4 Transforms

4.2: 2D Linear Filtering

Transforms

4.2

1. 2D Discrete Fourier Transform (2D-DCT)
   - Basic properties
   - Basic signal transforms
2. **2D Linear Filtering**
   - Filter design in the spatial domain
   - Filter design in the frequency domain
3. **Other transforms**
   - Discrete Cosine Transform (DCT)
   - Karhunen-Loeve Transform (KLT)
4. **Short-Term Fourier Transform (STFT)**
   - STFT as a filter bank
   - Spectogram: Time-frequency analysis

Unit Structure

4.2

1. **Introduction:**
   - Image convolution
   - Image padding
2. **Low-pass filters designed in the spatial domain:**
   - Example of application: Noise removal
   - Trade-off noise removal vs blurring
3. **High-pass filters designed in the spatial domain:**
   - Example of application: Contour detection
4. **Filters designed in the frequency domain:**
   - Ringing effect
5. **Summary and Conclusions**

Introduction

4.2

- Images can be modeled as **linear combinations** of simpler functions; typically:
  - **Impulse functions**, as in the canonical representation.
  - **Complex exponentials**, as in the Fourier representation.
  - …
- The Fourier representation (and its underlying space/frequency model) is a useful tool to define a natural set of operations such as **Linear Space-Invariant (LSI)** operators.
  - LSI operators linearly combine the pixel values in a given neighborhood of the pixel being processed (**impulse response** or **convolution mask**).
- Linear Space-Invariant (**LSI**) operators can be defined:
  - In the **original (space) domain**, through a **convolution**.
  - In the transformed (frequency) domain, through a **product**.

Image convolution

4.2

A Linear Space-Invariant operator is characterized by its impulse response $h[m, n]$:

$$x[m,n] \longrightarrow \boxed{h[m,n]} \longrightarrow y[m,n]$$

and the output can be computed by **convolution**:

$$\boxed{y[m,n] = \sum_{m'} \sum_{n'} x[m',n'] h[m-m',n-n'] = x[m,n] * h[m,n]}$$

$$\boxed{y[m,n] = \sum_{m'} \sum_{n'} x[m-m',n-n'] h[m',n'] = h[m,n] * x[m,n]}$$

In the frequency domain, the output transform is defined by the **product** of the input transform and the **frequency response** $H[k, l]$ of the system.

$$\tilde{y}[m,n] \leftrightarrow Y[k,l] = X[k,l] \cdot H[k,l]$$
$$\text{if } y[m,n] = x[m,n] * h[m,n]$$

Image convolution

For each pixel position (m, n):

- A **2D-mirrored version** of $h[m', n']$ ($h[-m', -n']$) is shifted at the pixel position (m, n) ($h[m - m', n - n']$)
- The output $y[m, n]$ is the sum of the pixel values included in the neighborhood defined by the filter and weighted by the co-located filter values ($h[m, n]$ is the **filter impulse response**)

$$y[m,n] = \sum_{m'} \sum_{n'} x[m',n'] h[m-m',n-n']$$

- A **similar interpretation** where the input signal is shifted is also possible

$$y[m,n] = \sum_{m'} \sum_{n'} x[m-m',n-n'] h[m',n']$$

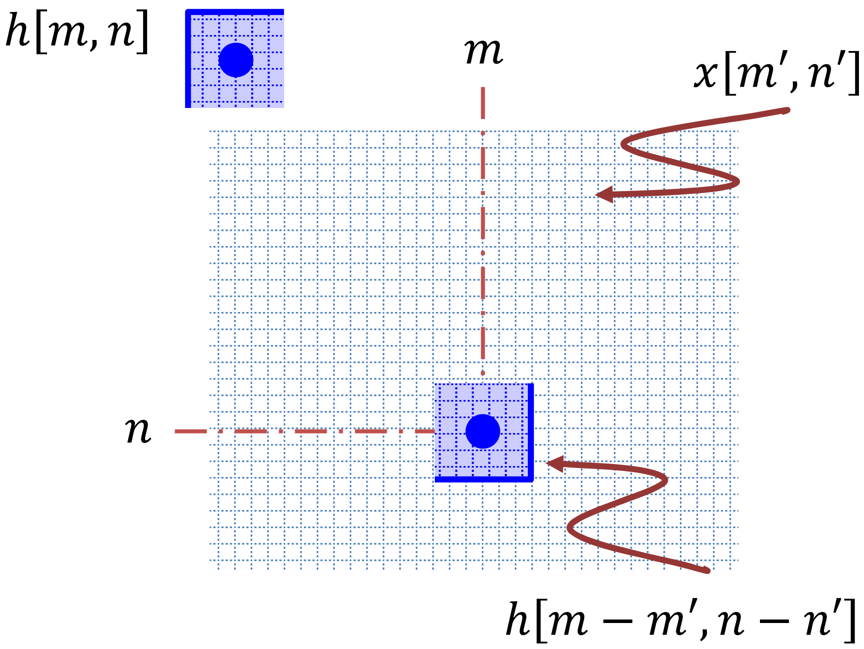
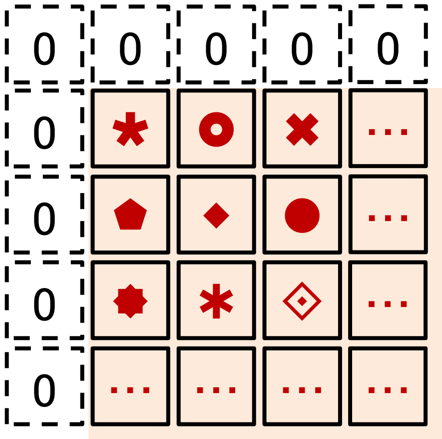


Image padding

The **computation of the output values close to the image border** requires the use of pixel values outside the input image support.

These values are commonly introduced by:

- **Zero padding**: Including a frame of zeros in the input signal:
  - It may introduce discontinuities in the signal



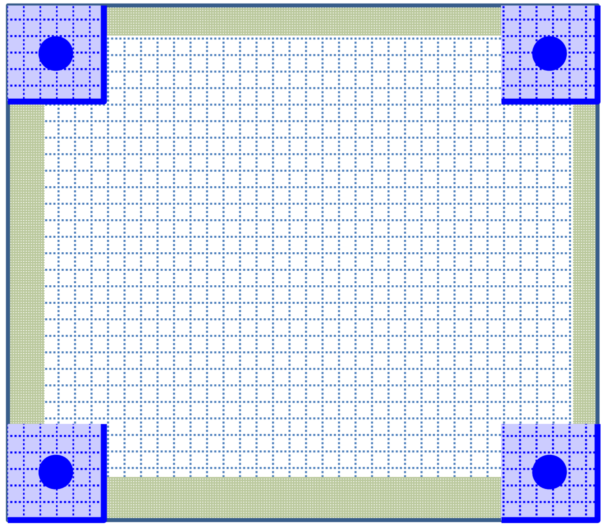


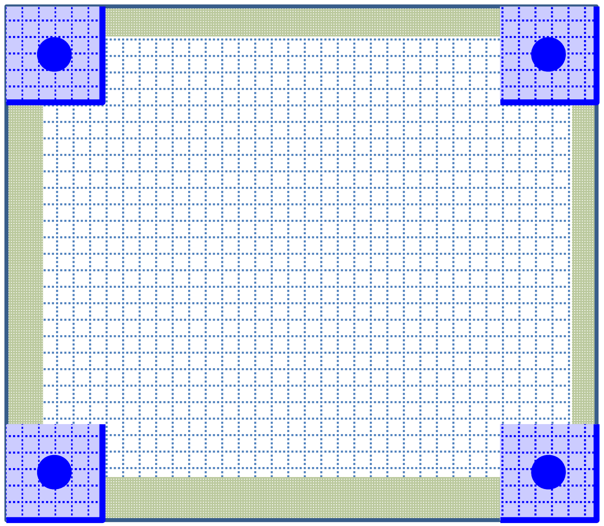
Image padding

4.2

The **computation of the output values close to the image border** requires the use of pixel values outside the input image support.

These values are commonly introduced by:

- **Mirroring**: Including pixel values that are a symmetric mirroring of the input:
    - It generates a smooth transition



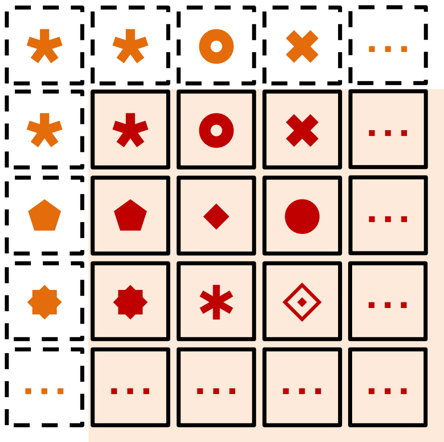
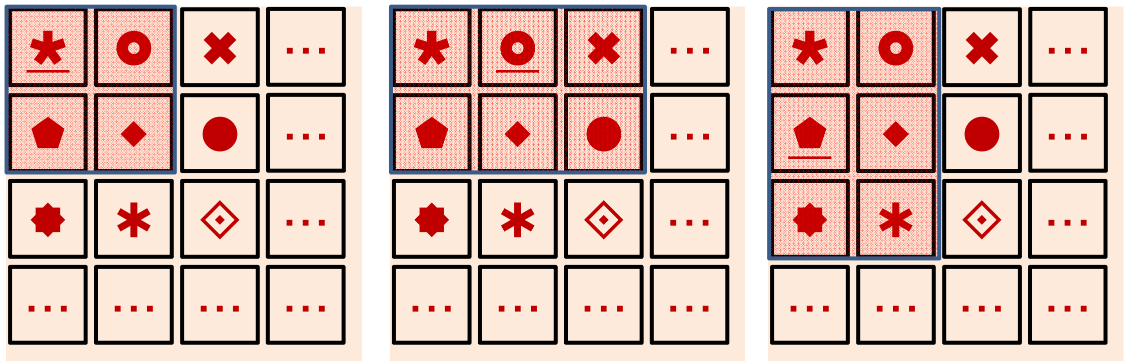


Image padding

The **computation of the output values close to the image border** requires the use of pixel values outside the input image support.

These values are commonly introduced by:

- **Translation variant filter:** To only use known pixel values:
  - Example with a 3x3 filter



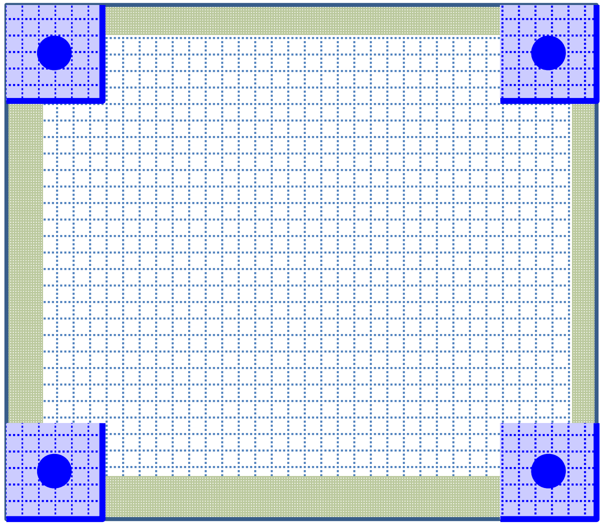
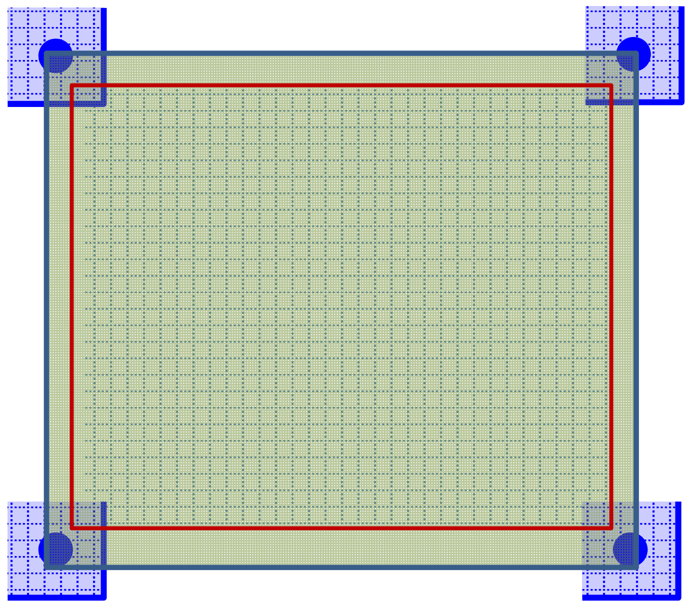


Image convolution and padding

- The result of the convolution of an image of size MxN and a filter with impulse response of size H_1xH_2 is an image of size $(M + H_1 - 1)x(N + H_2 - 1)$

- Nevertheless, in almost all practical cases, the output image support is **finally restricted** to the original image size.



$y[m, n]$

Separability

4.2

- In some cases, the 2D impulse response can be defined as the product of two 1D impulse responses (**separability**):
  - Separability allows **faster implementations**

$$\boxed{h[m,n] = h_1[m]h_2[n]}$$

$$y[m,n] = \sum_{m'}\sum_{n'} x[m',n']h[m-m',n-n'] = \sum_{m'}\sum_{n'} x[m',n']h_1[m-m']h_2[n-n']$$

$$y[m,n] = \sum_{n'} h_2[n-n'] \sum_{m'} x[m',n']h_1[m-m'] = x[m,n] *_{rows}^{1D} h_1[m] *_{cols}^{1D} h_2[n]$$

- The frequency response of a 2D separable filter can be defined as **the product of the two 1D frequency responses**:

$$\boxed{H[k,l] = H_1[k]H_2[l]}$$

Filters designed in the spatial domain

4.2

- Filters are defined in the spatial domain when their **size** (area of support) is relatively **small**.
  - In other cases, filters are usually defined in the frequency domain.
- Filters are commonly **square, odd in size and symmetric** (3x3, 5x5, …)
- Two different types of filters are going to be studied:
  - **Low-pass filters**, that perform a spatial average, and their application to noise removal.
  - **High-pass filters**, that perform an estimation of the derivative, and their application to contour detection.

Unit Structure

4.2

1. **Introduction:**
   - Image convolution
   - Image padding
2. **Low-pass filters designed in the spatial domain:**
   - Example of application: Noise removal
   - Trade-off noise removal vs blurring
3. **High-pass filters designed in the spatial domain:**
   - Example of application: Contour detection
4. **Filters designed in the frequency domain:**
   - Ringing effect
5. **Summary and Conclusions**

Low pass filters for noise reduction

4.2

We are going to study low-pass filters in the context of **noise reduction**.

In image processing, there are **three main types of noise**:

- **Gaussian noise**: It can appear in the sensors of a CCD camera.

$$p(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(n-\mu)^2}{\sigma^2}}$$

- **Uniform noise**: It is produced by the quantization process.

$$p(n) = \begin{cases} \frac{1}{b-a} & if \;\; a \le n \le b \\ 0 & otherwise \end{cases}$$

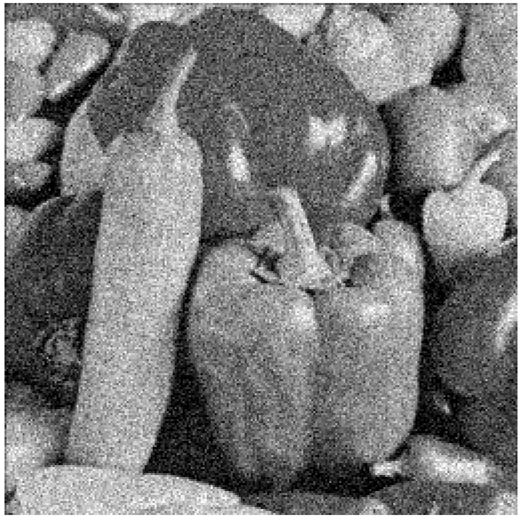
- **Salt and pepper noise**: Typical of error transmissions or damaged CCD sensors.

$$\tilde{i}(m,n) = \begin{cases} s & \text{withprobability}\, p \\ i(m,n) & \text{withprobability}\; 1-p \end{cases}$$

s is a random variable that takes values in {black (0), white (255)}

Noise examples

Original image ▶



Gaussian noise: (μ=0, σ^2=1)

Uniform noise

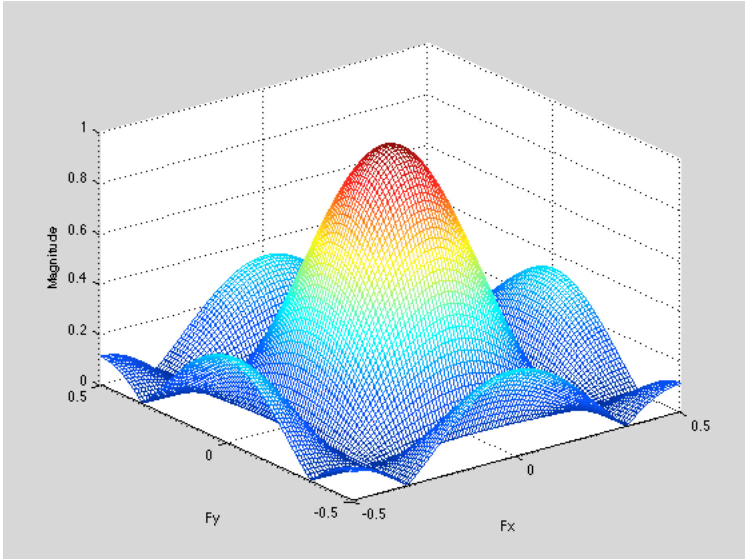
Salt and pepper noise (p=0.08)

Separability of the average filter

4.2

- The average filter is separable and, therefore, it allows for another type of analysis:

$$h[m,n]=\begin{bmatrix}1/9 & 1/9 & 1/9\\ 1/9 & \underline{1/9} & 1/9\\ 1/9 & 1/9 & 1/9\end{bmatrix}=h_1[m]h_2[n]$$



$$h_1[m]=\frac{1}{3}\Pi_3[m] \quad h_2[n]=\frac{1}{3}\Pi_3[n]$$

$$H[F_1,F_2]=H_1[F_1]H_2[F_2] \Rightarrow H[F_1,F_2]=\frac{1}{9}\frac{\sin[3\pi F_1]}{\sin[\pi F_1]}\frac{\sin[3\pi F_2]}{\sin[\pi F_2]}$$

Separability of the average filter

4.2

- The average filter is separable and it allows for another type of analysis:

$$H[F_1, F_2] = \frac{1}{9}\frac{\sin[3\pi F_1]}{\sin[\pi F_1]}\frac{\sin[3\pi F_2]}{\sin[\pi F_2]}$$

$$H_1(F) = \sum_{m=-\infty}^{\infty} h[m]\, e^{-i 2\pi F_1 m} = \frac{1}{3}\sum_{m=-1}^{1} e^{-i 2\pi F_1 m} =$$

$$= \frac{1}{3}\left[e^{i 2\pi F_1} + 1 + e^{-i 2\pi F_1}\right] = \frac{e^{i 2\pi F_1}}{3}\left[1 + e^{-i 2\pi F_1} + e^{-i 2\pi 2 F_1}\right] =$$

$$= \left[\sum_{n=0}^{P-1} a^n = \frac{a^0 - a^P}{1-a} \quad \forall a \neq 1\right] = \frac{e^{i 2\pi F_1}}{3} \cdot \frac{1 - e^{-i 2\pi 3 F_1}}{1 - e^{-i 2\pi F_1}} =$$

$$= \frac{e^{i 2\pi F_1}}{3} \cdot \frac{e^{-i 3\pi F_1}}{e^{-i \pi F_1}} \left[\frac{e^{i 3\pi F_1} - e^{-i 3\pi F_1}}{e^{i\pi F_1} - e^{-i\pi F_1}}\right] = \frac{1}{3}\frac{\text{sen}\, 3\pi F_1}{\text{sen}\, \pi F_1}$$

Spatial averaging filters

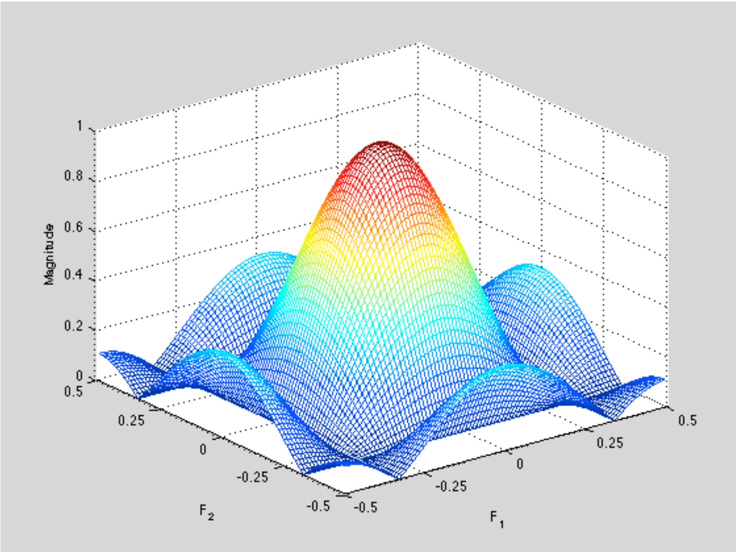
Average 3x3

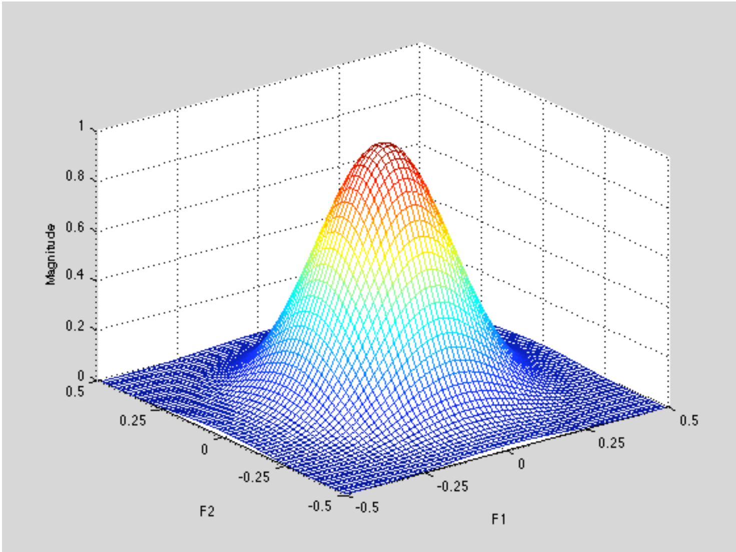
$$h=\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & \underline{1/9} & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Gaussian 5x5
σ=1

$$h=\begin{bmatrix} 0.00 & 0.01 & 0.02 & 0.01 & 0.00 \\ 0.01 & 0.06 & 0.10 & 0.06 & 0.01 \\ 0.02 & 0.10 & \underline{0.16} & 0.10 & 0.02 \\ 0.01 & 0.06 & 0.10 & 0.06 & 0.01 \\ 0.00 & 0.01 & 0.02 & 0.01 & 0.00 \end{bmatrix}$$

Note: In order to preserve the continuous component, the coefficients add up to 1



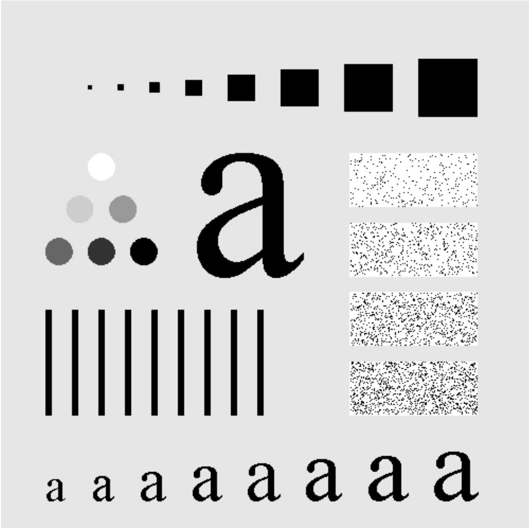


Plots of the frequency response of both filters

Filtering examples

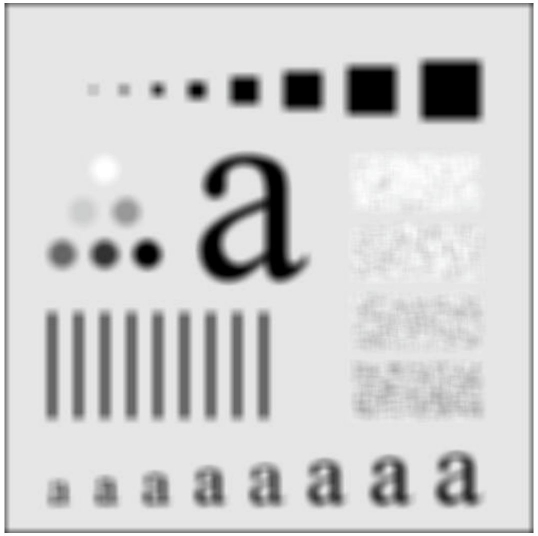
4.2

Original image



Gaussian σ=4

Average 5x5



Average 11x11

Example of noise reduction

Original image

Uniform noise

Averaging filter

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & \underline{1/9} & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Gaussian Filter (σ=0.5)

$$\begin{bmatrix} 0.011 & 0.084 & 0.011 \\ 0.084 & \underline{0.619} & 0.084 \\ 0.011 & 0.084 & 0.011 \end{bmatrix}$$

Unit Structure

4.2

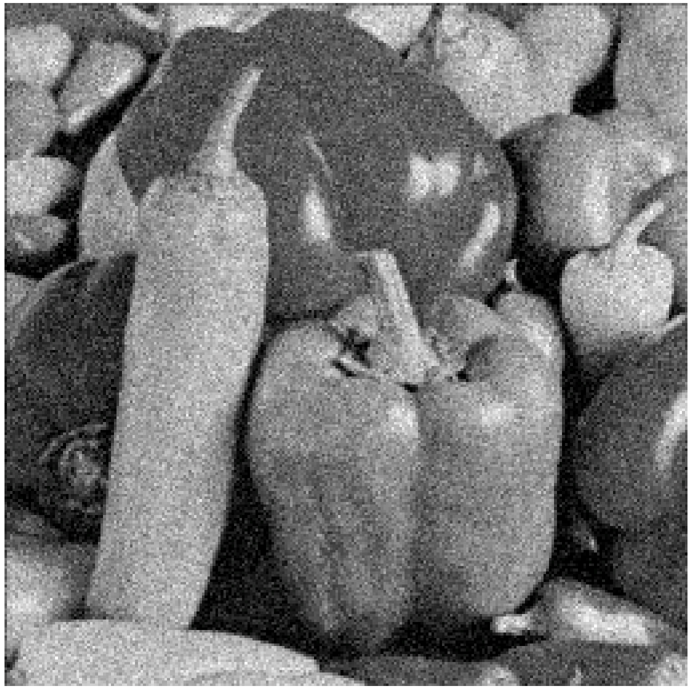
1. **Introduction:**
   - Image convolution
   - Image padding
2. **Low-pass filters designed in the spatial domain:**
   - Example of application: Noise removal
   - Trade-off noise removal vs blurring
3. **High-pass filters designed in the spatial domain:**
   - Example of application: Contour detection
4. **Filters designed in the frequency domain:**
   - Ringing effect
5. **Summary and Conclusions**

Filter design

4.2

Original image

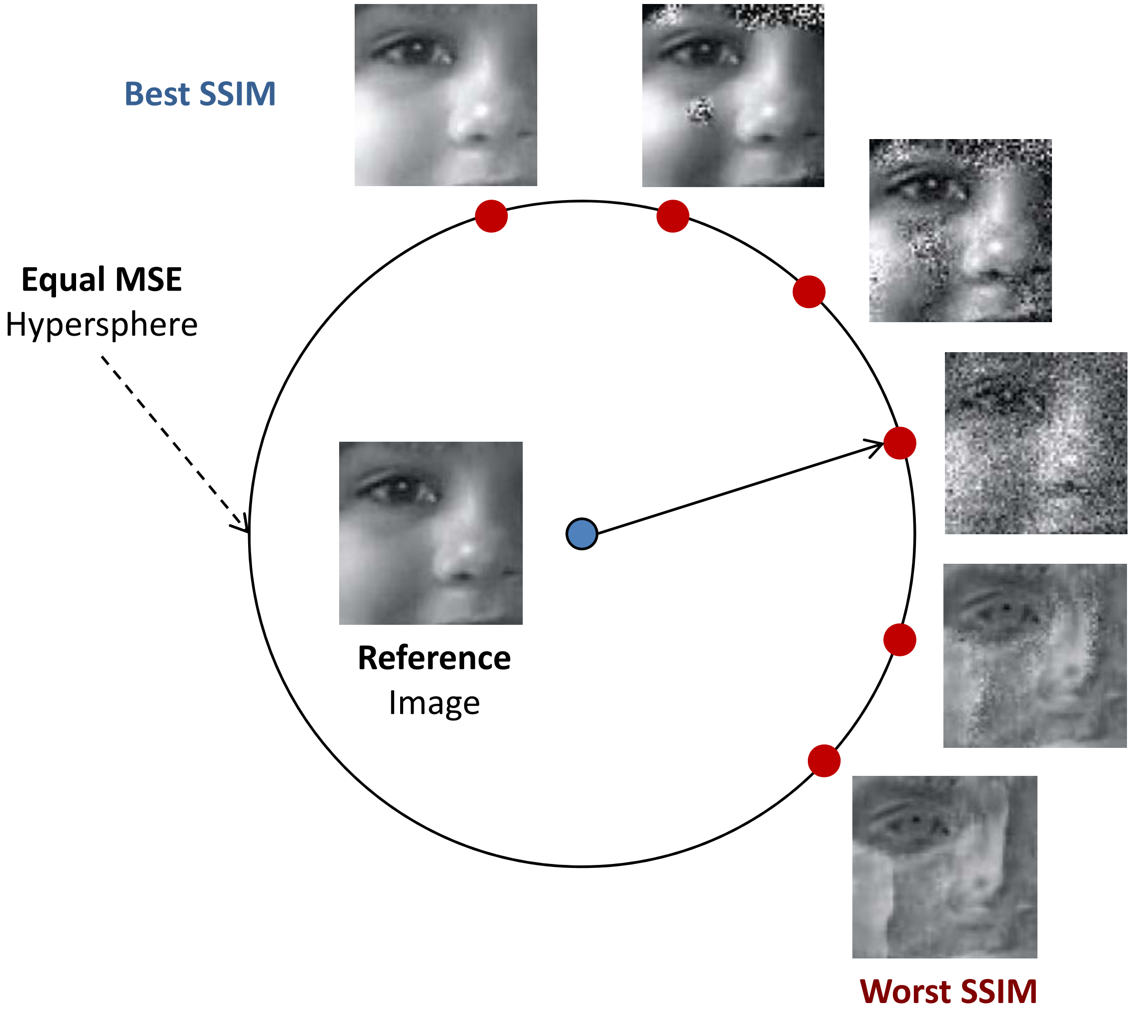
Image corrupted by additive Gaussian noise



PSNR=27,88 dB
SSIM=0,448

User oriented assessment

4.2



Objective and perceptual criteria for human: Researchers are continuously looking for objective measures that model better the subjective behavior of the Human Visual System.

For instance, the **Structural SIMilarity Index** (**SSIM**).

Z. Wang and A.C. Bovik, "Mean Square Error: Love it or Leave it?" *IEEE Signal Processing Magazine*, pp. 98 - 117, January 2009.

Filter design

4.2

Assume we use an averaging filter: how to define the size of the **impulse response** ?

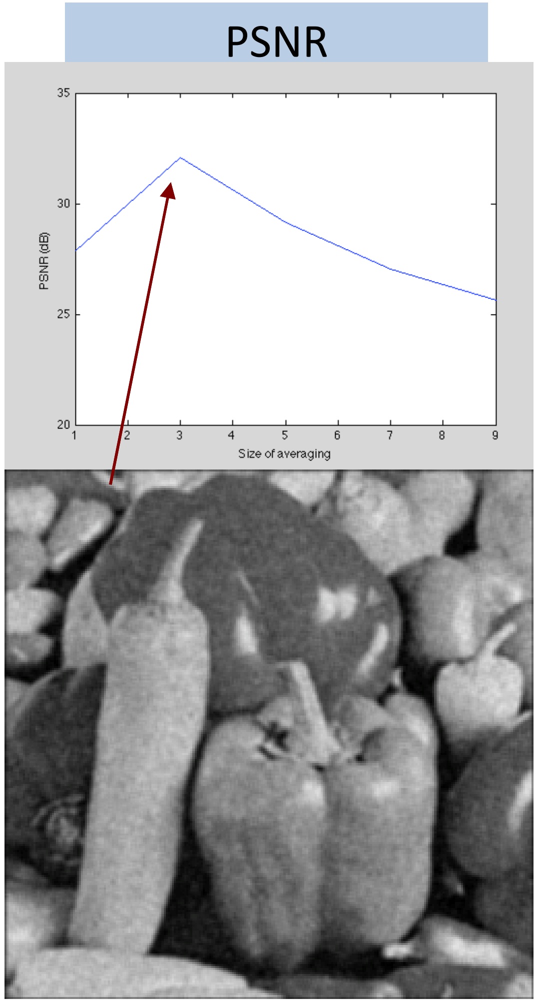
- Small impulse responses: do not remove much noise
- Large windows: blur edges

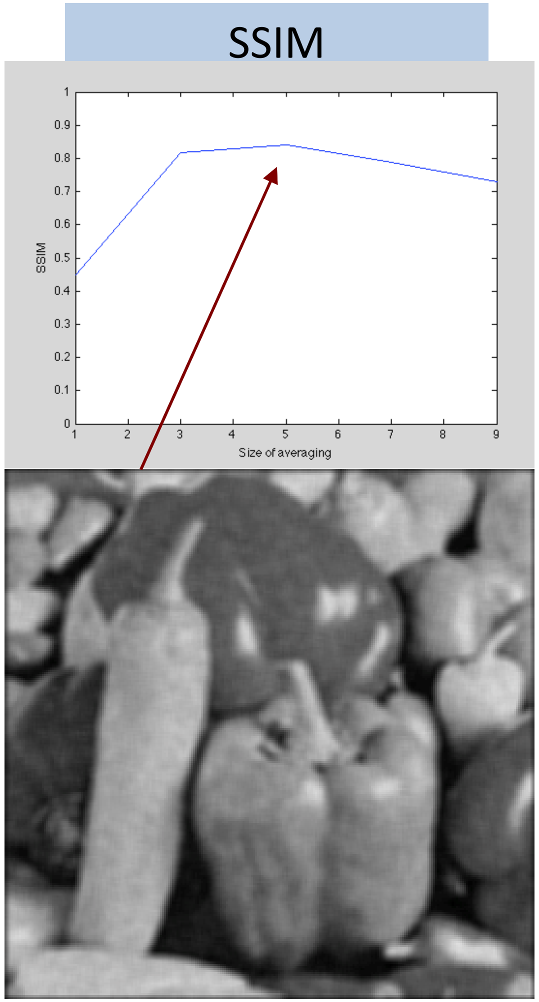
Short window

Large window

Trade-off: Noise reduction vs Blurring

4.2

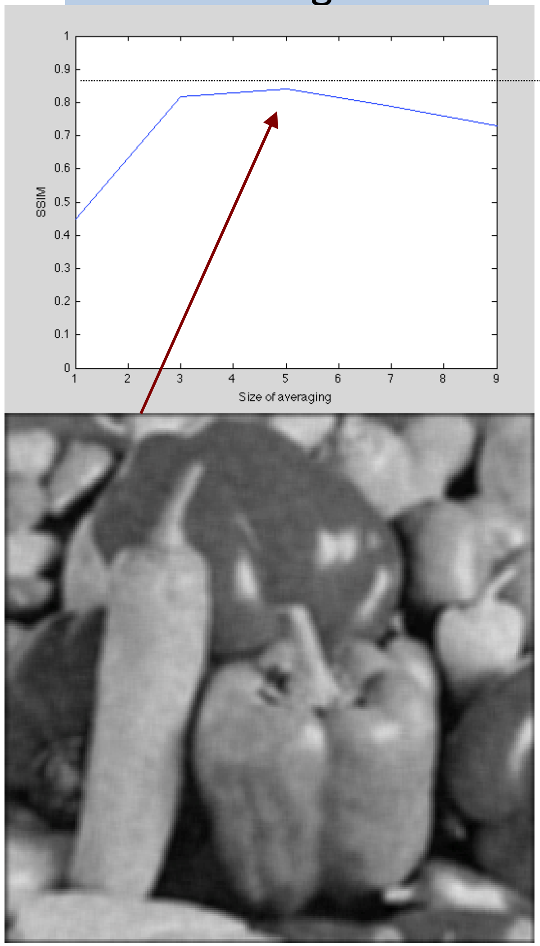




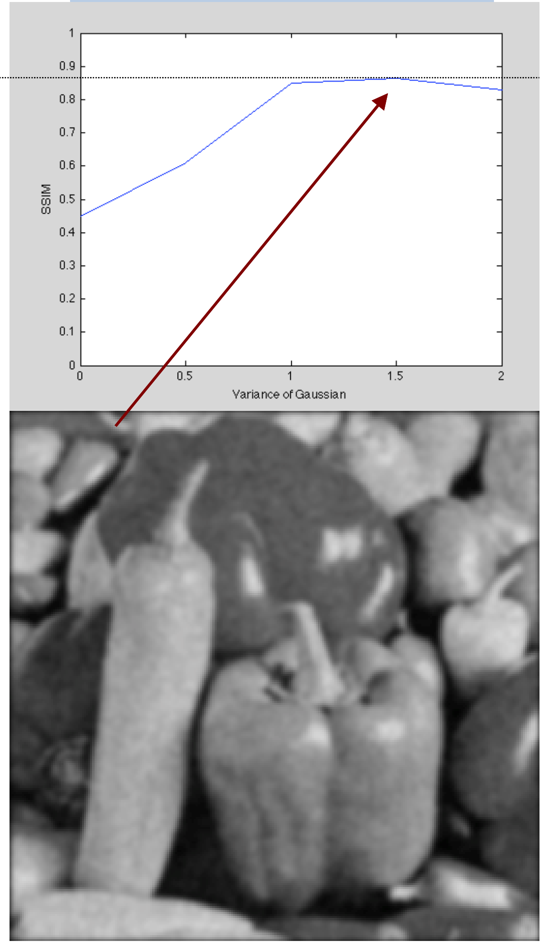
Averaging vs Gaussian filter

4.2

Average



Gaussian



Impulsive noise (salt and pepper)

4.2

Original image

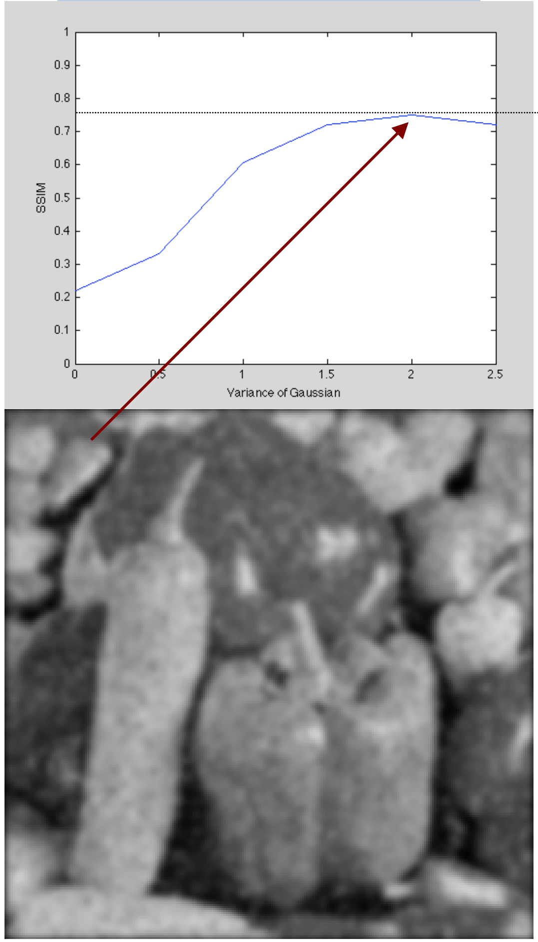
Image corrupted by impulsive noise

PSNR=19,22 dB
SSIM=0,22

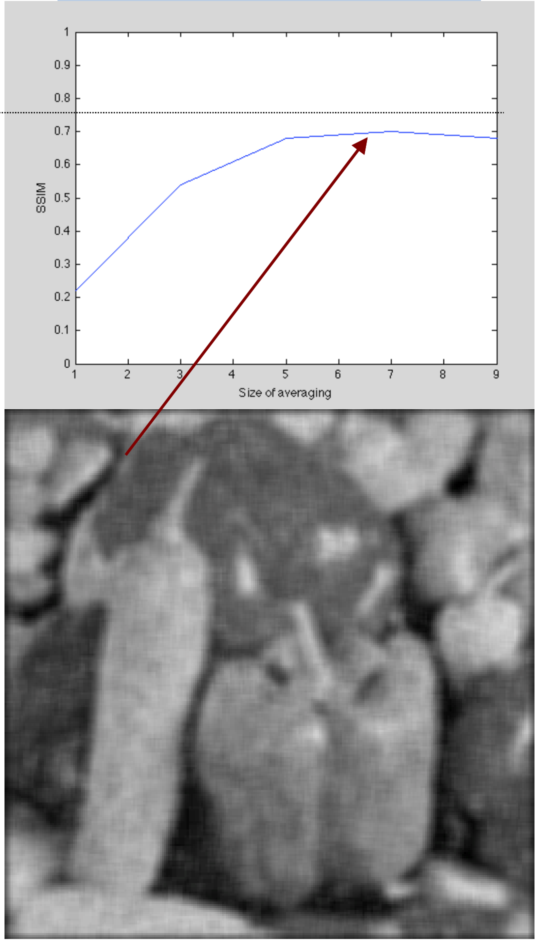
Impulsive (salt and pepper) noise

4.2

Gaussian



Average



Impulsive (salt and pepper) noise

4.2

The Gaussian filter is better than the Averaging filter…. **But can we do better**?

(Original noisy image: PSNR=19,22 dB; SSIM=0,22)

Gaussian

PSNR=27,50 dB SSIM=0,75

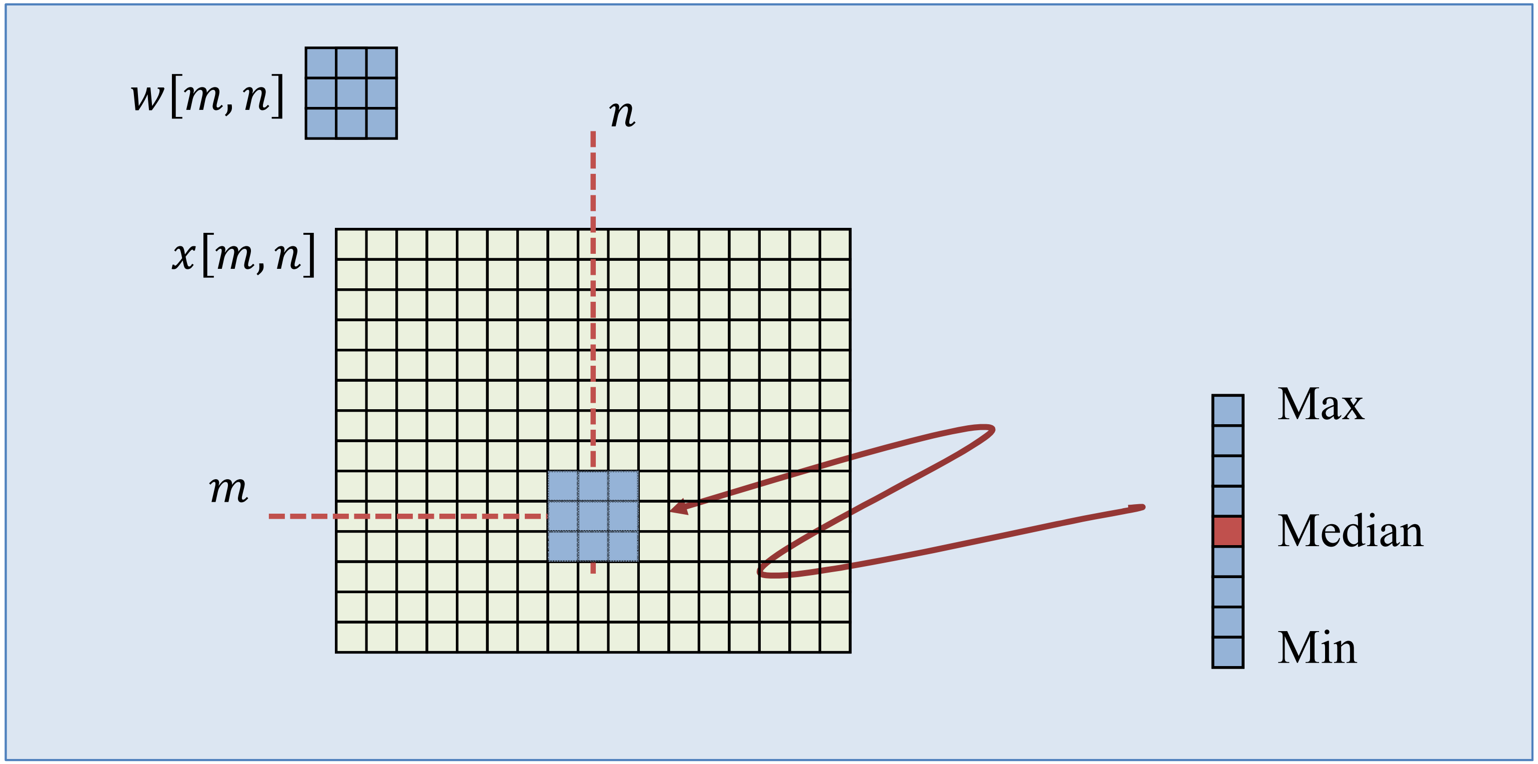
?

PSNR=34,91 dB SSIM=0,97

Impulsive noise (salt and pepper)

4.2

Median filter: a non linear filter that is very efficient for impulsive noise removal!



Unit Structure

4.2

1. **Introduction:**
   - Image convolution
   - Image padding
2. **Low-pass filters designed in the spatial domain:**
   - Example of application: Noise removal
   - Trade-off noise removal vs blurring
3. **High-pass filters designed in the spatial domain:**
   - Example of application: Contour detection
4. **Filters designed in the frequency domain:**
   - Ringing effect
5. **Summary and Conclusions**

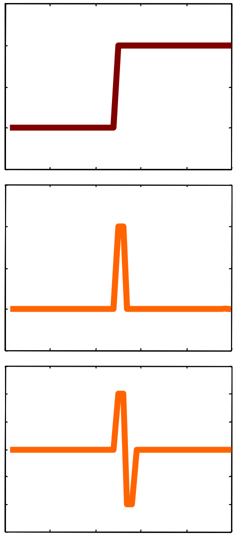
Contour detection

- We are going to study high-pass filters in the context of **contour detection**.

Contours (**transitions**) in a signal can be detected by means of :

- **First derivative**: There is a **local maximum** (or minimum) in the derivative in the position of the transition.

- **Second derivative**: There is a **zero crossing** in the second derivative in the position of the transition.



- In 2D, the **gradient** or the **Laplacian** of the image are studied:

$$\nabla f(x,y) = \left[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right]^T$$

$$\Delta f(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Gradient approximation

4.2

- In image processing, gradients can be approximated by means of **finite differences equations**, one for each direction (x, y).

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\Delta x \to 0}\left[\frac{f(x+\Delta x,y)-f(x,y)}{\Delta x}\right] \approx \frac{f(x+\Delta x,y)-f(x,y)}{\Delta x}$$

Forward difference

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\Delta x \to 0}\left[\frac{f(x,y)-f(x-\Delta x,y)}{\Delta x}\right] \approx \frac{f(x,y)-f(x-\Delta x,y)}{\Delta x}$$

Backward difference

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\Delta x \to 0}\left[\frac{f(x+\Delta x,y)-f(x-\Delta x,y)}{2\Delta x}\right] \approx \frac{f(x+\Delta x,y)-f(x-\Delta x,y)}{2\Delta x}$$

Symmetric difference

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{\substack{x=m\\y=n}} \approx f(m+1,n)-f(m,n)$$

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{\substack{x=m\\y=n}} \approx f(m,n)-f(m-1,n)$$

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{\substack{x=m\\y=n}} \approx \frac{f(m+1,n)-f(m-1,n)}{2}$$

Gradient approximation

- Each of these equations can be implemented as an **LTI filter**.

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{\substack{x=m\\y=n}} \approx f(m+1,n) - f(m,n) \qquad h_x^+[m] = [1, \underline{-1}] \;\Rightarrow\; \frac{\partial f}{\partial x} \approx f * h_x^+$$

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{\substack{x=m\\y=n}} \approx f(m,n) - f(m-1,n) \qquad h_x^-[m] = [\underline{1}, -1] \;\Rightarrow\; \frac{\partial f}{\partial x} \approx f * h_x^-$$

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{\substack{x=m\\y=n}} \approx \frac{f(m+1,n) - f(m-1,n)}{2} \qquad h_x^s[m] = [1, \underline{0}, -1] \;\Rightarrow\; \frac{\partial f}{\partial x} \approx f * h_x^s$$

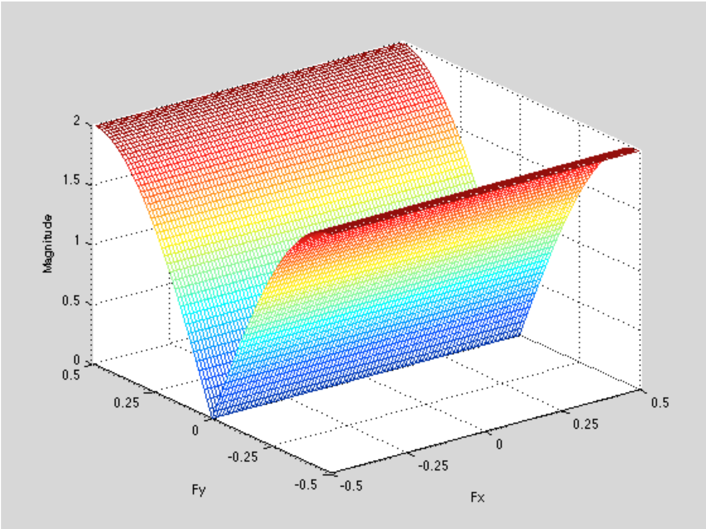
- In the symmetric filter, **the factor ½ has been removed** since usually the exact value of the gradient is not sought.
- Analogous filters can be defined in the **vertical direction**.

Gradient approximation

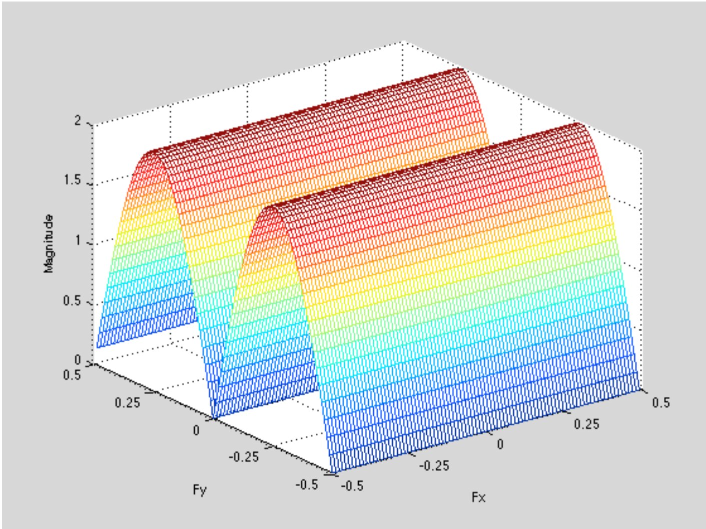
4.2

- Frequency response of the basic derivative approximations

$$h_y^-[m,n] = \begin{bmatrix} \underline{1} \\ -1 \end{bmatrix}$$



$$h_y^s[m,n] = \begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix}$$



Gradient approximation

4.2

- Frequency response of the basic derivative approximations

$$\left.\begin{array}{l} h[m,n] = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ \\ h[m,n] = h_1[m] \cdot h_2[n] \end{array}\right\} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & \underline{1} & 0 \\ 0 & -1 & 0 \end{bmatrix} \Rightarrow \begin{array}{l} h_1[m] = \delta[m] \\ \\ h_2[n] = \delta[n] - \delta[n-1] \end{array}$$

$$h_2[n] = \delta[n] - \delta[n-1] \Rightarrow H_2(F_2) = 1 - e^{-j2\pi F_2}$$

$$H_2(F_2) = e^{-j\pi F_2}\left[e^{j\pi F_2} - e^{-j\pi F_2}\right] = 2j\, e^{-j\pi F_2} \sin \pi F_2$$

$$|H(F_1, F_2)| = |H_1(F_1) \cdot H_2(F_2)| = 2\,|\sin \pi F_2| \quad \text{FIRST CASE}$$

$$|H(F_1, F_2)| = |H_1(F_1) \cdot H_2(F_2)| = 2\,|\sin 2\pi F_2| \quad \text{SECOND CASE}$$

Gradient approximation

- There exist **several implementations** of the filter that approximates the gradient in a given direction. Let us denote them, in general, as h_x (h_y).
- Typically, these filters do not estimate the gradient relying only on the data in one row (column) but in a **set of neighbor rows** (columns)
- This allows averaging among several rows (columns) and being more **robust against the presence of noise**

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

Horizontal derivative
Detects vertical contours

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

Vertical derivative
Detects horizontal contours

Gradient approximation

4.2

Let us see how the filter works in an image showing a perfectly defined contour.

Note: We use zero padding to study its effect

$$h_x[m,n]=\begin{bmatrix}1 & 0 & -1\\ 1 & \underline{0} & -1\\ 1 & 0 & -1\end{bmatrix} \Rightarrow h_x[-m,-n]=\begin{bmatrix}-1 & 0 & 1\\ -1 & \underline{0} & 1\\ -1 & 0 & 1\end{bmatrix}$$

Horizontal derivative
Detects vertical contours

0	0	0	0	0	0	0	0	0
0	0	···	0	1	1	···	1	0
0	0	···	0	1	1	···	1	0
0	···	···	···	···	···	···	···	0
0	0	···	0	1	1	···	1	0
0	0	···	0	1	1	···	1	0
0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0
0	0	···	2	2	0	···	−2	0
0	0	···	3	3	0	···	−3	0
···	···	···	···	···	···	···	···	···
0	0	···	3	3	0	···	−3	0
0	0	···	2	2	0	···	−2	0
0	0	0	0	0	0	0	0	0

Gradient approximation

4.2

Let us see how the filter works in an image showing a perfectly defined contour.

Note: We use zero padding to study its effect

$$h_y[m,n]=\begin{bmatrix}1 & 1 & 1\\ 0 & \underline{0} & 0\\ -1 & -1 & -1\end{bmatrix} \Rightarrow h_y[-m,-n]=\begin{bmatrix}-1 & -1 & -1\\ 0 & \underline{0} & 0\\ 1 & 1 & 1\end{bmatrix}$$

Vertical derivative
Detects horizontal contours

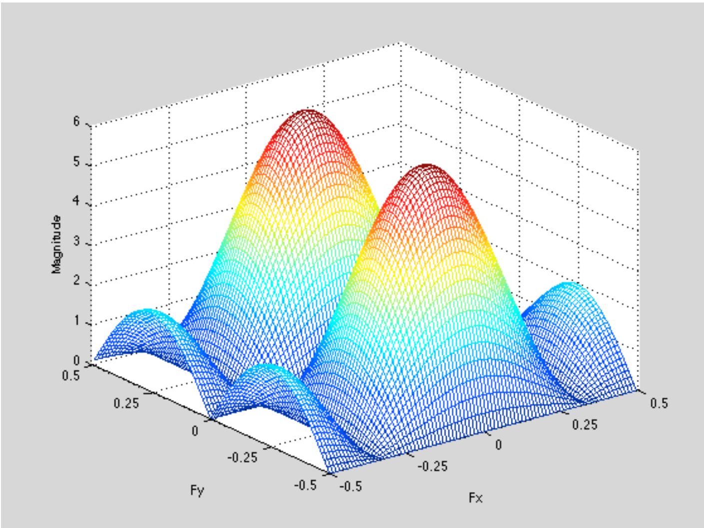
0	0	⋯	0	0	0	⋯	0	0
0	0	⋯	0	1	1	⋯	1	0
0	0	⋯	0	1	1	⋯	1	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
0	0	⋯	0	1	1	⋯	1	0
0	0	⋯	0	1	1	⋯	1	0
0	0	⋯	0	0	0	⋯	0	0

0	0	0	0	0	0	0	0	0
0	0	⋯	1	2	3	⋯	2	0
0	0	⋯	0	0	0	⋯	0	0
0	⋯	⋯	⋯	⋯	⋯	⋯	⋯	0
0	0	⋯	0	0	0	⋯	0	0
0	0	⋯	−1	−2	−3	⋯	−2	0
0	0	0	0	0	0	0	0	0

Gradient approximation

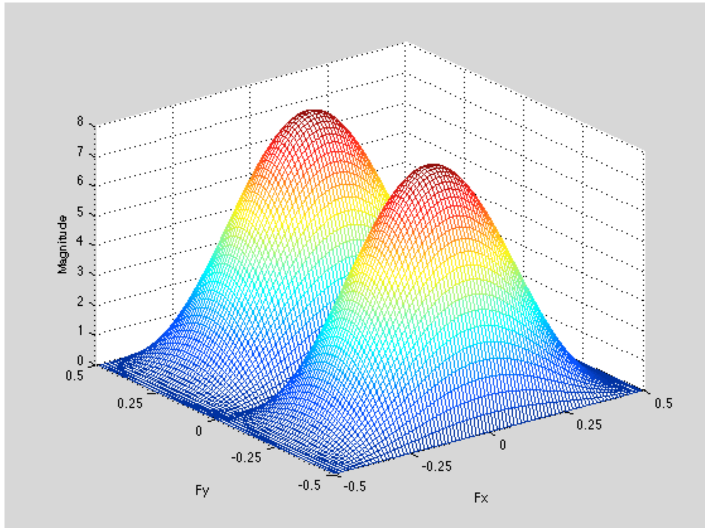
Prewitt

$$h_y[m,n] = \begin{bmatrix} 1 & 1 & 1 \\ 0 & \underline{0} & 0 \\ -1 & -1 & -1 \end{bmatrix}$$



Sobel

$$h_y[m,n] = \begin{bmatrix} 1 & 2 & 1 \\ 0 & \underline{0} & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



- Compute the frequency responses of these filters

Gradient approximation

- Compute the frequency responses of these filters

PREWITT

$$h_y[m,n] = \begin{bmatrix} 1 & 1 & 1 \\ \emptyset & \emptyset & \emptyset \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ \emptyset \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = h_{y_1}[m]\, h_{y_2}[n]$$

WHERE $h_{y_1}[n]$: GRADIENT APPROX. $h_{y_2}[m]$: AVERAGE

SOBEL

$$h_y[m,n] = \begin{bmatrix} 1 & 2 & 1 \\ \emptyset & \emptyset & \emptyset \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ \emptyset \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = h_{y_1}[m]\, h_{y_2}[n]$$

WHERE $h_{y_1}[n]$: GRADIENT APPROX. $h_{y_2}[m]$: TRIANGLE

Gradient approximation

Given the previous filters, a set of images can be defined that help analyzing the gradient of the input image:

$$\nabla f[m,n] = \begin{bmatrix} g_x[m,n] \\ g_y[m,n] \end{bmatrix}$$

Estimation of the gradient

$$g_x[m,n] = f[m,n] * h_x[m,n]$$

Image that stores the estimation of the horizontal gradient

$$g_y[m,n] = f[m,n] * h_y[m,n]$$

Image that stores the estimation of the vertical gradient

$$\left|\nabla f[m,n]\right| = \sqrt{g_x^2[m,n] + g_y^2[m,n]}$$

Image that stores the estimation of the gradient magnitude

$$\theta_{\nabla f}[m,n] = \arctan\left[\frac{g_y[m,n]}{g_y[m,n]}\right]$$

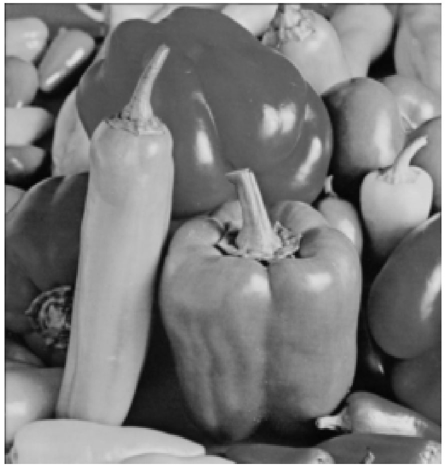
Image that stores the estimation of the gradient phase

Examples

4.2

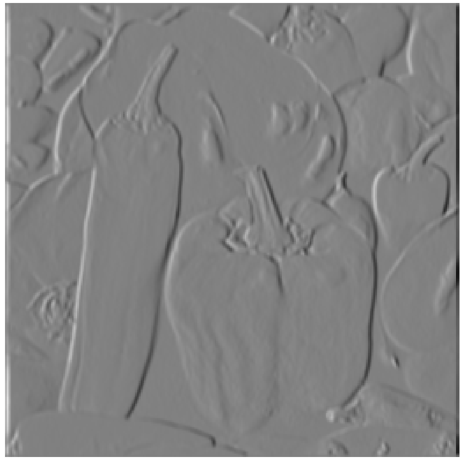
The **Prewitt filter** detects both horizontal and vertical contours and it is robust with respect to the noise.

Original image



Horizontal derivative
Detects vertical contours

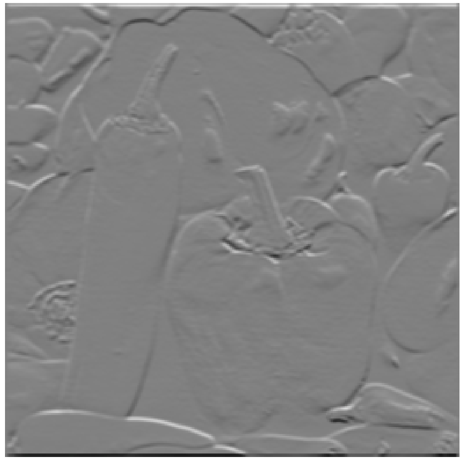
$$h_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & \underline{0} & -1 \\ 1 & 0 & -1 \end{bmatrix}$$



Gradient magnitude

Vertical derivative
Detects horizontal contours

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



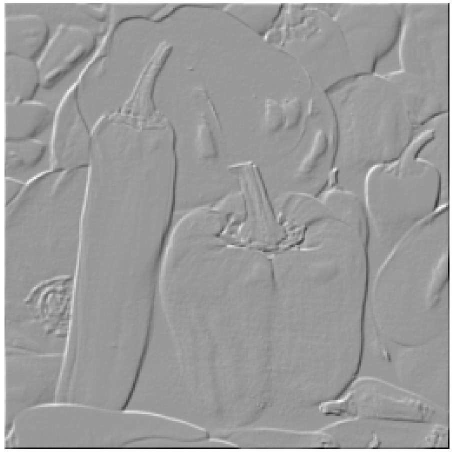
Examples

The **Roberts filter** detects contours in both diagonal directions but it is not robust with respect to the noise.

Original image

Diagonal direction
Detects contours at 45º

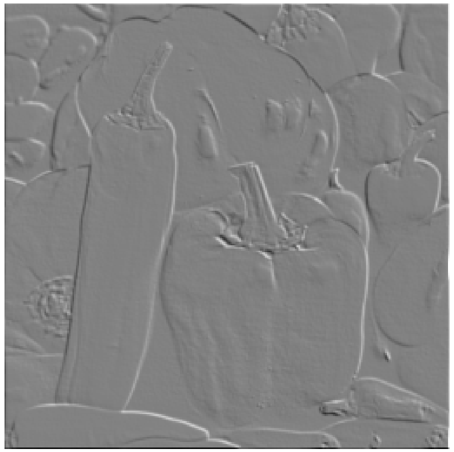
$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$



Gradient magnitude

Diagonal direction
Detects contours at 135º

$$h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$



Examples

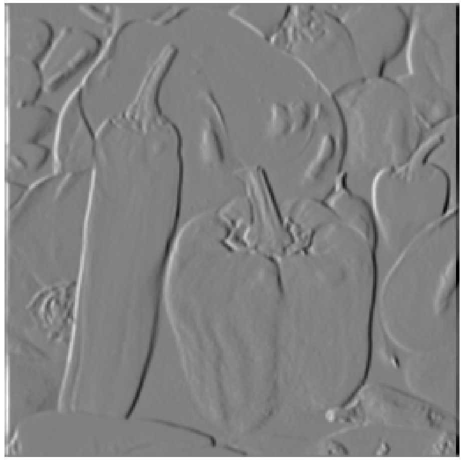
4.2

The **Sobel filter** detects both horizontal and vertical contours and increases the robustness with respect to the noise.

Original image

Horizontal derivative
Detects vertical contours

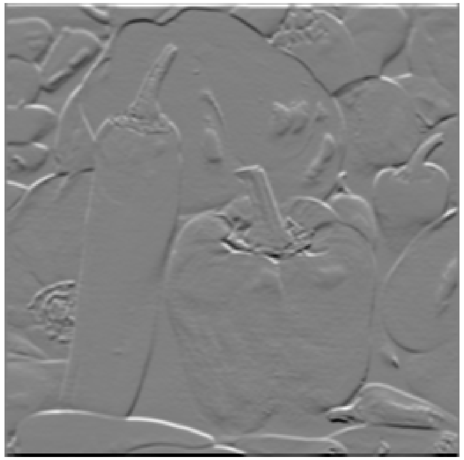
$$h_x[m,n] = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \underline{0} & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



Gradient magnitude

Vertical derivative
Detects horizontal contours

$$h_y[m,n] = \begin{bmatrix} 1 & 2 & 1 \\ 0 & \underline{0} & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



Laplacian approximation

4.2

- In image processing, the Laplacian operator can be approximated by means of **equations in finite differences**.

$$\left.\frac{\partial^2 f(x,y)}{\partial x^2}\right|_{\substack{x=m\\y=n}} \approx f(m+1,n) - 2f(m,n) + f(m-1,n)$$

$$\left.\frac{\partial^2 f(x,y)}{\partial y^2}\right|_{\substack{x=m\\y=n}} \approx f(m,n+1) - 2f(m,n) + f(m,n-1)$$

$$\Rightarrow \quad \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\left.\Delta f(x,y)\right|_{\substack{x=m\\y=n}} = \left.\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right|_{\substack{x=m\\y=n}} \approx f(m+1,n) + f(m-1,n) + f(m,n+1) + f(m,n-1) - 4f(m,n)$$

- This equation can be implemented by means of the convolution with the so-called **Laplacian filter**:

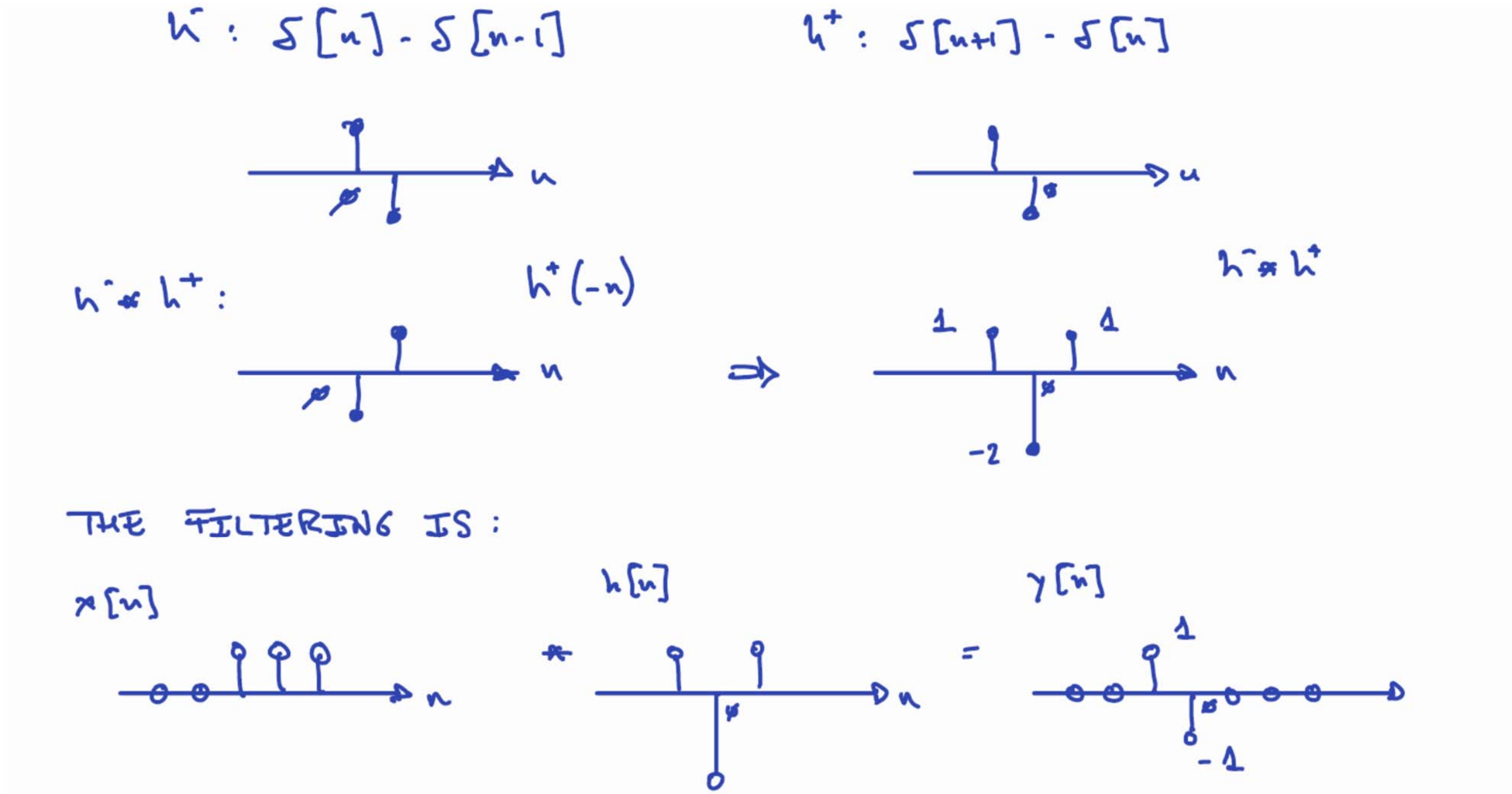
$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & \underline{-4} & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \underline{4} & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Laplacian approximation

4.2

- The Laplacian operator is approximated by **equations in finite differences**.

$$\left.\frac{\partial^2 f(x,y)}{\partial x^2}\right|_{\substack{x=m\\y=n}} \approx f(m+1,n)-2f(m,n)+f(m-1,n)$$

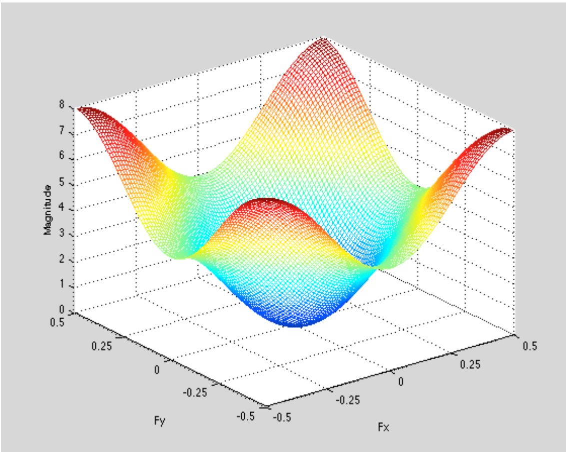


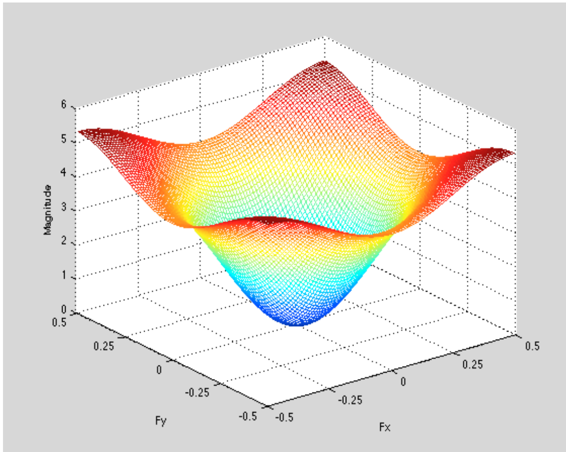
Laplacian approximation

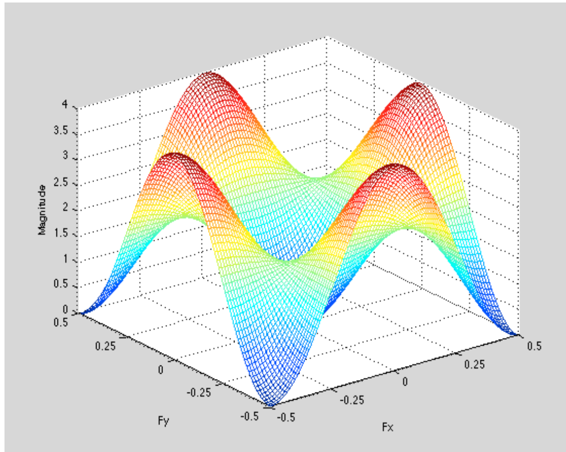
Several approximations can be used to get a more or less isotropic response.

$$h=\begin{bmatrix}0 & 1 & 0\\ 1 & \underline{-4} & 1\\ 0 & 1 & 0\end{bmatrix} \xrightarrow{2/3} h=\begin{bmatrix}1/6 & 2/3 & 1/6\\ 2/3 & \underline{-10/3} & 2/3\\ 1/6 & 2/3 & 1/6\end{bmatrix} \xleftarrow{1/3} h=\begin{bmatrix}0.5 & 0 & 0.5\\ 0 & \underline{-2} & 0\\ 0.5 & 0 & 0.5\end{bmatrix}$$

Impulse response
Matlab implementation







Frequency response

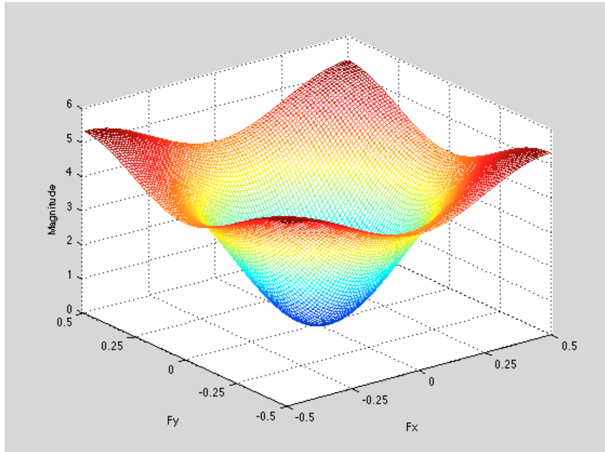
Laplacian approximation

4.2

- The **Laplacian filter** detects contours estimating the zeros of the Laplacian function. It is not very robust against noise:
  - Typically, a filter to remove noise is applied previous to the Laplacian filter.
- It produces positive and negative values and, to visualize the result, a **contrast transform** is applied.

Original image

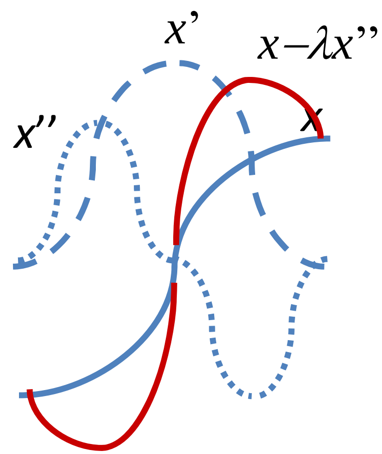
Laplacian image

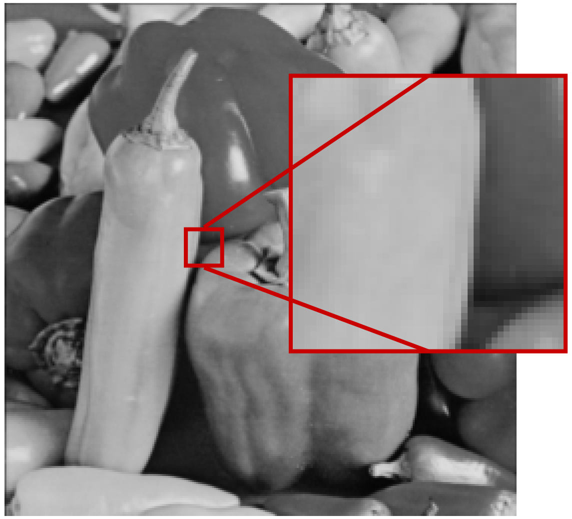


Frequency response

Application to enhancement

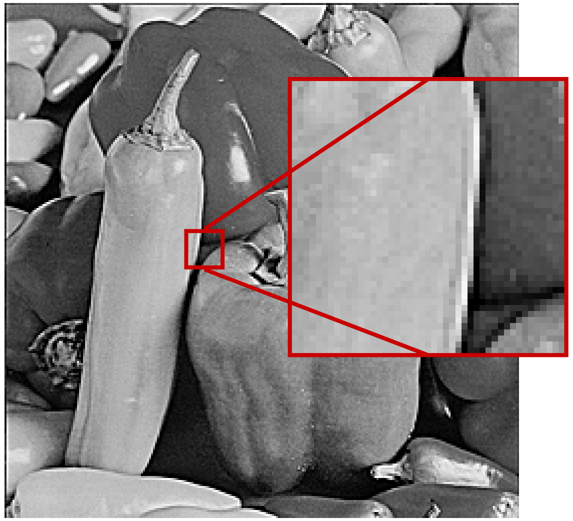
- For visualization purposes, it may be interesting to highlight the presence of contours in the image.
- This can be done by **unsharp masking**; that is, adding a weighted version of the Laplacian image to the original one.





Original image

Laplacian image



Enhanced image

Unit Structure

4.2

1. **Introduction:**
   - Image convolution
   - Image padding
2. **Low-pass filters designed in the spatial domain:**
   - Example of application: Noise removal
   - Trade-off noise removal vs blurring
3. **High-pass filters designed in the spatial domain:**
   - Example of application: Contour detection
4. **Filters designed in the frequency domain:**
   - Ringing effect
5. **Summary and Conclusions**

Ideal filters

Ideal filters (with **sharp transitions**) can be defined in the frequency domain.

$$x[m,n] \longrightarrow \boxed{h[m,n]} \longrightarrow y[m,n]$$

$$y[m,n] = x[m,n] * h[m,n]$$
$$\tilde{Y}[k,l] = X[k,l] \cdot H[k,l]$$

- The filtering process implemented in the frequency domain implies:
  - Computing the DFT of $x[m, n]$ and center it: $X[k, l]$.
    - Filters are commonly defined centered.
  - Multiply the image transform by the filter: $\tilde{Y}[k,l] = X[k,l] \cdot H[k,l]$
  - Compute the inverse DFT of the resulting signal: $\tilde{Y}[k,l]$
  - Compute the real part of the result and de-center it again: $\tilde{y}[m,n]$
    - Although usually the filter has real values and, therefore, the filtered signal should be real, due to rounding effects final values may not be real.
  - For visualization purposes, the image may need to be re-quantized.

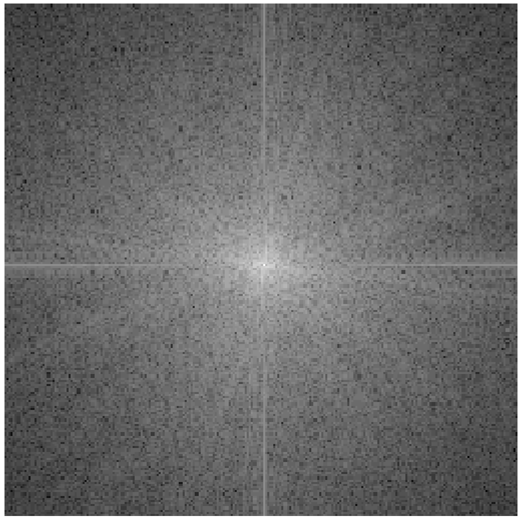
Ideal filters

4.2

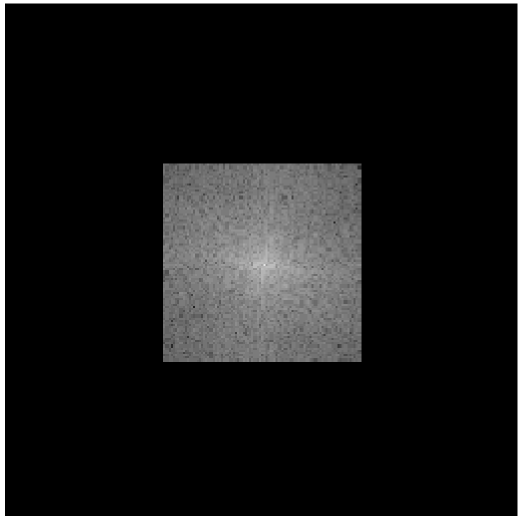
Original image

Ideal low-pass filtered image

Transform magnitude



Ideal low-pass filtered transform magnitude

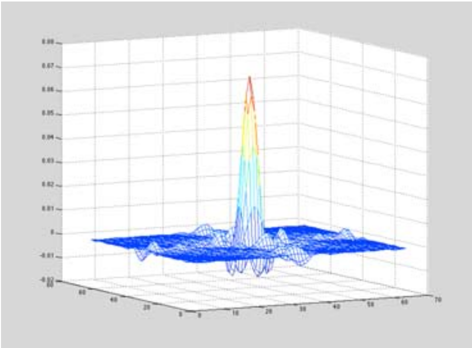


Ideal filters

4.2

- The ideal low-pass filtered image presents **ringing effect**.
- Ideal filters have as inverse transform **sinc functions**, whose lobes convolve with the transitions of the image producing these rings around contours.

Impulse response of ideal low-pass filter



Original image

Ideal low-pass filtered image

Unit Structure

4.2

1. **Introduction:**
   - Image convolution
   - Image padding
2. **Low-pass filters designed in the spatial domain:**
   - Example of application: Noise removal
   - Trade-off noise removal vs blurring
3. **High-pass filters designed in the spatial domain:**
   - Example of application: Contour detection
4. **Filters designed in the frequency domain:**
   - Ringing effect
5. **Summary and Conclusions**

Summary

4.2

- Image filters may be implemented in the **spatial domain** (typically, when the impulse response is short enough) or in the **frequency domain** (for long impulse responses or specific applications).

- Low-pass filters can be used for **noise removal**. They “average” the values within a neighborhood given by the impulse response of the filter. Filter values usually add up 1 to preserve the **image continuous component**.

- High-pass filters can be used for **contour detection**. They are designed as estimations of the **gradient** or the **Laplacian**. Gradient estimations approximate the derivative in the horizontal and vertical directions and, afterwards, combine these results into a single output.

- Ideal filters are designed in the frequency domain. Their sharp frequency transitions translates into the so-called **ringing effect** (which is the effect of the **spatial sinc function** convolving with the spatial transitions).