Image Processing

(Year III, 2-nd semester)

Lecture 9-10:

Digital filtering

Technical University of Cluj Napoca

Computer Science Department

Contents

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

Technical University of Cluj Napoca
Computer Science Department

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: $\mathbf{g(i,j) = h(i,j) \odot f\ (i,j)}$
  - In Fourier Space (using the convolution theorem) :

$$\boldsymbol{G(k,l) = H(k,l)F(k,l)}$$

- 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 the convolution kernels or filter functions
  - ***h*****(i,*j*)** - In real space
  - ***H*****(*k,l*)** - In Fourier space

Technical University of Cluj Napoca
Computer Science Department

Fourier Space Convolutions

- The result of a Fourier space convolution can be converted in spatial domain by an invers Fourier transform:

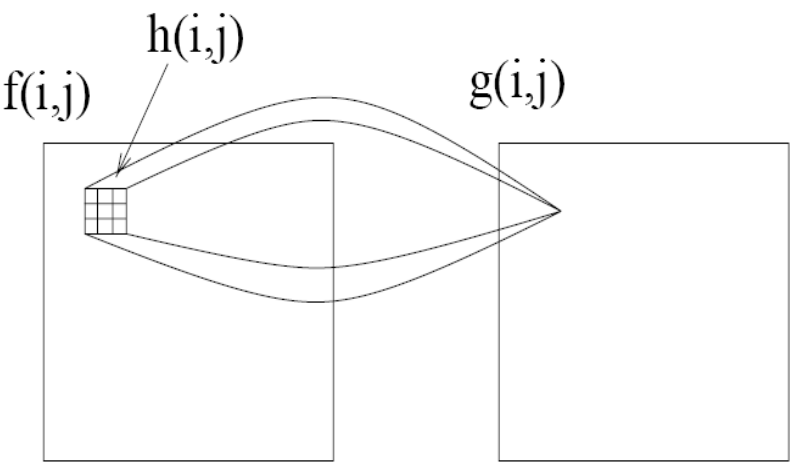
  $$\boldsymbol{g(\mathrm{i},j)=F^{-1}\{H(k,l)F(k,l)\}}$$

- Processing requires **TWO** FTs (one DFT and one IFT) and a complex multiply, (a third DFT required if $H(k,l)$ is formed from $h(\mathrm{i},j)$).
- **Note:** FTs and ‘x’ must be performed in floating point format.
- Computational time independent of filter type.

Real Space Convolutions

- For filter ***h*(i,*j*)** of size *MxM*:

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

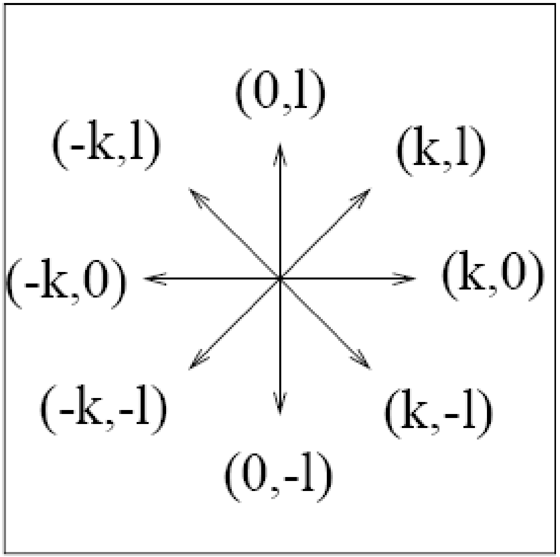


- "shift, multiply and accumulate" scheme
- Computation time ~ M^2
- All calculations can be integer or byte.
- Filter to 5x5 available in custom hardware at video rates.
- For serial machines, filters bigger than 9x9, are typically faster by Fourier space technique.

Technical University of Cluj Napoca
Computer Science Department

Fourier Space Filters

- Filtering operation applied to ***F(k,l)*** and determined by ***H*(*k,l*)** ➔ filtered Fourier domain image ***G(k,l)***.
- Most applications input & output images are **REAL.**
- The Fourier transform of a real image ***F(k,l)*** is complex and obeys the familiar properties of
  - *Real part symmetric* and
  - *Imaginary part anti-symmetric*
- To obtain a real output the Fourier filter ***H(k,l)*** must also obey these symmetry relations
- 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.
- Low pass filters:
  - Ideal
  - Gaussian
  - Smooth: Butterworth, Trapezoidal

Ideal Low Pass Filter

- Block all frequencies greater than some limit:

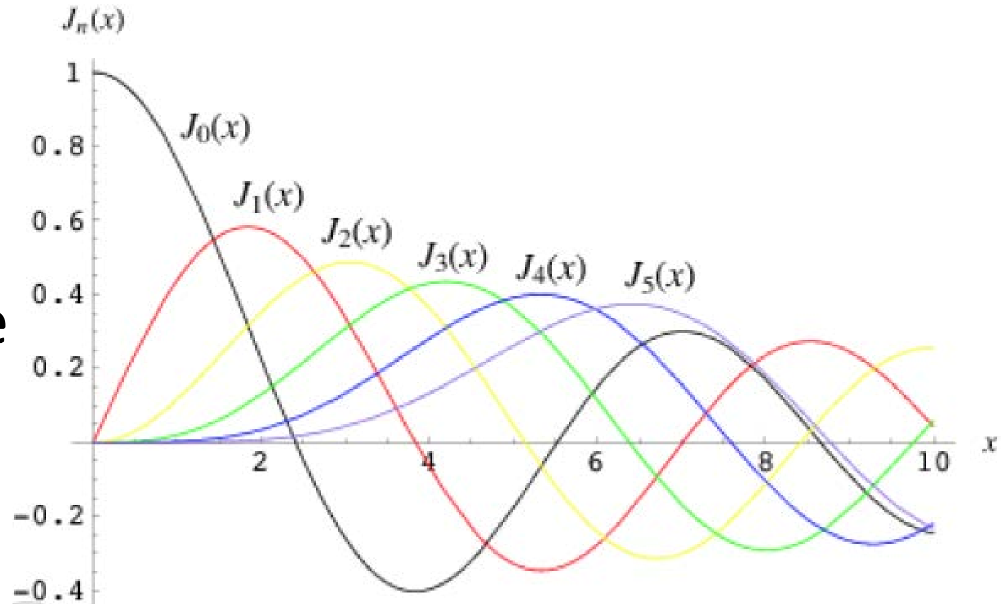
$$H(k,l)=\begin{cases}1, k^2+l^2 \le w_0^2 \\ 0, otherwise\end{cases}$$

- The ideal low pass filter in spatial domain:

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

with $r^2 = k^2 + l^2$

where J_1 is a Bessel function



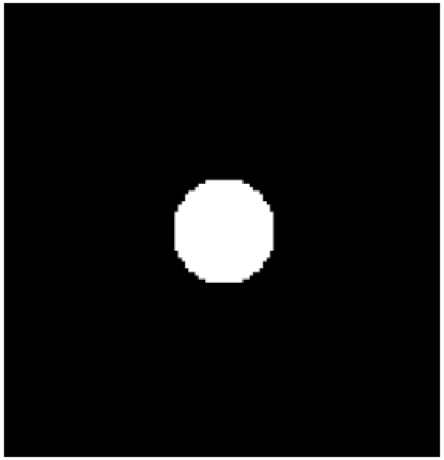
- Which results in "ringing" effects in the output image *g(x,y)* due to the lobes associated with *h*(*x,y*).

Technical University of Cluj Napoca
Computer Science Department

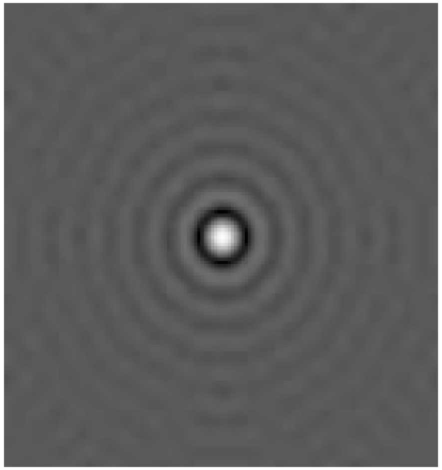
Digital Example

- 128x128 image, low-pass filter with w_0= 15:

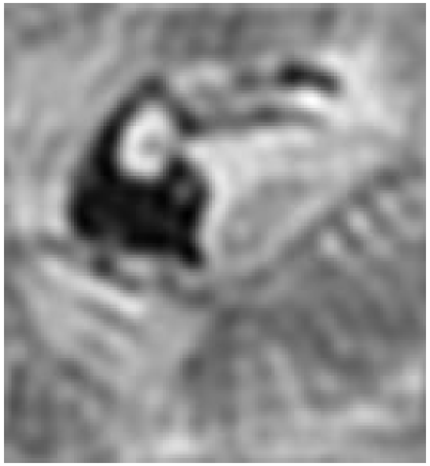
Input image



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 Filter

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

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

  where $w^2 = k^2+l^2$ and w_0 is the $1/e$ point of the Gaussian.
- In real space, h(i,j) is also Gaussian:

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

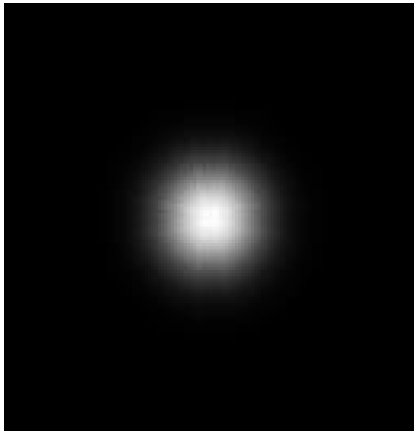
  where $r^2 = i^2+j^2$
- The filter is infinite in both Fourier and real space, so **attenuates** rather than totally removing the high spatial frequencies.
- H(u,v) does not remove high spatial frequencies
- **The image is smoothed but there is no edge ringing**

Technical University of Cluj Napoca
Computer Science Department

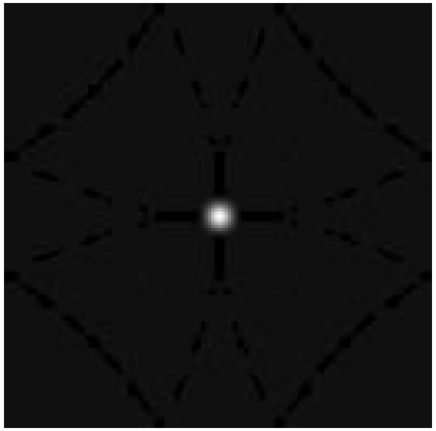
Digital example

- 128x128 image filtered with low-pass filter; w_0= 15

Input image



Low Pass filter

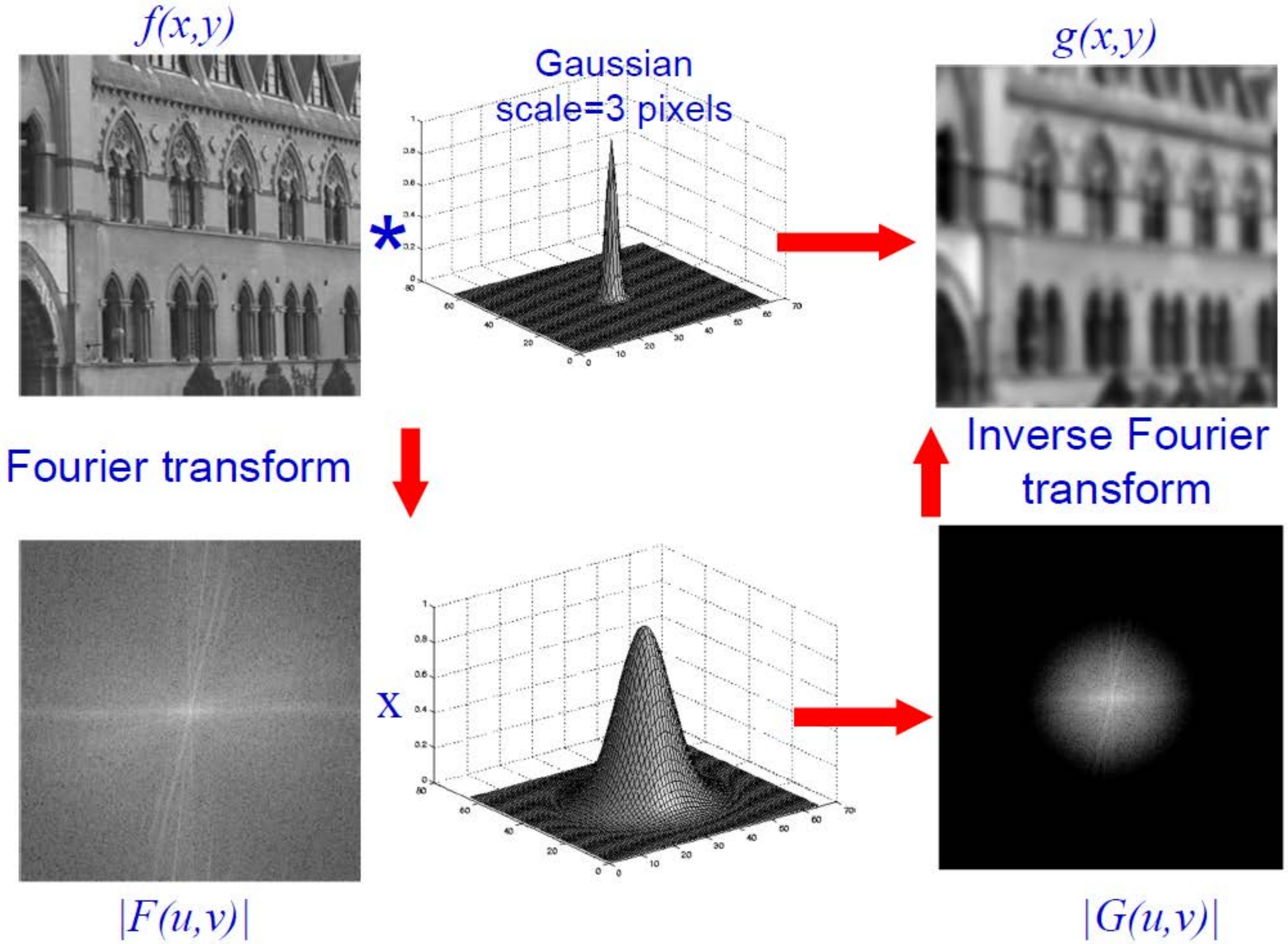


Real space filter

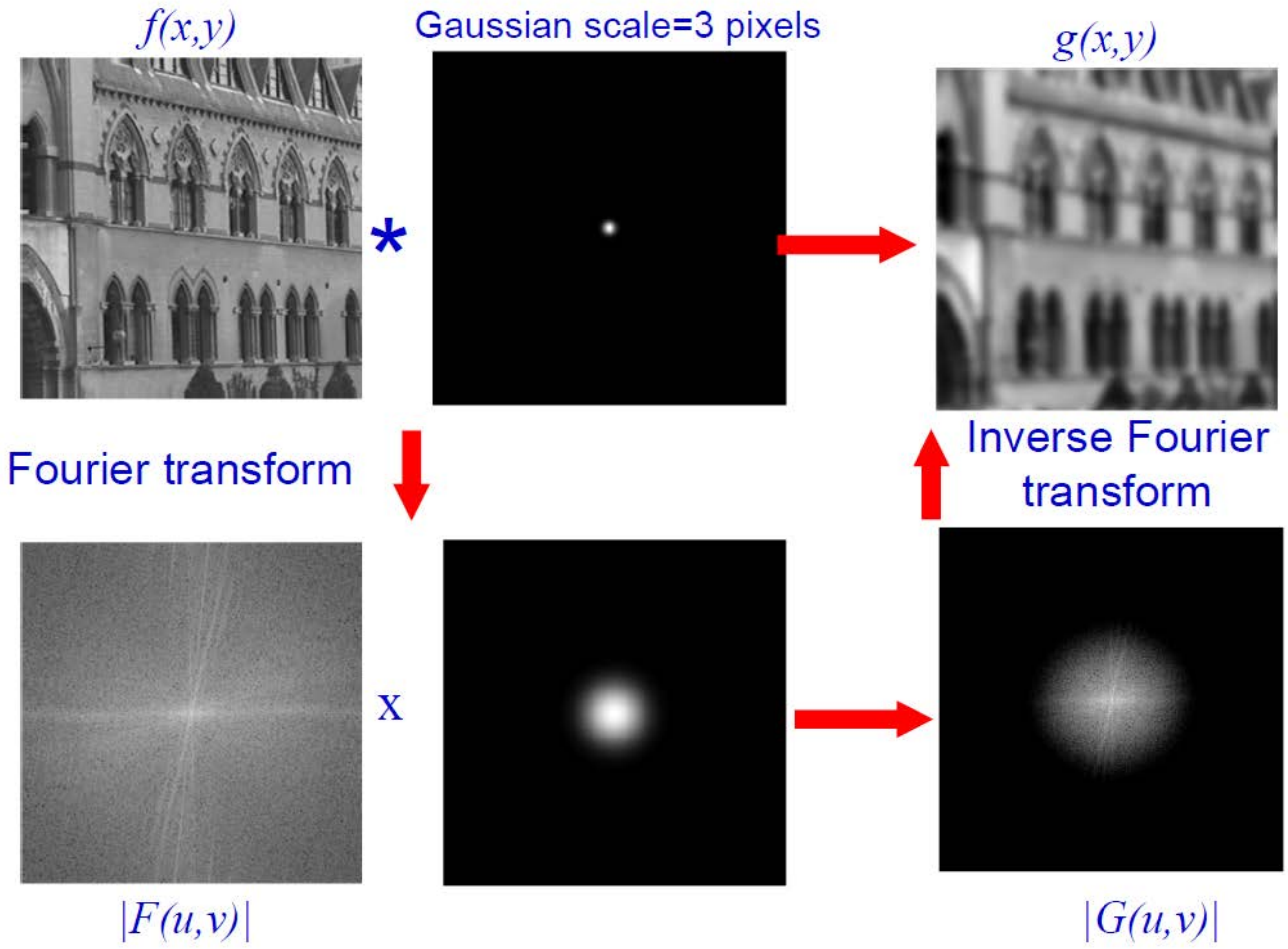
Filtered image

- Very useful digital filter for noise reduction giving a very "smooth" filtered image. Tends to severely smooth edges making further "edge detection" difficult.

Gaussian Low Pass filter - example 2



Gaussian Low Pass filter - example 2

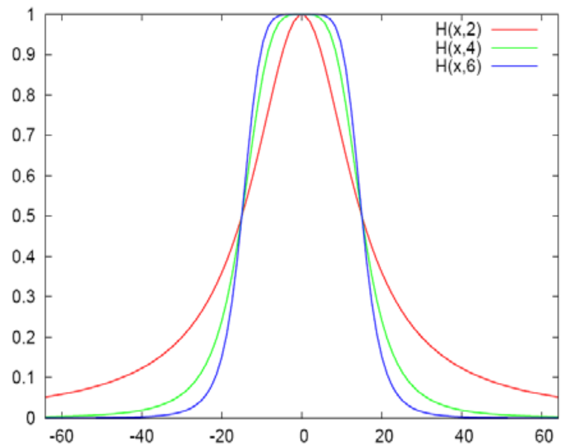


Other Smooth Low-Pass Filters

- **Butterworth Filter**:

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

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



- Plot with $w0 = 15$ and $n = 2;4;6$.
- Very similar properties to Gaussian, filter inherited from analogue signal processing.

- **Trapezoidal filter**

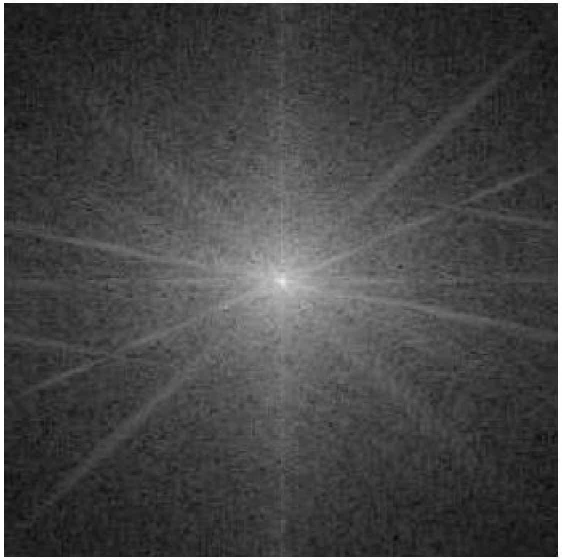
$$H(k,l) = \begin{cases} 1, & if \quad w < w_0 \\ \dfrac{w-w_1}{w_0-w_1}, & w_0 \le w \le w_1 \\ 0, & if \quad w > w_1 \end{cases}$$

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

Technical University of Cluj Napoca
Computer Science Department

Low Pass Butterworth filtering example

- Low Pass filtering removes high frequencies, blurs image
- Gentler cutoff eliminates ringing artifact



DFT of Image after low pass Butterworth filtering

Resulting image after inverse DFT

High Pass Filters

- High pass filters allow HIGH spatial frequencies to pass while attenuating or blocking LOW spatial frequencies. Used for the enhancement of high frequencies (and thus edges).
- Types:
  - Ideal
  - Gaussian
  - Smooth: Butterworth

Ideal High Pass filter

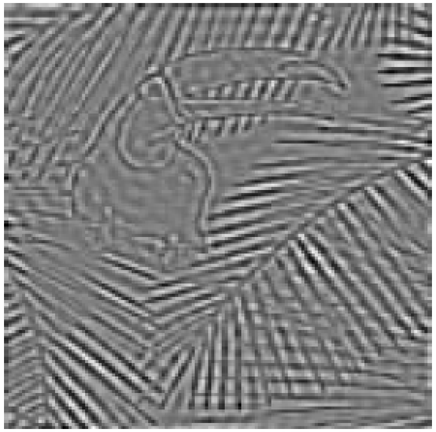
- Block all frequencies less than some limit,

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

- This filter suffers from such sever ringing artifacts that it is rarely used and much better operations can be obtained from the smooth high pass filters.

Example with w_0=25

Filter



Output Image

Technical University of Cluj Napoca
Computer Science Department

Gaussian High Pass Filter

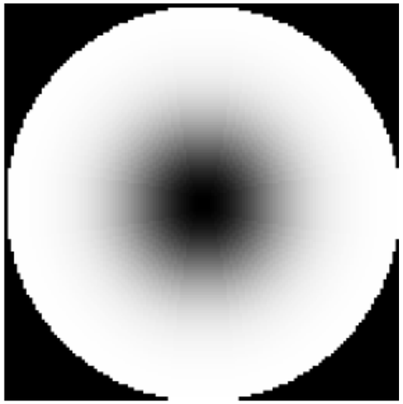
- smooth reduction of **low** spatial frequencies while the high spatial frequencies are pass unaltered.

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

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”

- Example with w_0 = 25.



Filter

Output Image

In practice often combined with the Gaussian Low Pass filter to form a composite Difference of Gaussians (DOG) filter.

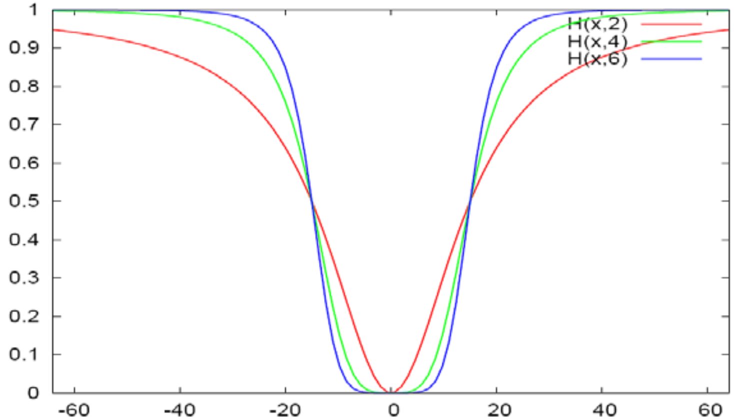
Other Smooth High Pass Filters

High Pass Butterworth:

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

where w_0 is half point and n is the order.

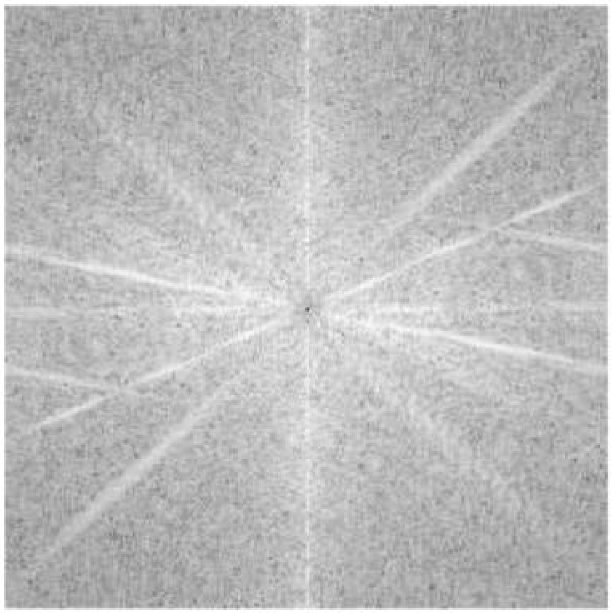
- Plot with w_0 = 25 and n = 2; 4; 6. Gives almost identical results to



- **Lowpass:** Ideal, Gaussian, Butterworth.
- **Highpass:** Ideal, Gaussian, Butterworth.
- **Bandpass:** Difference of Gaussians (DOG) filter.

Technical University of Cluj Napoca
Computer Science Department

High Pass Butterworth filter

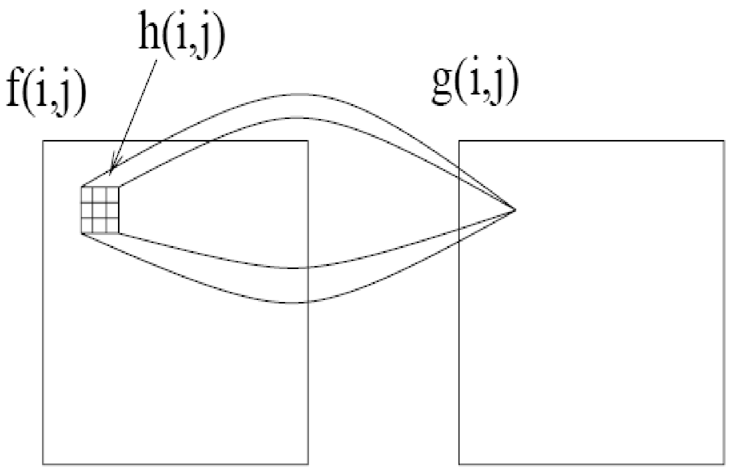


DFT of Image after high
pass Butterworth filtering

Resulting image
after inverse DFT

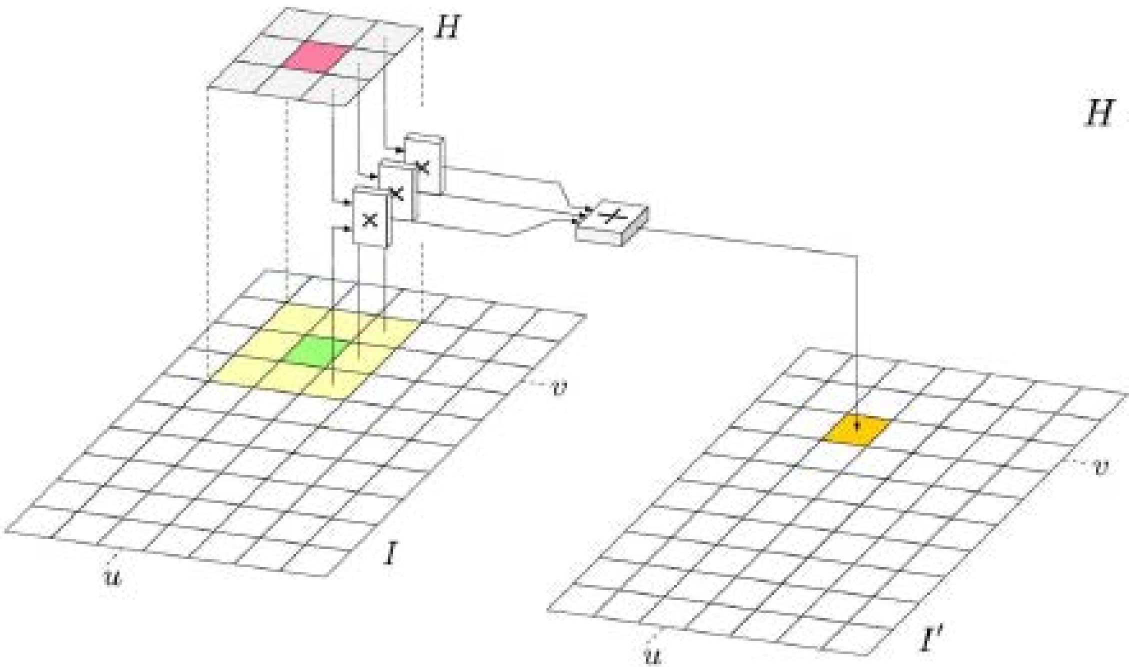
Real Space Filters

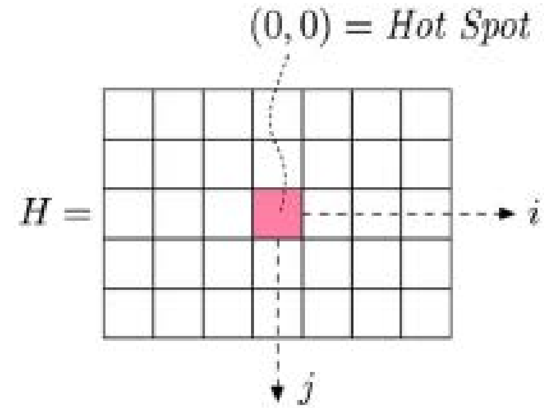
- Filter is specified in real space by the mask *h*(i,*j*) of finite size, typically 3x3, 5x5 or sometimes 7x7.



- 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 that 7x7, faster to use Fourier technique.
- **Note :** Convolution can be easily implemented by a parallel computer system, or custom parallel hardware.

Real Space Filters





$$I'(u,v) \leftarrow \sum_{(i,j)\in R_H} I(u+i, v+j) \cdot H(i,j)$$

$$I'(u,v) \leftarrow \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u+i, v+j) \cdot H(i,j)$$

Technical University of Cluj Napoca
Computer Science Department

Real Space Averaging

- Replace each pixel by the average of its neighbours, has the effect of “Low pass” filtering, so used to reduce the effect of noise.
- Effect or reducing high spatial frequencies, but not removing them, (actually removes a range of spatial frequencies).

- **5 Pixel Average**

  0 1 0
  1 1 1
  0 1 0

  - gives a filter with an effective radius of 1 pixel

- **9 Pixel Average**

  1 1 1
  1 1 1
  1 1 1

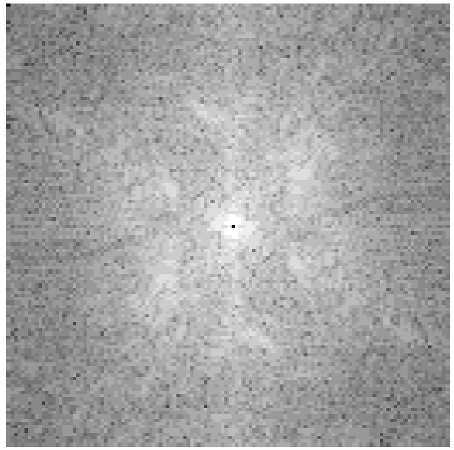
  - gives a filter with an effective radius of $\sqrt{2}$ pixels

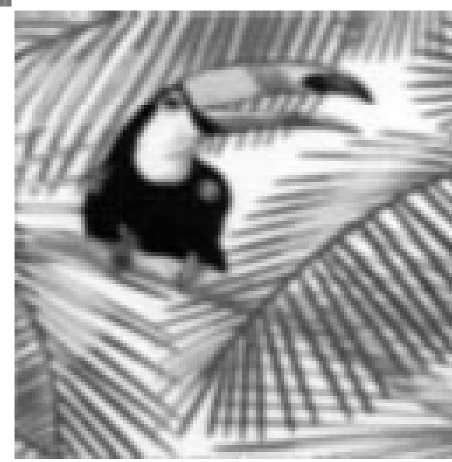
It is approximately equivalent to multiplication in Fourier space with a functions:
$H(k; l) = \mathrm{sinc}(Nk/3)\ \mathrm{sinc}(Nl/3)$ where NxN is the size of the image.

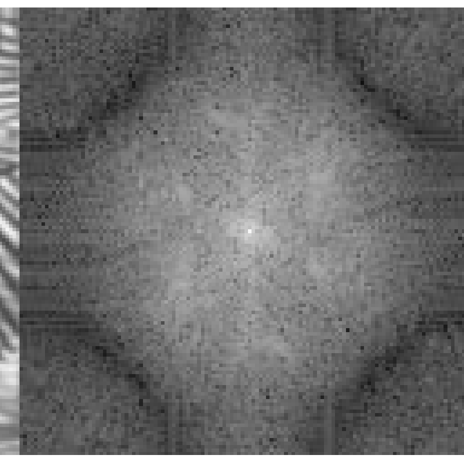
Technical University of Cluj Napoca
Computer Science Department

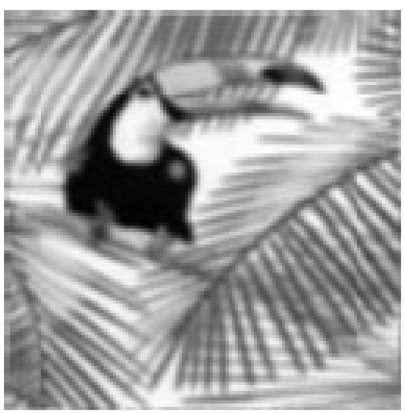
Digital Example

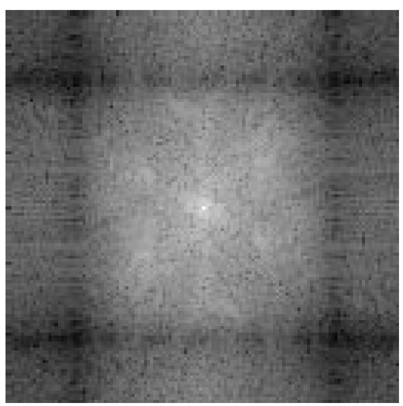

Input image


Fourier Transform


5 point averaging


Fourier Transform


9 point averaging


Fourier Transform

Real Space Gaussian Filters

Gaussian Kernel

1D Case $g_{\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$

2D Case $G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

Separability of 2D Gaussian

$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} = g_{\sigma}(x) \cdot g_{\sigma}(y)$$

Consequently, convolution with a Gaussian is separable: $I*G=I*G_x*G_y$ where G is the 2D discrete Gaussian Kernel and G_x *and* G_y are "horizontal" and "vertical" 1D discrete Gaussian kernels.

Technical University of Cluj Napoca
Computer Science Department

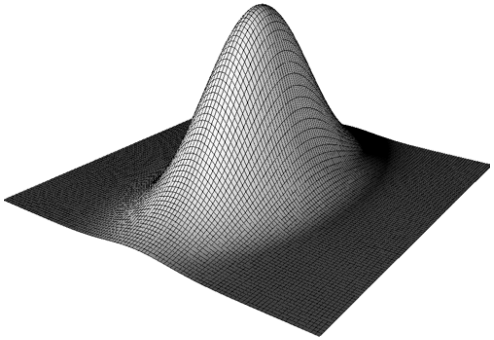
Real Space Gaussian Filters

$$G_{2D}(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

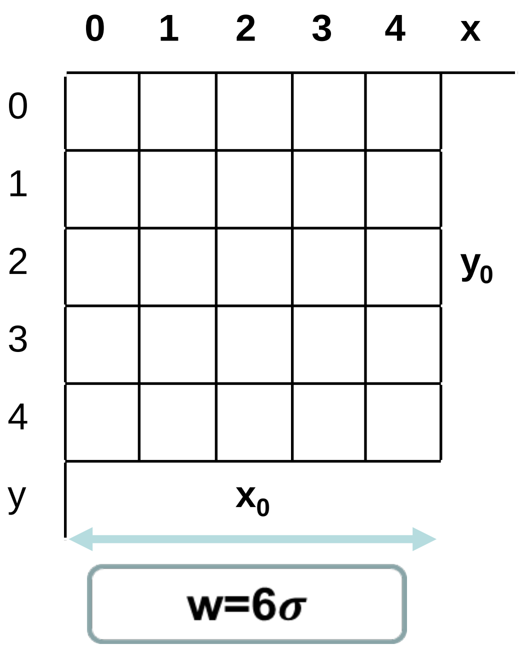
with $\boldsymbol{\mu}$**=o** and $\boldsymbol{\sigma = 1.0}$

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1



Design of a Gaussian kernel with given σ



1. Estimate the size of the kernel **w=6**$\boldsymbol{\sigma}$
2. Evaluate the values in each cell of the kernel:

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x_0-x)^2+(y_0-y)^2}{2\sigma^2}}$$

3. A rounding and a scaling is necessary
4. The middle row and column represent the 1 D kernels

Real Space Differentiation

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

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

- In the discrete case δ=1:

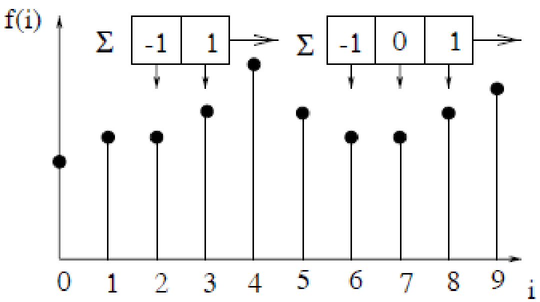
$$\frac{df(i)}{di} = f(i+1) - f(i)$$

- Which if we consider convolution as "**Shift-fold-multiply-add**" then differentiation can be written as:

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

- Simmilarly with δ=2 we get:

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



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

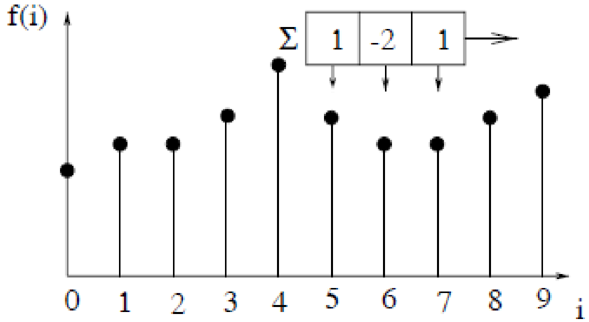
Second order differentials

- 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 -2 1] = [-1 1] ⊙ [-1 1]

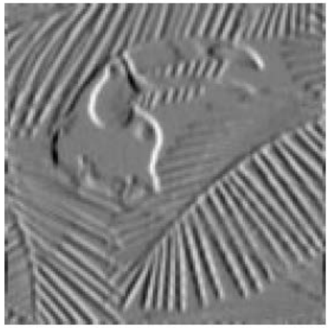
As would be expected since convolution is a linear operation.

Two dimensional differentials

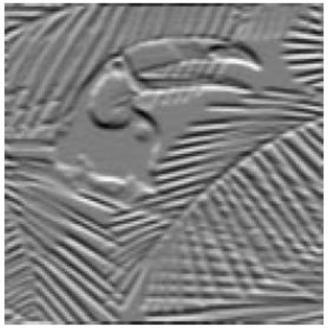
- In two dimensions we have:
- $\frac{\partial f(i,j)}{\partial i} = [-1 \;\; 0 \;\; 1] \odot f(i,j)$ and $\frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \odot f(i,j)$
- However to reduce the effect of noise, it is conventional to average the differential over 3 rows/columns respectively to give:

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

Which will enhance the vertical and horizontal edges respectively.



x-differential

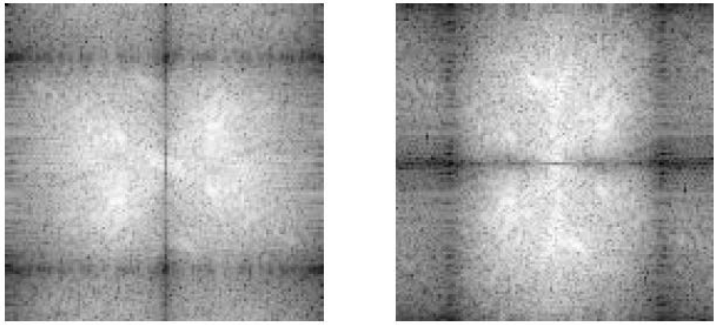


y-differential

Technical University of Cluj Napoca
Computer Science Department

Fourier space differentials

- Properties:
- $F\left\{\frac{\partial f(x,y)}{\partial x}\right\} = i2\pi u F(u,v)$ and $F\left\{\frac{\partial f(x,y)}{\partial y}\right\} = i2\pi v F(u,v)$
- 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.

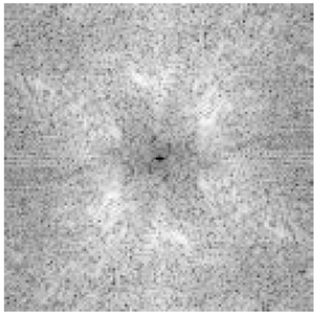


x-differential (FT)   y-differential (FT)

Note: the Fourier transforms show zero through vertical/horizontal lines as expected, plus additional line zeros due to averaging effect.

Second order differentials

- For the second order differentials we have:
  - $\frac{\partial^2 f(i,j)}{\partial i^2} = [1 \ \ -2 \ \ 1] \odot f(i,j)$ 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.
- We also have: $F\{\nabla^2 f(x,y)\} = -(2\pi w)^2 F(u,v)$ where $w^2 = u^2 + v^2$ giving:



Enhances edges in all directions.

Technical University of Cluj Napoca
Computer Science Department

Laplacian Variations

- We can form an 8 point Laplacian by using

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

which also takes the Laplacian, but is less sensitive to noise.

- **Edge Enhancement:**
  – Edges of an image may be enhanced by the subtraction of the Laplacian from and image, which can be formed by,

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

Input image

Edge Enhanced

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** is real space by $\odot$ operation.

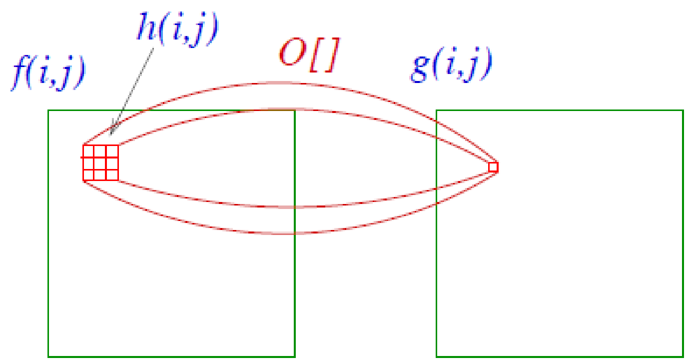
Technical University of Cluj Napoca
Computer Science Department

Non-Linear Real Space Filters

- The real space shift & multiply operation can be modified to:

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

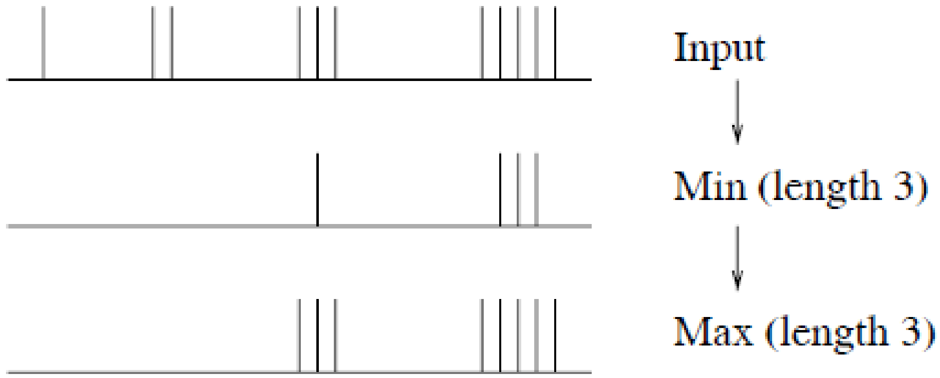
- Range of h(m,n) defined by w.
- The operation is now defined by the mask h(i,j) and operator O[]



In most non-linear filters we have: $h(i,j) = 1, i,j \in w$
With the operation of the filter controlled by O[] and the size of w only.

Shrink and Expand Filters

- Taking O[] = Min[] the operator will act as a Shrink operation with bright objects reduced in size by approximately the “size” of the filter.
- Taking O[] = Max[] the operator will act as an Expand filter, with bright objects increasing in size by approximately the “size” of the filter.
- These operators typically used as a pair on binary image to remove small, isolated regions.
- These filters are not commutative, i.e. $E[S[f(i,j)]] \neq S[E[f(i,j)]]$



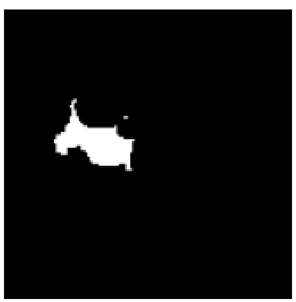
Two Dimensional Case

- 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

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



Binary Shrink

Binary Expand

Two Dimensional Case

(a) (b) (c)

Original Image with Salt-and-pepper noise

Minimum filter removes bright spots (maxima) and widens dark image structures

Maximum filter (opposite effect): Removes dark spots (minima) and widens bright image structures

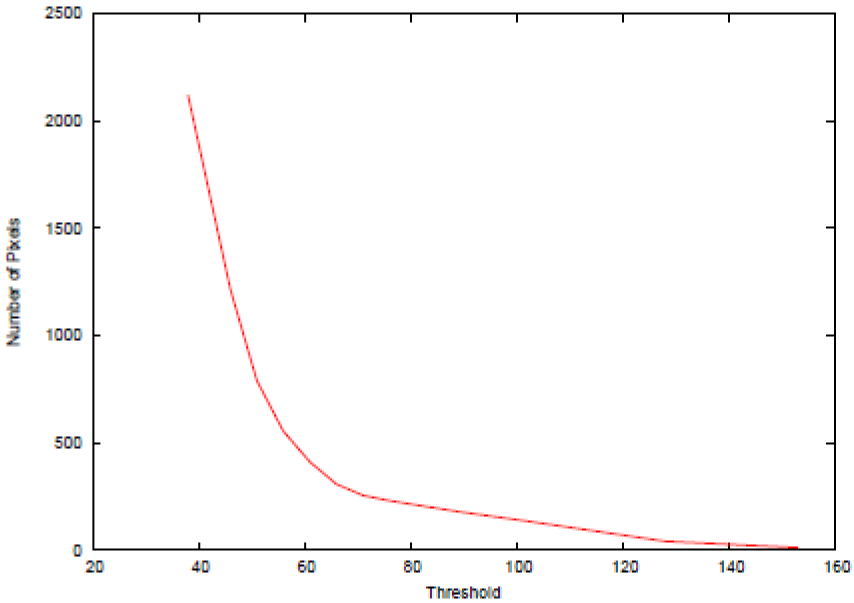
Threshold Average Filter

- For “data-dropout” noise we have isolated “noise points” that differ from the neighbour pixels.
- Compare 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 3x3 filter we have:
- $h(i,j) = \begin{bmatrix} k & k & k \\ k & 0 & k \\ k & k & k \end{bmatrix}$ where k = 1/(M^2-1) = 0.25, then output is:
- $g(i,j) = \begin{cases} A \ if \ |A - f(i,j)| > T \\ \quad f(i,j) \ otherwise \end{cases}$
- 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
- Typical threshold value

  $T=0.25\ f_{MAX}$

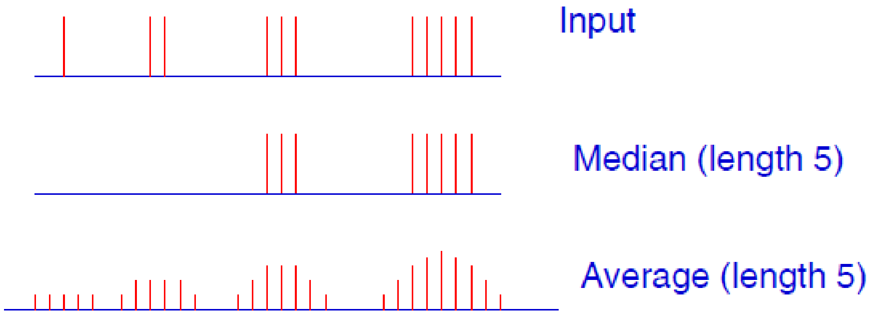


1:50 Bit Error

Threshold of 66

Median Filter

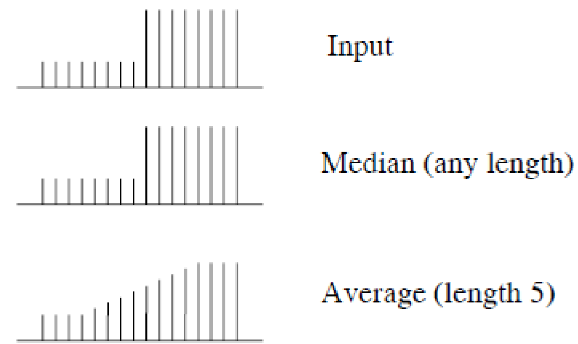
- The Median filter is formed by setting *O*[] =Median[]
- Where the median is defined as the *middle* value.
  - Eg. for the 5 values *f* (*i*) = 61,10,9,11,9 then Median[*f* (*i*)] = 10
- **Note:** it effectively ignores the out-of-place large values, so removes noise points.
- In 1D the median filter 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 property

- The most useful feature of the Median filter is its edge preserving property
- In 2D it removes all feature of size $< M/2-1$ while retaining all other features, **and** retaining edges.
- Very useful noise reduction filter used throughout image processing.
- Filter effectively smoothens the image into regions of constant intensity but retains edges.
- So acts as a selective Low-Pass filter.




3×3 Median


5×5 Median

Median Filter Example

(a)

Original Image with Salt-and-pepper noise

(b)

Linear filter removes some of the noise, but not completely. Smears noise

(c)

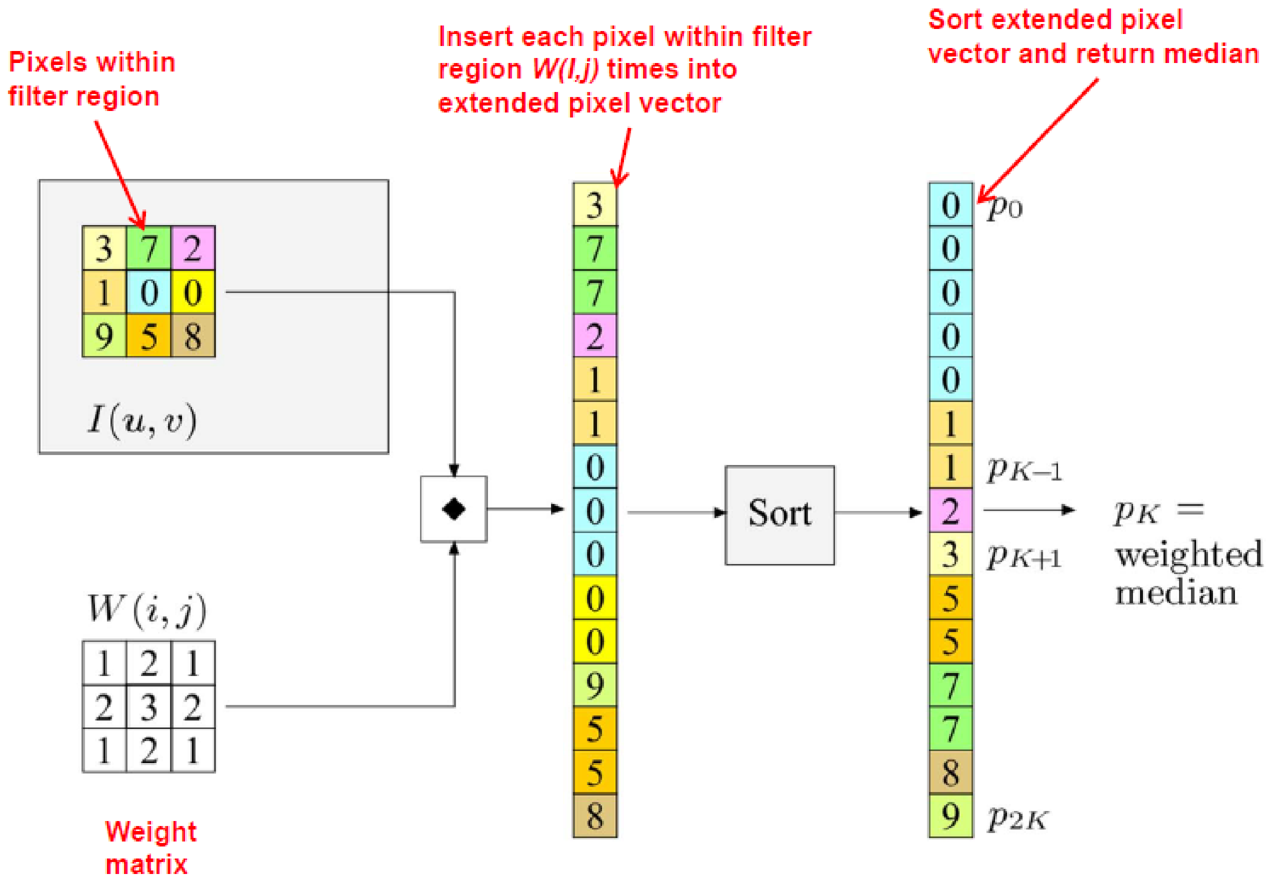
Median filter salt-and-pepper noise and keeps image structures largely intact. But also creates small spots of flat intensity, that affect sharpness

Technical University of Cluj Napoca
Computer Science Department

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. Fast sorting techniques should be used.*
- One of the most useful real space filters available.

Weighted Median Filter



Median filter assigns weights (number of "votes") to filter positions *W(i,j)*

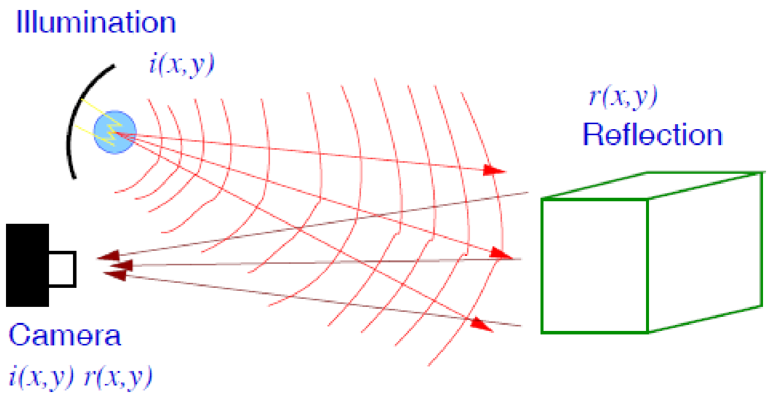
To compute result, each pixel value within filter region is inserted *W(i,j)* times to create **extended pixel vector**

Extended pixel vector then sorted and median returned

Weighting can be applied to implement non-rectangular filters and/or to increase the influence of a specific area.

Homomorphic Filtering

- For the case of a multiplicative process in Real space,
  - $f(x,y) = i(x,y)r(x,y)$
  - Where i(x,y) = Illumination and r(x,y) = reflectance.



Homomorphic Filtering cont'd

- Apply ln() to separate terms:

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

- Apply Fourier transform

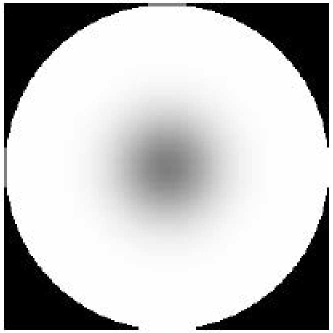
  $Z(u,v)= F(ln(i(x,y))) + F(ln(r(x,y)))$ known as Cepstrum

- Consider the frequency characteristic of each term:
  - $i(x,y)$ is smooth then ln(i(x,y)) is smooth
  - $r(x,y)$ is rought then ln(r(x,y)) is rought
- Filter $Z(u,v)$ to get $Y(u,v)=Z(u,v)H(u,v)$ where
  - High Pass filtering improves $i(x,y)$
  - Low Pass filtering improves $r(x,y)$
- Then the improved image $g(x,y) = exp(F^{-1}(Y(u,v)))$
- Typically used to correct the illumination but can also be used for dealing with multiplicative noise

Homomorphic Filtering example

Input Image

Log of Input



Filter

Output

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

Bibliography

- https://www2.ph.ed.ac.uk/~wjh/teaching/dia/documents/filtering.pdf
- https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture4.pdf
- https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture10.pdf
- https://www.ibiology.org/talks/fourier-transform/
- http://www.robots.ox.ac.uk/~az/lectures/ia/lect1.pdf
- http://www.robots.ox.ac.uk/~az/lectures/ia/lect2.pdf
- https://www.pearson.com/us/higher-education/program/Gonzalez-Digital-Image-Processing-4th-Edition/PGM241219.html