

Topic 6: Digital Filtering

Aim These two lectures cover Digital Filtering which is the main tool of image processing. We will consider both *linear* and *non-linear* filters and their operation on images.

Contents:

- Linear Digital Filtering
- Fourier Space Filters
- Real Space Filters
- Use of Linear Filters
- Real Space Non-Linear Filters
- Homomorphic Filtering
- Summary

Linear Digital Filtering

Main image processing operation, used for 90% of image processing operations.

Objective is to *Convolve* image $f(i, j)$, with filter function $h(i, j)$. In Real Space,

$$g(i, j) = h(i, j) \odot f(i, j)$$

or by the *convolution theorem* in Fourier Space

$$G(k, l) = H(k, l) F(k, l)$$

This operation can be performed in *Real OR Fourier* space. Mathematical operation is identical, but computational cost varies.

The operation of the filter is controlled by

$$\begin{aligned} h(i, j) &\rightarrow \text{In real space} \\ H(k, l) &\rightarrow \text{In Fourier space} \end{aligned}$$

Fourier Space Convolutions

Convolution can be written as,

$$g(i, j) = \mathcal{F}^{-1} \{H(k, l) F(k, l)\}$$

Processing requires **TWO** DFTs and a complex multiply, (a third DFT required if $H(k, l)$ is formed from $h(i, j)$).

Note: DFTs and \times must be performed in floating point format.

Computational time **independent** of filter type.

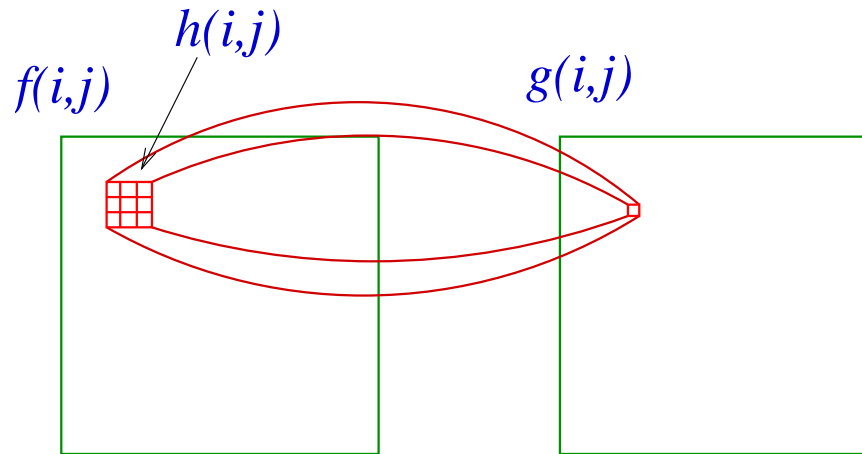
Time will depend on the computer system: for example 512×512 images, Pentium 4, computational time about 0.25 seconds.

Real Space Convolutions

For filter $h(i, j)$ of size $M \times M$,

$$g(i, j) = \sum_{m, n = -M/2}^{M/2} h(m, n) f(i - m, j - n)$$

shift & multiply scheme.



Computation $\propto M^2$

All calculations can be integer or byte.

For serial machine filters bigger than 9×9 are typically faster by Fourier space technique.

Fourier Space Filters

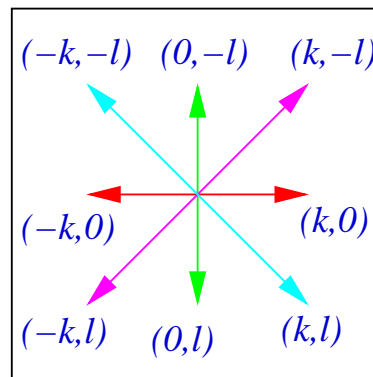
Filtering operation determined by $H(k,l)$, modified FT of image.

Most applications input & output images are **REAL**, so filter must preserve Symmetry conditions,

Real Part \Rightarrow Symmetric
Imaginary Part \Rightarrow Anti-symmetric

so $H(k,l)$ must obey these conditions.

In practice $H(k,l)$ is Real and Symmetric.



Low Pass Filter

Low pass filters allow *LOW* spatial frequencies to pass while attenuating or blocking *HIGH* spatial frequencies. Used extensively in the Reduction of Noise (see last lecture).

Ideal Low-Pass Filter

Block all frequencies greater than some limit,

$$H(k, l) = \begin{cases} 1 & k^2 + l^2 \leq w_0^2 \\ 0 & \text{else} \end{cases}$$

This Fourier transforms to give

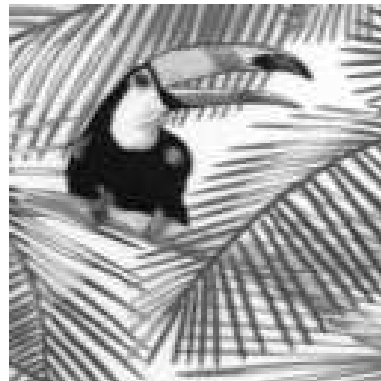
$$h(i, j) = \frac{J_1(r/w_0)}{r/w_0}$$

which results in “*ringing*” effects in the output image due to the lobes associated with $h(x, y)$.

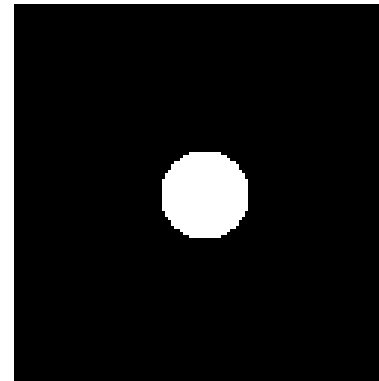
Not really useful.

Digital Example

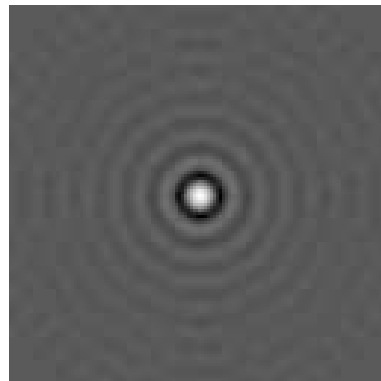
128×128 image with low-pass filter with $w_0 = 15$.



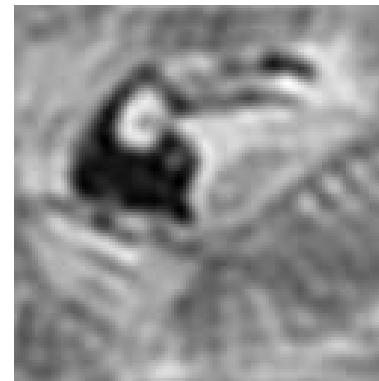
Input image



Low-pass filter



Real space filter



Filtered Image

Useful to reduce the effect of random noise, but too much “ringing” to be actually useful.

Gaussian Low Pass Filters

Filter profile in Fourier space is a two dimensional Gaussian of the type:

$$H(k, l) = \exp\left(-\frac{w}{w_0}\right)^2$$

where $w^2 = k^2 + l^2$ and w_0 is the $1/e$ point of the Gaussian.

$H(u, v)$ is *infinite* attenuates but does not remove high spatial frequencies.

In real space $h(i, j)$ is also Gaussian, begin, (see Fourier Booklet),

$$h(i, j) = \frac{\pi}{w_0^2} \exp(-\pi^2 w_0^2 r^2)$$

where $r^2 = i^2 + j^2$.

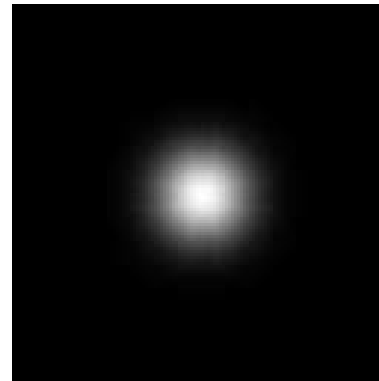
This gives a smooth filtered image without any ringing.

Digital Example

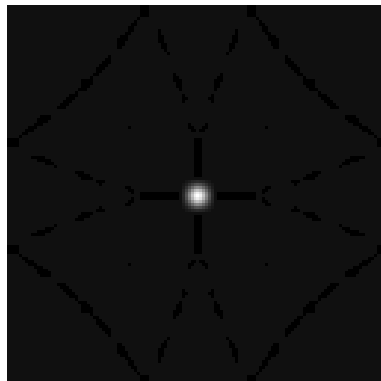
128×128 image with lowpass filter with $w_0 = 15$.



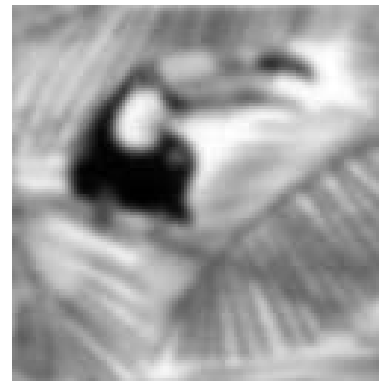
Input image



Lowpass filter



Real space filter



Filtered Image

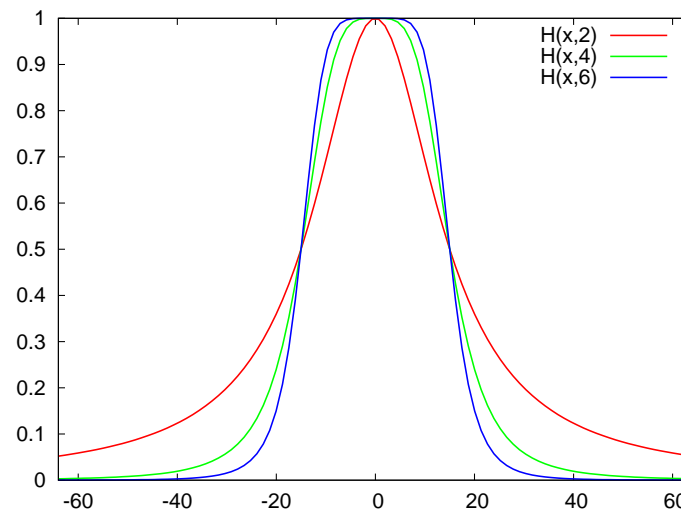
Very useful digital filter for noise reduction giving a very “smooth” filtered image.

Other Smooth Low-Pass Filters

Butterworth Filter:

$$H(k, l) = \frac{1}{1 + \left(\frac{w}{w_0}\right)^n}$$

where w_0 is *half point* and n is the *order*.



Plot with $w_0 = 15$ and $n = 2, 4, 6$.

Very similar properties to Gaussian, filter inherited from analogue signal processing.

Other Smooth Low-Pass Filters I

Trapezoidal filter:

$$\begin{aligned} H(k,l) &= 1 & w < w_0 \\ &= \frac{w-w_1}{w_0-w_1} & w_0 \leq w \leq w_1 \\ &= 0 & w > w_1 \end{aligned}$$

This will exhibit more ringing than Gaussian or Butterworth, but less than ideal filter.

Use the filtering program to “play” with these.

High Pass Filter

Pass *HIGH* spatial frequencies while attenuating or blocking *LOW* spatial frequencies.

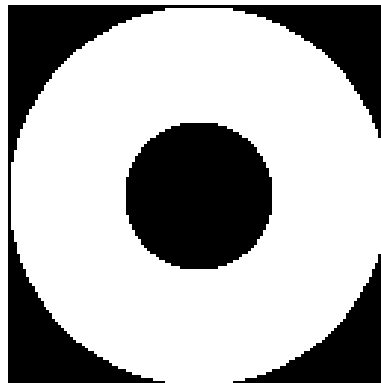
Used for the enhancement of high frequencies (and thus edges)

Ideal High-Pass Filter

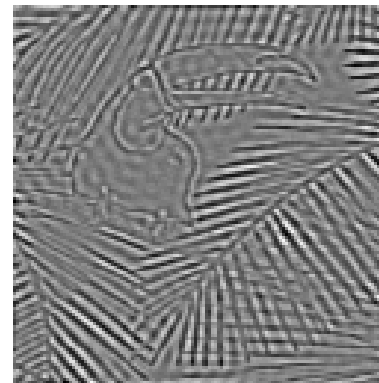
Block all frequencies less than some limit,

$$H(l, k) = \begin{cases} 0 & l^2 + k^2 \leq w_0^2 \\ 1 & \text{else} \end{cases}$$

This filter suffers from such severe ringing artifacts pass filters.



Filter



Output Image

Example with $w_0 = 25$.

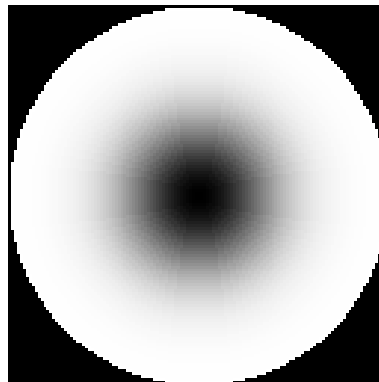
Gaussian High Pass Filter

Smooth reduction of *Low* spatial frequencies while the high spatial frequencies are pass unaltered.

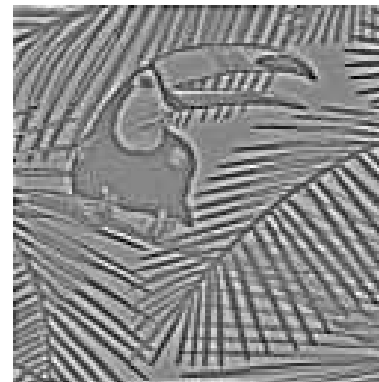
$$H(k, l) = 1 - \exp\left(-\frac{w}{w_0}\right)^2$$

again with $w^2 = k^2 + l^2$, which is a smooth filter in Fourier space.

This gives a smooth $h(i, j)$ in real space, so enhances edges without introduction of *ringing*.



Filter



Output Image

Example with $w_0 = 25$.

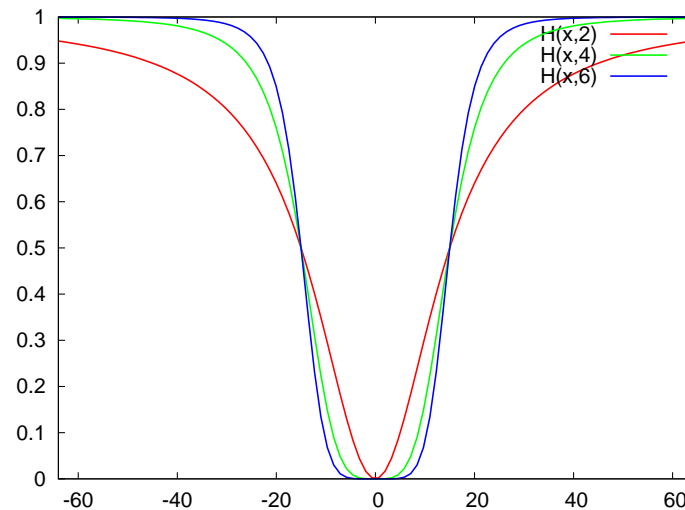
Other Smooth High Pass Filters

We can modify the smooth low pass filters to give High Pass, for example:

Highpass Butterworth:

$$H(k, l) = 1 - \frac{1}{1 + \left(\frac{w}{w_0}\right)^n} = \frac{1}{1 + \left(\frac{w_0}{w}\right)^n}$$

where w_0 is *half point* and n is the *order*.

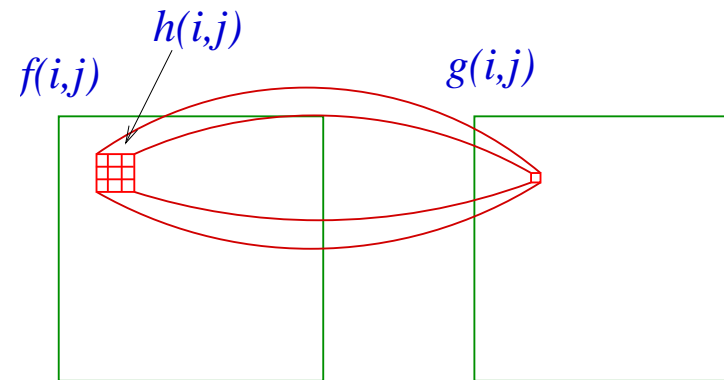


Plot with $w_0 = 25$ and $n = 2, 4, 6$. This give almost identical results to the highpass Gaussian.

Use the filtering program to “play” with these.

Real Space Filters

Filter is specified in real space by the mask $h(i, j)$ of finite size, typically 3×3 , 5×5 or sometimes 7×7 .



The filter operation is then specified by the mask elements $h(i, j)$.

- Mask elements are Real, usually integer.
- Able to use integer, or fixed point arithmetic.
- For masks bigger than $\approx 7 \times 7$, faster to use Fourier technique.

Real Space Averaging

Replace each pixel by the average of its neighbours, effect of *Low pass* filtering.

5 Pixel Average

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

gives a filter with an effective radius of 1 pixel.

9 Pixel Average

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

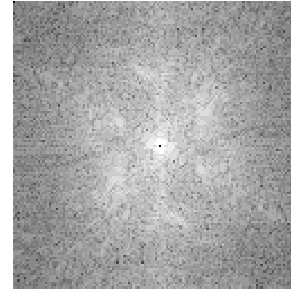
This is equivalent of multiplying with $H(k, l)$ in Fourier space, (see tutorial for detail).

Effect of reducing high spatial frequencies, but not removing them, (actually removes a range of spatial frequencies).

Digital Example



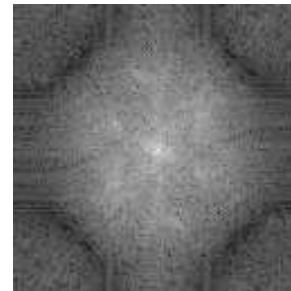
Input Image



Fourier Transform



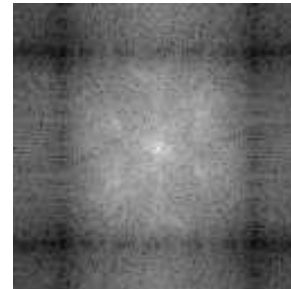
5 point ave



Fourier Transform



9 point ave



Fourier Transform

Real Space Differentiation

For a one-dimensional continuous function we have the definition of differentiation being:

$$\frac{df(x)}{dx} = \lim_{\delta \rightarrow 0} \frac{f(x + \delta) - f(x)}{\delta}$$

Discrete case of $\delta = 1$,

$$\frac{df(i)}{di} = f(i + 1) - f(i)$$

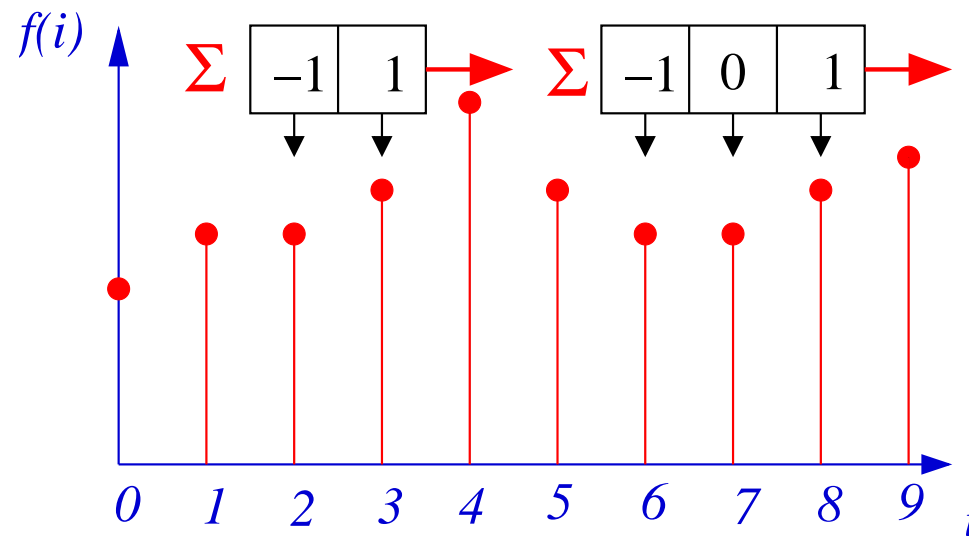
which if we consider convolution as **Shift-Fold-Multiply-Add** then differentiation can be written as:

$$\frac{df(i)}{di} = [-1 \quad 1] \odot f(i)$$

Real Space Differentiation I

Similarly with $\delta = 2$ we get

$$\frac{df(i)}{di} = [-1 \ 0 \ 1] \odot f(i)$$



Either of these convolutions can be used to approximate the first order differential of a sampled function.

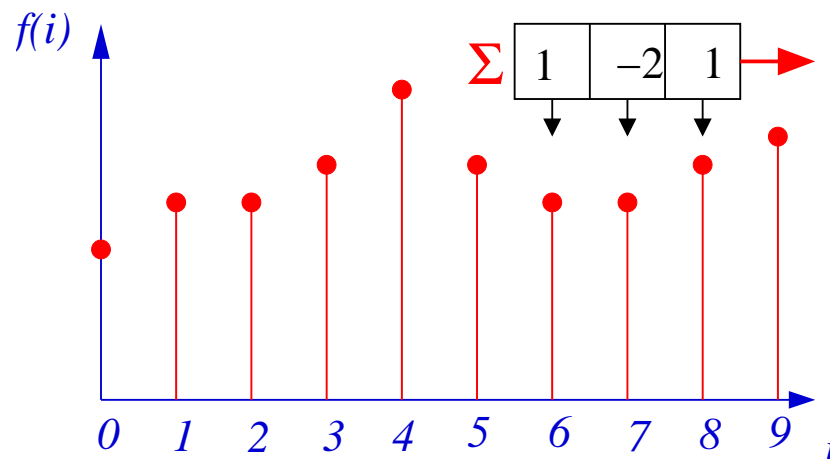
Second Order Differentials

Similarly the second order differential is given by,

$$\frac{d^2 f(i)}{di^2} = f(i+1) - 2f(i) + f(i-1)$$

which can be written as

$$\frac{d^2 f(i)}{di^2} = [1 \quad -2 \quad 1] \odot f(i)$$



Note also that

$$[1 \quad -2 \quad 1] = [-1 \quad 1] \odot [-1 \quad 1]$$

as would be expected since convolution is a linear operation.

Two-Dimensional Differentials

In Two Dimensions we have

$$\frac{\partial f(i, j)}{\partial i} = [-1 \ 0 \ 1] \odot f(i, j)$$

and similarly

$$\frac{\partial f(i, j)}{\partial j} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \odot f(i, j)$$

However to reduce the effect of noise, it is conventional to **average** the differential over 3 rows/columns respectively to give,

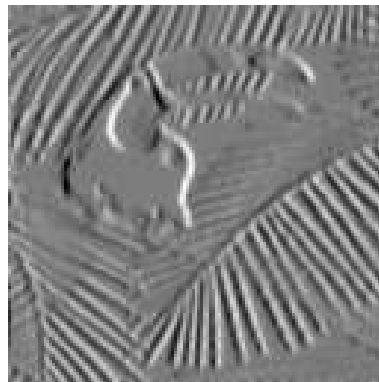
$$\frac{\partial f(i, j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \odot f(i, j)$$

and

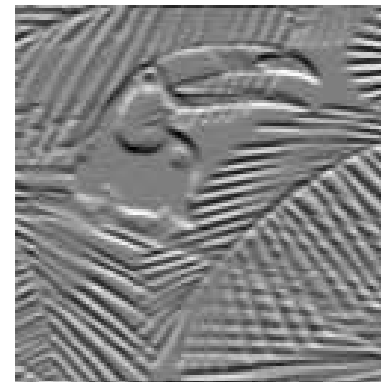
$$\frac{\partial f(i, j)}{\partial j} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \odot f(i, j)$$

which will enhance the vertical and horizontal edges respectively.

Two-Dimensional Differentials



x-differential



y-differential

X-differential enhances vertical edges

Y-differential enhances vertical edges.

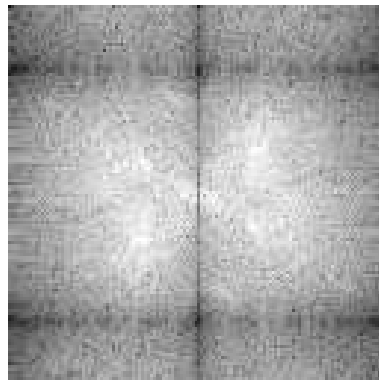
Fourier Space Differentials

From *Fourier Transform Booklet*, we have that

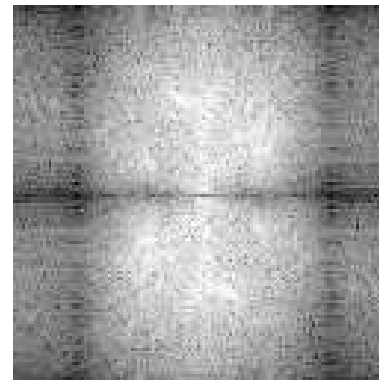
$$\mathcal{F} \left\{ \frac{\partial f(x,y)}{\partial x} \right\} = i2\pi u F(u,v) \quad \text{and} \quad \mathcal{F} \left\{ \frac{\partial f(x,y)}{\partial y} \right\} = i2\pi v F(u,v)$$

so that differential is equivalent to Fourier space multiplication by $i2\pi u/v$.

This has the effect of enhancing high frequency at the expense of low frequencies, so is essentially a **high-pass** filter.



x-differential (FT)



y-differential (FT)

Other zeros result from the three-row/col averaging.

Second Order Differentials

For the second order differentials,

$$\frac{\partial^2 f(i, j)}{\partial i^2} = [1 \quad -2 \quad 1] \odot f(i, j)$$

and

$$\frac{\partial^2 f(i, j)}{\partial j^2} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \odot f(i, j)$$

so that the Laplacian,

$$\nabla^2 f(i, j) = \frac{\partial^2 f(i, j)}{\partial i^2} + \frac{\partial^2 f(i, j)}{\partial j^2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \odot f(i, j)$$

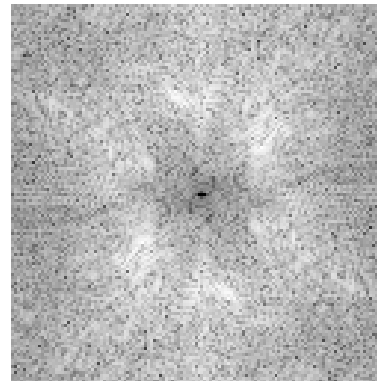
which forms the Laplacian of the 2-dimensional image.

Second Order Differentials I

While in Fourier space, we have from *Fourier Transform* booklet equation (17) we have that:

$$\mathcal{F} \{ \nabla^2 f(x, y) \} = -(2\pi w)^2 F(u, v)$$

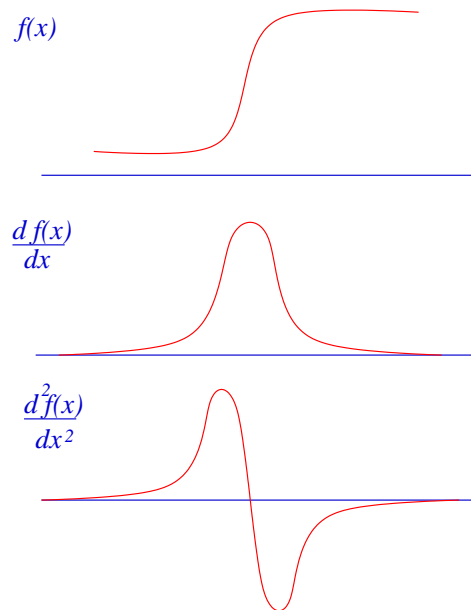
where $w^2 = u^2 + v^2$, giving



Enhances edges in all directions.

Edge Shapes

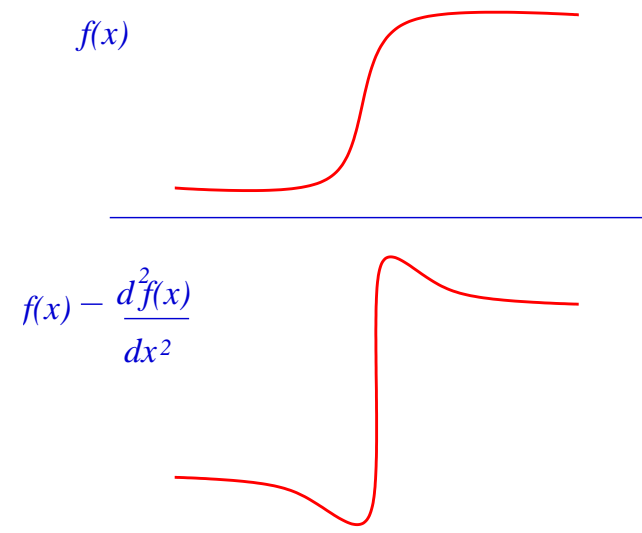
For a positive edge the shape of the differentials are:



Peak in the first differentail and **zero crossing** in the second. (see later)

Edge Shapes I

Form and *edge enhancement* by subtraction of the laplacian



which can be implemented by

$$f(i, j) - \nabla^2 f(i, j) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \odot f(i, j)$$

Edge Shapes II

Whis gives



Input image



Edge Enhanced

This is the most common **edge sharpening** filter.

Use of Linear Filters

- **Low Pass Filters:** are used to smooth images and reduce the effect of noise, in particular used to smooth image prior to edge detection.
- **High Pass Filter:** (also differentiations filters), have the effect of enhancing high frequencies and thus edges.

Filters can be combined to form *Bandpass* that attenuates both low & high spatial frequencies allowing middle frequencies to pass..

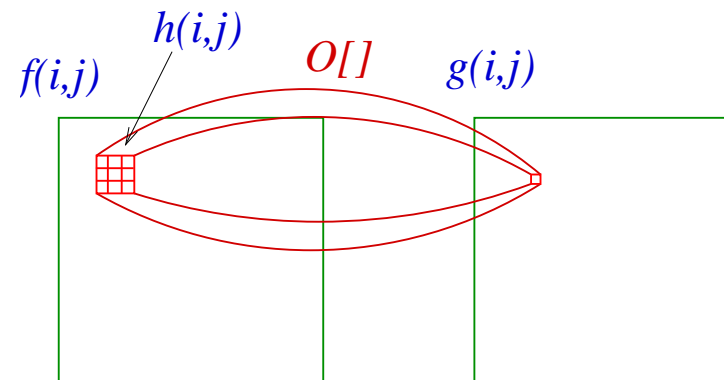
Due to linear nature, filters can be combined in Fourier space by \times or in real space by \odot operation.

Non-Linear Real Space Filters

The real space **shift & multiply** operation can be modified to

$$g(i, j) = O_{m,n \in w} [h(m, n) f(i - m, j - n)]$$

range of $h(m, n)$ defined by w . Operation now defined by mask $h(i, j)$ and operator $O[]$



In **most** non-linear filters we have

$$h(i, j) = 1 \quad i, j \in w$$

with the operation of the filter controlled by $O[]$ and the size of w only.

Shrink & Expand Filters

Taking

$$O[] = \text{Min}[]$$

the operator will act as a *Shrink* operation with bright objects reduced in size by approximately the *size* of the filter.

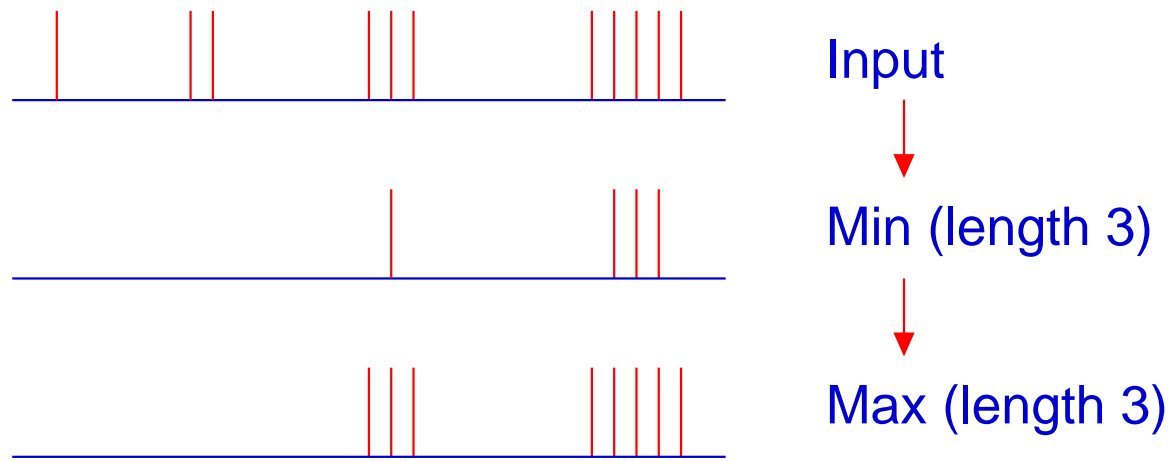
Taking

$$O[] = \text{Max}[]$$

the operator will act as an *Expand* filter, with bright objects increasing in size by approximately the *size* of the filter.

Shrink & Expand Filters

These operators typically used as a *pair* on binary image to remove small, isolated regions.



Note: These filters NOT commutative, ie.

$$E[S[f(i, j)]] \neq S[E[f(i, j)]]$$

Two-Dimensional Case

In two dimensions the *Min* and *Max* will selectively remove small **bright** objects.

Very useful in “cleaning-up” isolated points in a binary thresholded image



Input



Binary Threshold



Binary Shrink



Binary Expand

Can be used with Grey Scale image, but you tend to get funny results.

Threshold Average Filter

For “data-dropout” noise we have isolated “noise points” that differ from the neighbour pixels.

compares each pixel with average of neighbours and smoothes only if pixel deviates significantly

For each point form

$$A = \sum_{m,n=-M/2}^{M/2} h(m,n) f(i-m, j-n)$$

for 3 by 3 filter

$$h(i, j) = \begin{bmatrix} k & k & k \\ k & 0 & k \\ k & k & k \end{bmatrix}$$

where $k = 1/(M^2 - 1) = 0.125$, then output is

$$g(i, j) = \begin{cases} A & |A - f(i, j)| > T \\ f(i, j) & \text{else} \end{cases}$$

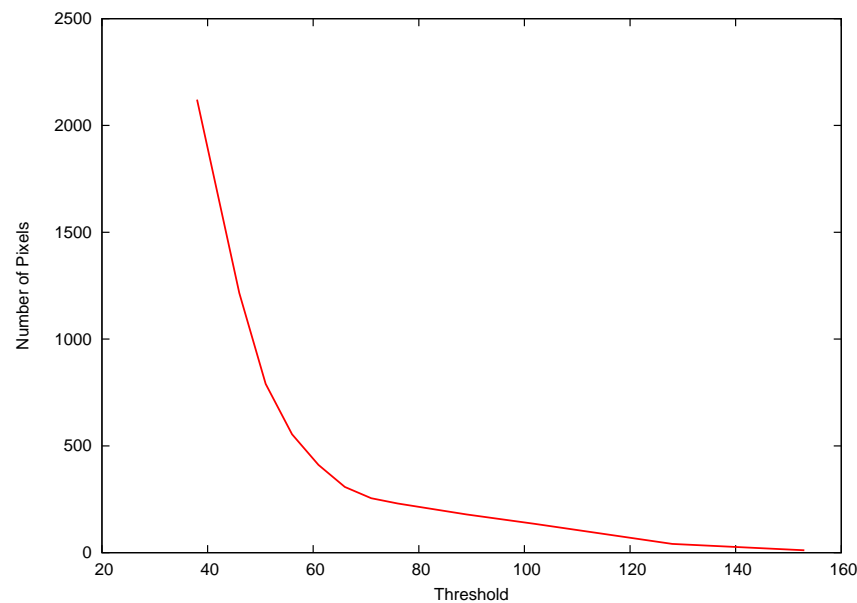
Selectively removes points that differ from neighbours.

Random Bit Error Example

8 bit image and we corrupt 1:50 bits. Large corruption when *most significant bit* is corrupted.

For 128×128 pixel image expect about 325 seriously corrupted pixels.

Apply *Average Threshold Filter* and count number of changed pixels.



Shows rapid rise at about 60.

Random Bit Error Example



1:50 Bit Error



Threshold of 66

Threshold will depend of the im,age, but $\approx 0.25 f_{\max}$ is typical.

Median Filter

The Median filter is formed by setting

$$O[] = \text{Median}[]$$

where the median is defined as the *middle* value. Eg. for the 5 values

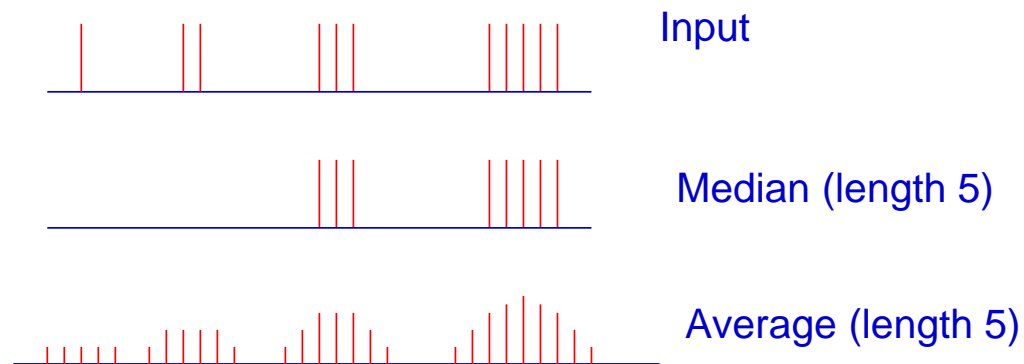
$$f(i) = 61, 10, 9, 11, 9$$

then

$$\text{Median}[f(i)] = 10$$

Note: is effectively *ignores* the out-of-place large value, so *removes noise points*.

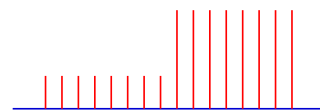
In one-dimensions the median filter removes all features of less than $M/2 + 1$ in size but preserves all other features.



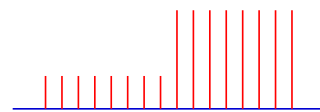
Similar to Shrink/Expand, but is also valid for Grey Level images.

Edge Preserving Properties

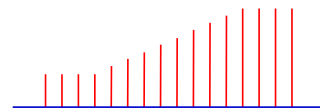
The *most* useful feature of the Median filter is its property at edges:



Input



Median (any length)



Average (length 5)

Median Filter of Images

In Two-dimensions it removes all feature of size $< M^2/2 - 1$ while retaining all other features, **and** retaining edges.



3×3 Median



5×5 Median

Very useful noise reduction filter used throughout image processing.

Filter effectively **smooths** the image into regions of constant intensity but retains edges.

So acts as a selective Low-Pass filter.

Implementation of Median Filter

To calculate **Median** over each window the data must be (partly) sorted.

Computationally expensive, and typically 5×5 Median filter about the same computational time as DFT.

Aside: Medians of large arrays are very slow to calculated by “thick” (SelectSort) way. For fast technique, see Numerical Recipes in <language> section 8.5

One of the most useful real space filters available.

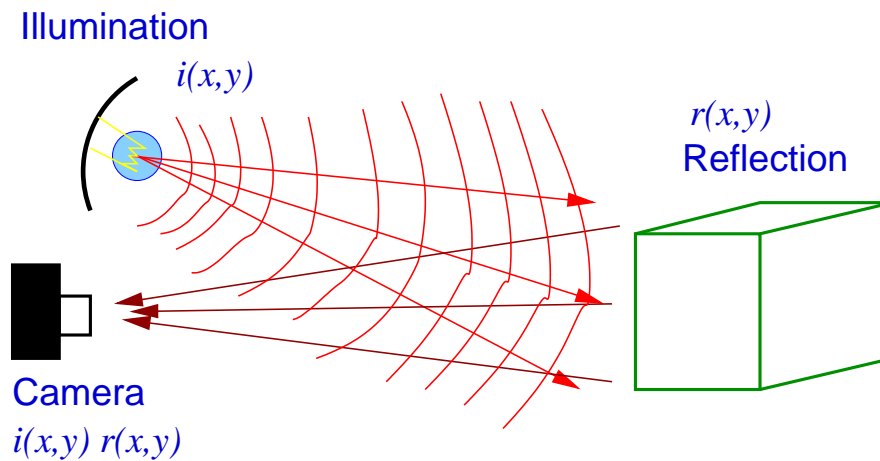
Homomorphic Filtering

For the case of a multiplicative process in Real space,

$$f(x,y) = i(x,y) r(x,y)$$

where

$$\begin{aligned} i(x,y) &= \text{Illumination} \\ r(x,y) &= \text{Reflectance} \end{aligned}$$



Homomorphic Filtering I

Form $\ln[]$ to separate terms go give,

$$z(x,y) = \ln(i(x,y)) + \ln(r(x,y))$$

Fourier transform this to give,

$$Z(u,v) = \mathcal{F}\{\ln(i(x,y))\} + \mathcal{F}\{\ln(r(x,y))\}$$

known as the *Cepstrum* of $f(x,y)$. Consider the frequency characteristics of each term:

$$\begin{aligned} i(x,y)\text{smooth} &\Rightarrow \ln(i(x,y))\text{smooth} \\ r(x,y)\text{rough} &\Rightarrow \ln(r(x,y))\text{rough} \end{aligned}$$

Filter $Z(u,v)$ to get

$$Y(u,v) = Z(u,v)H(u,v)$$

where

$$\begin{aligned} \text{High pass} &\Rightarrow \text{Reduces } i(x,y) \\ \text{Low pass} &\Rightarrow \text{Reduces } r(x,y) \end{aligned}$$

then reform “filtered” image by,

$$g(x,y) = \exp[\mathcal{F}^{-1}\{Y(u,v)\}]$$

Typically used to correct for illumination effect.

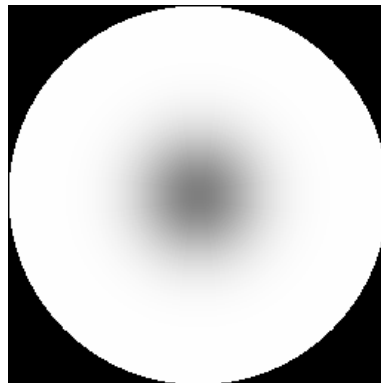
Homomorphic Example



Input Image



Log of Input



Filter



Output

Low frequency variation in illumination has been (partially) removed.

See Gonzalez & Woods for better example

Summary

In this section we have covered:

1. Linear filtering in both real and Fourier space.
2. Examples low and high pass Fourier filters and their basic properties.
3. Example in real space linear filters for image smoothing and formation of differentials.
4. Method of combining linear filters in both real and Fourier space.
5. Real space non-linear filters.
6. Shrink and expand filters for image segmentation.
7. Average threshold filters for data drop-out noise removal.
8. Median filters and its edge preserving properties.
9. Homomorphic filtering for correction of illumination variation.

This is the main image processing technique.