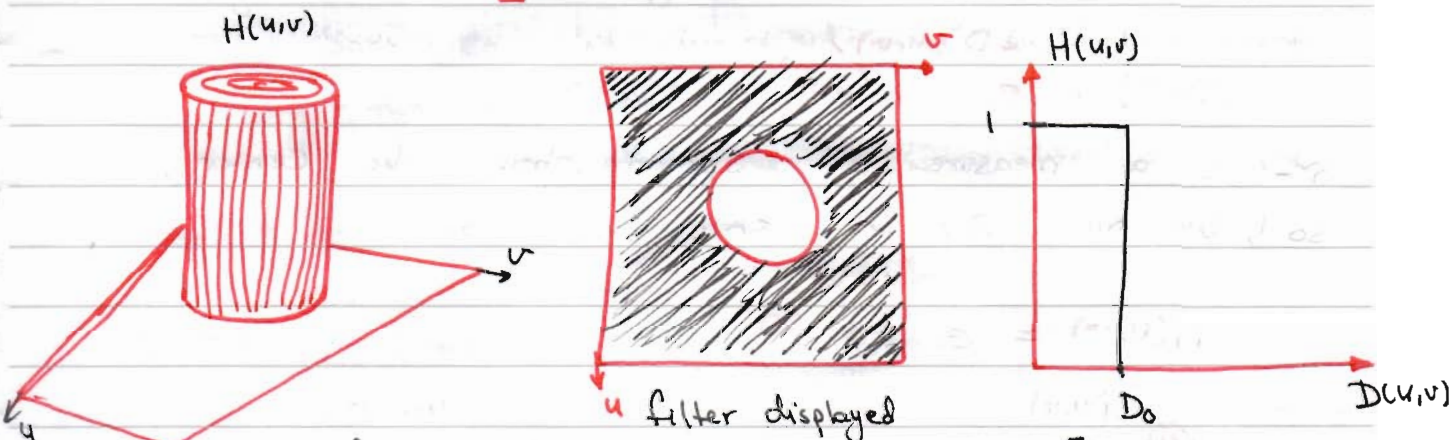
A 2-D low pass filter that passes without attenuation all frequencies within a circle of radius D_0 from the origin and "cut off" all frequencies outside this circle is called an Ideal Lowpass filter (ILPF);

it is specified by the function :

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

D_0 - positive Constant., $D(u,v)$ - distance between a point (u,v) in the frequency domain and the center of the frequency rectangle, that is

$$D(u,v) = \left[\left(u - \frac{P}{2}\right)^2 + \left(v - \frac{Q}{2}\right)^2 \right]^{\frac{1}{2}},$$ P, Q - padded size.



ILPF transfer function — filter displayed as an image — filter radial cross section.

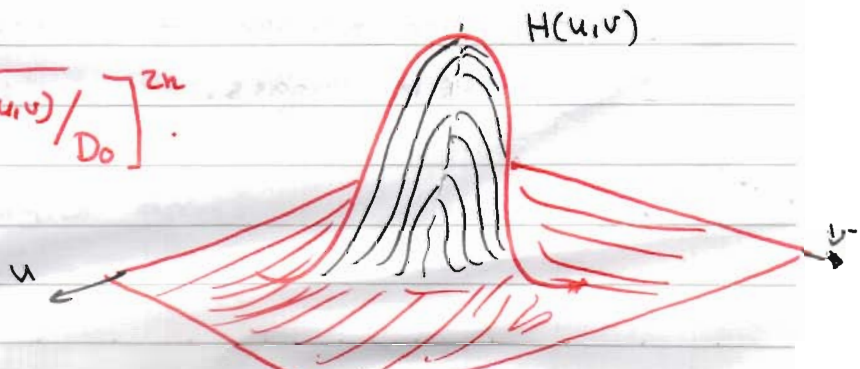
D_0 - Cutoff frequency →

the sharp cutoff frequencies of an ILPF cannot be realized with electronic components, it can be just simulated in a computer.

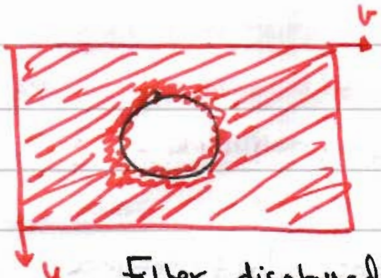
B) Butterworth Lowpass Filters :

BLPF transfer function of order n, and with cutoff frequency at a distance D_0 from the origin, is defined as ; n - the order of the filter.

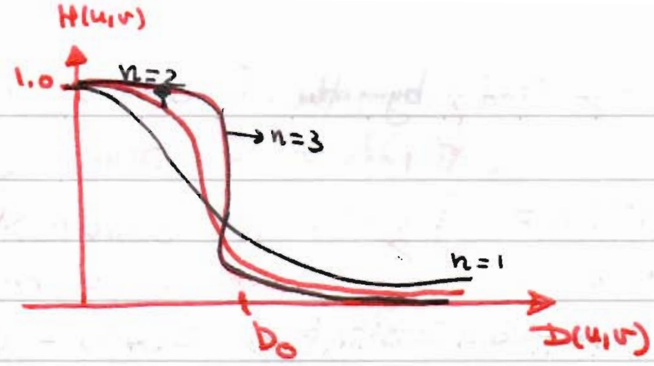
$$H(u,v) = \frac{1}{1 + \left[D(u,v)/D_0 \right]^{2n}}.$$



BLPF transfer function.



Filter displayed as an image



Filter radial cross section for orders 1, 2, and 3.

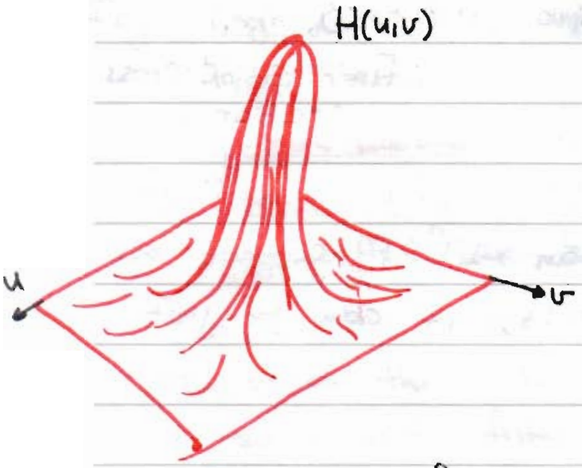
c) Gaussian Lowpass filters:

the transfer function :

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

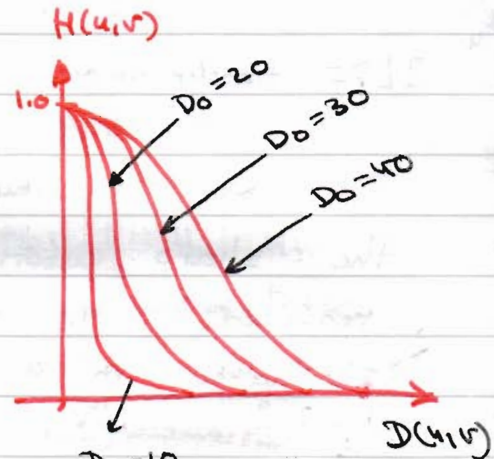
σ - is a measure of spread about the center, so, we put, $\sigma = D_0$, so:

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



GLPF transfer function

Filter displayed as an image



Filter radial cross sections for various values of Do

* Practical application of LPF:

- Character Recognition (machine recognition systems)
- Satellite images.

MatLab Implementation of LPF:

1) Creating Mesh grid Array for Use in implementing Filters in the Frequency domain.

- We need a function that computes the distance functions from any point to a specified point in the frequency rectangle.
- our distance computations are with respect to the origin (top, left point), because of FFT function.
- The data can be rearranged for visualization purposes using function fftshif.

function dftuv provides the necessary mesh grid array for use in distance computations.

```
function [u,v] = dftuv (M,N)
% DFTUV Computes meshgrid frequency matrices.
% [u,v] = DFTUV (M,N) Computes meshgrid frequency
% matrices U and V. U and V are useful for
% Computing frequency-domain filter.
% functions that can be used with DFTUV. U and V
% are both M-by-N.

% Set up range of variables.
u = 0: (M-1);
v = 0: (N-1);
% Compute the indices for use in meshgrid.
idx = find(u > M/2);
u(idx) = u(idx) - M;
idy = find(v > N/2);
v(idy) = v(idy) - N;
% Compute the meshgrid arrays.
[U, V] = meshgrid (v, u);
```

* using function dftuv.

```
>> [U, V] = dftuv(8,5);
>> D = U.^2 + V.^2.
```

Note: the distance is 0 at the top, left, and the larger distances are in the center of the frequency rectangle.

% We can use function fftshift to obtain the distances with respect to the center of the frequency rectangle:

```
>> fftshift(D).
```

example 1:

Apply a Gaussian Lowpass filter to the 500X 500- pixel image, f, use a value of Do equal to 5% of the padded image width.

```
PQ = paddedsize(size(f));
[U, V] = dftuv(PQ(1), PQ(2));
Do = 0.05 * PQ(2);
F = fft2(f, PQ(1), PQ(2));
H = exp(-(U.^2 + V.^2)/(2*(Do^2)));
g = dftfilt(f, H);
```

% we can view the filter as an image

```
>> figure, imshow(fftshift(H), [ ]);
```

% the spectrum can be displayed as an image

```
>> figure, imshow(log(1 + abs(fftshift(F))), [ ]);
>> figure, imshow(g, [ ]);
```

example 2:

The following function generates the transfer functions of all the Lowpass filters (Gaussian, Ideal and Butterworth).

```
function H = lpfilter(type, M, N, Do, n)
% LPFILTER(TYPE, M, N, Do, n) Creates the transfer
% function of a Lowpass filter, H, of the specified
% Type and size (M-by-N). To view the filter as
% an image or mesh plot, it should be centered
% using H = fftshift(H).
```

```
% Valid values for TYPE, Do and n are:
% 'Ideal'   Ideal lowpass filter with cutoff frequency
% Do. n need not be supplied. Do must be positive.
% 'btw'   Butterworth Lowpass filter of order n, and
% Cutoff Do. The default value for n is 1.0.
% Do must be positive.
% 'gaussian'   Gaussian Lowpass filter with cutoff
% (Standard deviation) Do. n need not be supplied.
% Do must be positive.
% Use function dftuv to set up the meshgrid arrays need
% for computing the required distances.
  [U, V] = dftuv(M,N);
% Compute the distances D(U,V).
  D = sqrt(U.^2 + V.^2);
% Begin filter computations.
    switch type
    case 'Ideal'
          H = double(D <= Do);
    case 'btw'
         if nargin == 4
            n = 1
         end
         H = 1./(1+(D./Do).^(2*n));
    Case 'gaussian'
         H = exp(-(D.^2)./(2*(Do^2)));
   otherwise
         error('Unknown filter type.')
   end
```

This function 'Lpfilter' is used as the basis for generating highpass filters.