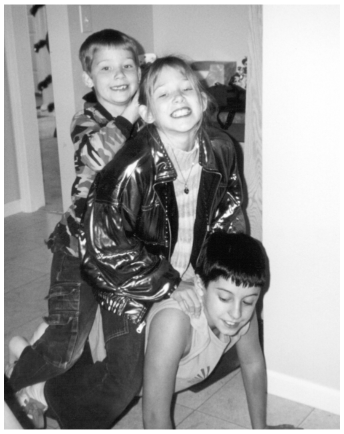
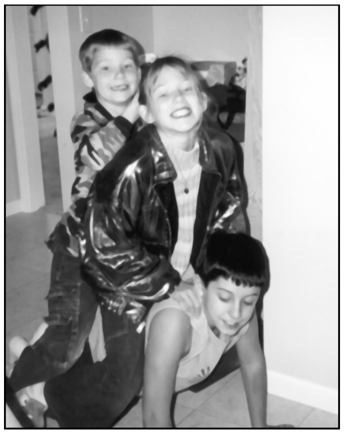
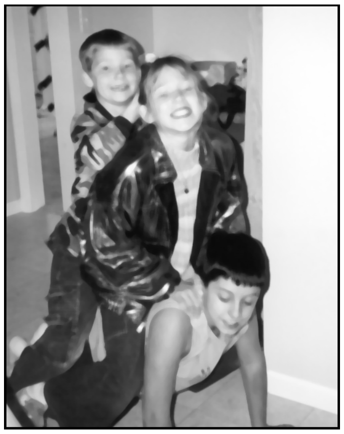
Smoothing using Median Filtering

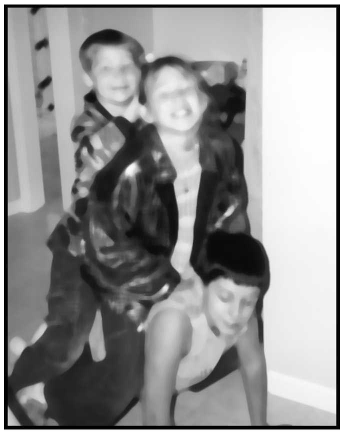
- Median filters create a smoothing effect, and large mask sizes create a painted (and blurred) look.


a) Original image


c) 5x5 median filter

Caiani 13/04/2012


d) 7x7 median filter


f) 11x11 median filter

Note: As the filter size increases above a 5x5, the resulting image acquires a painted effect.

Caiani 13/04/2012

With a large mask the median filter takes a long time to process. Fast algorithms for median filtering operate by efficiently maintaining the sorting of the data as we move across the image. An alternative to using a fast algorithm is the ***pseudomedian filter***, which approximates the operation of a median filter, but is simpler and more computationally efficient:

- The pseudomedian is defined is as follows:

$$PMED(S_L) = (1/2)MAXIMIN(S_L) + (1/2)MINIMAX(S_L)$$

where S_L *denotes a sequence of elements* $s_1, s_2, \ldots, s_L$

where for $M = \frac{(L+1)}{2}$

$$MAXIMIN(S_L) = \mathrm{MAX}\left[[\mathrm{MIN}(s_1,\ldots,s_M)],[\mathrm{MIN}(s_2,\ldots,s_{M+1})],\ldots,[\mathrm{MIN}(s_{L-M+1},\ldots,s_L)]\right]$$

$$MINIMAX(S_L) = \mathrm{MIN}\left[[\mathrm{MAX}(s_1,\ldots,s_M)],[\mathrm{MAX}(s_2,\ldots,s_{M+1})],\ldots,[\mathrm{MAX}(s_{L-M+1},\ldots,s_L)]\right]$$

- For example, the pseudomedian of length five is defined as:

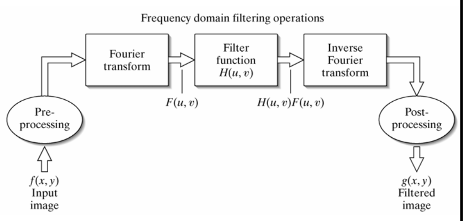
$$PMED(a,b,c,d,e) = (1/2)\,\mathrm{MAX}[\mathrm{MIN}(a,b,c),\mathrm{MIN}(b,c,d),\mathrm{MIN}(c,d,e)]$$
$$+(1/2)\,\mathrm{MIN}[\mathrm{MAX}(a,b,c),\mathrm{MAX}(b,c,d),\mathrm{MAX}(c,d,e)]$$

- The MIN followed by MAX in the first part always results in the actual median or a value smaller, while the MAX followed by the MIN results in the actual median or a value larger: their average tends to cancel out the biases, thus creating a valid approximation to the median filtering process.

Caiani 13/04/2012

Again on filtering in the frequency space

Remember: as H(n,m) and F(n,m) are complex, G(n,m) will be complex. Filtering with a complex function modifies the phase of the original image.To preserve the phase, <u>H(n,m) must be real</u>, without imaginary components. Such type of filters are called “zero-phase” and they are spatially symmetrical.



As F and H are periodic functions, convolution in the discrete frequency domain will be periodic (**circular convolution**). Convolving periodic functions can cause interference (wraparound error) between adjacent periods if the periods are close with respect to the duration of the nonzero parts of the functions.
To guarantee that spatial and circular convolution give the same result is to use appropriate zero-padding: given f(x,y) and h(x,y) of size AxB and CxD, they are both padded to PxQ where P>= A+C-1 and Q>=B+D-1

Apply the filtering in the frequency space, instead than using convolution, verifying the correspondence of the results or not, according to padding.

Caiani 13/04/2012

Enhancement filters

- Implemented with convolution masks having alternating positive and negative coefficients
- Enhance the image by sharpening

Image Sharpening

- Image sharpening deals with enhancing detail information in an image, typically edges and, in general, correspond to image features that are small spatially
- Most of the techniques include some form of highpass filtering
- The information is visually important, because it delineates object and feature boundaries, and is important for textures in objects

- Many image sharpening algorithms consist of three general steps:
  - Extracting high frequency information (High pass filtering)
  - Combining the high frequency image with the original image to emphasize image detail
  - Maximizing image contrast via histogram manipulation

Caiani 13/04/2012

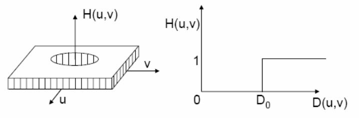
Image Sharpening - High Pass Filter

H(u,v) - Ideal Filter

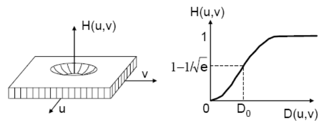
$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_0 \\ 1 & D(u,v) > D_0 \end{cases}$$

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

D_0 = cut off frequency



H(u,v) - Gaussian High Pass Filter

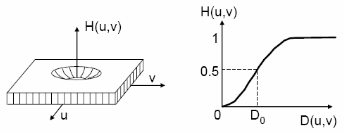


$H(u,v) = 1 - e^{-D^2(u,v)/(2D_0^2)}$

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

- Highpass filtering for image enhancement typically requires some form of post-processing, such as histogram equalization, to create an acceptable image
- Highpass filtering, in the form of edge detection, is often used in computer vision applications to delineate object outlines.

H(u,v) - Butterworth High Pass Filter

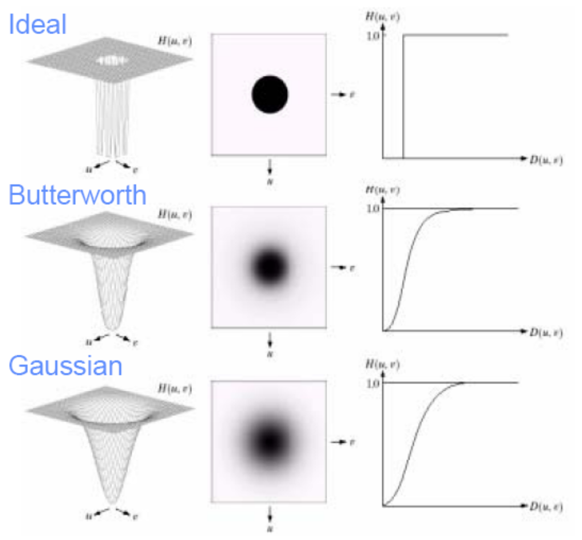


$$H(u,v) = \frac{1}{1 + (D_0/D(u,v))^{2n}}$$

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

Caiani 13/04/2012

High Pass Filters - Comparison



Caiani 13/04/2012

High Frequency Emphasis

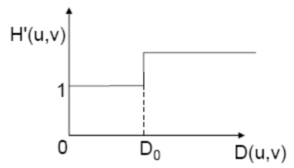
- A high frequency emphasis (HFE) filter is essentially a highpass filter with an offset in the filter function to boost high frequencies and retain some of the low frequency information

Emphasize High Frequency.
Maintain Low frequencies and Mean.

$$\boxed{H'(u,v) = K_0 + H(u,v)}$$

(Typically K_0 =1)

- Care must be taken to avoid overflow in the resulting image, as it may create noise in the image in the form of black and white points



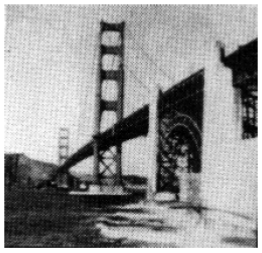
- This problem can be avoided by careful use of proper data types, correct data conversion when necessary, and appropriate remapping of the data before display.

Caiani 13/04/2012

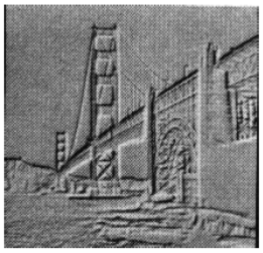
High Frequency Emphasis

Original

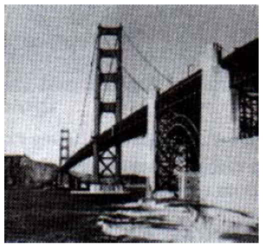
High Pass Filtered



+



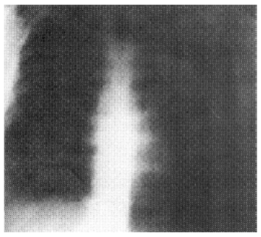
=



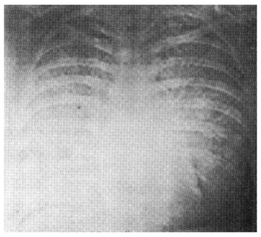
Caiani 13/04/2012

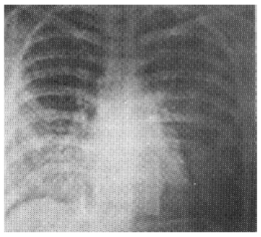
High Pass Filtering - Examples

Original

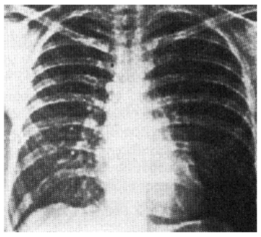


High pass Butterworth Filter





High Frequency Emphasis

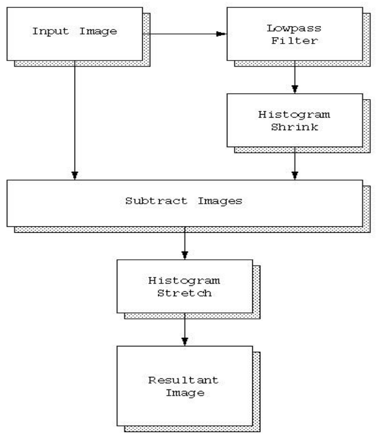


High Frequency Emphasis + Histogram Equalization

Caiani 13/04/2012

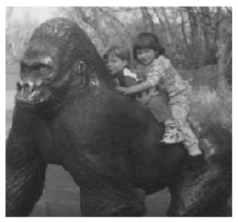
Unsharp Masking

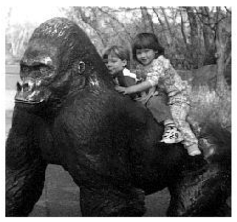
- Sharpens the image by subtracting a blurred (lowpass) version of the original image
- The process is similar to *adding* a detail enhanced (highpass) version of the image to the original.
- This process works because subtracting a slowly changing edge (the lowpass filtered image) from faster changing edges (in the original), has the visual effect of causing overshoot and undershoot at the edges, which has the effect of emphasizing the edges
- By the scaling the lowpassed image with a histogram shrink we can control the amount of edge emphasis desired

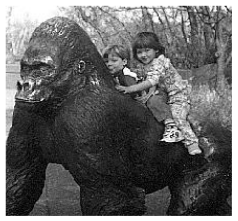


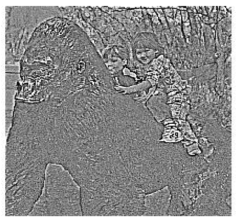
Caiani 13/04/2012

Unsharp Masking


a) Original image


b) Unsharp masking with lower limit = 0, upper = 100, with 2% low and high clipping


c) Unsharp masking with lower limit = 0, upper = 150, with 2% low and high clipping


d) Unsharp masking with lower limit = 0, upper = 200, with 2% low and high clipping

Caiani 13/04/2012

In fact:
it is an high pass filter, that sharpens a blurried image by enhancing the high frequencies in the image. Each image can be viewed as:

$$f(x, y) = f_{lowpass}(x, y) + f_{highpass}(x, y)$$

As the mean filter preserves low frequency components, we can write it as:

$$f(x, y) = f_{mean}(x, y) + f_{highpass}(x, y)$$

Solving for the high pass filter, we obtain:

$$f_{highpass}(x, y) = f(x, y) - f_{mean}(x, y)$$

If it's desired to enhance the high frequency components without attenuating the low frequencies, it's possible to proceed by adding a proportional amount of the HF component to the original image by:

$$f_h(x, y) = (1 + G) \cdot f(x, y) - G \cdot f_{mean}(x, y)$$

How to implement it? The linear property of spatial convolution can be used. Given two masks $h_1(x,y)$ and $h_2(x,y)$, the difference of the two filter operations can be written as:

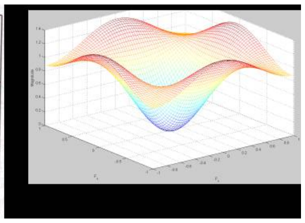
$$g(x, y) = g_1(x, y) - g_2(x, y) = \sum_{i,j \in H}\sum f(x-i, y-j) \cdot h_1(i, j) - \sum_{i,j \in H}\sum f(x-i, y-j) \cdot h_2(i, j)$$

Caiani 13/04/2012

$$g(x, y) = \sum_{i,j \in H}\sum f(x-i, y-j) \cdot [\, h_1(i, j) - h_2(i, j) \,]$$

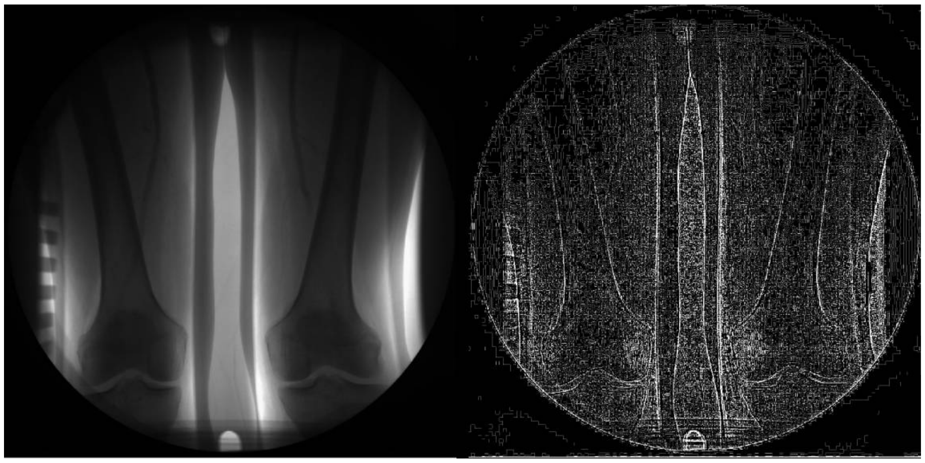
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \begin{bmatrix} \frac{-1}{9} & \frac{-1}{9} & \frac{-1}{9} \\ \frac{-1}{9} & \frac{8}{9} & \frac{-1}{9} \\ \frac{-1}{9} & \frac{-1}{9} & \frac{-1}{9} \end{bmatrix}$$

$$\begin{bmatrix} \frac{-G}{9} & \frac{-G}{9} & \frac{-G}{9} \\ \frac{-G}{9} & \frac{9+8G}{9} & \frac{-G}{9} \\ \frac{-G}{9} & \frac{-G}{9} & \frac{-G}{9} \end{bmatrix}$$



In Matlab, such filter of 3x3 size can be obtained using **fspecial**:
h=**fspecial**('unsharp',ALPHA)   [ALPHA between 0 and 1, default=0.2]

Caiani 13/04/2012



*Select an image. Filter it with the high frequency enphasis filter, varying G, using **imfilter**. Also, create a unsharp mask by **fspecial** and, for different alpha values, filter the image and see the effect.*

Caiani 13/04/2012

High Boost spatial filter

- The high boost spatial filter mask is of the following form:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & x & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

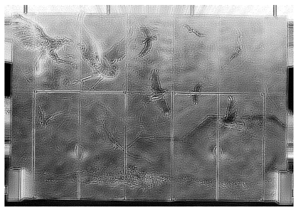
- The mask is convolved with the image, and the value of x determines the amount of low frequency information retained in the resulting image
- A value of 8 will result in a highpass filter (the output image will contain only the edges), while larger values will retain more of the original image
- If values of <u>less than 8</u> are used for x, the resulting image will appear as a <u>negative</u> of the original
- A spatial domain high boost filter provides results similar to the frequency domain HFE filter

Caiani 13/04/2012

High Boost spatial filter

Original image

Results of performing a high boost spatial filter with a 3x3 mask and x=6


Histogram stretched version; Note : Image is a negative of the original

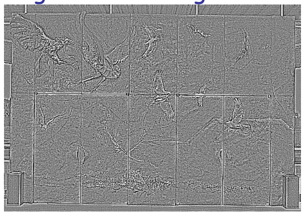
Caiani 13/04/2012

High Boost spatial filter

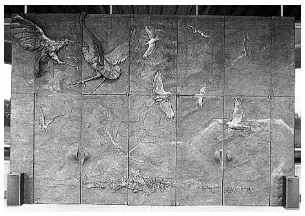
Histogram stretched version; Note: Image contains edge information only

Results of performing a high boost

spatial filter with a 3x3 mask and x = 8



Results of performing a highboost spatial filter with a 3x3 mask and x=12



Histogram stretched version

Caiani 13/04/2012

High Boost spatial filter

- The high boost mask can be extended with 1's and a corresponding increase in the value of x
- Larger masks will emphasize the edges more (make them wider), and help to mitigate the effects of any noise in the original image
- An example a 5x5 version of this mask is:

$$\begin{matrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & x & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{matrix}$$

- If we create an N x N mask, the value for x for a highpass filter is NxN-1, in this case 24 (5x5-1)

On the image used in the previous exercise, apply high boost spatial filtering with different values of x, and different size.

Caiani 13/04/2012

FIR Filter design (non recursive)

ADVANTAGES

- Increased flexibility
- |H(z)| more constant in pass band (< distorsions).
- **Linear phase** $\Longleftrightarrow$ h(n) symmetric
- Always steady (poles within the unit radius circle)
- Use of fast alghoritms

DISADVANTAGES

- Better performance only with a large number of samples
- Increased computation time
- **Poles in the origin only**
- Imprecisions in the definition of ω_c

Caiani 13/04/2012

Time domain filtering

Moving average filtering (MA)
$$y(n) = \sum_{k=0}^{N} b_k \cdot x(n-k)$$

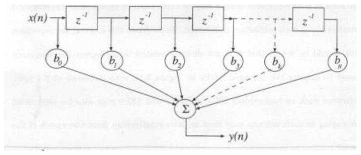
$$= b_0 \cdot x(n) + b_1 \cdot x(n-1) + b_2 \cdot x(n-2) + ... + b_N \cdot x(n-N)$$

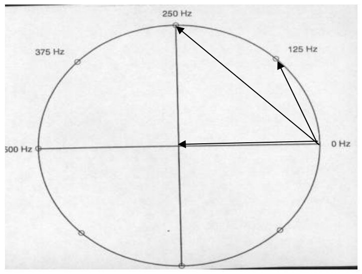
Z-Trasform:

$$Y(z) = (b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + ... + b_N \cdot z^{-N}) \cdot X(z)$$

$$H(z) = \frac{Y(Z)}{X(Z)} = \sum_{k=0}^{N} b_k \cdot z^{-k}$$

- FIR filter with a finite number of terms in the impulse response;
- Steady with poles in the origin only;
- Linear phase (step-wise)



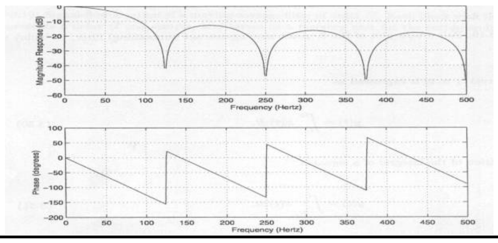


$$y(n) = \frac{1}{8} \cdot \sum_{k=0}^{7} x(n-k)$$

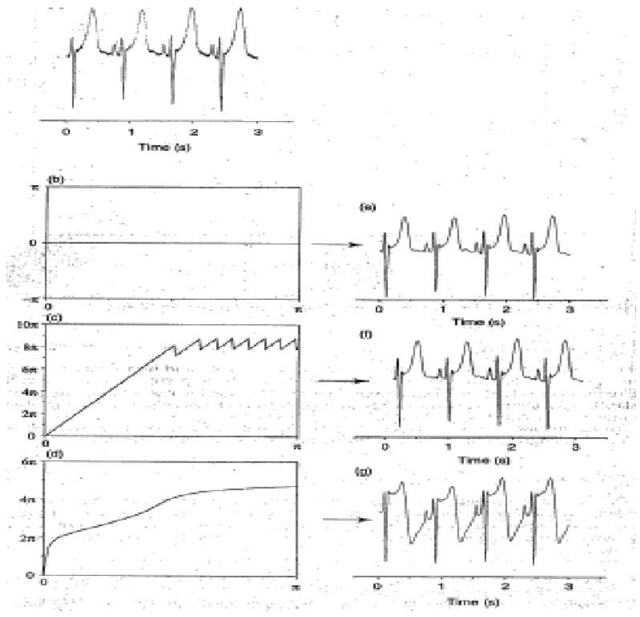
$$H(z) = \frac{1}{8} \cdot \sum_{k=0}^{7} z^{-k}$$

Frequency responce:

$$H(w) = \frac{1}{8} \cdot \left[1 + e^{-j4w} \cdot (1 + 2\cos w + 2\cos 2w + 2\cos 3w)\right]$$



Linear and non linear phase filters



2-D Filter design

Finite impulse response filter (FIR):
• linear
• can be represented by matrix of coefficients
• natural extension of 1-D FIR filters
• easy to implement
• linear phase

Frequency transformation method.
It transforms a 1-D FIR filter in a 2-D one, using a transformation matrix, and preserving most of the characteristics of the 1-D filter (transition bandwidth and ripple characteristics).
In Matlab, it is performed by the command **ftrans2**, which by default produces a nearly circular simmetry. This method produces good results, because it is easier to design a 1-D filter with particular characteristics than a 2-D filter.

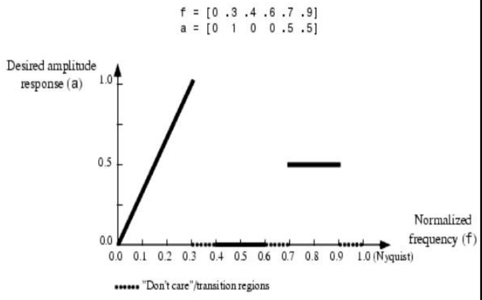
Then, to implement a 2D FIR in Matlab with this method, we have to define first the characteristics of the 1-D filter, to be used to generate the 2D one by circular simmetry.

The 1D FIR implementation can be done using **firpm** (or **remez**):

> b = **firpm**(n,f,a);
The row vector b will contain the n+1 FIR filter coefficients of order n, which frequency and amplitude characteristicsare described in vectors f and a.
The frequency values in f must be included between 0 and 1, where 1 is the Nyquist frequency (Fc/2) and n must be even.

The desired response is given by the line joining the points (F(k),A(k)) and (F(k+1),A(k+1)) for k odd; the intervals between F(k+1) and F(k+2) are considered "transition bands“.

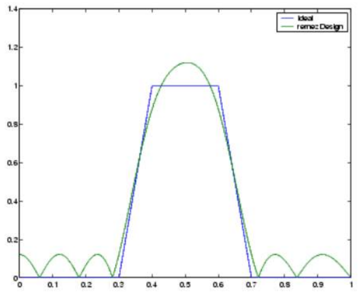
By Parks-McClellan algorithm, the optimal filter, in respect to the desired frequency responce, is generated.
Such filters have a equiripple responce.



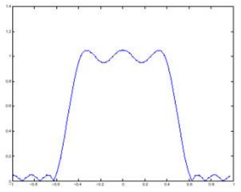
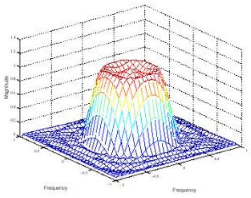
By **freqz**(b), gain and phase of the 1-D filter can be visualized.

Caiani 13/04/2012

f = [0 0.3 0.4 0.6 0.7 1];
a = [0 0 1 1 0 0];
b = **firpm**(17,f,a);
[h,w] = **freqz**(b,1,512);
plot(f,a,w/pi,abs(h))
legend('Ideal','firpm Design')



```
b = remez(10,[0 0.4 0.6 1],[1 1 0 0]);
h = ftrans2(b);
[H,w] = freqz(b,1,64,'whole');
colormap(jet(64))
plot(w/pi-1,fftshift(abs(H)))
figure, freqz2(h,[32 32])
```

Try these examples.
Then, design a high-pass filter with cut-off frequency at .7, and transition band from .6 to .7 (in respect of Nyquist).

Caiani 13/04/2012

Frequency sampling method.
It creates a filter based on a desired 2-D frequency response. Given a matrix of points which defines the shape of the frequency response, this method creates a filter whose frequency response passes through those points.

To this aim, the commands **meshgrid** and **freqspace** are very useful:
[C,R]=**meshgrid**(c,r)
It transforms the coordinate vectors into two arrays C and R that can be used to compute a function of two variables.

```
[X,Y]=meshgrid(1:3,10:14)
Z=X+Y
```

```
function f=twodsin2(A, u0, v0, M, N)
r=0:M-1; %row
c=0:N-1; % columns
[C,R]=meshgrid(c,r);
f=A*sin(u0*R+v0*C);
```

Try this example.

Between the given points, no constraint is defined, obtaining ripple as a result when a sharp transition is present. To minimize ripples, a larger filter is needed.

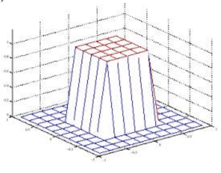
In Matlab, this method is performed by the command **fsamp2**, which returns a filter h (m x n) with a frequency response that passes through the points defined in the matrix Hd (m x n).
Hd is a matrix containing the desired frequency response sampled at equally spaced points between -1.0 and 1.0 along the x and y frequency axes, where 1.0 corresponds to half the sampling frequency, or PI radians.

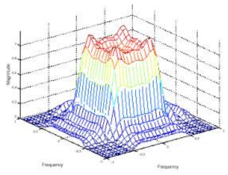
For accurate results, use frequency points returned by **freqspace** to create Hd.

```
>[f1,f2] = freqspace(n)              > [f1,f2] = freqspace([m n])
```

It returns the 2-D frequency vectors f1 and f2 for an n-by-n matrix, or an m-by-n matrix.

```
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2] = freqspace(11,'meshgrid');
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
h = fsamp2(Hd);
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
```



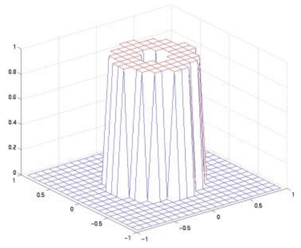


Use **fsamp2** to design an approximately symmetric two-dimensional bandpass filter with passband between 0.1 and 0.5 (normalized frequency).

1) Create a matrix Hd that contains the desired bandpass response. Use **freqspace** to create the frequency range vectors f1 and f2.

[f1,f2] = **freqspace**(21,'meshgrid');
Hd = **ones**(21);
r = **sqrt**(f1.^2 + f2.^2);
Hd((r<0.1)|(r>0.5)) = 0;
colormap(jet(64))
mesh(f1,f2,Hd)

Try these examples.



2) Design the filter that passes through this response.
h = **fsamp2**(Hd);
freqz2(h)

Caiani 13/04/2012