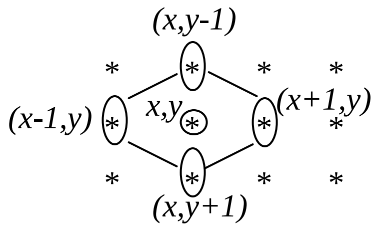
Image Smoothing

Type of image enhancement, smoothing operations are used primarily for diminishing spurious effects that may be present in a digital image as a result of poor sampling system or transmission channel. Smoothing techniques are used for blurring and for noise reduction.

Neighborhood Averaging *(spatial domain)*

Neighborhood averaging is a straight forward spatial domain technique for image smoothing. The idea behind this approach is accomplished by replacing the value of every pixel in an image by the average of the graylevels of its neighborhood.



Given an $N*N$ image, $f(x,y)$, the procedure is to generate a smoothed image $g(x,y)$, whose gray level at point (x,y) is obtain by averaging the graylevel values of the pixels of f contained in a predefined neighborhood of (x,y).
In other words , the smoothed image is obtained by using the relation:

$$g(x,y) = \frac{1}{M} \sum_{(n,m)\in S} f(n,m)$$

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

S : is the set of coordinates of points in the neighborhood (but not including) the point (x,y).

M : is the total number of points defined by the coordinates in S.

Average = [(x,y-1)+(x-1,y)+(x+1,y)+(x,y+1]/4

This approach results is an image with reduced sharp transitions in graylevels.

Example: *Applying neighborhood averaging method to smooth the following points:*

$$\begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & 1 & 1 & 0 \\ 1 & 3 & 2 & 4 \\ 4 & 0 & 3 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 6/4 & 7/4 & 11/4 & 13/4 \\ 5/4 & 8/4 & 6/4 & 10/4 \\ 10/4 & 4/4 & 11/4 & 7/4 \\ 9/4 & 10/4 & 6/4 & 9/4 \end{pmatrix} \Rightarrow \begin{pmatrix} 1.5 & 1.75 & 2.75 & 3.25 \\ 1.25 & 2 & 1.5 & 2.5 \\ 2.5 & 1 & 2.75 & 1.75 \\ 2.25 & 2.5 & 1.5 & 2.25 \end{pmatrix} \Rightarrow \begin{pmatrix} 2 & 2 & 3 & 3 \\ 1 & 2 & 2 & 3 \\ 3 & 1 & 3 & 2 \\ 2 & 3 & 2 & 2 \end{pmatrix}$$

Lowpass Filtering

Edges and other sharp transitions (such as noise) in the graylevels of an image contribute heavily to the high frequency content of its fourier transform. The blurring can be achieved in the frequency domain by attenuating a specified range of high frequency components in the transform of a given image. Basic model for frequency domain filtering of *f(x,y)* is given by the equation:

$$G(u,v) = H(u,v)F(u,v)$$

where *F(u,v)* is the transform of the image we wish to smooth. The problem is to select a function *H(u,v)* which yields *G(u,v)* by attenuating the high frequency components of *F(u,v)*. The inverse transform of *G(u,v)* will then yield the desired smoothed image *g(x,y)*. Since high frequency components are *"filtered out"* and information in the low frequency range *"passed"* without attenuation, this method is commonly referred to as lowpass filtering. The function *H(u,v)* is referred to filter transform function.

1. Ideal Filter

A two dimensional ideal lowpass filter *(ILPF)* is one whose transfer function satisfies the relation:

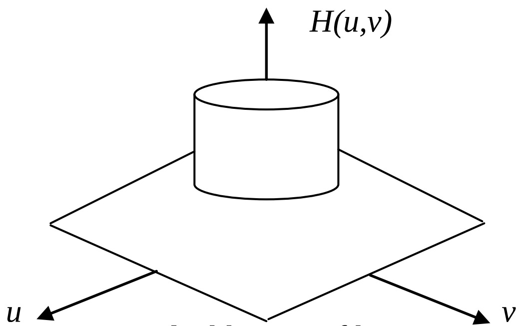
$$H(u,v) = \begin{cases} 1 & if \;\; D(u,v) \le D_0 \\ 0 & if \;\; D(u,v) > D_0 \end{cases}$$

where D_0 is a specified non negative quantity and *D(u,v)* is the distance from point *(u,v)* to the origin of the frequency plane that is:

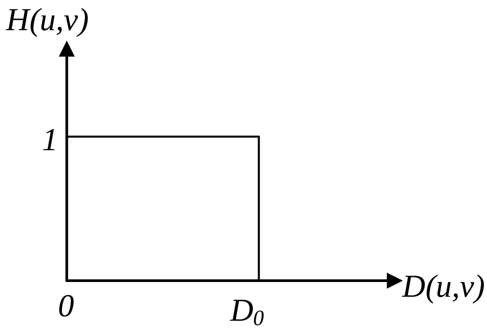
$$D(u,v) = \{u^2 + v^2\}^{1/2}$$

the name ideal filter arises from the fact that all frequencies inside a circle of radius D_0 are passed with no attenuation while all frequencies outside this circle are completely attenuated.

For an ideal lowpass filter cross section the point of transition between *H(u,v)=1 and H(u,v)=0* is often called the cut off frequency .



Ideal lowpass filter transfer function



Filter cross section

<u>Example:</u> Design a digital lowpass filter:

Type: Ideal $D_0 = 5$ *size =7*7*

D	H(D)
0	1
1	1
2	1
3	1
4	1
5 ← D_0	1
6	0

	0	1	2	3	4	5	6
0	1	1	1	1	1	1	0
1	1	1	1	1	1	0	0
2	1	1	1	1	1	0	0
3	1	1	1	1	1	0	0
4	1	1	1	1	0	0	0
5	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0

2. *Butterworth Filter*

The transfer function of the Butterworth lowpass filter *(BLPF)* of order n and with cut off frequency locus at a distance D_0 from the origin is defined by the relation:

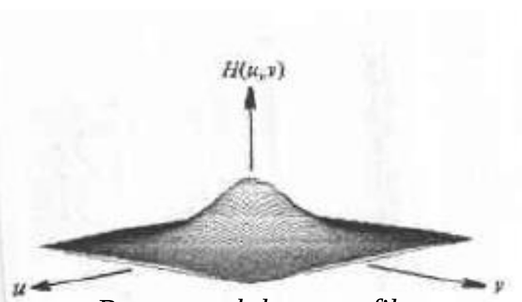
$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$

Where:

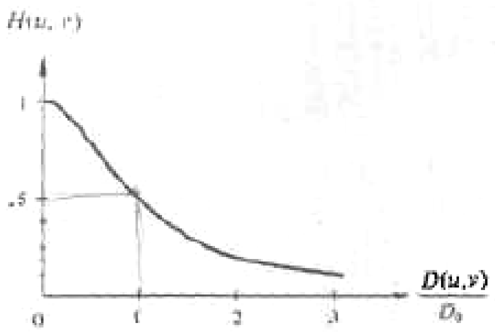
$$D(u,v) = \{u^2 + v^2\}^{1/2}$$

Unlike the *ILPF* the *BLPF* transfer function does not have a sharp discontinuity that establishes a clear cutoff between passed and filtered frequencies. For filters with smooth transfer functions it is customary to define a *"cut off"* frequency locus at points for which *H(u,v)* is down to a certain fraction of its maximum value. In the case of above eq. we see that $H(u,v)=0.5$ when $D(u,v)=D_0$. another value commonly used is $1/\sqrt{2}$ of the maximum value of *H(u,v)*. The following simple modification yields the desired value when $D(u,v)=D_0$.

$$H(u,v) = \frac{1}{1 + 0.414[D(u,v)/D_0]^{2n}}$$



Butterworth lowpass filter transfer function



Radial cross sections for n=1

<u>Example:</u> Design a butterworth filter (normal)

D_0=5 size=9*9 n=1

D	H(D)
0	1
1	0.962
2	0.862
3	0.735
4	0.609
5 (← D_0)	0.5
6	0.409
7	0.338
8	0.281

	0	1	2	3	4	5	6	7	8
0	1	0.962	0.862	0.735	0.609	0.5	0.409	0.338	0.281
1	0.962	0.927	0.833	0.715	0.595	0.490	0.403	0.333	0.278
2	0.862	0.833	0.758	0.658	0.556	0.463	0.385	0.321	0.268
3	0.735	0.715	0.658	0.581	0.5	0.424	0.357	0.321	0.255
4	0.609	0.595	0.556	0.5	0.439	0.379	0.325	0.278	0.238
5	0.5	0.490	0.463	0.424	0.379	0.333	0.291	0.253	0.219
6	0.409	0.403	0.385	0.357	0.325	0.291	0.258	0.227	0.200
7	0.338	0.333	0.321	0.301	0.278	0.253	0.227	0.203	0.181
8	0.281	0.278	0.268	0.255	0.238	0.219	0.200	0.181	0.163

<u>Example:</u> Design a butterworth filter (modified)

D_0=3 size=5*5 n=2

D	H(D)
0	1
1	0.9949
2	0.9244
3	0.7072
4	0.4332

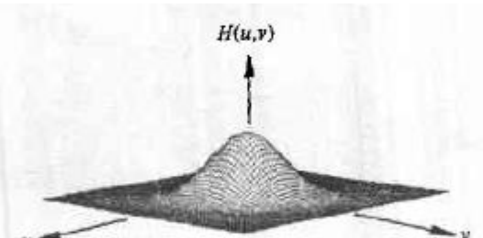
	0	1	2	3	4
0	1	0.9949	0.9244	0.7072	0.4332
1	0.9949	0.9800	0.8867	0.6618	0.4037
2	0.9244	0.8867	0.7535	0.5365	0.3285
3	0.7072	0.6618	0.5365	0.3765	0.2384
4	0.4332	0.4037	0.3285	0.2384	0.1604

3. Exponential Filter

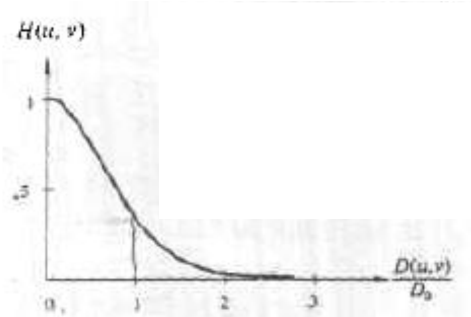
The exponential lowpass filter *(ELPF)* is another smooth filter commonly used in image processing. The *ELPF* with cut off frequency locus at a distance D_0 from the origin has a transfer function given by the relation :

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

where: $D(u,v) = \{u^2 + v^2\}^{1/2}$, and n controls the rate of decay of the exponential function. The plot and cross section of the *ELPF* are shown in the following figure.



An exponential LPF



Radial cross sections for n=2

When $D(u,v)=D_0$ we have from above equation that $H(u,v)=1/e$, a simple modification given by:

$$H(u,v) = e^{-0.347[D(u,v)/D_0]^n}$$

forces $H(u,v)$ to be equal to $1/\sqrt{2}$ of its maximum value at frequencies in the cut off locus.

Example: Design exponential lowpass filter (normal):

*$D_0=3$ size = 5*5 n=1*

D	*H(D)*
0	*1*
1	*0.7165*
2	*0.5134*
3	*0.3679*
4	*0.2636*

	0	1	2	3	4
0	1	0.7165	0.5134	0.3679	0.2636
1	0.7165	0.6241	0.4746	0.3485	0.2530
2	0.5134	0.4746	0.3895	0.3006	0.2252
3	0.3679	0.3485	0.3006	0.2431	0.1889
4	0.2636	0.2530	0.2252	0.1889	0.1517

Example: Design exponential lowpass filter (modified):

*$D_0=3$ size = 5*5 n=2*

D	*H(D)*
0	*1*
1	*0.9622*
2	*0.8571*
3	*0.7068*
4	*0.5396*

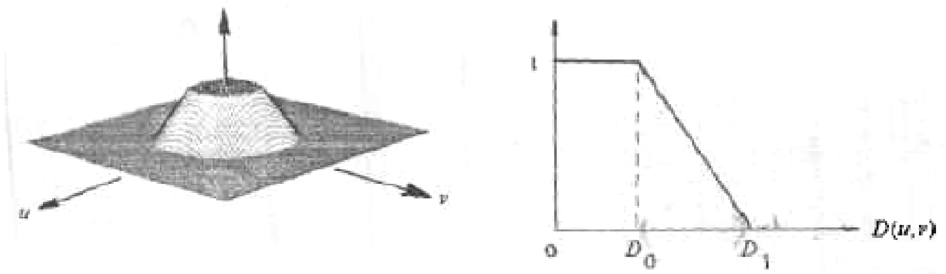
	0	1	2	3	4
0	1	0.9622	0.8571	0.7068	0.5396
1	0.9622	0.9258	0.8247	0.6801	0.5192
2	0.8571	0.8247	0.7346	0.6058	0.4625
3	0.7068	0.6801	0.6058	0.4996	0.3814
4	0.5396	0.5192	0.4625	0.3814	0.2912

4. Trapezoidal Filter

A trapezoidal lowpass filter *(TLPF)* is a compromise between the *ILPF* and a completely smooth filter. *TLPFs* can be defined by the relation:

$$H(u,v) = \begin{cases} 1 & if \;\; D(u,v) < D_0 \\ \dfrac{1}{[D_0 - D_1]}[D(u,v) - D_1] & if \;\; D_0 \le D(u,v) \le D_1 \\ 0 & if \;\; D(u,v) > D_1 \end{cases}$$

where $D(u,v) = \{u^2 + v^2\}^{1/2}$, D_0 and D_1 are specified , and it is assumed that $D_0 < D_1$. a plot and cross section of a typical *TLPF* transfer function are shown below.



Plot and cross section of a typical TLPF

For convenience in implementation, we define the cut off locus to be at the first breakpoint *(D_0)* of the transfer function. The second variable D_1 is arbitrary as long as it is greater than D_0.

Example: Design a trapezoidal lowpass filter:

*$D_0=5$ $D_1=7$ size=9*9*

	D	*H(D)*
	0	*1*
	1	*1*
	2	*1*
	3	*1*
	4	*1*
D_0 →	*5*	*1*
	6	*0.5*
D_1 →	*7*	*0*
	8	*0*

	0	1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	0.5	0	0
1	1	1	1	1	1	0.9505	0.4586	0	0
2	1	1	1	1	1	0.8074	0.3377	0	0
3	1	1	1	1	1	0.5845	0.1459	0	0
4	1	1	1	1	0.6716	0.2984	0	0	0
5	1	0.9505	0.8074	0.5845	0.2984	0	0	0	0
6	0.5	0.4586	0.3377	0.1459	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0