Digital Image Processing
Image Filtering

O. Le Meur
olemeur@irisa.fr

Univ. of Rennes 1
http://www.irisa.fr/temics/staff/lemeur/

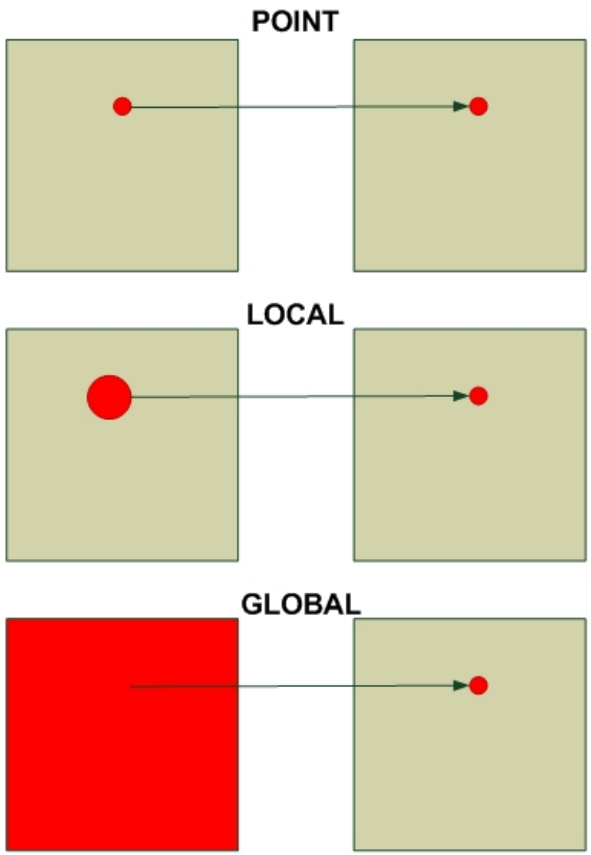
January 2, 2011

Introduction

Introduction: image transformation

There exist 3 types of transformation:

- Point to point transformation:
  The output value at a specific coordinate is dependent only on one input value but not necessarily at the same coordinate;
- Local to point transformation:
  The output value at a specific coordinate is dependent on the input values in the neighborhood of that same coordinate;
- Global to point transformation:
  The output value at a specific coordinate is dependent on all the values in the input image.



Note that the complexity increases with the size of the considered neighborhood...

Point-to-point transformation

Spatial coordinates-based transformations

Remark:

This section is composed of several pictures extracted from `http://eeweb.poly.edu/~onur/lectures/lectures.html`.

Let $im[x, y]$ be an input image of size $N \times N$.

- A spatial coordinates-based transformation, also called warping, aims at providing an image $IM[k, l]$ from the input image $im[x, y]$:

$$IM[k, l] = im[x(k, l), y(k, l)]$$

- $x(k, l)$ and $y(k, l)$ are the transformations or the pixel warping functions. These functions just modify the spatial coordinates of a pixel not its value;

- Special cases to take into consideration:

  - the new coordinates $[x(k, l), y(k, l)]$ is out of the image IM. In this case, $IM[k, l] = 0$;
  - the new coordinates must be integers (Rounding operation, nearest integers...).

Transpose

The transpose tansformation is given by

$$\begin{aligned} x(k,l) &= l \\ y(k,l) &= k \end{aligned}$$

Vertical Flip

The vertical flip tansformation is given by

$$\begin{aligned} x(k,l) &= N-k \\ y(k,l) &= l \end{aligned}$$

The horizontal flip tansformation is given by

$$\begin{aligned} x(k,l) &= k \\ y(k,l) &= N-l \end{aligned}$$

Translation

The translation tansformation is given by

$$\begin{aligned} x(k,l) &= k + T_k \\ y(k,l) &= l + T_l \end{aligned}$$

where T_k and T_l are the transalation values for the x-axis and the y-axis respectively.

In the example below, we have ($T_l = -50$)

$$\begin{aligned} IM[k,l] &= im[x(k,l), y(k,l)] \\ IM[k,l] &= im[k, l + T_l] \end{aligned}$$

Different transformations can be obtained depending on T_k and T_l values.

Rotation

The rotation tansformation is given by

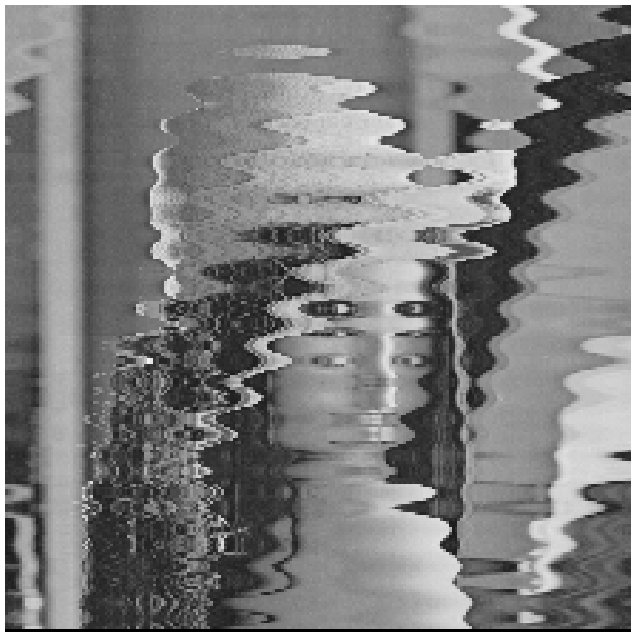
$$\begin{aligned} x(k,l) &= (k-x_0)cos\theta - (l-y_0)sin\theta + x_0 \\ y(k,l) &= (k-x_0)sin\theta + (l-y_0)cos\theta + y_0 \end{aligned}$$

where $[x_0, y_0]$ is the spatial coordinates of the center of the rotation and θ the angle.

Extracted from http://eeweb.poly.edu/ onur/lectures/lectures.html.

Wave

$$\begin{aligned} x(k,l) &= k \\ y(k,l) &= l + \alpha \times sin(\beta \times l) \end{aligned}$$



$$\begin{aligned} x(k,l) &= k + \alpha \times sin(\beta \times k) \\ y(k,l) &= l \end{aligned}$$

α and β can be used to strengthen the effect.

Warp and swirl

$$
\begin{aligned}
x(k,l) &= k \\
y(k,l) &= \frac{sign(l-y_0)}{y_0} \times (l-y_0)^2 + y_0
\end{aligned}
$$

The swirl effect is a rotation but the angle of the rotation θ varies with the pixel distance from the center of the image $[x_0, y_0]$:

$$
\begin{aligned}
d &= \sqrt{(k-x_0)^2 + (l-y_0)^2} \\
\theta &= \frac{\pi}{512} r
\end{aligned}
$$

If $r \longrightarrow 0$, θ is small...

Glass effect

A glass effect is obtained by adding a small and random displacement to each pixel:

$$\begin{aligned} x(k,l) &= k + (RAND(1,1) - 1/2) \times 10 \\ y(k,l) &= l + (RAND(1,1) - 1/2) \times 10 \end{aligned}$$

Extracted from http://eeweb.poly.edu/ onur/lectures/lectures.html.

Summary

All 2D linear transformations:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} k \\ l \end{bmatrix}$$

Scale, rotation, mirror...

Properties:

- Origin maps to origin;
- Lines map to lines;
- Ratios are preserved...

Affine transformations (linear transf. + translation):

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} k \\ l \\ w \end{bmatrix}$$

Properties:

- Origin does not necessarily map to origin;
- Lines map to lines;
- Ratios are preserved...

Summary

Affine transformations (linear transf. + translation):

- Translation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} k \\ l \\ 1 \end{bmatrix}$$

- Scale:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} k \\ l \\ 1 \end{bmatrix}$$

- 2D in-plane rotation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} k \\ l \\ 1 \end{bmatrix}$$

Pixel values-based transformations

Let $im[x, y]$ be an input image of size $N \times N$.

- A pixel values-based transformation aims at providing an image $IM[k, l]$ from the input image $im[x, y]$:
$$IM[k, l] = f\left(im[k, l]\right)$$
- Noticed that the spatial coordinates of pixel are not modified. The function f is used to modify the pixel values.

The simplest one is the identity function: $f(p) = p$:

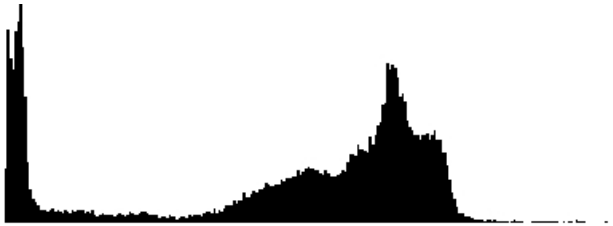
$$\begin{aligned} IM[k, l] &= f\left(im[k, l]\right) \\ IM[k, l] &= im[k, l] \end{aligned}$$

Histogram

An image histogram is a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value.

- Histogram gives information about the global distribution of an image.
- Histogram plots the number of pixels in the image (vertical axis) with a particular brightness value (horizontal axis).

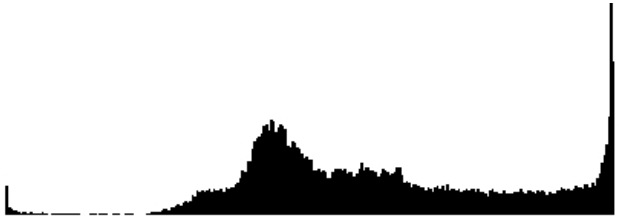
(a) Original



(b) Histogram (intensity)

Histogram

(a) Original

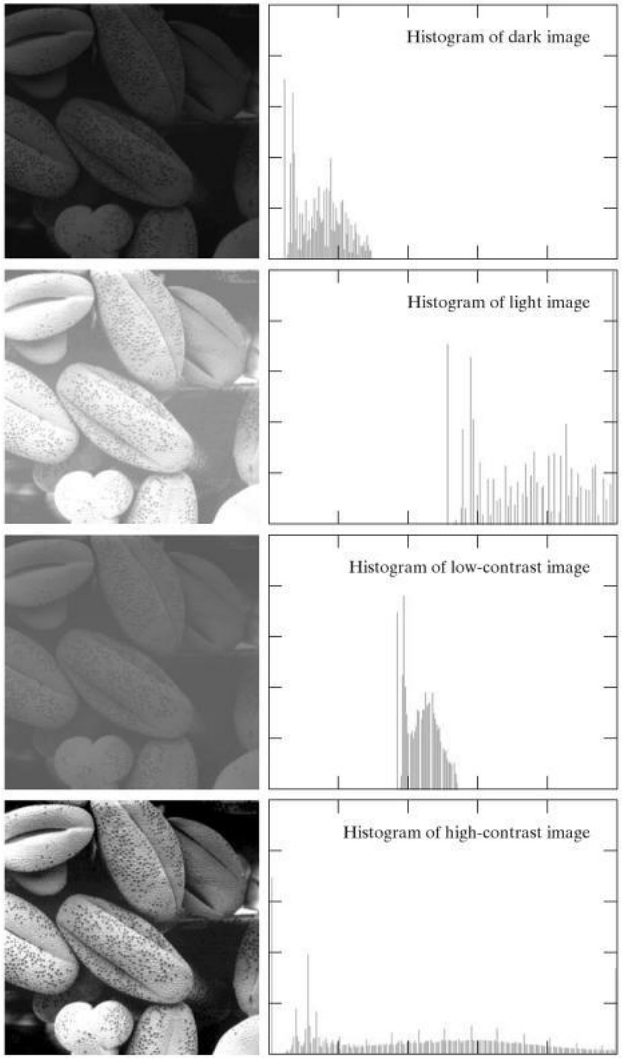


(b) Histogram (intensity)

(c) Original

(d) Histogram (intensity)

Histogram



From R.C. Gonzalez, R.E. Woods, Digital image processing.

High dynamic range in the last case providing the best quality.

Histogram equalization

The goal is to increase the global contrast of images, especially when the usable data of the image is represented by close contrast values.

Consider a discrete grayscale image $\{x\}$ and let n_i be the number of occurrences of gray level i.

- The probability of an occurrence of a pixel of level i in the image is $p_x(i) = p(x = i) = \frac{n_i}{n}$, $0 \leq i < L$, L being the total number of gray levels in the image, n being the total number of pixels in the image.
- Let us also define the cumulative distribution function: $cdf_x(i) = \sum_{j=0}^{i} p_x(j)$.

We want to produce an image $\{y\}$, such that $cdf_y(i) = iK$.

$$y = cdf_x(x) \times (max\{x\} - min\{x\}) + min\{x\}$$

Negative image

The negative image is obtained by $f(p) = 255 - p$ (pixel values are coded on 8 bits).

$$\begin{aligned} IM[k,l] &= f\left(im[k,l]\right) \\ IM[k,l] &= 255 - im[k,l] \end{aligned}$$

Piece continuous transformation

The objective of such transformation is:

- to compress pre-determined ranges of values: Range compression;
- to accentuate pre-determined ranges of values: Range stretching.

$$f(p) = \begin{cases} \alpha_1 \times p & 0 \leq p < a_1 \\ \alpha_2 \times (p - a_1) & a_1 \leq p < a_2 \\ \vdots & \\ \alpha_i(p - a_{i-1}) + \left(\sum_{j=1}^{i-1} \alpha_j(\alpha_j - \alpha_{j-1}\right) & a_{i-1} \leq p < a_i \\ \vdots & \end{cases}$$

Obviously, we have:

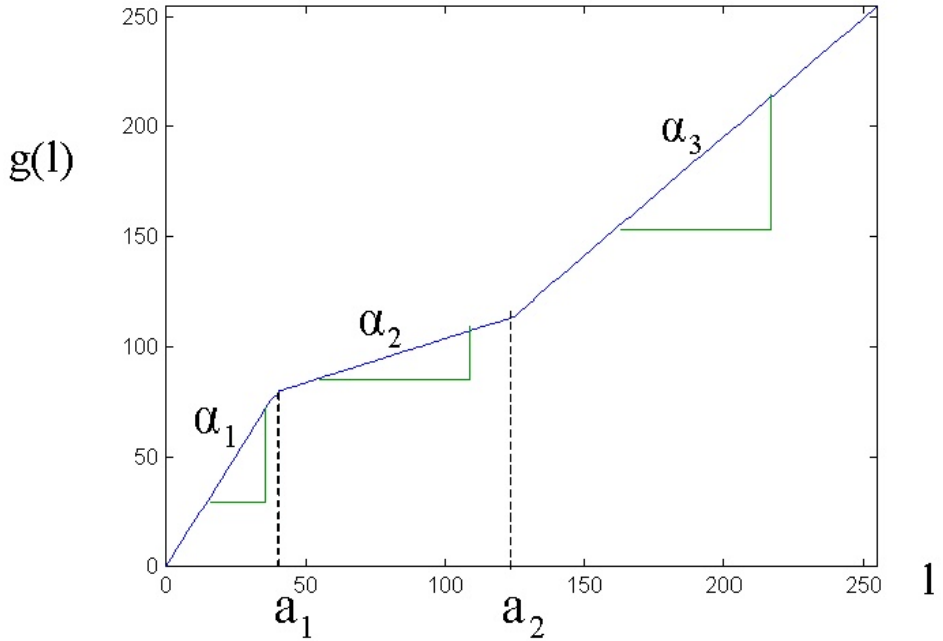
- $\alpha_i < 1$, compression;
- $\alpha_i > 1$, stretching.

Piece continuous transformation

Example of contrast stretching:

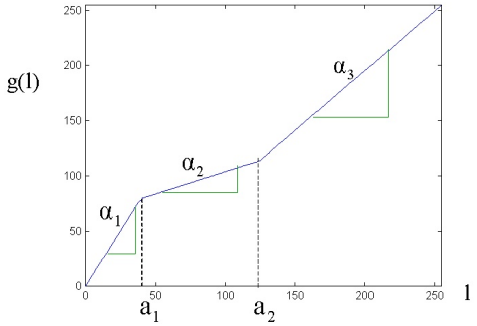
- Let f a function defined on three pieces:

$$f(p) = \begin{cases} \alpha_1 p, & 0 \leq p < a_1 \\ \alpha_2(p - a_1) + \alpha_1 a_1, & a_1 \leq p < a_2 \\ \alpha_3(p - a_2) + (\alpha_2(a_2 - a_1) + \alpha_1 a_1), & a_2 \leq p \leq 255 \end{cases}$$

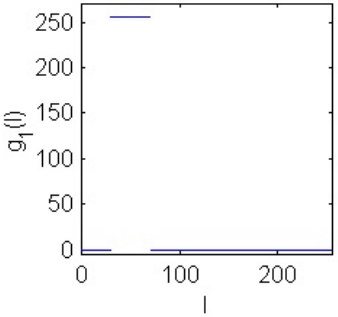


Piece continuous transformation

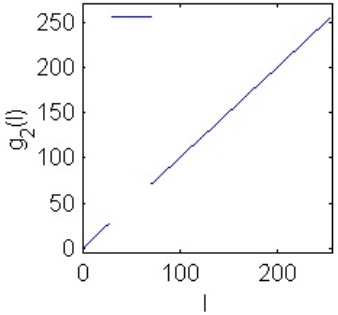
Example of contrast stretching:



- Let imaging that α_1 and α_2 are null. The filtered image contains only grey level belonging to $[a_1, a_2]$. We just keep a slice of the image.



without background



with background

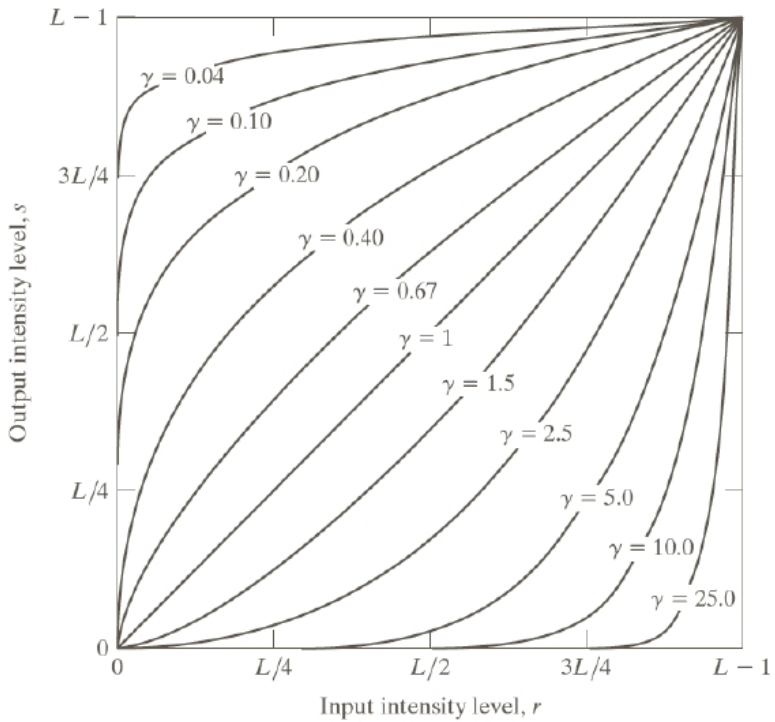
Piece continuous transformation

- Binary thresholding:

$$IM[x,y] = \begin{cases} 0 & \text{im[x,y]<T} \\ 255 & otherwise \end{cases}$$

- Gamma correction:

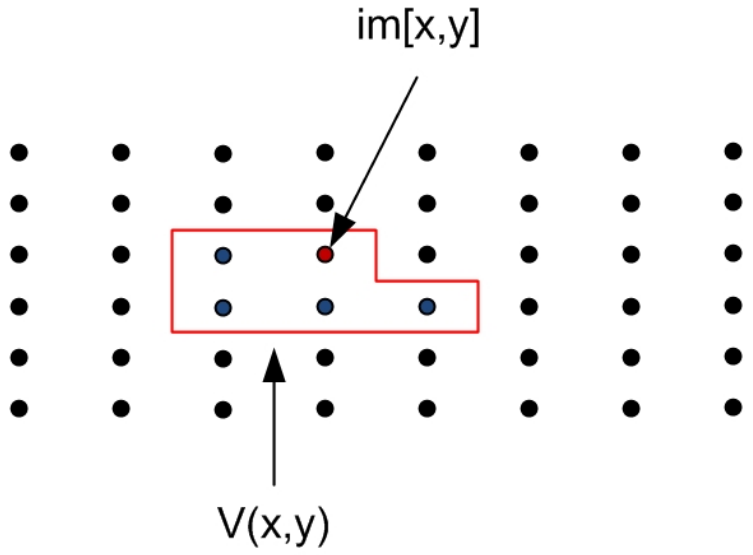
$$IM[x,y] = im[x,y]^{\gamma}$$



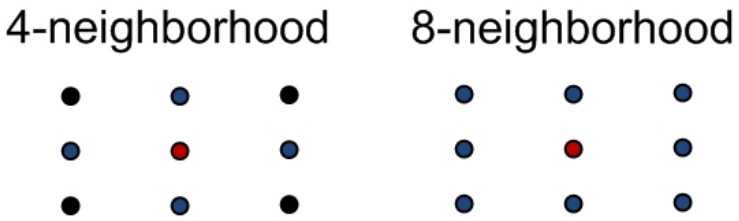
Linear filtering with neighborhoods operator

Definition

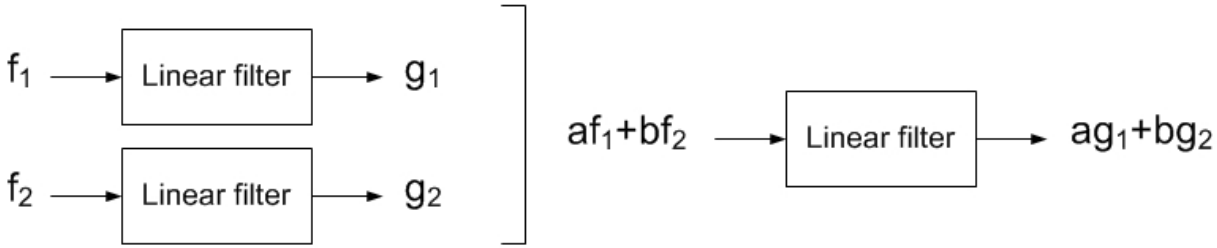
Definition of neighborhood around a pixel of spatial coordinate (x, y). The neighborhood is called $V(x, y)$:



Two examples:



Definition



2D Finite Impluse Response (FIR) filter:

$$
\begin{aligned}
IM[x,y] &= (im * h)[x,y] \\
\underbrace{IM[x,y]}_{\text{Output}} &= \sum_{k \in V(x,y)} \sum_{l \in V(x,y)} \underbrace{h(k,l)}_{\text{Filter coeff.}} \underbrace{im[x-k, y-l]}_{\text{Input}}
\end{aligned}
$$

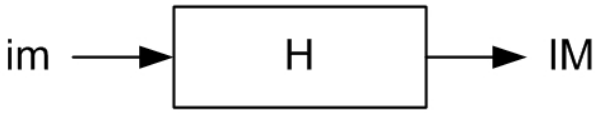
with h is the 2D impulse response called also the Point Spread Function (PSF) or the kernel of the transform. It is composed of the filter coefficients (finite length).

The gain of the filter is equal to

$$
g = \sum_{k \in V(x,y)} \sum_{l \in V(x,y)} h(i,j)
$$

Definition

$$\underbrace{IM[x,y]}_{\text{Output}} = \sum_{k \in V(x,y)} \sum_{l \in V(x,y)} \underbrace{h(k,l)}_{\text{Filter coeff.}} \underbrace{im[x-k, y-l]}_{\text{Input}}$$



where H is the convolution kernel.

Example: the filter support is (3×5). The convolution kernel is:

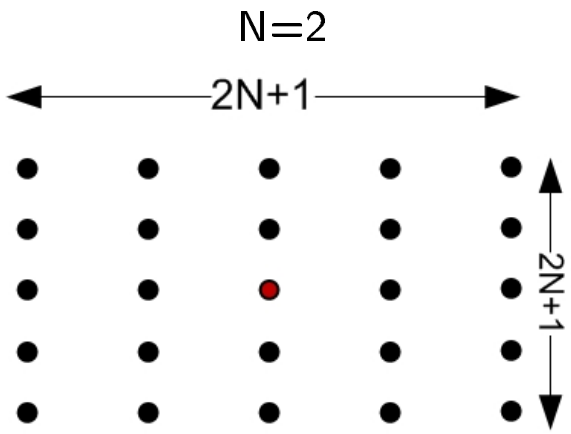
$$H = \begin{bmatrix} h(-2,-1) & h(-1,-1) & h(0,-1) & h(1,-1) & h(2,-1) \\ h(-2,0) & h(-1,0) & h(0,0) & h(1,0) & h(2,0) \\ h(-2,-1) & h(-1,-1) & h(0,-1) & h(1,-1) & h(2,-1) \end{bmatrix}$$

Definition

Example for a neigborhood of size $(2N+1) \times (2N+1)$:

$$IM[x,y] = \sum_{k=-N}^{N} \sum_{l=-N}^{N} h(k,l) im[x-k, y-l]$$

Number of multiplications per output point: $\mathcal{O}(N^2)$



Average filter

The most simple low-pass filter is the local averaging operation. The main effect of a low-pass filter is a blurring. The size of the kernel is $(2N+1) \times (2N+1)$:

$$h(k,l) = \begin{cases} \frac{1}{(2N+1)^2} & -N \leq k, l \leq -N \\ 0 & \text{Otherwise} \end{cases}$$

For $N = 1$, the convolution kernel is given by:

$$H = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Average filter

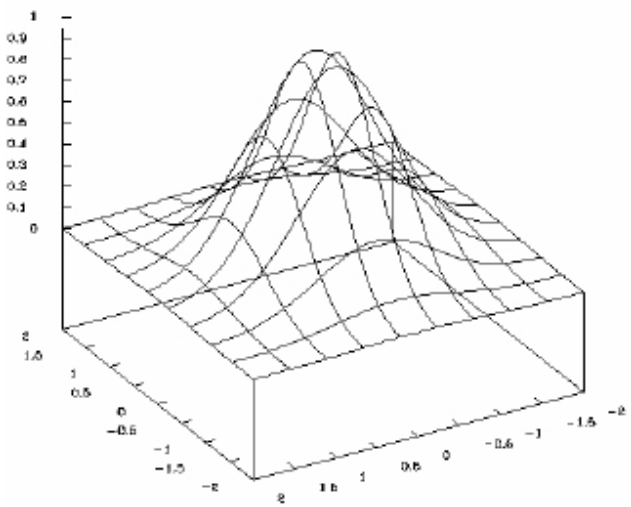
Three examples of averaging for different sizes of kernel. From the left-hand side to the right-side, $N = \{1, 3, 8\}$:

The amount of blur increases with the size of the kernel (the number of operation too $\mathcal{O}(N^2)$). In order to filter pixels located near the edges of the image, edge pixel values are replicated to give sufficient data (this is not the case here).

Gaussian filter

The kernel h is given by the following function

$$h(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$



Each pixel's new value is set to a weighted average of that pixel's neighborhood. The original pixel's value receives the heaviest weight (having the highest Gaussian value) and neighboring pixels receive smaller weights as their distance to the original pixel increases. This results in a blur that preserves boundaries and edges better than other, more uniform blurring filters.
Note that the filter support is truncated...

Gaussian filter

$$h(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Example of kernel:

- $\sigma = 0.84089642$, $N = 3$:
  Note that the center element (at [4, 4]) has the largest value, decreasing symmetrically as distance from the center increases.

$$\boldsymbol{H} = \begin{bmatrix} 0.00000067 & 0.00002292 & 0.00019117 & 0.00038771 & 0.00019117 & 0.00002292 & 0.00000067 \\ 0.00002292 & 0.00078633 & 0.00655965 & 0.01330373 & 0.00655965 & 0.00078633 & 0.00002292 \\ 0.00019117 & 0.00655965 & 0.05472157 & 0.11098164 & 0.05472157 & 0.00655965 & 0.00019117 \\ 0.00038771 & 0.01330373 & 0.11098164 & 0.22508352 & 0.11098164 & 0.01330373 & 0.00038771 \\ 0.00019117 & 0.00655965 & 0.05472157 & 0.11098164 & 0.05472157 & 0.00655965 & 0.00019117 \\ 0.00002292 & 0.00078633 & 0.00655965 & 0.01330373 & 0.00655965 & 0.00078633 & 0.00002292 \\ 0.00000067 & 0.00002292 & 0.00019117 & 0.00038771 & 0.00019117 & 0.00002292 & 0.00000067 \end{bmatrix}$$

Note that 0.22508352 (the central one) is 1177 times larger than 0.00019117 which is just outside 3σ.

Gaussian filter

$$h(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Example of kernel:

- $\sigma = 0.6,\ N = 1$:

$$H = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- $\sigma = 1,\ N = 2$:

$$H = \frac{1}{1444}\begin{bmatrix} 1 & 9 & 18 & 9 & 1 \\ 9 & 81 & 162 & 81 & 9 \\ 18 & 162 & 324 & 162 & 18 \\ 9 & 81 & 162 & 81 & 9 \\ 1 & 9 & 18 & 9 & 1 \end{bmatrix}$$

Gaussian filter

Example of Gaussian filtering with $\sigma = 2$:

Gaussian filter

Increasing sigma increases the smoothing

$\sigma = 1$

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

Other low pass filters

- 2D Pyramidal filter:

$$H = \frac{1}{81}\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

- Conic filter:

$$H = \frac{1}{25}\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 1 & 2 & 5 & 2 & 1 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

High-pass filter

The high-pass filtered image can be thought of as the original image minus the low pass filtered image.

$$IM[x,y] = im[x,y] - \sum_{k=-N}^{N} \sum_{l=-N}^{N} h(k,l) im[x-k, y-l]$$

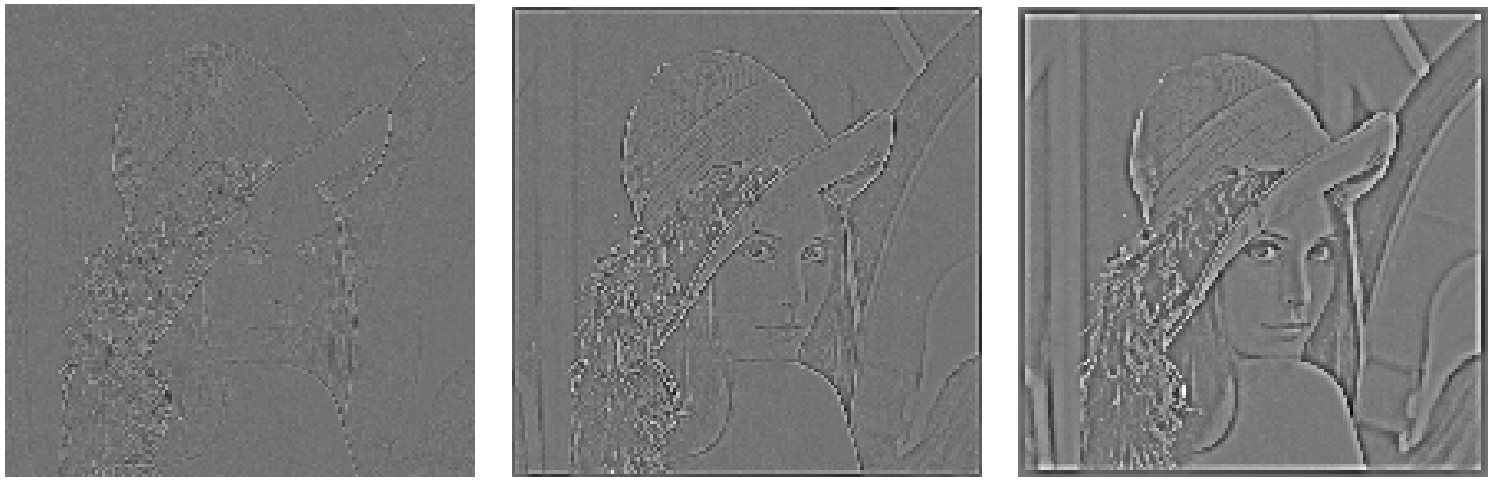
$$IM[x,y] = \sum_{k=-N}^{N} \sum_{l=-N}^{N} h(k,l) im[x-k, y-l]$$

with h the convolution kernel.

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

High-pass filter

Three examples of high-pass filtering for different sizes of kernel. From the left-hand side to the right-side, $N = \{1, 3, 8\}$:



When the kernel's size increases, the filtering is more important and then the result is less noisy.

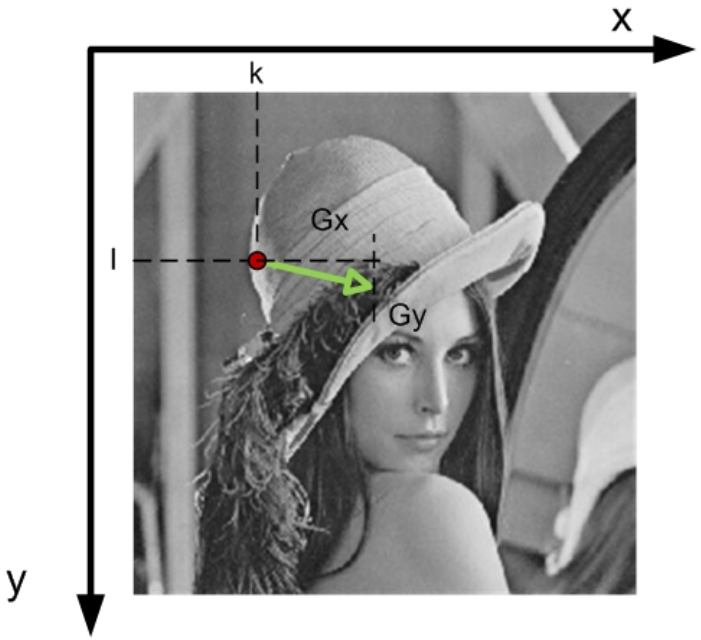
Differentiation filter

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Local variations of intensity are an important source of information in image processing. These local variations are gradient (it measures the rate of change of the function):

$$\nabla im[k, l] = \left[\frac{\partial im}{\partial x}[k, l], \frac{\partial im}{\partial y}[k, l] \right]$$

In the illustration below: $G_X = \frac{\partial im}{\partial x}[k, l]$ and $G_Y = \frac{\partial im}{\partial y}[k, l]$.



Differentiation filter

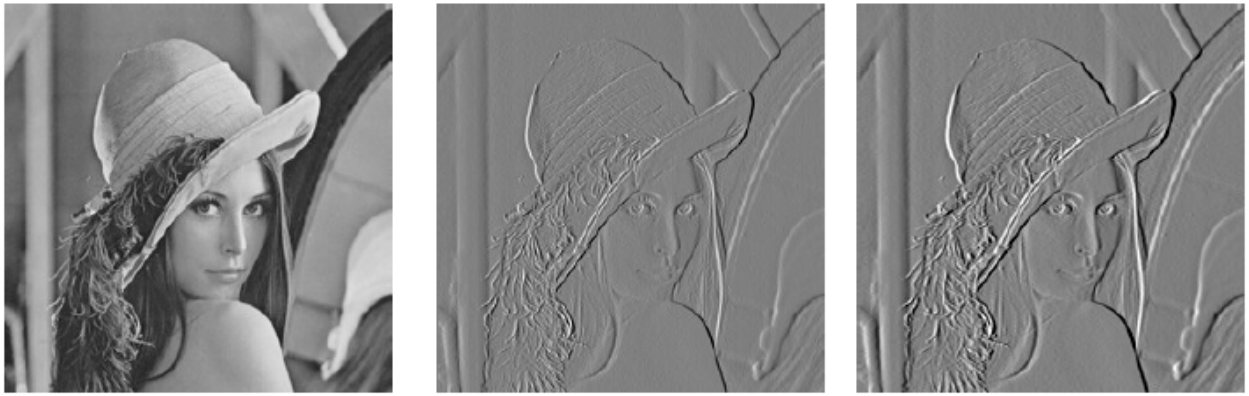
- $G_X = \frac{\partial im}{\partial x}[k,l]$: Kernel $= h_x = \begin{bmatrix} -1 & 1 \end{bmatrix}$

$$\frac{\partial im}{\partial x}[k,l] \approx im[k+1,l] - im[k,l]$$

- $G_Y = \frac{\partial im}{\partial y}[k,l]$ Kernel $= h_y = \begin{bmatrix} -1 & 1 \end{bmatrix}^T$

$$\frac{\partial im}{\partial y}[k,l] \approx im[k,l+1] - im[k,l]$$

However, most of the time, we use the following kernel $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$ (phase =zero). Below, from the left-hand side to the right: original, $\begin{bmatrix} -1 & 1 \end{bmatrix}$, $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$.



Differentiation filter

However, these filters are very sensitive to the noise. In order to enhance the robustness, these filters are combined with a blurring filter:

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

These kernel are the Sobel's kernel. The blurring filter is $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$.

$$\begin{aligned} IM_x[k,l] &= (im * h_x)[k,l] \\ IM_y[k,l] &= (im * h_y)[k,l] \end{aligned}$$

In the same vein, Prewitt and Robert's filters.

Differentiation filter

- Norm of the gradient: $\|\nabla IM[k,l]\|^2 = \sqrt{IM_x[k,l]^2 + IM_y[k,l]^2}$;
- Its orientation: $arg(\nabla IM[k,l]) = arctan\left(\frac{IM_y[k,l]}{IM_x[k,l]}\right)$

From left-hand side to the right: IM_x, IM_y and the norm.

Differentiation filter

The Laplacian of a picture is the second derivative:

$$\begin{aligned}\frac{\partial^2 im}{\partial^2 x}[k,l] &\approx im[k+1,l]+im[k-1,l]-2\times im[k,l]\\ \frac{\partial^2 im}{\partial^2 y}[k,l] &\approx im[k,l+1]+im[k,l-1]-2\times im[k,l]\end{aligned}$$

$$\begin{aligned}\nabla^2 im[k,l] &= \frac{\partial^2 im}{\partial^2 x}[k,l]+\frac{\partial^2 im}{\partial^2 y}[k,l]\\ \nabla^2 im[k,l] &\approx im[k+1,l]+im[k-1,l]+im[k,l+1]+im[k,l-1]-4\times im[k,l]\end{aligned}$$

For a 4-neighborhood, the kernel is given by

$$h = \begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$$

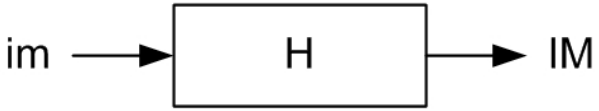
We can extent this kernel to compute the laplacian in all directions (8-neighborhood):

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

Differentiation filter

The second order derivatives have a stronger response to fine details (e.g. thin lines) than the first order derivatives.

Frequency domain filtering



where H is the convolution kernel.

$$IM[x,y] = (im * h)[x,y]$$

$$im_1[x,y] * im_2[x,y] \xrightarrow{\mathcal{F}} IM_1[u,v] \times IM_2[u,v]$$
$$im_1[x,y] \times im_2[x,y] \xrightarrow{\mathcal{F}} IM_1[u,v] * IM_2[u,v]$$

When the size of the kernel is large, it is better to apply the filter in the frequency domain.

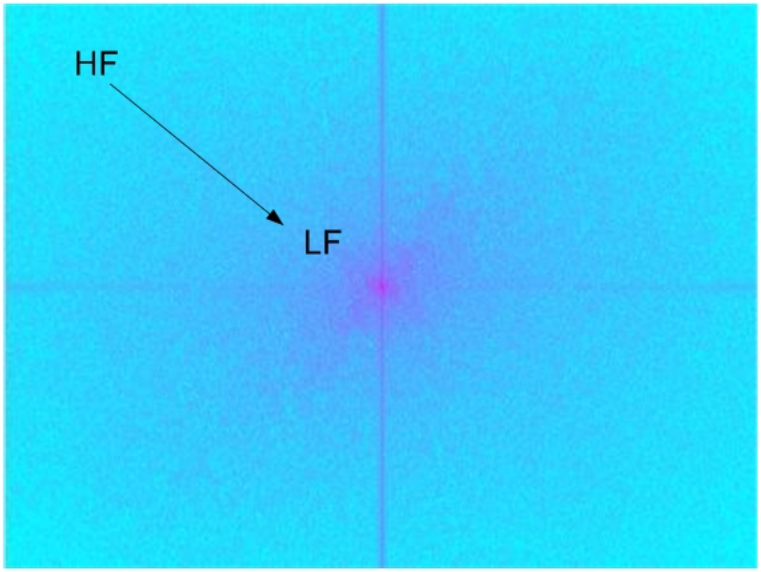
For more information:
Digital Image Processing, by R. C. Gonzalez and R. E. Woods, 3rd edition, Pearson Prentice Hall, 2008.

Frequency domain filtering

We can spatially filter an image by Fourier transforming and applying a frequency filter:

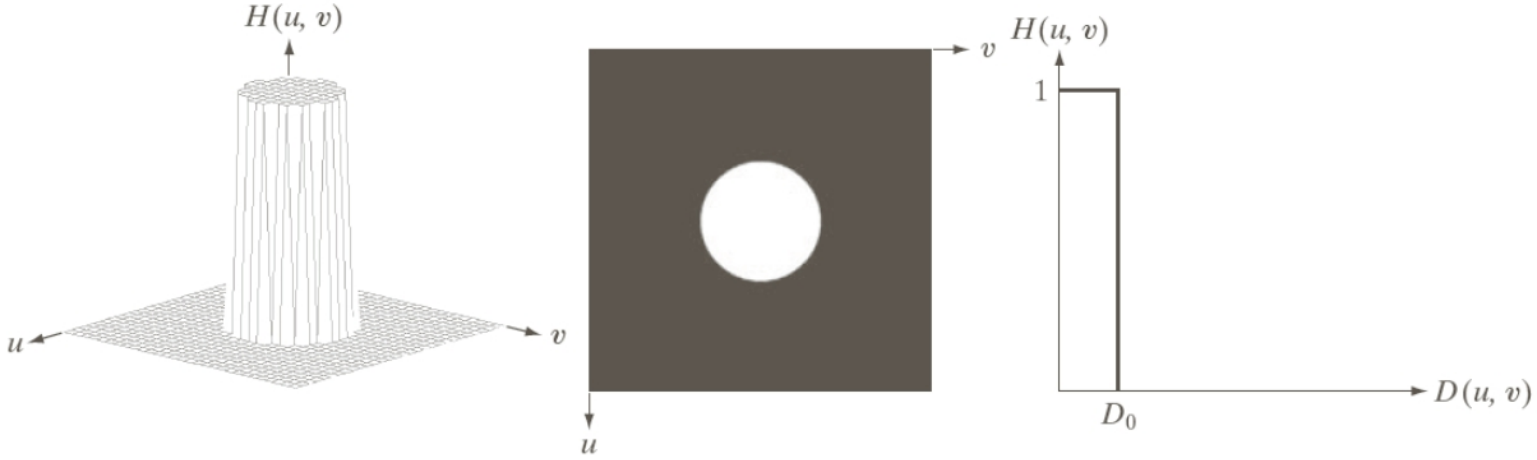
$$\begin{aligned} IM[x,y] &= im[x,y] * h[x,y] \\ I\tilde{M}[u,v] &= IM[u,v] \times H[u,v] \end{aligned}$$

where, $H[u,v]$ is the filter in the frequency function.



Ideal low pass filter

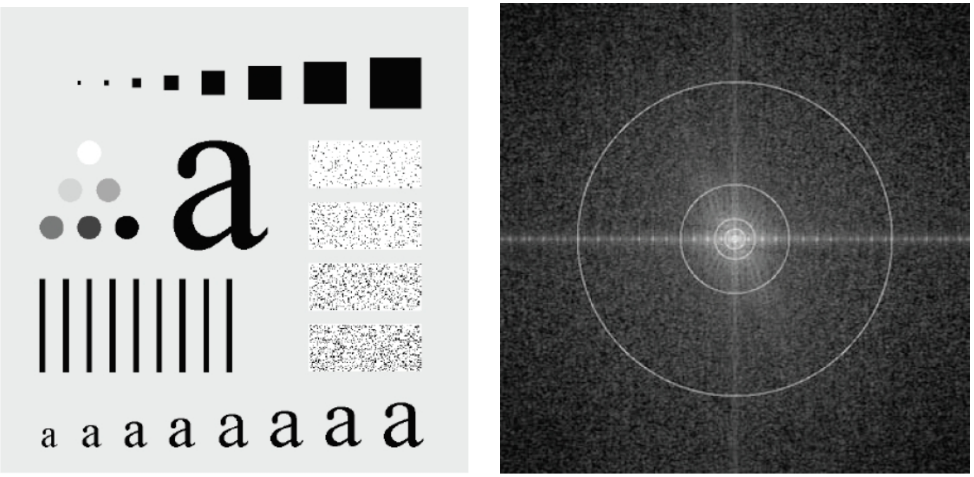
From the left-hand side to the right: Ideal low pass filter transfert function, filter displayed as an image, filter radial cross section.



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

With D the euclidean distance from the spectrum center $(\frac{N}{2}, \frac{N}{2})$.

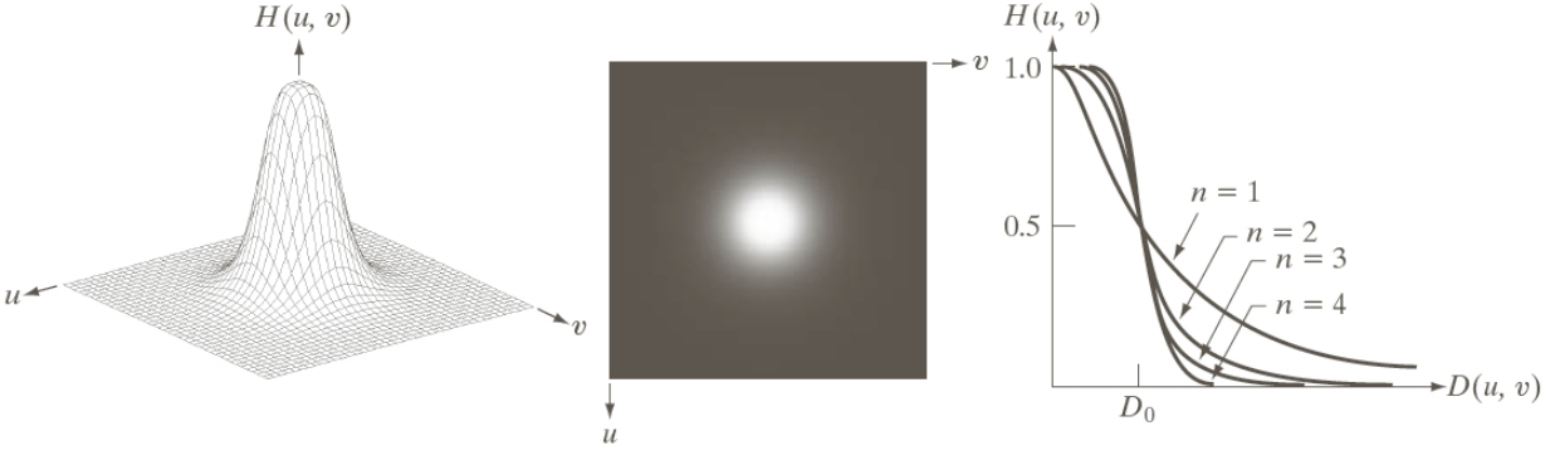
Ideal low pass filter



Low pass filtering:

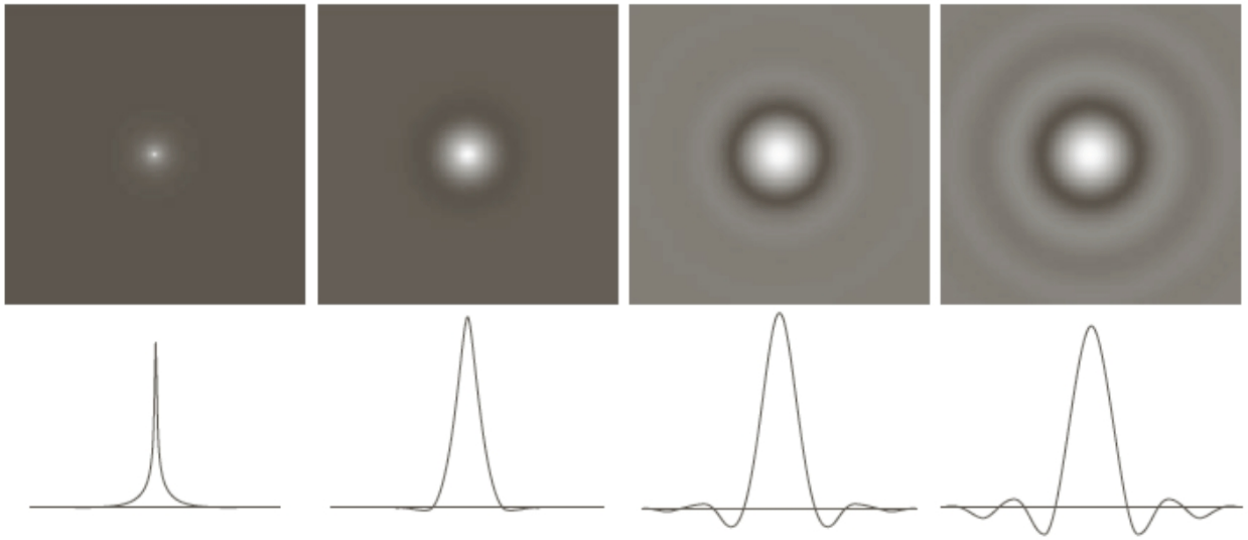
Ringing and blurring

Butterworth low pass filter



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

Butterworth low pass filter

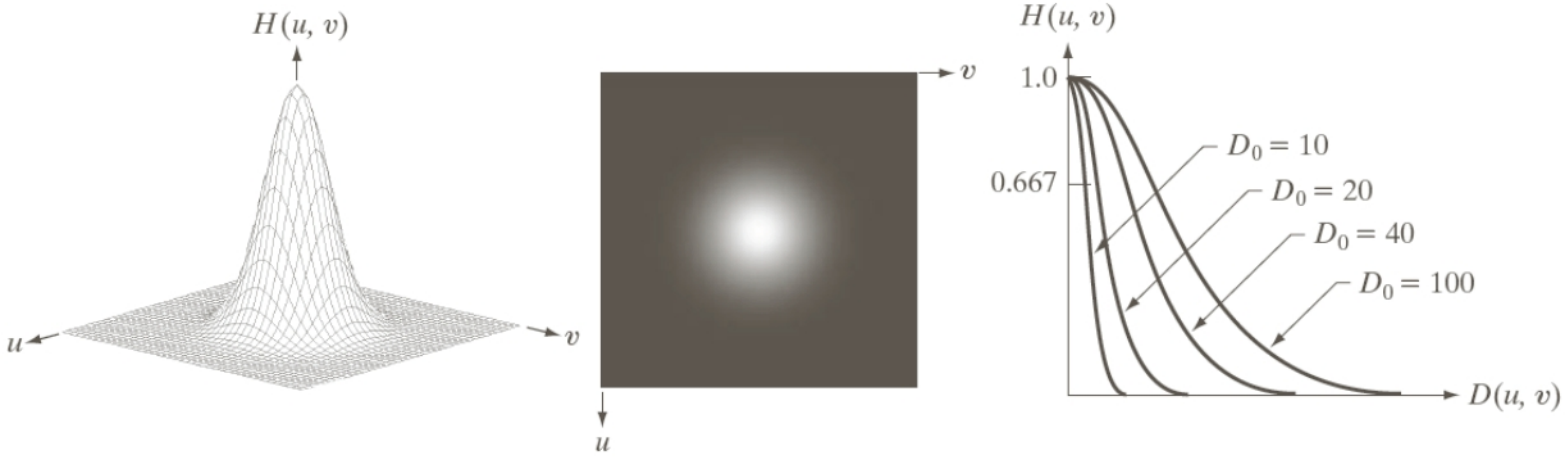


Top: spatial representation of the filter for different orders;
Bottom: intensity profiles through the center of the filters.

Butterworth low pass filtering:

Smooth transition in blurring, no ringing is present.

Gaussian low pass filter



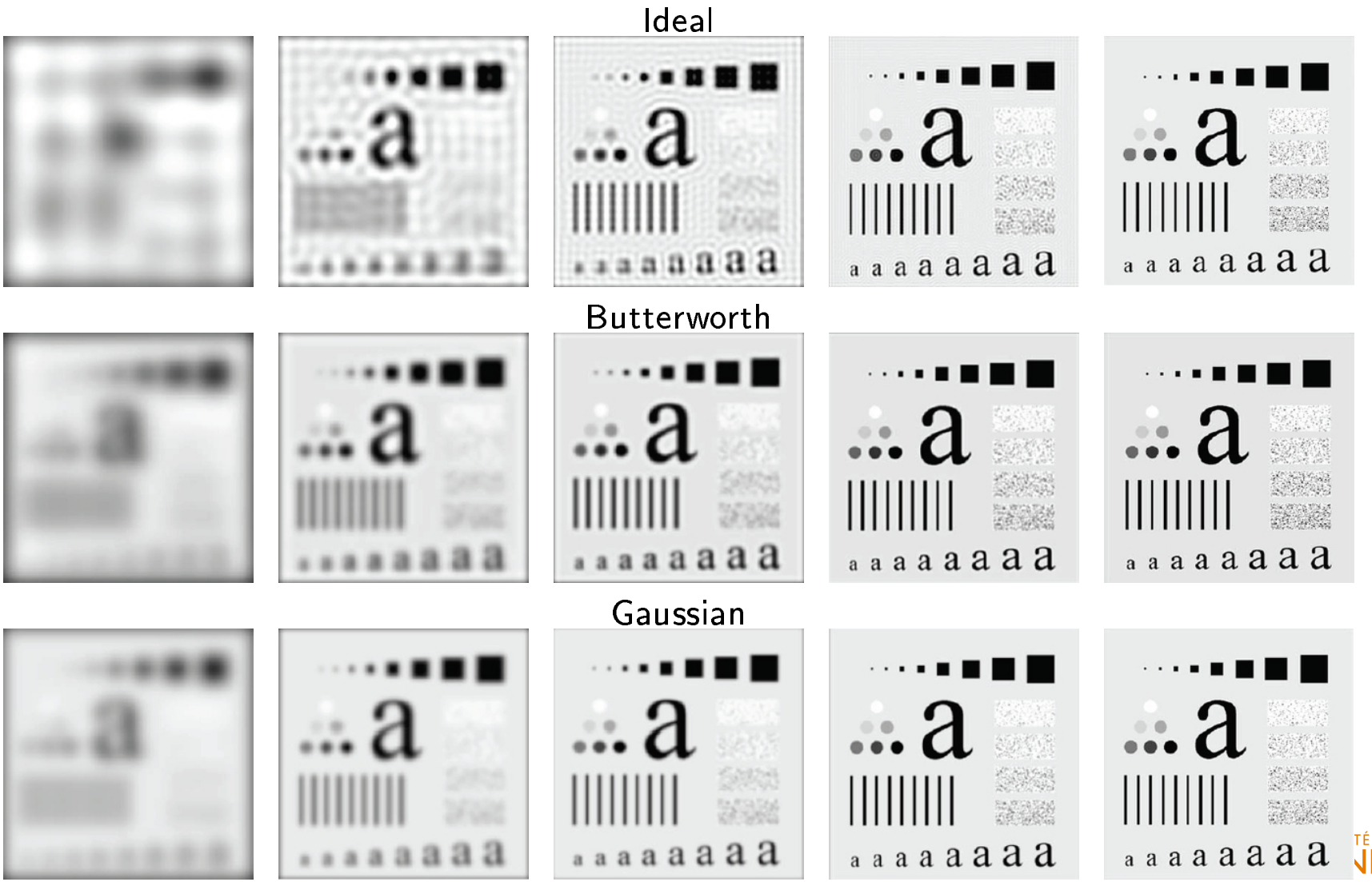
$$H(u,v) = exp\left(-\frac{D^2(u,v)}{2D_0^2}\right)$$

with $D_0 = \sigma$. Gaussian low pass filtering:

Smooth transition in blurring, no ringing is present.

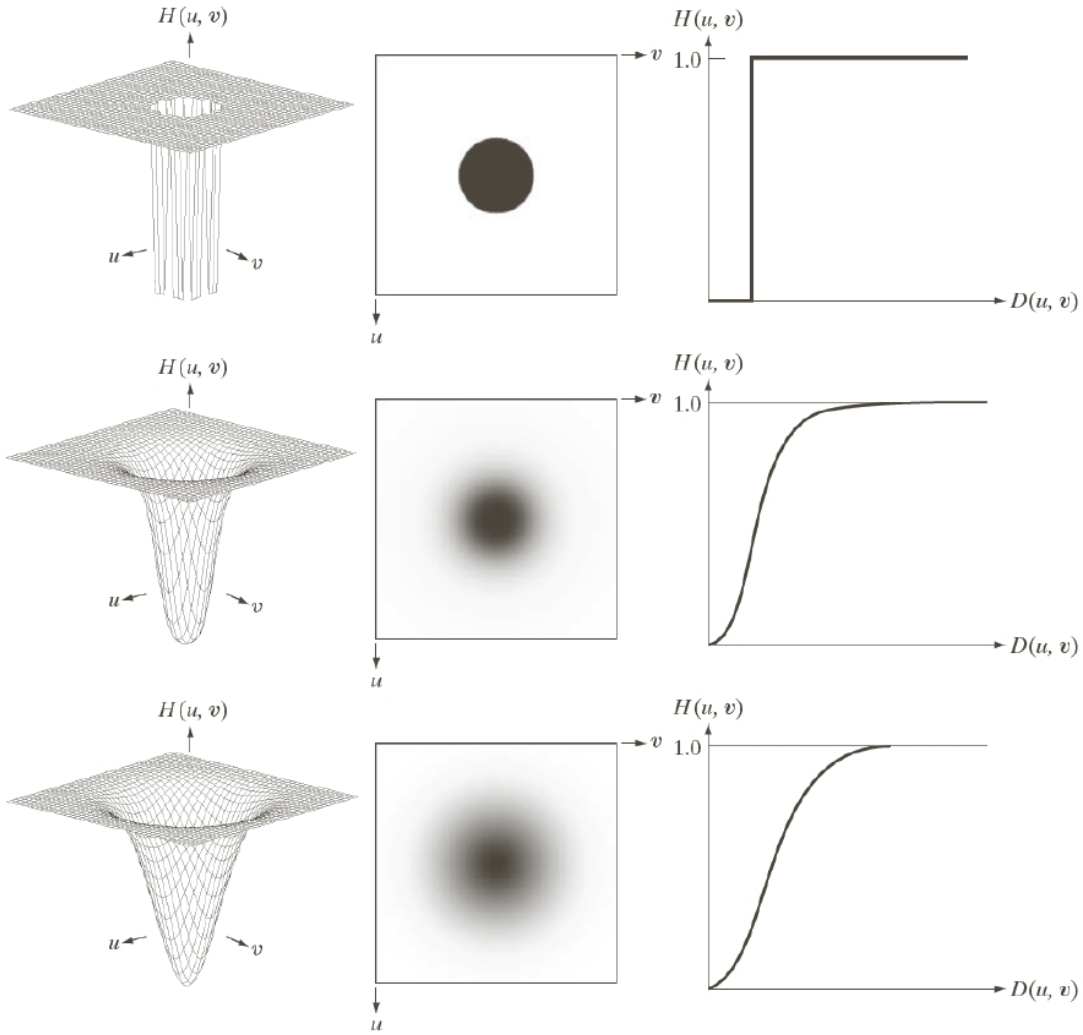
Low pass filter

Comparison between the ideal, the Butterworth and the gaussian low pass filter:

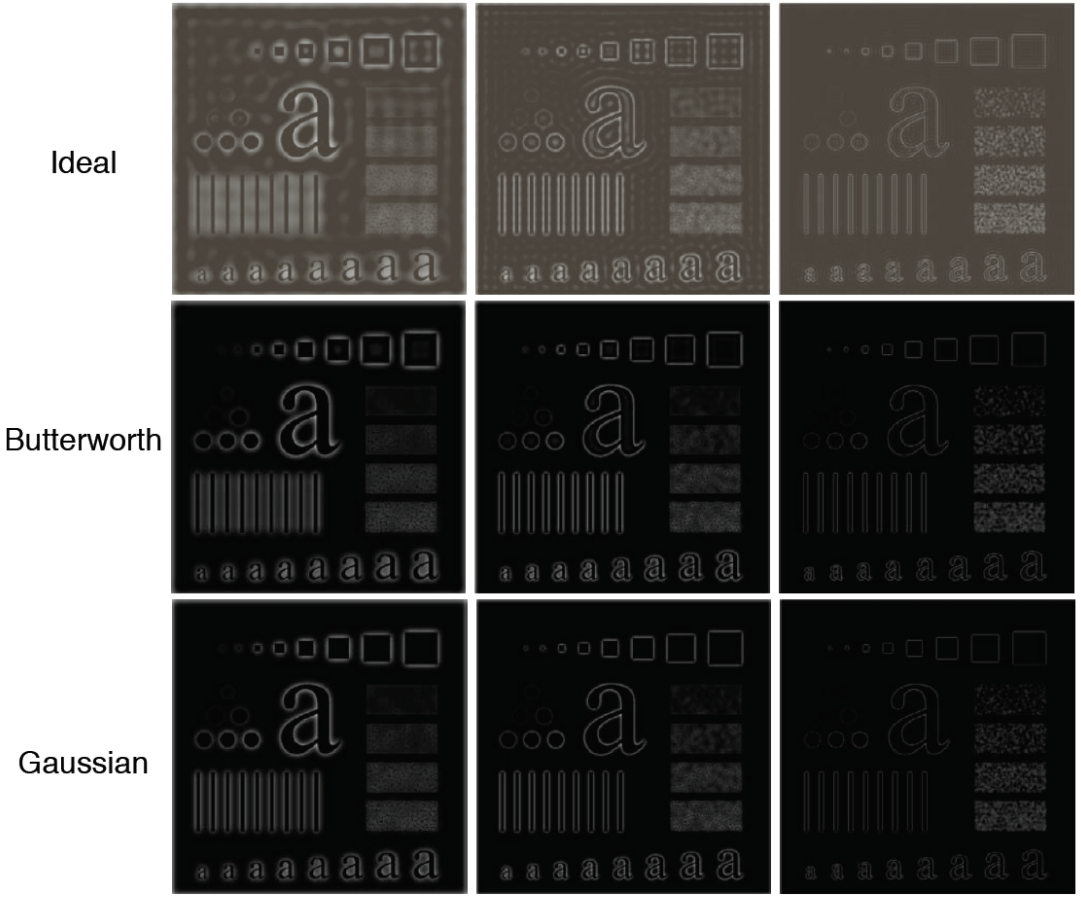


High pass filtering in the frequency domain

$$H_{HP}(u,v) = 1 - H_{LP}(u,v)$$



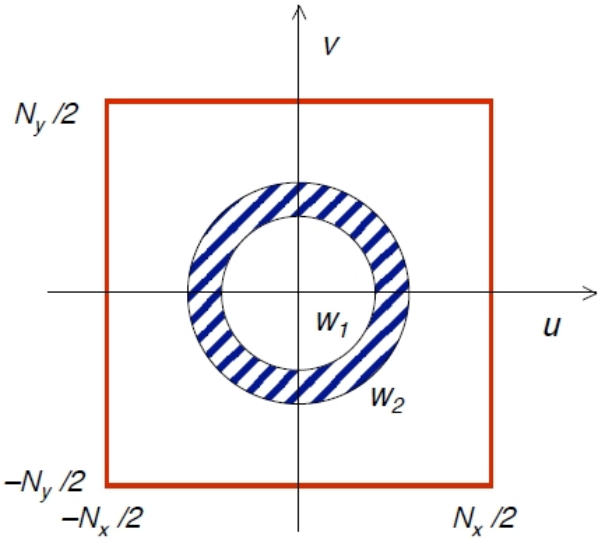
High pass filtering in the frequency domain

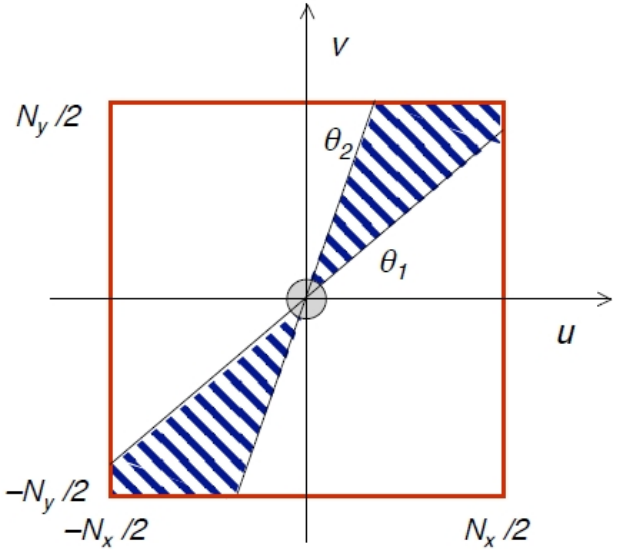


- Ideal high-pass filters enhance edges but suffer from ringing artefacts, just like Ideal LPF;
- smoother results with the two others.

Frequency domain filtering

Example of two band-pass filters:





Laplacian in the frequency domain

We remind (see previous lecture):

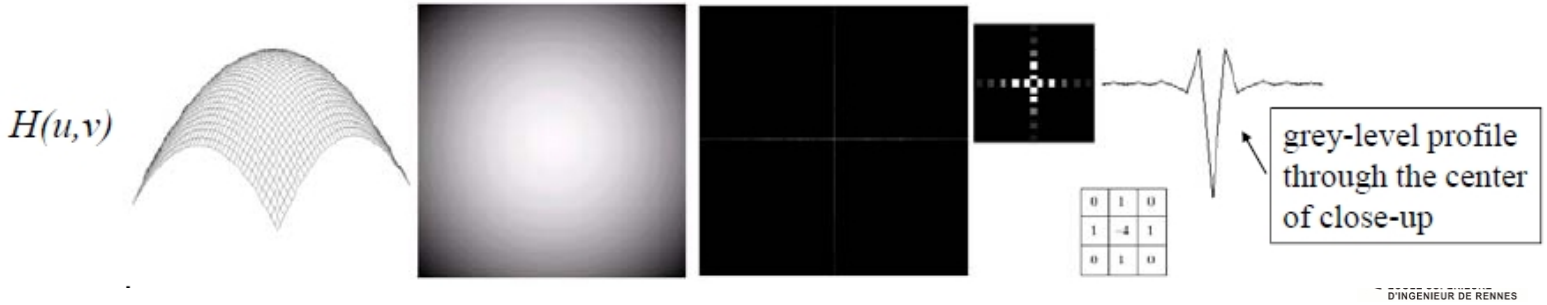
- $$\nabla^2 im[k,l] = \frac{\partial^2 im}{\partial^2 x}[k,l] + \frac{\partial^2 im}{\partial^2 y}[k,l]$$
- $\frac{d^n x(t)}{dt^n} \xrightarrow{\mathcal{F}} (j2\pi f)^n X(f)$.

From this, it follows that

$$\frac{\partial^2 im}{\partial^2 x}[k,l] + \frac{\partial^2 im}{\partial^2 y}[k,l] \xrightarrow{\mathcal{F}} -(u^2+v^2)IM[u,v]$$

The Laplacian filter is then implemented in the frequency domain by

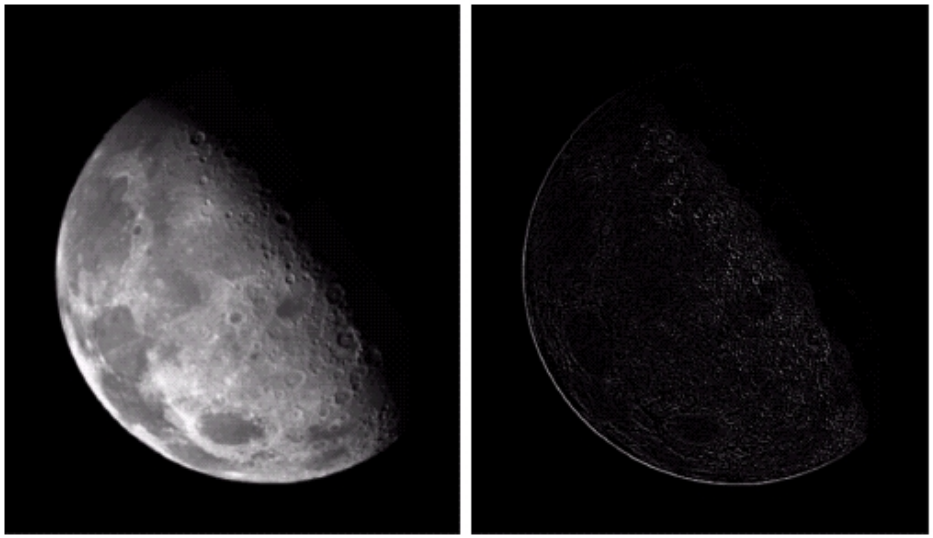
$$H(u,v) = -(u^2+v^2)$$



Laplacian in the frequency domain

Finally, to compute the Laplacian, we need :

1. to compute the Fourier transform of the picture;
2. to multiply the spectrum by $-(u^2 + v^2)$;
3. to compute the inverse Fourier transform.



The cortex transform

The cortex transform is first described by A. Watson as the modeling of the neural response of retinal cells to visual stimuli.

The cortex filter in the frequency domain is

$$cortex_{[b_i,\theta_i]}(\rho,\theta) \quad = \quad dom_{b_i} \times fan_{\theta_i}(\theta)$$

Where,

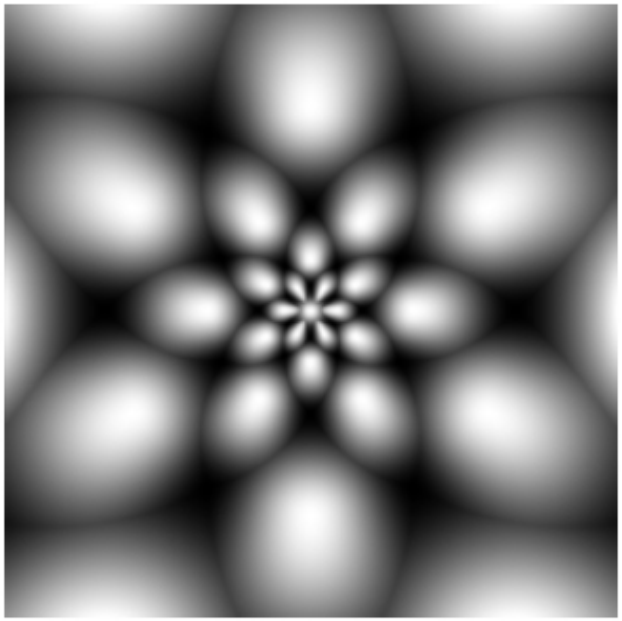
- b_i and θ_i represent the frequency band and the index of orientation, respectively;
- (ρ,θ) are polar coordinates.

The cortex transform decomposes the input image $im[x,y]$ into a set of subband images $B_{[b_i,\theta_i]}[k,l]$:

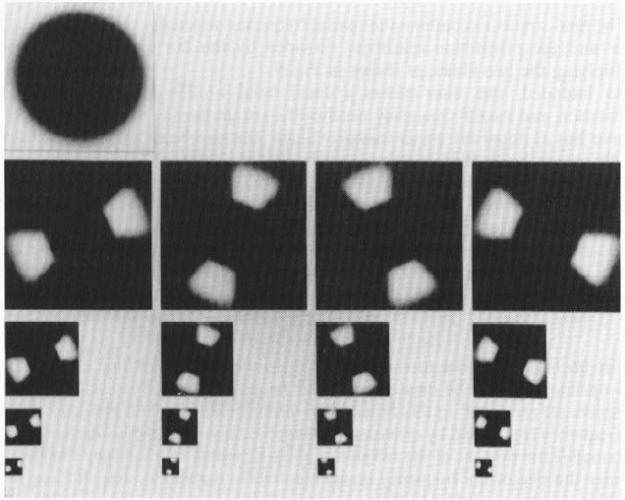
$$B_{[b_i,\theta_i]}[k,l] \quad = \quad \mathcal{F}^{-1}\left\{cortex_{[b_i,\theta_i]}(\rho,\theta) \times \mathcal{F}\left\{im[x,y]\right\}\right\}$$

The cortex transform

Frequency responses of several cortex filters (brightness represents gain for the given spatial frequency).



Complete set of cortex filters

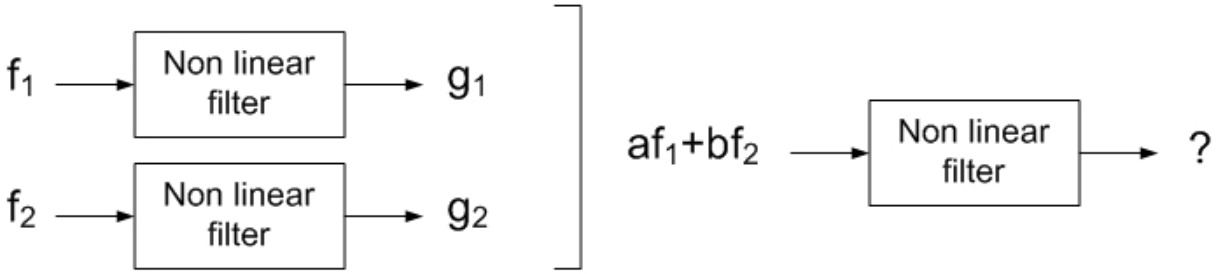


Introduction

Definition

The most important drawback of the linear filtering is that all pixels in the image are modified by the filtering process.

To overcome this problem, non linear filtering is used. It aims, for instance, to protect some parts of the picture having particular features (edges...) or to remove data without blurring the whole image (impluse noise).



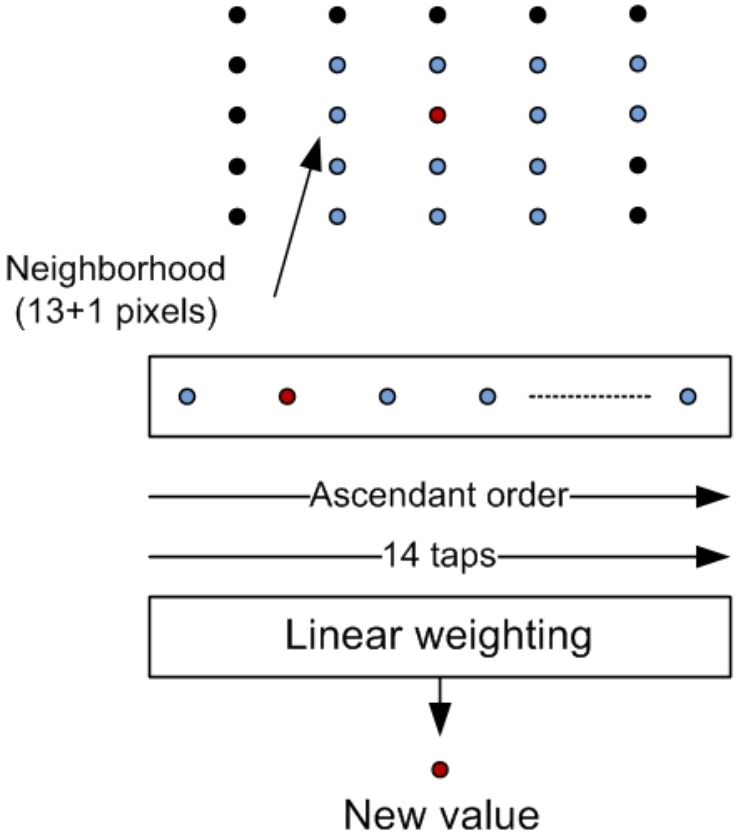
Three kinds of filters:

- Rank filtering;
- Adaptive linear filtering;
- Morphological operators.

Rank filtering

Rank filters are based on three steps:

1. Data selection, also called windowing;
2. Data ranking (in ascendant order);
3. 1D linear weighting of the ordered data list.



Special case of a generalized rank filter

If all weights of the linear filter are null, except one in the median position. This filter is called a median filter.

$$IM[x, y] = MED\left(im[x_i, y_i] | [x_i, y_i] \in V[x, y]\right)$$

$V[x, y]$ is the neighborhood (a set of N samples).

- if the size of the neighborhood is odd, the output value is the median value;
- if the size is even, the output value is the average of the two middles values.

The median filter is very efficient in filtering signals corrupted by impulsive noise but it is not very efficient in gaussian noise environment.

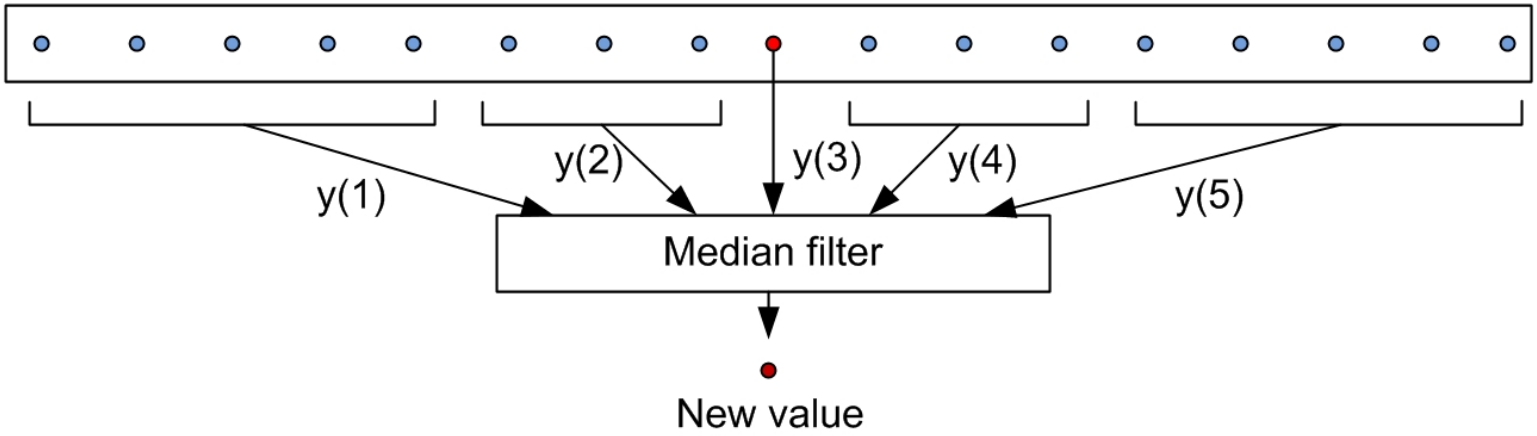
Special case of a generalized rank filter

However, when the number of the samples is large, the ordering procedure becomes cumbersome.

Idea: the median filter is taken over the outputs of several FIR substructures and the number of the substructures is much smaller than the number of the data samples inside the filter window.

$$IM[x, y] = MED\left(y(1), \ldots, y(m)\right)$$

where, m is linear FIR filters.



Homomorphic filtering

Homomorphic filtering is a generalized technique for signal and image processing, involving a nonlinear mapping to a different domain in which linear filter techniques are applied, followed by mapping back to the original domain.

In many case, we want to remove shading effects from an image. The objective is then

- to enhance high frequencies;
- to attenuate low frequencies (but fine details have to be preserved).

Consider the following model of image formation:

$$im[x,y] = \underbrace{i(x,y)}_{illumination} \times \underbrace{r(x,y)}_{reflection}$$

- The illumination component varies slowly and then affects low frequencies mostly;
- The reflection component varies faster and then affects the high frequencies mostly.

Homomorphic filtering

What is the solution to separate LF and HF?

$$im[x,y] = i[x,y] \times r[x,y]$$

Fourier transform

$$IM[u,v] = I[u,v] * R[u,v]$$

Due to the convolution in the frequency domain, LF and HF from $i[x,y]$ and $r[x,y]$ are mixed together...

Too complex to filter LF and HF in such condtion.

Homomorphic filtering

What is the solution to separate LF and HF? We can take the log !

$$\begin{aligned} im[x,y] &= i[x,y] \times r[x,y] \\ log(im[x,y]) &= log(i[x,y] \times r[x,y]) \\ log(im[x,y]) &= log(i[x,y]) + log(r[x,y]) \end{aligned}$$

1. Take the log and apply the Fourier tansform to the new signal:

$$\begin{aligned} log(im[x,y]) &= log(i[x,y]) + log(r[x,y]) \\ \mathcal{F}\left(log(im[x,y])\right) &= \mathcal{F}\left(log(i[x,y])\right) + \mathcal{F}\left(log(r[x,y])\right) \\ Z[u,v] &= I_{log}[u,v] + R_{log}[u,v] \end{aligned}$$

2. Filtering in the frequency domain: $H[u,v]$

$$Z[u,v] \times H[u,v] = I_{log}[u,v] \times H[u,v] + R_{log}[u,v] \times H[u,v]$$

3. Take the inverse Fourier transform and apply the exponential function:

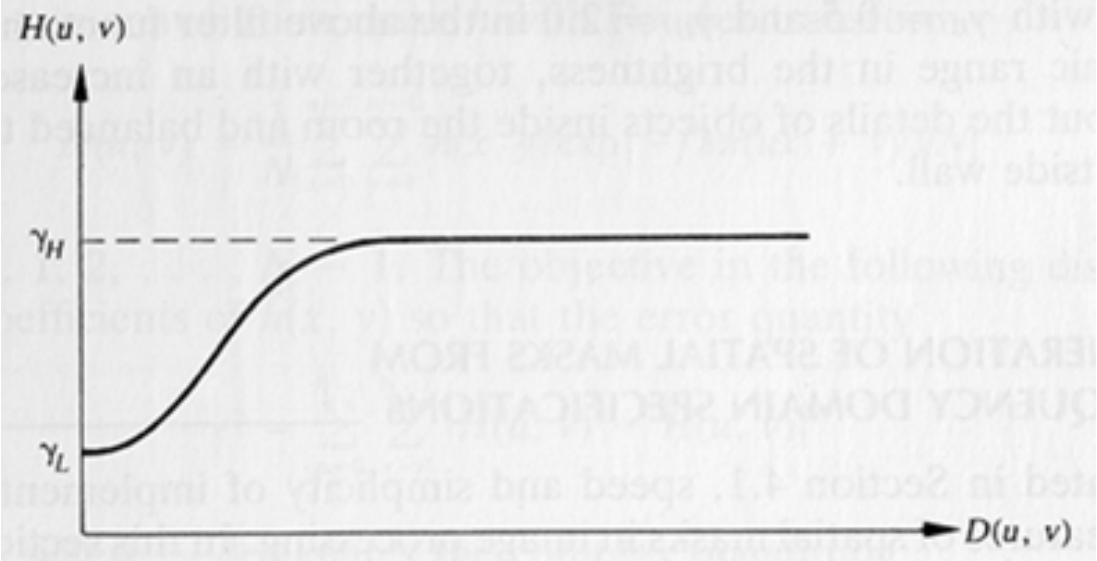
$$\begin{aligned} \mathcal{F}^{-1}\left(Z[u,v] \times H[u,v]\right) &= \mathcal{F}^{-1}\left(I_{log}[u,v] \times H[u,v]\right) + \mathcal{F}^{-1}\left(R_{log}[u,v] \times H[u,v]\right) \\ z[x,y] &= \widetilde{i}[x,y] + \widetilde{r}[x,y] \\ exp\left(z[x,y]\right) &= exp\left(\widetilde{i}[x,y]\right) + exp\left(\widetilde{r}[x,y]\right) \end{aligned}$$

Homomorphic filtering

$$H[u,v] = \gamma_L + \frac{\gamma_H}{1 + \left[\frac{D_0}{\sqrt{u^2+v^2}}\right]^{2n}}$$

with,

- γ_L a parameter that affects low frequencies;
- γ_H a parameter that affects high frequencies;



Adaptive filtering

An adaptive filter is a filter that self-adjusts its transfer function according to an optimizing algorithm.

The goal is still to smooth the signal. However, we want to preserve edges...

- Filtering by pixel grouping;
- Conditional mean, Bilateral filtering and mean shift filter;
- Diffusion.

Conditional mean

$$\underbrace{IM[x,y]}_{\text{Output}} = \sum_{k \in V(x,y)} \sum_{l \in V(x,y)} \underbrace{h(k,l)}_{\text{Filter coeff.}} \underbrace{im[x-k, y-l]}_{\text{Input}}$$

Principle: pixels in a neighbourhood are averaged only if they differ from the central pixel by less than a given threshold.

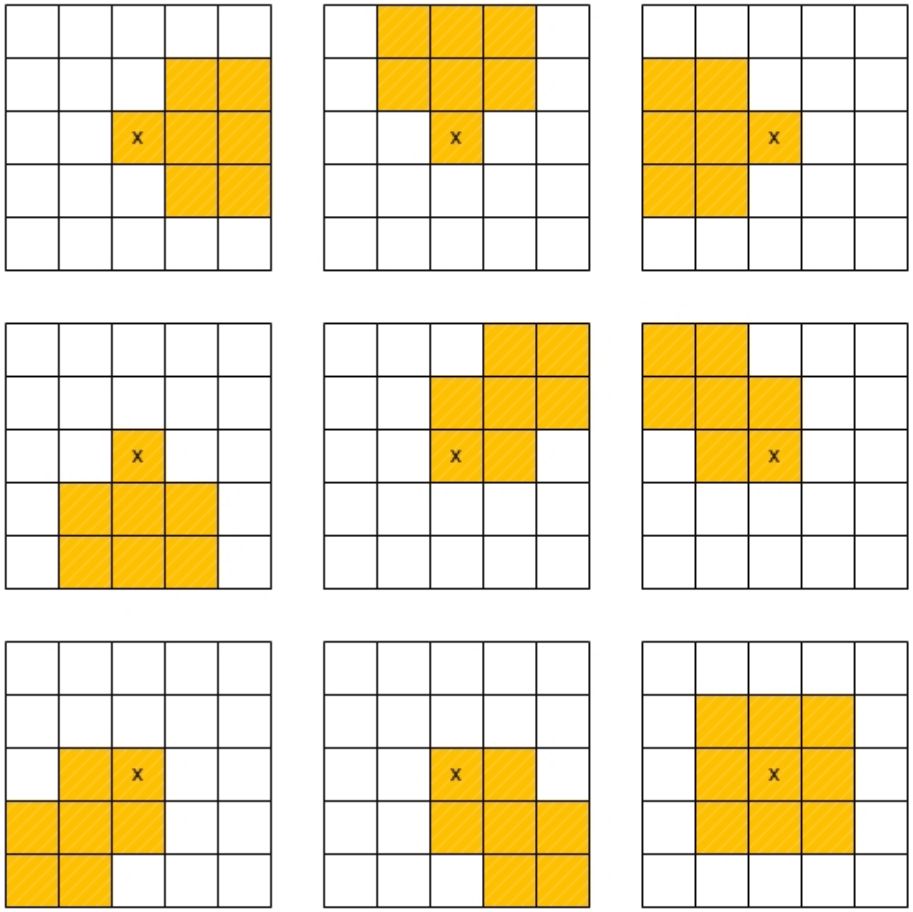
$$h(k,l) = \begin{cases} 1 & \text{if } |IM[x-k, y-k] - IM[k,l]| < TH \\ 0 & \text{Otherwise.} \end{cases}$$

Example with a neighbourhood equal $(2 \times 3 + 1)(2 \times 3 + 1), TH = 32$:

Anisotropic Kuwahara filtering

Method proposed by Kuwahara and adapted by Nagao in 1980.

Example for a window 5×5:



- Selection of the sub-domain that has the minimum variance (9 windows for Nagao);
- Replace the value of the central pixel by the average value of the sub-domain having the minimum variance.

Anisotropic Kuwahara filtering

Bilateral filtering

The idea is to use a weighted filtering but with an outlier rejection.
Pixels that are very different in intensity from the central pixel are weighted less even though they may be in close proximity to the central pixel.

This is applied as two Gaussian filters at a localized pixel neighborhood:

$$IM[x,y] = \frac{1}{\sum_{k,l} c_{[x,y]}[k,l] s_{[x,y]}[k,l]} \sum_{k,l} im[k,l] \underbrace{c_{[x,y]}[k,l] s_{[x,y]}[k,l]}_{h[k,l]}$$

- One in the spatial domain named the domain filter:

$$c_{[x,y]}[k,l] = exp\left(-\frac{d([x,y],[k,l])}{2\sigma_d}\right)$$

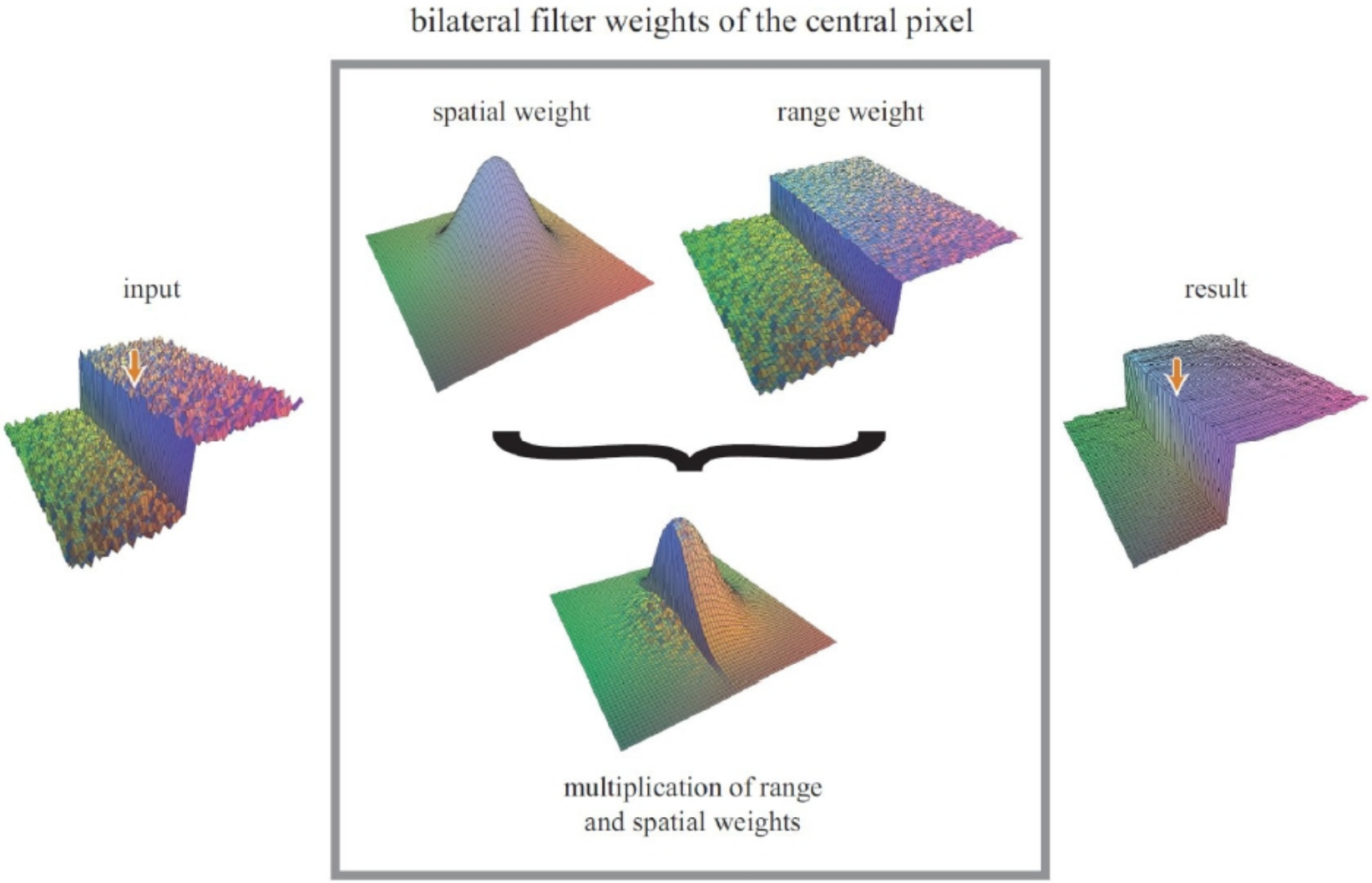
  where d is Euclidean distance.

- One in the intensity space named the range filter:

$$s_{[x,y]}[k,l] = exp\left(-\frac{\phi(im[x,y], im[k,l])}{2\sigma_r}\right)$$

  where ϕ is a suitable measure of distance in intensity space.

Bilateral filtering



Bilateral filtering

Low contrast texture has been removed and edges are well preserved.

Conclusion

Conclusion

- Image Transformation: global to point; point to point; local to point;
- Histogram;
- Linear Filtering;
- Non linear filtering.