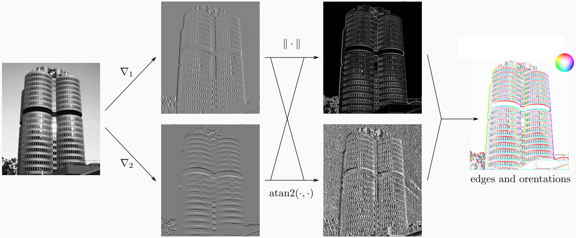
ECE 285

Image and video restoration

Chapter II – Basics of filtering I

Charles Deledalle

May 30, 2019



Basics of filtering

Definition (Collins dictionary)

filter, *noun*: any electronic, optical, or acoustic device that <u>blocks</u> signals or radiations of certain frequencies while allowing others to pass.

Basics of filtering

Definition (Collins dictionary)

filter, *noun*: any electronic, optical, or acoustic device that <u>blocks</u> signals or radiations of certain frequencies while allowing others to pass.

Refers to the direct model (observation/sensing filter)

$$y = \boldsymbol{H}x \quad \begin{cases} \bullet\ y\text{: observed image} \\ \bullet\ x\text{: image of interest} \end{cases}$$

$\boldsymbol{H}$ is a linear filter, may act only on frequencies (*e.g.*, blurs) or may not, but can only remove information (*e.g.*, inpainting).

(a) Unknown image x

$\xrightarrow{\boldsymbol{H}}$

(b) Observation y

Basics of filtering

Definition (Oxford dictionary)

filter, *noun*: a function used to <u>alter</u> the overall appearance of an image in a specific manner: ‘many other apps also offer filters for enhancing photos’

Basics of filtering

Definition (Oxford dictionary)

filter, *noun*: a function used to alter the overall appearance of an image in a specific manner: 'many other apps also offer filters for enhancing photos'

Refers to the inversion model (restoration filter)

$$\hat{x} = \psi(y) \quad \begin{cases} \bullet\ y\text{: observed image} \\ \bullet\ \hat{x}\text{: estimate of } x \end{cases}$$

ψ is a filter, linear or non-linear, that may act only on frequencies or may not, and usually attempts to add information.

(a) Observation y

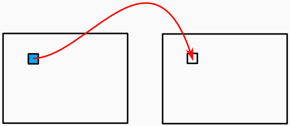
$\xrightarrow{\psi}$

(b) Estimate $\hat{x}$

Action of filters

Perform punctual, local and/or global transformations of pixel values

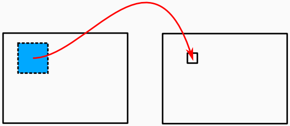
Punctual:



New pixel value depends only on the input one

e.g., change of contrast

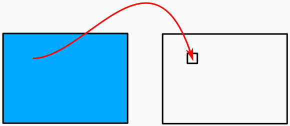
Local:



New pixel value depends on the surrounding input pixels

e.g., averaging/convolutions

Global:



New pixel value depends on the whole input image

e.g., sigma filter

Basics of filtering

Filters

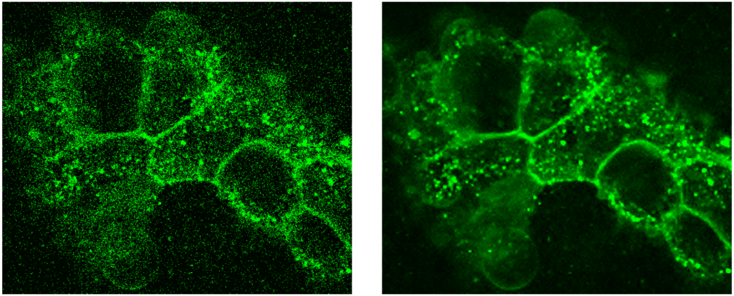
- Often one of the first steps in a processing pipeline,
- Goal: improve, simplify, denoise, deblur, detect objects...

CV pipeline

Source: Mike Thompson

Improve/denoise/detect

Improve/**denoise**/detect

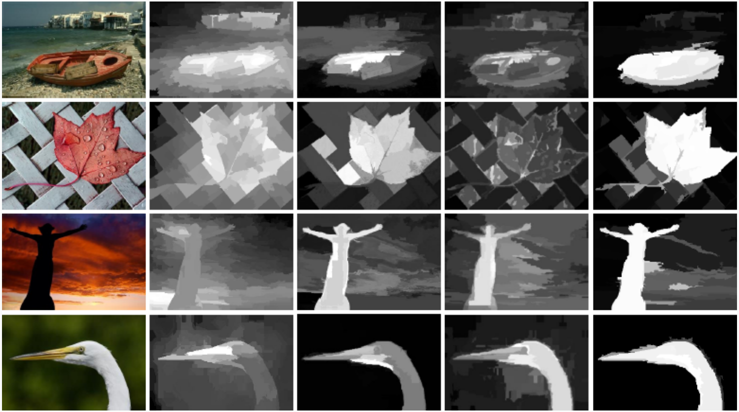


Fibroblast cells and microbreads (fluorescence microscopy)

Source: F. Luisier & C. Vonesch

Basics of filtering

Improve/denoise/**detect**



Foreground/Background separation

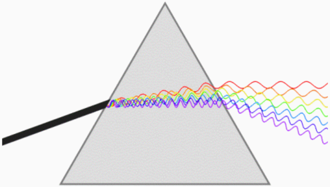
Source: H. Jiang, et al.

Standard filters

Two main approaches:

- **Spatial domain:** use the pixel grid / spatial neighborhoods
- **Spectral domain:** use Fourier transform, cosine transform, ...

Spatial



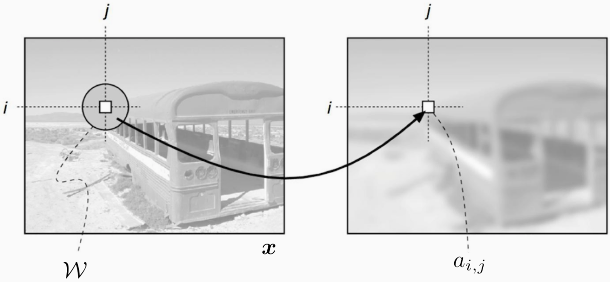
Spectral

Spatial filtering

Spatial filtering – Local filters

Local / Neighboring filters

- Combine/select values of y in the neighborhood $\mathcal{N}_{i,j}$ of pixel (i, j)
- Following examples: moving average filters, derivative filters, median filters



Moving average

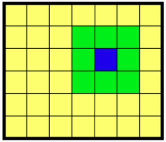
$$\hat{x}_{i,j} = \frac{1}{\mathrm{Card}(\mathcal{N})} \sum_{(k,l)\in\mathcal{N}_{i,j}} y_{k,l}$$

Examples:

- Boxcar filter: $\mathcal{N}_{i,j} = \{(k,l)\ ;\ |i-k| \leqslant \tau \quad \text{and} \quad |j-l| \leqslant \tau\}$
- Diskcar filter: $\mathcal{N}_{i,j} = \left\{(k,l)\ ;\ |i-k|^2 + |j-l|^2 \leqslant \tau^2\right\}$

3×3 boxcar filter

$$\hat{x}_{i,j} = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} y_{k,l}$$



Parameters:

- Size: 3×3, 5×5, ...
- Shape: square, disk
- Centered or not

Moving average

$$\hat{x}_{i,j} = \frac{1}{\mathrm{Card}(\mathcal{N})} \sum_{(k,l)\in\mathcal{N}_{i,j}} y_{k,l} \quad \text{or} \quad \hat{x}_{i,j} = \frac{1}{\mathrm{Card}(\mathcal{N})} \sum_{(k,l)\in\mathcal{N}} y_{i+k,j+l}$$

$\mathcal{N} = \mathcal{N}_{0,0}$

Examples:

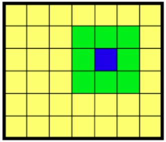
- Boxcar filter: $\mathcal{N}_{i,j} = \{(k,l) \; ; \; |i-k| \leqslant \tau \quad \text{and} \quad |j-l| \leqslant \tau\}$
- Diskcar filter: $\mathcal{N}_{i,j} = \left\{(k,l) \; ; \; |i-k|^2 + |j-l|^2 \leqslant \tau^2\right\}$

3×3 boxcar filter

$$\hat{x}_{i,j} = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} y_{k,l}$$

or

$$\hat{x}_{i,j} = \frac{1}{9} \sum_{k=-1}^{+1} \sum_{l=-1}^{+1} y_{i+k,j+l}$$

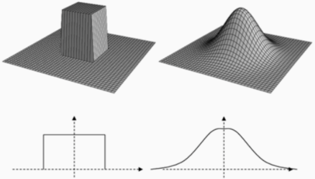


Parameters:

- Size: 3×3, 5×5, ...
- Shape: square, disk
- Centered or not

Moving weighted average

$$\hat{x}_{i,j} = \frac{\displaystyle\sum_{(k,l)\in\mathbb{Z}^2} w_{k,l}\, y_{i+k,j+l}}{\displaystyle\sum_{(k,l)\in\mathbb{Z}^2} w_{k,l}}$$



0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

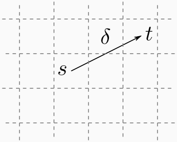
0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

- Neighboring filter: $w_{i,j} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathcal{N} \\ 0 & \text{otherwise} \end{cases}$
- Gaussian kernel: $w_{i,j} = \exp\left(-\frac{i^2+j^2}{2\tau^2}\right)$
- Exponential kernel: $w_{i,j} = \exp\left(-\frac{\sqrt{i^2+j^2}}{\tau}\right)$

Spatial filtering – Moving average

- Rewrite $\hat{x}$ as a function of $s = (i, j)$, and let $\delta = (k, l)$ and $t = s + \delta$

$$\hat{x}(s) = \frac{\displaystyle\sum_{\delta \in \mathbb{Z}^2} w(\delta) y(s + \delta)}{\displaystyle\sum_{\delta \in \mathbb{Z}^2} w(\delta)} = \frac{\displaystyle\sum_{t \in \mathbb{Z}^2} w(t - s) y(t)}{\displaystyle\sum_{t \in \mathbb{Z}^2} w(\underbrace{t - s}_{\delta})}$$



Local average filter

- Weights are functions of the distance between t and s (length of δ) as

$$w(t - s) = \varphi(\text{length}(t - s))$$

- $\varphi : \mathbb{R}^+ \to \mathbb{R}$: kernel function (⚠ $\neq$ convolution kernel)
- Often, φ satisfies
  - $\varphi(0) = 1$,
  - $\lim\limits_{\alpha \to \infty} \varphi(\alpha) = 0$,
  - φ non-increasing: $\alpha > \beta \Rightarrow \varphi(\alpha) \leqslant \varphi(\beta)$.

Example

- Box filter

$$\varphi(\alpha) = \begin{cases} 1 & \text{if} \quad \alpha \leqslant \tau \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \text{length}(\delta) = \|\delta\|_\infty$$

- Disk filter

$$\varphi(\alpha) = \begin{cases} 1 & \text{if} \quad \alpha \leqslant \tau \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \text{length}(\delta) = \|\delta\|_2$$

- Gaussian filter

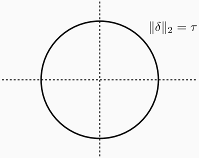
$$\varphi(\alpha) = \exp\left(-\frac{\alpha^2}{2\tau^2}\right) \quad \text{and} \quad \text{length}(\delta) = \|\delta\|_2$$

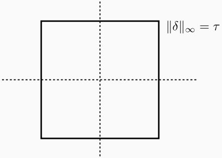
- Exponential filter

$$\varphi(\alpha) = \exp\left(-\frac{\alpha}{\tau}\right) \quad \text{and} \quad \text{length}(\delta) = \|\delta\|_2$$

Reminder:

$$\|v\|_p = \left(\sum_{k=1}^{d} v_k^p\right)^{1/p}$$



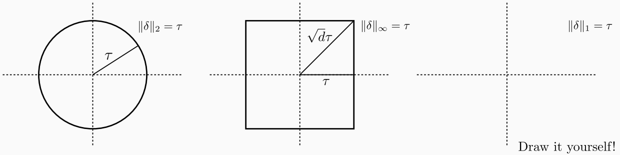


Spatial filtering – Moving average

- φ often depends on (at least) one parameter τ
    - τ controls the amount of filtering
    - $\tau \to 0$: no filtering (output = input)
    - $\tau \to \infty$: average everything in the same proportion (output = constant signal)

- φ often depends on (at least) one parameter τ
  - τ controls the amount of filtering
  - $\tau \to 0$: no filtering (output = input)
  - $\tau \to \infty$: average everything in the same proportion (output = constant signal)

What would provide $\varphi(\alpha) = \begin{cases} 1 & \textbf{if} \quad \alpha \leqslant \tau \\ 0 & \textbf{otherwise} \end{cases}$ **and** $\mathrm{length}(\delta) = \|\delta\|_1$**?**



d: dimension ($d = 2$ for pictures, $d = 3$ for videos, ...)

Spatial filtering – Moving average

Box filter

(a) $\tau = 1$

(b) $\tau = 20$

(c) $\tau = 40$

(d) $\tau = 10^3$

Diamond filter

(e) $\tau = 1$

(f) $\tau = 20$

(g) $\tau = 40$

(h) $\tau = 10^3$

Disk filter

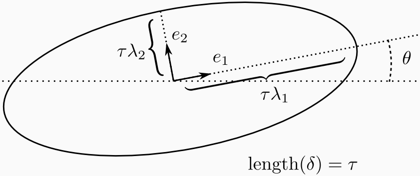
(i) $\tau = 1$

(j) $\tau = 20$

(k) $\tau = 40$

(l) $\tau = 10^3$

How to express anisotropy?

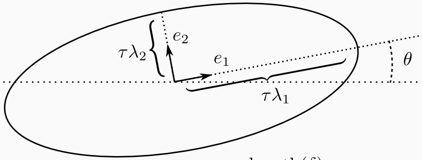


- $\|e_1\|_2 = 1$
- $\|e_2\|_2 = 1$
- $\langle e_1,\, e_2 \rangle = 0$

$$\text{length}(\delta) =$$

Spatial filtering

How to express anisotropy?

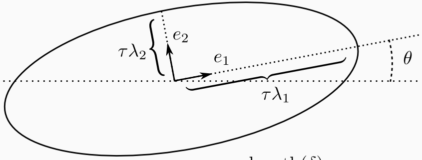


$\mathrm{length}(\delta) = \tau$

- $\|e_1\|_2 = 1$
- $\|e_2\|_2 = 1$
- $\langle e_1, \, e_2 \rangle = 0$

$$\mathrm{length}(\delta) = \sqrt{\delta^T \boldsymbol{\Sigma}^{-1} \delta} \quad \text{where} \quad \boldsymbol{\Sigma} = \underbrace{\begin{pmatrix} e_1 & e_2 \end{pmatrix} \begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix} \begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix}}_{\text{eigen-decomposition}}$$

How to express anisotropy?

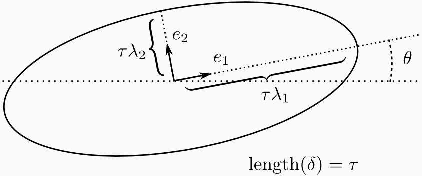


- $\|e_1\|_2 = 1$
- $\|e_2\|_2 = 1$
- $\langle e_1,\, e_2 \rangle = 0$

$$\mathrm{length}(\delta) = \sqrt{\delta^T \boldsymbol{\Sigma}^{-1} \delta} \quad \text{where} \quad \boldsymbol{\Sigma} = \underbrace{\begin{pmatrix} e_1 & e_2 \end{pmatrix} \begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix} \begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix}}_{\text{eigen-decomposition}}$$

$$= \|SR\delta\|_2 \quad \text{where} \quad SR = \underbrace{\begin{pmatrix} \lambda_1^{-1} & 0 \\ 0 & \lambda_2^{-1} \end{pmatrix}}_{S\text{: scaling}} \underbrace{\begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix}}_{R\text{: rotation}}$$

How to express anisotropy?



- $\|e_1\|_2 = 1$
- $\|e_2\|_2 = 1$
- $\langle e_1, e_2 \rangle = 0$

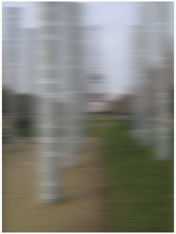
$$\text{length}(\delta) = \sqrt{\delta^T \boldsymbol{\Sigma}^{-1} \delta} \quad \text{where} \quad \boldsymbol{\Sigma} = \underbrace{\begin{pmatrix} e_1 & e_2 \end{pmatrix} \begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix} \begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix}}_{\text{eigen-decomposition}}$$

$$= \|SR\delta\|_2 \quad \text{where} \quad SR = \underbrace{\begin{pmatrix} \lambda_1^{-1} & 0 \\ 0 & \lambda_2^{-1} \end{pmatrix}}_{S\text{: scaling}} \underbrace{\begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix}}_{R\text{: rotation}}$$

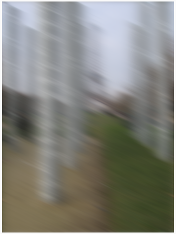
indeed, $e_1 = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}, e_2 = \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix}$ i.e. $R = \underbrace{\begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}}_{\text{rotation of } -\theta}$

Spatial filtering – Moving average

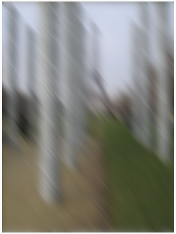
(a) y



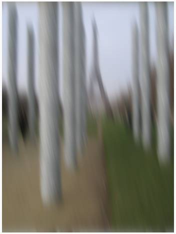
(b) $\hat{x}$, $\theta = 0°$



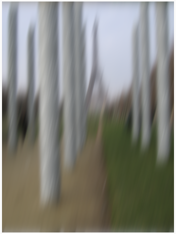
(c) $\theta = 26°$



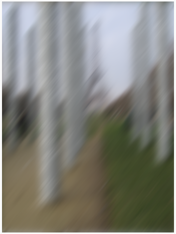
(d) $\theta = 51°$



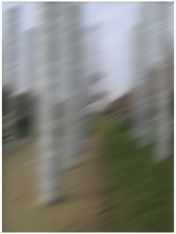
(e) $\theta = 77°$



(f) $\theta = 103°$



(g) $\theta = 129°$



(h) $\theta = 154°$

Spatial filtering – Moving average for denoising

Moving average for denoising?

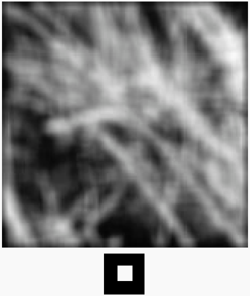
Figure 1 – (left) Gaussian noise $\sigma = 10$. (right) Gaussian filter $\tau = 3$.

Moving average for denoising?

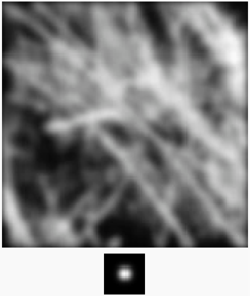
Figure 1 – (left) Gaussian noise $\sigma = 30$. (right) Gaussian filter $\tau = 5$.

Spatial filtering – Moving average for denoising

Input image



Boxcar filter



Gaussian filter

- Boxcar: oscillations/artifacts in vertical and horizontal directions
- Gaussian: no artifacts
- Moving average: reduces noise ☺,
  but loss of resolution, blurry aspect, removes edges ☹

Spatial filtering – Moving average for denoising

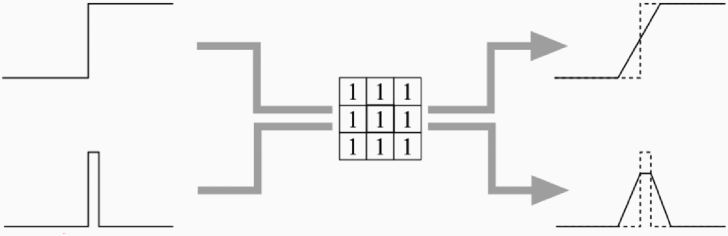


Image blur ⇒ No more edges ⇒ Structure destruction
⇒ Reduction of image quality

Spatial filtering – Moving average for denoising

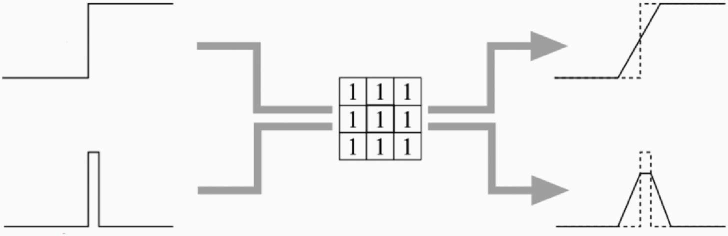
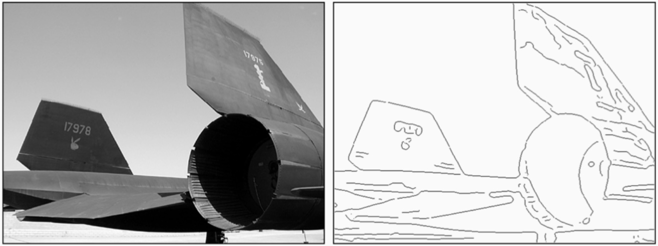


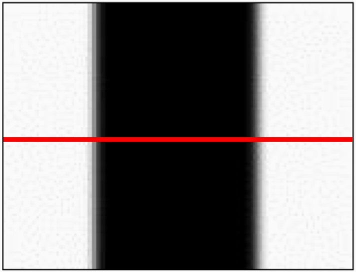
Image blur $\Rightarrow$ No more edges $\Rightarrow$ Structure destruction
$\Rightarrow$ Reduction of image quality

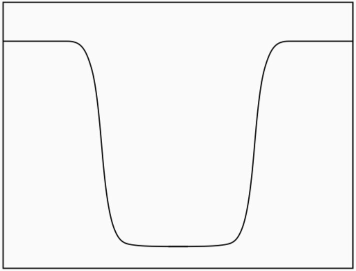
What is an edge?

Edges?

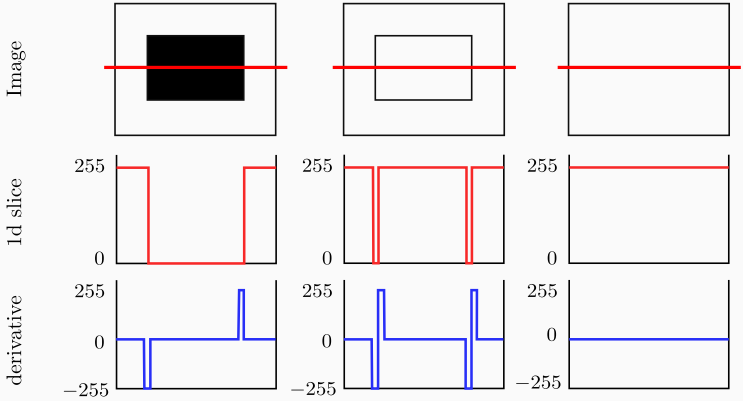
- Separation between objects, important parts of the image
- Necessary for vision in order to reconstruct objects







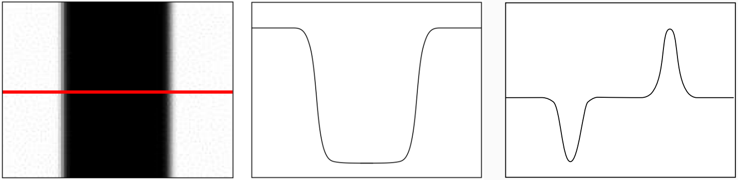
Edge: More or less brutal change of intensity



- no edges $\equiv$ no objects in the image
- abrupt change $\Rightarrow$ gap between intensities $\Rightarrow$ large derivative

How to detect edges?

- Look at the derivative
- How? Use derivative filters
- What? Filters that behave somehow as the derivative of real functions



How to design such filters?

Derivative of 1d signals

- Derivative of a function $x : \mathbb{R} \to \mathbb{R}$, if exists, is:

$$x'(t) = \underbrace{\lim_{h \to 0} \frac{x(t+h) - x(t)}{h} \quad \text{or} \quad \lim_{h \to 0} \frac{x(t) - x(t-h)}{h} \quad \text{or} \quad \lim_{h \to 0} \frac{x(t+h) - x(t-h)}{2h}}_{\text{equivalent definitions}}$$

- For a 1d discrete signal, **finite differences** are

$x'_k = x_{k+1} - x_k$	$x'_k = x_k - x_{k-1}$	$x'_k = \frac{x_{k+1} - x_{k-1}}{2}$
Forward	Backward	Centered

Derivative of 1d signals

- Can be written as a filter

$$x'_i = \sum_{k=-1}^{+1} \kappa_k y_{i+k}, \quad \text{with}$$

$\kappa = (0, -1, 1)$	$\kappa = (-1, 1, 0)$	$\kappa = (-\frac{1}{2}, 0, \frac{1}{2})$
Forward	Backward	Centered

Derivative of 2d signals

- Gradient of a function $x : \mathbb{R}^2 \to \mathbb{R}$, if exists, is:

$$\nabla x = \begin{pmatrix} \dfrac{\partial x}{\partial s_1} \\ \dfrac{\partial x}{\partial s_2} \end{pmatrix}$$

with

$$\frac{\partial x}{\partial s_1}(s_1, s_2) = \lim_{h \to 0} \frac{x(s_1 + h, s_2) - x(s_1, s_2)}{h}$$
$$\frac{\partial x}{\partial s_2}(s_1, s_2) = \lim_{h \to 0} \frac{x(s_1, s_2 + h) - x(s_1, s_2)}{h}$$

Derivative of 2d signals

- Gradient for a 2d discrete signal: **finite differences** in each direction

$$(\nabla_1 x)_{i,j} = \sum_{k=-1}^{+1} \sum_{l=-1}^{+1} (\kappa_1)_{k,l} y_{i+k,j+l}$$

$$(\nabla_2 x)_{i,j} = \sum_{k=-1}^{+1} \sum_{l=-1}^{+1} (\kappa_2)_{k,l} y_{i+k,j+l}$$

Forward:

$$\kappa_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad \kappa_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Backward:

$$\kappa_1 = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \kappa_2 = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Centered:

$$\kappa_1 = \begin{pmatrix} 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{pmatrix} \qquad \kappa_2 = \begin{pmatrix} 0 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{pmatrix}$$

Second order derivative of 1d signals

- Second order derivative of a function $x : \mathbb{R} \to \mathbb{R}$, if exists, is:

$$x''(t) = \lim_{h \to 0} \frac{x(t-h) - 2x(t) + x(t+h)}{h^2}$$

- For a 1d discrete signal: $x''_k = x_{k-1} - 2x_k + x_{x+1}$
- Corresponding filter: $h = (1, -2, 1)$

Laplacian of 2d signals

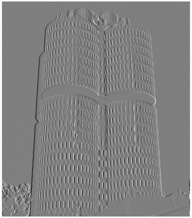
- Laplacian of a function $x : \mathbb{R}^2 \to \mathbb{R}$, if exists, is:

$$\Delta x = \frac{\partial^2 x}{\partial s_1^2} + \frac{\partial^2 x}{\partial s_2^2}$$

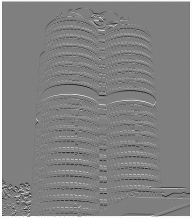
- For a 2d discrete signal: $x''_{i,j} = x_{i-1,j} + x_{i,j-1} - 4x_{i,j} + x_{i+1,j} + x_{i,j+1}$
- Corresponding filter: $h = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$

Spatial filtering – Derivative filters

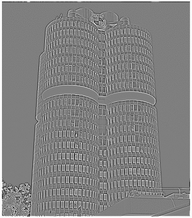
(a) x



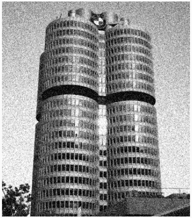
(b) $\nabla_1 x$



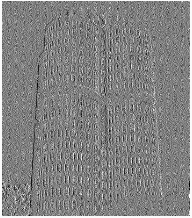
(c) $\nabla_2 x$



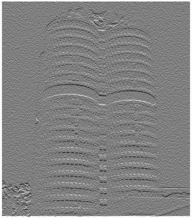
(d) Δx



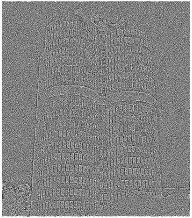
(e) x



(f) $\nabla_1 x$



(g) $\nabla_2 x$



(h) Δx

Derivative filters detect edges ☺ but are sensitive to noise ☹

Other derivative filters

- Roberts cross operator (1963)

$$\kappa_{\searrow} = \begin{pmatrix} +1 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{and} \quad \kappa_{\swarrow} = \begin{pmatrix} 0 & +1 \\ -1 & 0 \end{pmatrix}$$

- Sobel operator (1968)

$$\kappa_1 = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} \quad \text{and} \quad \kappa_2 = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$$

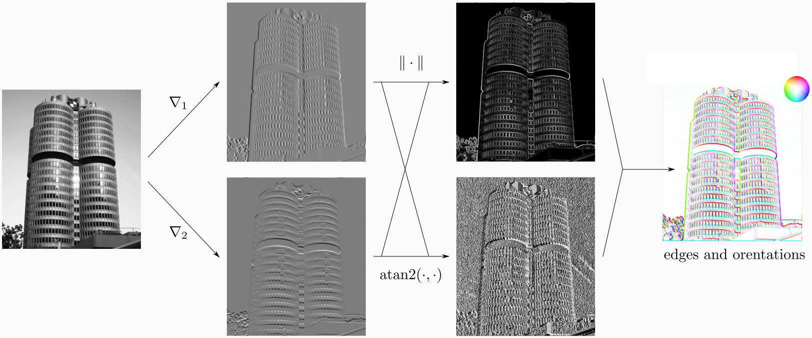
- Prewitt operator (1970)

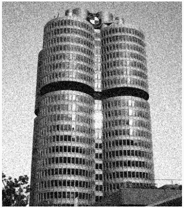
$$\kappa_1 = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \kappa_2 = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$$

Edge detection

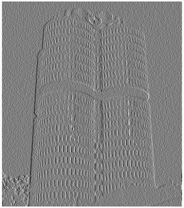
Based on the norm (and angle) of the discrete approximation of the gradient

$$\|(\nabla x)_k\| = \sqrt{(\nabla_1 x)_k^2 + (\nabla_2 x)_k^2} \quad \text{and} \quad \angle(\nabla x)_k = \text{atan2}((\nabla_2 x)_k, (\nabla_1 x)_k)$$

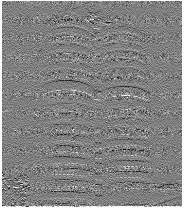




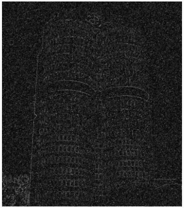
(a) x



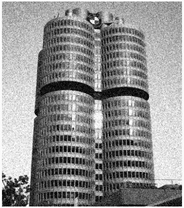
(b) $\nabla_1 x$



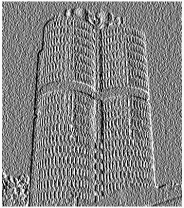
(c) $\nabla_2 x$



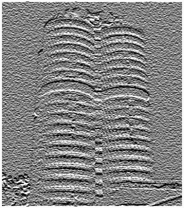
(d) $\|\nabla x\|$



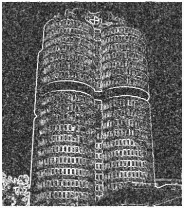
(e) x



(f) $\nabla_1 x$ (Prewitt)



(g) $\nabla_2 x$ (Prewitt)



(h) $\|\nabla x\|$ (Sobel)

Sobel & Prewitt: average in one direction, and differentiate in the other one

$\Rightarrow$ More robust to noise

Comparison between averaging and derivative filters

- Moving average

$$\hat{x}_{i,j} = \frac{\displaystyle\sum_{(k,l)\in\mathbb{Z}^2} w_{k,l}\, y_{i+k,j+l}}{\displaystyle\sum_{(k,l)\in\mathbb{Z}^2} w_{k,l}} = \sum_{(k,l)\in\mathbb{Z}^2} \underbrace{\frac{w_{k,l}}{\displaystyle\sum_{(p,q)\in\mathbb{Z}^2} w_{p,q}}}_{\kappa_{k,l}} y_{i+k,j+l}$$

$$= \sum_{(k,l)\in\mathbb{Z}^2} \kappa_{k,l}\, y_{i+k,j+l} \quad \text{with} \quad \sum_{(k,l)\in\mathbb{Z}^2} \kappa_{k,l} = 1 \qquad \text{(preserve mean)}$$

- Derivative filter

$$\hat{x}_{i,j} = \sum_{(k,l)\in\mathbb{Z}^2} \kappa_{k,l}\, y_{i+k,j+l} \quad \text{with} \quad \sum_{(k,l)\in\mathbb{Z}^2} \kappa_{k,l} = 0 \qquad \text{(remove mean)}$$

- They share the same expression

Do all filters have such an expression?

Spatial filtering – Linear translation-invariant filters

No, only linear translation-invariant (LTI) filters

Let ψ satisfying

1. **Linearity** $\quad \psi(ax + by) = a\psi(x) + b\psi(y)$
2. **Translation-invariance** $\quad \psi(y^\tau) = \psi(y)^\tau \quad$ where $\quad x^\tau(s) = x(s + \tau)$

Then, there exist coefficients $\kappa_{k,l}$ such that

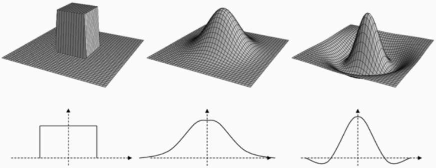
$$\psi(y)_{i,j} = \sum_{(k,l) \in \mathbb{Z}^2} \kappa_{k,l} y_{i+k,j+l}$$

The reciprocal holds true

Note: Translation-invariant = Shift-invariant = Stationary
= Same weighting applied everywhere
= Identical behavior on identical structures, whatever their location

Linear translation-invariant filters

$$\hat{x}_{i,j} = \psi(y)_{i,j} = \sum_{(k,l)\in\mathbb{Z}^2} \kappa_{k,l}\, y_{i+k,j+l}$$



$$\frac{1}{9}\begin{bmatrix} 0&0&0&0&0\\ 0&1&1&1&0\\ 0&1&1&1&0\\ 0&1&1&1&0\\ 0&0&0&0&0 \end{bmatrix} \qquad \frac{1}{57}\begin{bmatrix} 0&1&2&1&0\\ 1&3&5&3&1\\ 2&5&9&5&2\\ 1&3&5&3&1\\ 0&1&2&1&0 \end{bmatrix} \qquad \begin{bmatrix} 0&0&-1&0&0\\ 0&-1&-2&-1&0\\ -1&-2&16&-2&-1\\ 0&-1&-2&-1&0\\ 0&0&-1&0&0 \end{bmatrix}$$

Box — Gaussian — Laplacian

- Weighted average filters:

$$\sum \kappa_{k,l} = 1$$

Ex.: Box, Gaussian, Exponential, ...

- Derivative filters:

$$\sum \kappa_{k,l} = 0$$

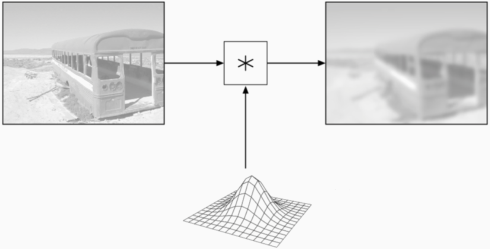
Ex.: Laplacian, Sobel, Roberts, ...

LTI filter $\equiv$ Moving weighted sum $\equiv$ Cross-correlation $\equiv$ Convolution

$$\hat{x}_{i,j} = \sum_{(k,l)\in\mathbb{Z}^2} \kappa^*_{k,l} y_{i+k,j+l} = \kappa \star y \quad \text{(for } \kappa \text{ complex)}$$

$$= \sum_{(k,l)\in\mathbb{Z}^2} \nu_{k,l} y_{i-k,j-l} = \nu * y \quad \text{where} \quad \nu_{k,l} = \kappa^*_{-k,-l}$$

ν called convolution kernel (impulse response of the filter)



Spatial filtering – LTI filters and convolution

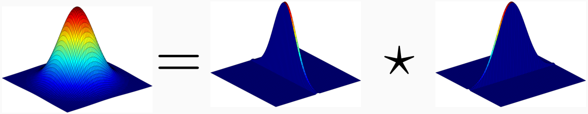
Properties of the convolution product

- **Linear** $f * (\alpha g + \beta h) = \alpha(f * g) + \beta(f * h)$
- **Commutative** $f * g = g * f$
- **Associative** $f * (g * h) = (f * g) * h$
- **Separable**

$$
\begin{aligned}
& h = h_1 * h_2 * \ldots * h_p \\
\Rightarrow \quad & f * h = (((f * h_1) * h_2) \ldots * h_p)
\end{aligned}
$$

Spatial filtering – LTI filters and convolution

- **Directional separability of (isotrope) Gaussians:**



$$\mathcal{G}_\tau^{\text{2d}} = \mathcal{G}_\tau^{\text{1d horizontal}} * \mathcal{G}_\tau^{\text{1d vertical}}$$

Directional separability of Gaussians.

$$(y * \mathcal{G}_\tau^{\text{2d}})_{i,j} = \frac{1}{Z} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \exp\left(-\frac{k^2+l^2}{2\tau^2}\right) y_{i-k,j-l}$$

$$\approx \underbrace{\frac{1}{Z} \sum_{k=-q}^{q} \sum_{l=-q}^{q} \exp\left(-\frac{k^2+l^2}{2\tau^2}\right) y_{i-k,j-l}}_{\text{Restriction to a } s \times s \text{ window, } s = 2q+1}$$

(Complexity $O(s^2 n_1 n_2)$)

$$\approx \underbrace{\frac{1}{Z} \sum_{k=-q}^{q} \exp\left(-\frac{k^2}{2\tau^2}\right) \underbrace{\sum_{l=-q}^{q} \exp\left(-\frac{l^2}{2\tau^2}\right) y_{i-k,j-l}}_{\propto (y * \mathcal{G}^{\text{1d horizontal}})_{i-k,j}}}_{\propto (y * \mathcal{G}^{\text{1d horizontal}}) * \mathcal{G}^{\text{1d vertical}}}$$

(Complexity $O(\qquad)$)

$\square$

Directional separability of Gaussians.

$$(y * \mathcal{G}_\tau^{\text{2d}})_{i,j} = \frac{1}{Z} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \exp\left(-\frac{k^2+l^2}{2\tau^2}\right) y_{i-k,j-l}$$

$$\approx \underbrace{\frac{1}{Z} \sum_{k=-q}^{q} \sum_{l=-q}^{q} \exp\left(-\frac{k^2+l^2}{2\tau^2}\right) y_{i-k,j-l}}_{\text{Restriction to a } s \times s \text{ window, } s = 2q+1}$$

(Complexity $O(s^2 n_1 n_2)$)

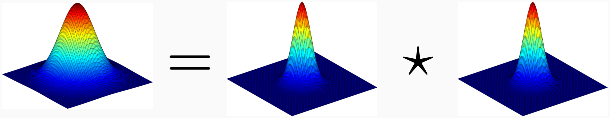
$$\approx \frac{1}{Z} \underbrace{\sum_{k=-q}^{q} \exp\left(-\frac{k^2}{2\tau^2}\right) \underbrace{\sum_{l=-q}^{q} \exp\left(-\frac{l^2}{2\tau^2}\right) y_{i-k,j-l}}_{\propto (y * \mathcal{G}^{\text{1d horizontal}})_{i-k,j}}}_{\propto (y * \mathcal{G}^{\text{1d horizontal}}) * \mathcal{G}^{\text{1d vertical}}}$$

(Complexity $O(s n_1 n_2)$)

$\square$

Spatial filtering – LTI filters and convolution

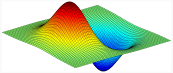
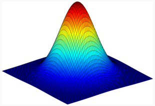
- **Multi-scale separability of Gaussians:** (Continuous case)

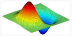
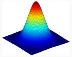


$$\mathcal{G}_{\tau_1^2} * \mathcal{G}_{\tau_2^2} = \mathcal{G}_{\tau_1^2+\tau_2^2}$$

Spatial filtering – LTI filters and convolution

- **Separability of Derivatives of Gaussian (DoG):** (Continuous case)

 $=$  $\star \; \dfrac{\partial}{\partial s_1}$

 $\star$  $=$  $\star$  $\star \; \dfrac{\partial}{\partial s_1}$

$$\mathcal{G}'_\tau * f = \frac{\partial \mathcal{G}_\tau}{\partial s} * f = \mathcal{G}_\tau * \frac{\partial f}{\partial s}$$

Spatial filtering – LTI filters and convolution

Separability of other LTI filters

	Directional sep.	Multi-scale sep.
Gaussian filter	$\surd \quad (\downarrow * \rightarrow)$	$\surd$
Exponential filter		
Box filter		
Disk filter		
Diamond filter		
Laplacian		-
Sobel		-
Prewitt		-

Separability of other LTI filters

	Directional sep.	Multi-scale sep.
Gaussian filter	$\surd$ $(\downarrow * \rightarrow)$	$\surd$
Exponential filter	x	x
Box filter	$\surd$ $(\downarrow * \rightarrow)$	x
Disk filter	x	x
Diamond filter	x	x
Laplacian	$\surd$ $(\downarrow + \rightarrow)$	-
Sobel	$\surd$ $(\downarrow * \rightarrow)$	-
Prewitt	$\surd$ $(\downarrow * \rightarrow)$	-

LTI filters can be written as a matrix vector product

Functional representation

$$\hat{x}(s_i) = \sum_{s_j \in \mathbb{Z}^2} \nu(s_i - s_j) y(s_j)$$

Vector representation

$$\hat{x} = \boldsymbol{H} y \quad \text{with} \quad h_{i,j} = \nu(s_i - s_j)$$

- Vectors represent objects (here: images)
- Matrices represent linear processings (here: convolution)

Proof in the periodical case.

- Assuming periodical boundary conditions, we get

$$\hat{x}(s_i) = \sum_{j=0}^{n-1} \nu(s_i - s_j) y(s_j)$$

- Let $h_{i,j} = \nu(s_i - s_j)$, $\hat{x}_i = \hat{x}(s_i)$ and $y_j = y(s_j)$:

$$\hat{x}_i = \sum_{j=0}^{n-1} h_{i,j} y_j$$

- Define the matrix $\boldsymbol{H} = (h_{i,j})$, then $\hat{x} = \boldsymbol{H} y$.

$\square$

What does $\boldsymbol{H}$ look like?

1d periodical case

- In 1d, LTI filter stands for linear time invariant filters and reads

$$\hat{x}(t_i) = \sum_{j=0}^{n-1} \nu(t_i - t_j) y(t_j)$$

- Consider $t_i - t_j = i - j$, and let $h_{i,j} = \nu(t_i - t_j) = \nu_{i-j[n]}$.
- $\boldsymbol{H}$ is a **circulant matrix** given by

$$\boldsymbol{H} = \begin{pmatrix} \nu_0 & \nu_{n-1} & \nu_{n-2} & \dots & \nu_2 & \nu_1 \\ \nu_1 & \nu_0 & \nu_{n-1} & \nu_{n-2} & \dots & \nu_2 \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ \nu_{n-1} & \nu_{n-2} & \dots & \nu_2 & \nu_1 & \nu_0 \end{pmatrix}$$

What does $\boldsymbol{H}$ look like?

2d periodical case

- In 2d, $\boldsymbol{H}$ is a **doubly block circulant matrix** given by

$$
\begin{array}{c}
\overbrace{\qquad\qquad\qquad\qquad}^{\text{First line}} \qquad \overbrace{\qquad\qquad\qquad\qquad}^{\text{Second line}} \qquad \cdots \qquad \overbrace{\qquad\qquad\qquad\qquad}^{\text{Last line}} \\
\boldsymbol{H}x = \left(\begin{array}{ccccc|ccccc|c|ccccc}
\nu_{0,0} & \nu_{0,-1} & & & \nu_{0,1} & \nu_{-1,0} & \nu_{-1,-1} & & & \nu_{-1,1} & & \nu_{1,0} & \nu_{1,-1} & & & \nu_{1,1} \\
\nu_{0,1} & \nu_{0,0} & \nu_{0,-1} & & & \nu_{-1,1} & \nu_{-1,0} & \nu_{-1,-1} & & & & \nu_{1,1} & \nu_{1,0} & \nu_{1,-1} & & \\
& \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & & \cdots & & \ddots & \ddots & \ddots & \\
& & \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & & & & & \ddots & \\
\nu_{0,-1} & & & \nu_{0,1} & \nu_{0,0} & \nu_{-1,-1} & & & \nu_{-1,1} & \nu_{-1,0} & & \nu_{1,-1} & & & \nu_{1,1} & \nu_{1,0} \\
\hline
\nu_{1,0} & \nu_{1,-1} & & & \nu_{1,1} & \nu_{0,0} & \nu_{0,-1} & & & \nu_{0,1} & & \nu_{2,0} & \nu_{2,-1} & & & \nu_{2,1} \\
\nu_{1,1} & \nu_{1,0} & \nu_{1,-1} & & & \nu_{0,1} & \nu_{0,0} & \nu_{0,-1} & & & & \nu_{2,1} & \nu_{2,0} & \nu_{2,-1} & & \\
& \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & & \cdots & & \ddots & \ddots & \ddots & \\
& & \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & & & & & \ddots & \\
\nu_{1,-1} & & & \nu_{1,1} & \nu_{1,0} & \nu_{0,-1} & & & \nu_{0,1} & \nu_{0,0} & & \nu_{2,-1} & & & \nu_{2,1} & \nu_{2,0} \\
\hline
& & \vdots & & & & & \vdots & & & \ddots & & & \vdots & & \\
\hline
\nu_{-1,0} & \nu_{-1,-1} & & & \nu_{-1,1} & \nu_{-2,0} & \nu_{-2,-1} & & & \nu_{-2,1} & & \nu_{0,0} & \nu_{0,-1} & & & \nu_{0,1} \\
\nu_{-1,1} & \nu_{-1,0} & \nu_{-1,-1} & & & \nu_{-2,1} & \nu_{-2,0} & \nu_{-2,-1} & & & & \nu_{0,1} & \nu_{0,0} & \nu_{0,-1} & & \\
& \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & & \cdots & & \ddots & \ddots & \ddots & \\
& & \ddots & \ddots & \ddots & & & \ddots & \ddots & \ddots & & & & & \ddots & \\
\nu_{-1,-1} & & & \nu_{-1,1} & \nu_{-1,0} & \nu_{-2,-1} & & & \nu_{-2,1} & \nu_{-2,0} & & \nu_{0,-1} & & & \nu_{0,1} & \nu_{0,0}
\end{array}\right)
\left(\begin{array}{c}
x_{0,0} \\ x_{0,1} \\ \vdots \\ \vdots \\ x_{0,n_2-1} \\
\hline
x_{1,0} \\ x_{1,1} \\ \vdots \\ \vdots \\ x_{1,n_2-1} \\
\hline
\vdots \\
\hline
x_{n_1-1,0} \\ x_{n_1-1,1} \\ \vdots \\ \vdots \\ x_{n_1-1,n_2-1}
\end{array}\right)
\end{array}
$$

Properties of circulant matrices

- Recall that the convolution is commutative: $f * g = g * f$

  $\Rightarrow$ Idem for (doubly block) circulant matrices: $\boldsymbol{H}_1 \boldsymbol{H}_2 = \boldsymbol{H}_2 \boldsymbol{H}_1$

- Two matrices commute if they have the same eigenvectors

  $\Rightarrow$ All circulant matrices share the same eigenvectors

  $\Rightarrow$ **LTI filters acts in the same eigenspace**

What eigenspace is that?

Theorem

- *The n eigenvectors, with unit norm, of any circulant matrix $\boldsymbol{H}$ reads as*

$$e_k = \frac{1}{\sqrt{n}}\left(1, \exp\left(\frac{2\pi i k}{n}\right), \exp\left(\frac{4\pi i k}{n}\right), \ldots, \exp\left(\frac{2(n-1)\pi i k}{n}\right)\right)$$

for $k = 0$ to $n - 1$. *(Note: here i is the imaginary number)*

- *Recall that the eigenvectors (e_k) with unit norm must satisfy:*

$$\boldsymbol{H} e_k = \lambda_k e_k, \quad e_k^* e_l = 0 \quad \textit{if} \quad k \neq l \quad \textit{and} \quad \|e_k\|_2 = 1$$

Challenge: try to prove it yourself.

The (e_k) form a basis in which LTI filters modulates each element pointwise. This will be at the heart of Chapter 3.

Spatial filtering – LTI filters – Limitations

Limitations of LTI filters

- Derivative filters:
  - Detect edges, but
  - Sensitive to noise
- Moving average:
  - Decrease noise, but
  - Do not preserve edges

Difficult object/background separation

LTI filters cannot achieve a good trade-off in terms of noise vs edge separation

Spatial filtering – LTI filters – Limitations

Weak robustness against outliers

Figure 2 – (left) Impulse noise. (center) Gaussian filter $\tau = 5$. (right) $\tau = 11$.

- Even less efficient for impulse noise
- For the best trade-off: structures are lost, noise remains
- Do not adapt to the signal.

Can we achieve better performance by designing an adaptive filter?

Adaptive filtering

Spatial filtering – Adaptive filtering

Linear filter $\Rightarrow$ Non-adaptive filter

- Linear filters are non-adaptive
- The operation does not depend on the signal

☺ Simple, fast implementation

☹ Introduce blur, do not preserve edges

Spatial filtering – Adaptive filtering

Linear filter $\Rightarrow$ Non-adaptive filter

- Linear filters are non-adaptive
- The operation does not depend on the signal

☺ Simple, fast implementation

☹ Introduce blur, do not preserve edges

Adaptive filter $\Rightarrow$ Non-linear filter

- Adapt the filtering to the content of the image
- Operations/decisions depend on the values of y
- Adaptive $\Rightarrow$ non-linear:

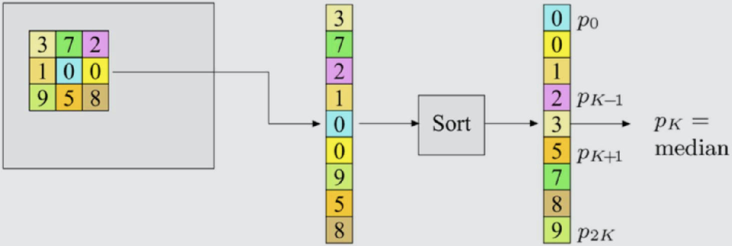
$$\psi(\alpha x + \beta y) \neq \alpha\psi(x) + \beta\psi(y)$$

Since adapting to x or to y is not the same as adapting to $\alpha x + \beta y$.

Median filters

- Try to denoise while respecting main structures

$$\hat{x}_{i,j} = \mathrm{median}(y_{i+k,j+l} \,|\, (k,l) \in \mathcal{N}), \quad \mathcal{N} : \text{neighborhood}$$



Behavior of median filters

- Remove isolated points and thin structures
- Preserve (staircase) edges and smooth corners

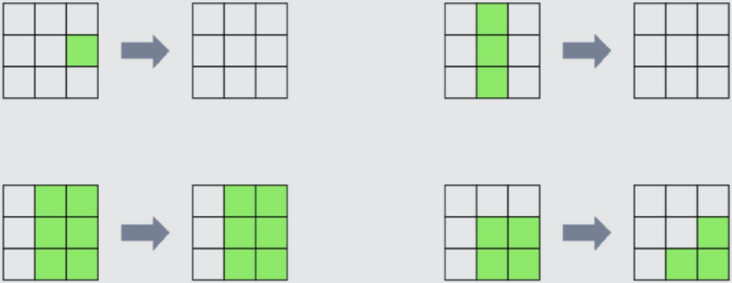


Figure 3 – (left) Impulse noise. (center) 3×3 median filter. (right) 9×9.

Spatial filtering – Median vs Gaussian

Figure 4 – (left) Impulse noise. (center) 9×9 median filter. (right) Gaussian $\tau = 4$.

Spatial filtering – Median vs Gaussian

Figure 5 – (left) Gaussian noise. (center) 5×5 median filter. (right) Gaussian $\tau = 3$.

Morphological operators

- Erosion

$$\hat{x}_{i,j} = \min(y_{i+k,j+l} \,|\, (k,l) \in \mathcal{N})$$

- Dilation

$$\hat{x}_{i,j} = \max(y_{i+k,j+l} \,|\, (k,l) \in \mathcal{N})$$

- $\mathcal{N}$ called structural element

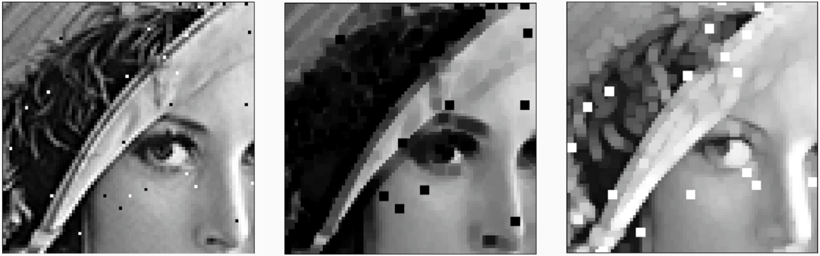


Figure 6 – (left) Salt-and-pepper noise, (center) Erosion, (right) Dilation

Figure 7 – (top) Opening, (bottom) Closing. *(Source: J.Y. Gil & R. Kimmel)*

- Opening: erosion and next dilation (remove small bright elements)
- Closing: dilation and next erosion (remove small dark elements)

Spatial filtering – Global filtering

Local filter

- The operation depends only on the local neighborhood
- ex: Gaussian filter, median filter

☺ Simple, fast implementation

☹ Do not preserve textures (global context)

Global filter

- Adapt the filtering to the global content of the image
- Result at each pixel may depend on all other pixel values
- Idea: Use non-linearity and global information

Local average filter

$$\hat{x}_i = \frac{\sum_{j=1}^{n} w_{i,j} y_j}{\sum_{j=1}^{n} w_{i,j}} \quad \text{with} \quad w_{i,j} = \varphi(\|s_i - s_j\|_2^2)$$

weights depend on the distance between **pixel positions** (linear)

Spatial filtering – Global average filters

Sigma filter [Lee, 1981] / Yaroslavsky filter [Yaroslavsky, 1985]

$$\hat{x}_i = \frac{\sum_{j=1}^{n} w_{i,j} y_j}{\sum_{j=1}^{n} w_{i,j}} \quad \text{with} \quad w_{i,j} = \varphi(\|y_i - y_j\|_2^2)$$

weights depend on the distance between **pixel values** (non-linear)

$$\text{Sigma filter: } \varphi(\alpha) = \begin{cases} 1 & \text{if} \quad \alpha \leqslant \tau^2 \\ 0 & \text{otherwise} \end{cases}$$

Note: is called sigma filter because the threshold τ^2 was called σ in the original paper.

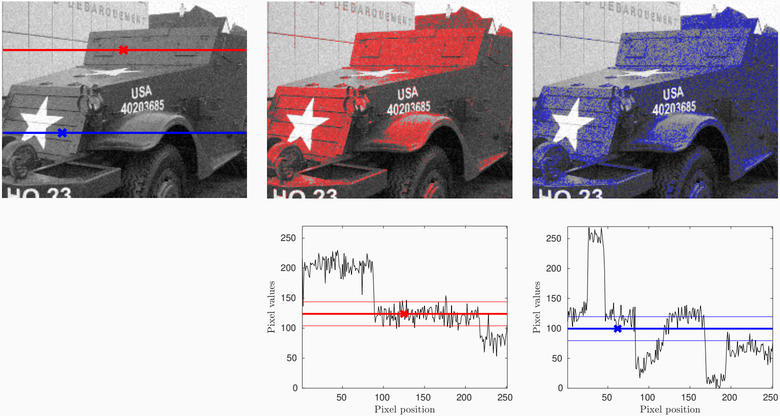


Figure 8 – Selection of pixel candidates in the sigma filter

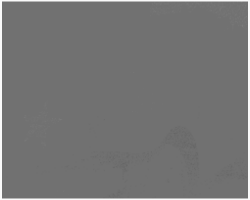
(a) Noisy image $\sigma = 10$

(b) Sigma filter $\tau = 50$

(c) $\tau = 100$

(d) $\tau = 150$

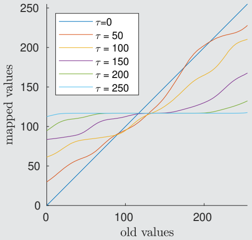
(e) $\tau = 200$



(f) $\tau = 250$

Limitations of Sigma filter

- ☺ Respects edges
- ☹ Produces a loss of contrast: dull effect
- ☹ Does not reduce noise as much
- ☹ Equivalent to a change of histogram:
  - each value is mapped to another one
  - the mapping depends on the image (adaptive/non-linear filtering)



- ☹ Naive implementation: $O(n^2)$
- ☺ Back to $O(n)$ by using histograms

Idea: apply the sigma filter on moving windows
$\equiv$ Mix moving average with sigma filter

Bilateral filter [Tomasi & Manduchi, 1998]

$$\hat{x}_i = \frac{\displaystyle\sum_{j=1}^{n} w_{i,j} y_j}{\displaystyle\sum_{j=1}^{n} w_{i,j}} \quad \text{with} \quad w_{i,j} = \varphi_{\text{space}}(\|s_i - s_j\|_2^2) \times \varphi_{\text{color}}(\|y_i - y_j\|_2^2)$$

Weights depend on both the distance

- between **pixel positions**, and
- between **pixel values**.

- Consider the influence of space and color,
- Closer positions affect more the average,
- Closer intensities affect more the average.

Properties

- Generalization of moving averages and sigma filters.
    - $\varphi_{\text{space}}(\cdot) = 1$: sigma filter
    - $\varphi_{\text{color}}(\cdot) = 1$: moving average
- Spatial constraint: avoid dull effects
- Color constraint: avoid blur effects

Spatial filtering – Bilateral filter

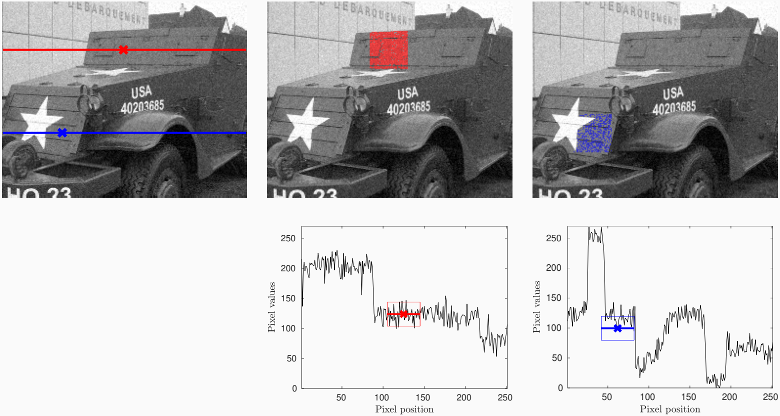


Figure 9 – Selection of pixel candidates in the bilateral filter

Spatial filtering – Bilateral filter

(a) Noisy image $\sigma = 10$

(b) Bilateral filter $\tau_{\text{color}} = 5$

(c) $\tau_{\text{color}} = 20$

(d) $\tau_{\text{color}} = 40$

(e) $\tau_{\text{color}} = 100$

(f) $\tau_{\text{color}} = 200$

$$\varphi_{\text{color}}(\alpha) = \exp\left(-\frac{\alpha}{2\tau_{\text{color}}^2}\right)$$

Spatial filtering – Bilateral filter

(a) Noisy image $\sigma = 10$

(b) Bilateral filter $\tau_{\text{space}} = 5$

(c) $\tau_{\text{space}} = 10$

(d) $\tau_{\text{space}} = 20$

(e) $\tau_{\text{space}} = 50$

(f) $\tau_{\text{space}} = \infty$

$$\varphi_{\text{space}}(\alpha) = \begin{cases} 1 & \text{if} \quad \alpha \leqslant \tau_{\text{space}^2} \\ 0 & \text{otherwise} \end{cases}$$

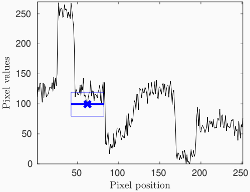
Spatial filtering – Bilateral vs moving average

Figure 10 – (left) Gaussian noise. (center) Moving average. (right) Bilateral filter.

Bilateral filter

- ☺ suppresses more noise while respecting the textures
- ☹ still remaining noises and dull effects

Spatial filtering – Bilateral vs moving average



Why are there remaining noises?

- Below average pixels are mixed with other below average pixels
- Above average pixels are mixed with other above average pixels

Why are there dull effects?

- To counteract the remaining noise effect, τ_{color} should be large

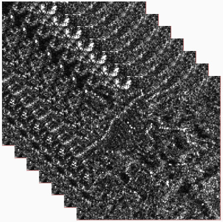
$\Rightarrow$ different things get mixed up together

What is missing? **A more robust way to measure similarity,**

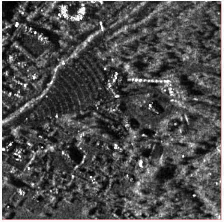
but similarity of what exactly?

Patches and non-local filters

Spatial filtering – Looking for other views



$\underbrace{\hspace{6em}}_{T \text{ noisy observations } y^{(t)}}$



$\underbrace{\hspace{6em}}_{\text{Estimation } \hat{x} \text{ of the unknown signal } x}$

- Sample averaging of T noisy values:

$$\mathbb{E}[\hat{x}_i] = \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} y_i^{(t)}\right] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[y_i^{(t)}] = \frac{1}{T}\sum_{t=1}^{T} x_i = x_i \qquad \text{(unbiased)}$$

$$\text{and} \quad \mathrm{Var}[\hat{x}_i] = \mathrm{Var}\left[\frac{1}{T}\sum_{t=1}^{T} y_i^{(t)}\right] = \frac{1}{T^2}\sum_{t=1}^{T}\mathrm{Var}[y_i^{(t)}] = \frac{1}{T^2}\sum_{t=1}^{T}\sigma^2 = \frac{\sigma^2}{T}$$

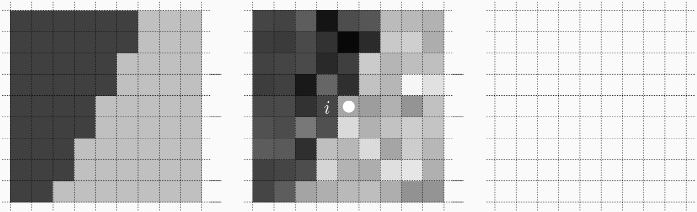
(reduce noise)

- . . . only if the selected values are iid.

similar = close to being iid

$\rightarrow$ How can we select them on a single image?

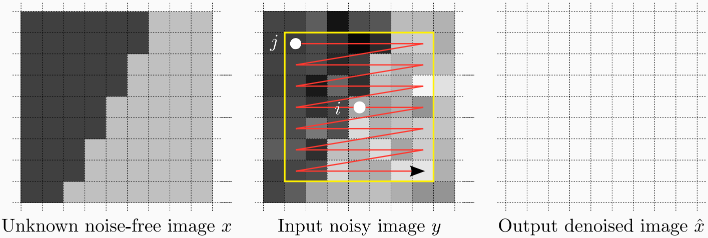
Spatial filtering – Selection-based filtering



Unknown noise-free image x — Input noisy image y — Output denoised image $\hat{x}$

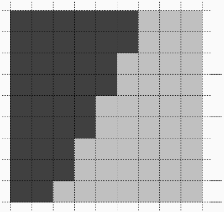
General idea

- Goal: estimate the image x from the noisy image y
- Choose a pixel i to denoise

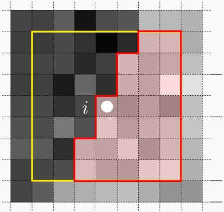


General idea

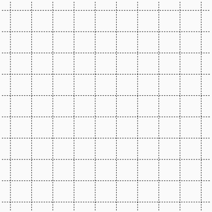
- Goal: estimate the image x from the noisy image y
- Choose a pixel i to denoise
  - Inspect the pixels j around the pixel of interest i
  - Select the suitable candidates j
  - Average their values and update the value of i



Unknown noise-free image x



Input noisy image y

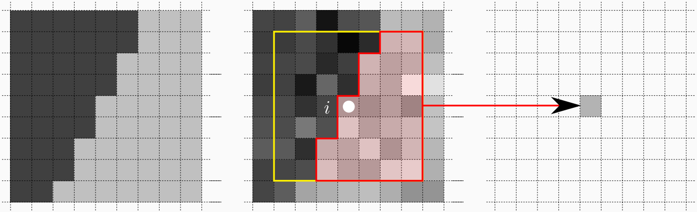


Output denoised image $\hat{x}$

General idea

- Goal: estimate the image x from the noisy image y
- Choose a pixel i to denoise
    - Inspect the pixels j around the pixel of interest i
    - Select the suitable candidates j
    - Average their values and update the value of i

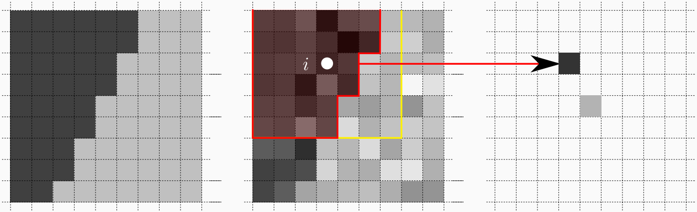
Spatial filtering – Selection-based filtering



Unknown noise-free image x Input noisy image y Output denoised image $\hat{x}$

General idea

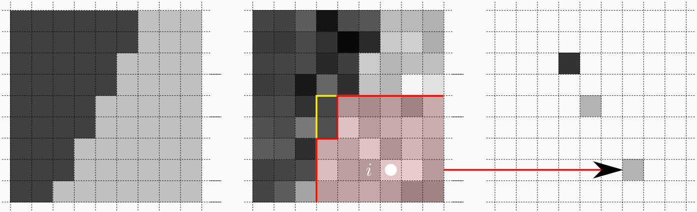
- Goal: estimate the image x from the noisy image y
- Choose a pixel i to denoise
  - Inspect the pixels j around the pixel of interest i
  - Select the suitable candidates j
  - Average their values and update the value of i



Unknown noise-free image x — Input noisy image y — Output denoised image $\hat{x}$

General idea

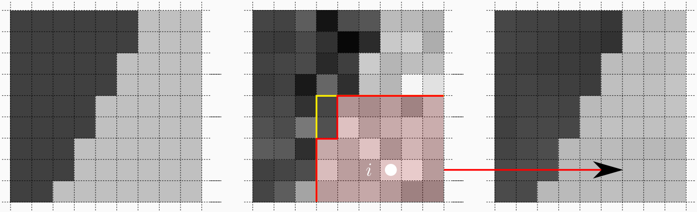
- Goal: estimate the image x from the noisy image y
- Choose a pixel i to denoise
  - Inspect the pixels j around the pixel of interest i
  - Select the suitable candidates j
  - Average their values and update the value of i
- Repeat for all pixels i



Unknown noise-free image x Input noisy image y Output denoised image $\hat{x}$

General idea

- Goal: estimate the image x from the noisy image y
- Choose a pixel i to denoise
  - Inspect the pixels j around the pixel of interest i
  - Select the suitable candidates j
  - Average their values and update the value of i
- Repeat for all pixels i



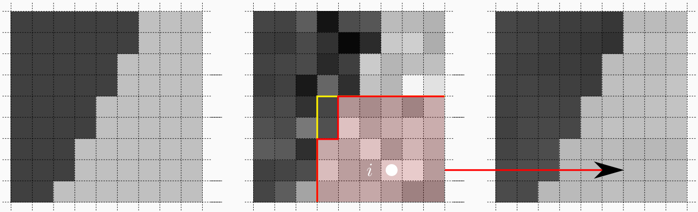
Unknown noise-free image x

Input noisy image y

Output denoised image $\hat{x}$

General idea

- Goal: estimate the image x from the noisy image y
- Choose a pixel i to denoise
    - Inspect the pixels j around the pixel of interest i
    - Select the suitable candidates j
    - Average their values and update the value of i
- Repeat for all pixels i



Unknown noise-free image x | Input noisy image y | Output denoised image $\hat{x}$

Selection rules

$$\hat{x}_i = \frac{\sum_j w_{i,j} y_j}{\sum_j w_{i,j}} \quad \text{where} \quad w_{i,j} = \begin{cases} 1 & \text{if } \boxed{\|s_i - s_j\| \leqslant \tau} \\ 0 & \text{otherwise} \end{cases} \leftarrow \text{Moving average}$$

$$w_{i,j} = \begin{cases} 1 & \text{if } \boxed{\|y_i - y_j\| \leqslant \tau} \\ 0 & \text{otherwise} \end{cases} \leftarrow \text{Sigma filter}$$

$$w_{i,j} = \begin{cases} 1 & \text{if } \boxed{x_i = x_j} \\ 0 & \text{otherwise} \end{cases} \leftarrow \text{Oracle}$$

How to choose suitable pixels j to combine?

Spatial filtering – Patches

Definition [Oxford dictionary]

patch (noun): *A small area or amount of something.*

Image patches: sub-regions of the image

- shape: typically rectangular
- size: much smaller than image size

$\rightarrow$ most common use:
square regions between
5×5 and 21×21 pixels

$\rightarrow$ trade-off:
size $\nearrow$ $\Rightarrow$ more distinctive/informative
size $\searrow$ $\Rightarrow$ easier to model/learn/match

---

non-rectangular / deforming shapes:
computational complexity $\nearrow$

patches capture local context: geometry and texture

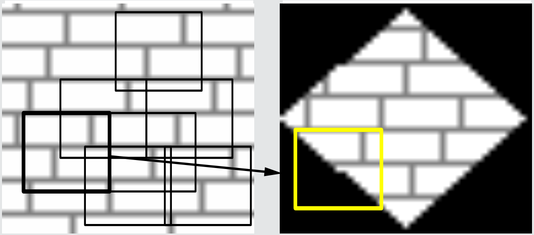
Spatial filtering – Patches for texture synthesis

Copying/pasting similar patches yields impressive texture synthesis:

Texture synthesis method by Efros and Leung (1999)

To generate a new pixel value:

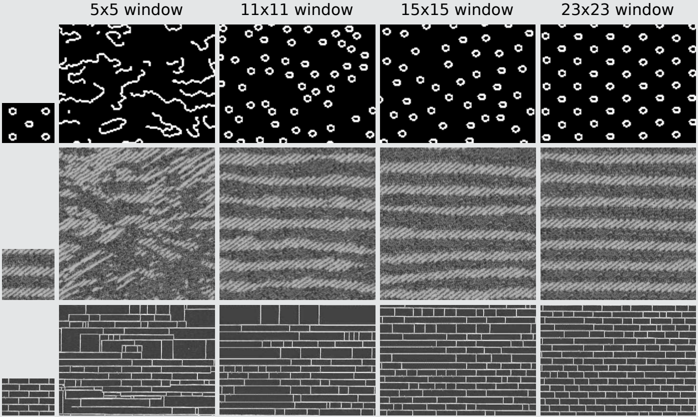
- extract the surrounding patch (yellow)
- find similar patches in the reference image
- randomly pick one of them
- use the value of the central pixel of that patch



Spatial filtering – Patches for texture synthesis

Copying/pasting similar patches yields impressive texture synthesis:

Texture synthesis method by Efros and Leung (1999)



Spatial filtering – Non-local means

Bilateral filter [Tomasi & Manduchi, 1998]

$$\hat{x}_i = \frac{\sum_{j \in \mathcal{N}_i} w_{i,j} y_j}{\sum_{j \in \mathcal{N}_i} w_{i,j}} \quad \text{with} \quad w_{i,j} = \varphi_{\text{space}}(\|s_i - s_j\|_2^2) \times \varphi_{\text{color}}(\|y_i - y_j\|_2^2)$$

weights depend on the distance between **pixel positions** and **pixel values**

Non-local means [Buades at al, 2005, Awate et al, 2005]

$$\hat{x}_i = \frac{\sum_{j \in \mathcal{N}_i} w_{i,j} y_j}{\sum_{j \in \mathcal{N}_i}^{n} w_{i,j}} \quad \text{with} \quad w_{i,j} = \varphi(\|\mathcal{P}_i y - \mathcal{P}_j y\|_2^2)$$

- $\mathcal{N}_i$: large neighborhood of i, called search window (typically 21×21)
- $\mathcal{P}_i$: operator extracting a small window, *patch*, at i (typically 7×7)

weights in a **large search window** depend on the distance between **patches**

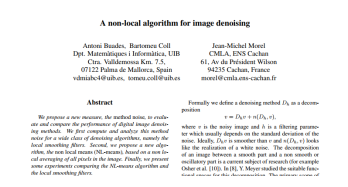
Remarks

The term *non-local* refers to that disconnected pixels are mixed together.

The Sigma, Yaroslavsky and Bilateral filters are then also non-local.

But $\underbrace{\textit{Non-Local means}}_{\text{(or } \textit{NL-means}\text{)}}$ always refers to the one using patches.

A similar algorithm was concurrently proposed under the name UINTA.



A non-local algorithm for image denoising

Antoni Buades, Bartomeu Coll
Dpt. Matemàtiques i Informàtica, UIB
Ctra. Valldemossa Km. 7.5,
07122 Palma de Mallorca, Spain
vdmiabc4@uib.es, tomeu.coll@uib.es

Jean-Michel Morel
CMLA, ENS Cachan
61, Av du Président Wilson
94235 Cachan, France
morel@cmla.ens-cachan.fr

Abstract

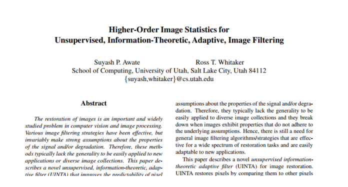
We propose a new measure, the method noise, *to evaluate and compare the performance of digital image denoising methods. We first compute and analyze this method noise for a wide class of denoising algorithms, namely the local smoothing filters. Second, we propose a new algorithm, the* non local means (NL-means), *based on a non local averaging of all pixels in the image. Finally, we present some experiments comparing the NL-means algorithm and the local smoothing filters.*

Formally we define a denoising method D_h as a decomposition

$$v = D_h v + n(D_h, v),$$

where v is the noisy image and h is a filtering parameter which usually depends on the standard deviation of the noise. Ideally, $D_h v$ is smoother than v and $n(D_h, v)$ looks like the realization of a white noise. The decomposition of an image between a smooth part and a non smooth or oscillatory part is a current subject of research (for example Osher et al. [10]). In [8], Y. Meyer studied the suitable func-

NL-means [Buades et al, CVPR 2005]



Higher-Order Image Statistics for Unsupervised, Information-Theoretic, Adaptive, Image Filtering

Suyash P. Awate   Ross T. Whitaker
School of Computing, University of Utah, Salt Lake City, Utah 84112
{suyash,whitaker}@cs.utah.edu

Abstract

The restoration of images is an important and widely studied problem in computer vision and image processing. Various image filtering strategies have been effective, but invariably make strong assumptions about the properties of the signal and/or degradation. Therefore, these methods typically lack the generality to be easily applied to new applications or diverse image collections. This paper describes a novel unsupervised, information-theoretic, adap-

assumptions about the properties of the signal and/or degradation. Therefore, they typically lack the generality to be easily applied to diverse image collections and they break down when images exhibit properties that do not adhere to the underlying assumptions. Hence, there is still a need for general image filtering algorithms/strategies that are effective for a wide spectrum of restoration tasks and are easily adaptable to new applications.

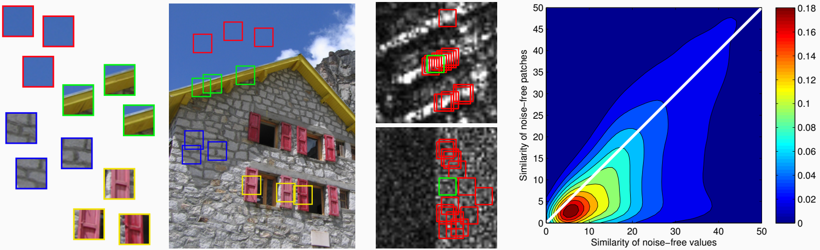
This paper describes a novel *unsupervised information-theoretic adaptive filter* (UINTA) for image restoration. UINTA restores pixels by comparing them to other pixels

UINTA [Awate et al, CVPR 2005]

Non-local approach **[Buades at al, 2005, Awate et al, 2005]**

- Local filters: average neighborhood pixels
- Non-local filters: average pixels being in a similar context

$$\hat{x}_i = \frac{\sum_j w_{i,j} y_j}{\sum_j w_{i,j}}$$



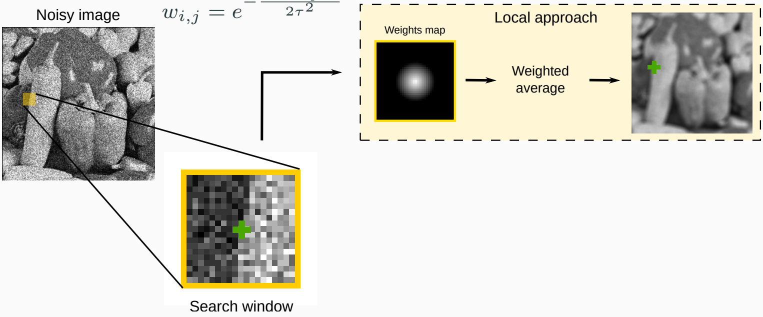
Patches are redundant in most types of images (large noise reduction)
and **similar ones tend to share the same underlying noise-free values** (unbiasedness)

Non-local approach [Buades at al, 2005, Awate et al, 2005]

- Local filters: average neighborhood pixels
- Non-local filters: average pixels being in a similar context

$$\hat{x}_i = \frac{\sum_j w_{i,j} y_j}{\sum_j w_{i,j}}$$

$$w_{i,j} = e^{-\frac{\|s_i - s_j\|_2^2}{2\tau^2}}$$



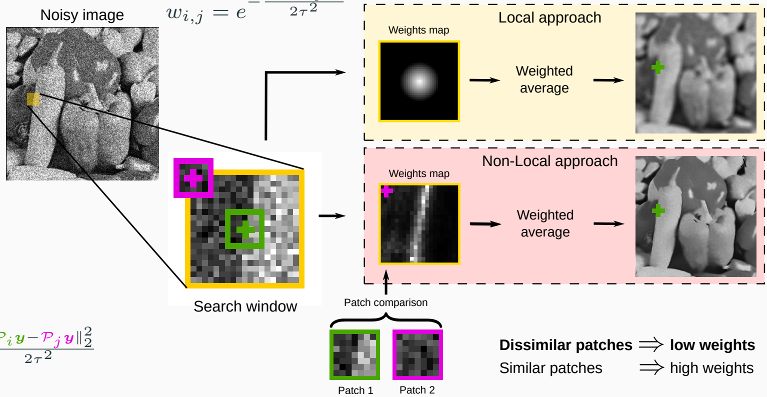
Spatial filtering – Non-local means

Non-local approach [Buades at al, 2005, Awate et al, 2005]

- Local filters: average neighborhood pixels
- Non-local filters: average pixels being in a similar context

$$\hat{x}_i = \frac{\sum_j w_{i,j} y_j}{\sum_j w_{i,j}}$$

$$w_{i,j} = e^{-\frac{\|s_i - s_j\|_2^2}{2\tau^2}}$$



$$w_{i,j} = e^{-\frac{\|\mathcal{P}_i y - \mathcal{P}_j y\|_2^2}{2\tau^2}}$$

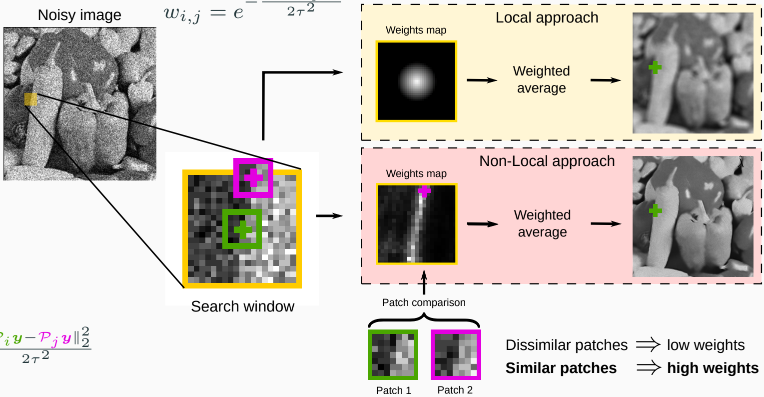
Spatial filtering – Non-local means

Non-local approach [Buades at al, 2005, Awate et al, 2005]

- Local filters: average neighborhood pixels
- Non-local filters: average pixels being in a similar context

$$\hat{x}_i = \frac{\sum_j w_{i,j} y_j}{\sum_j w_{i,j}}$$

$$w_{i,j} = e^{-\frac{\|s_i - s_j\|_2^2}{2\tau^2}}$$



$$w_{i,j} = e^{-\frac{\|\mathcal{P}_i \boldsymbol{y} - \mathcal{P}_j \boldsymbol{y}\|_2^2}{2\tau^2}}$$

Example (Map of non-local weights)

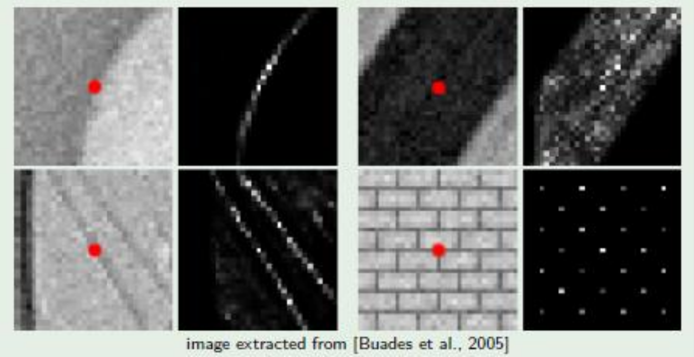
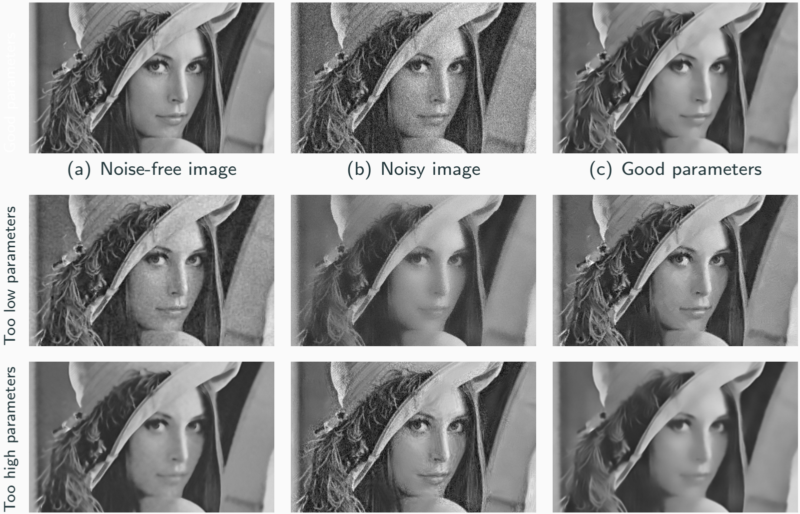


Figure 11 – Image extracted from [Buades et al., 2005]



(a) Noise-free image (b) Noisy image (c) Good parameters

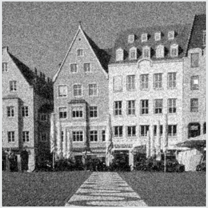
(d) Search window size (e) Patch size (f) Bandwidth τ

Figure 12 – Influence of the three main parameters of the NL means on the solution.

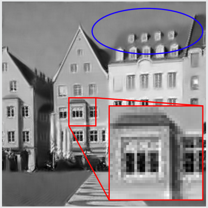
Spatial filtering – Non-local means

Limitations of NL-means

- ☺ Respects edges
- ☺ Good for texture
- ☹ Remaining noise around rare patches
- ☹ Loses/blurs details with low SNR



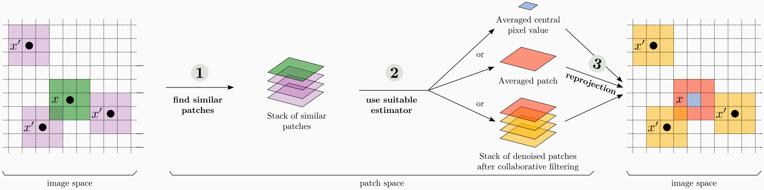
(a) Noisy image



(b) NL-means

(c) BM3D

- ☹ Naive implementation: $O(n|\mathcal{N}||\mathcal{P}|)$ ($\sim$ 1 minute for 256×256 image)
- ☺ Using integral tables: $O(n|\mathcal{N}|)$ (few seconds for 256×256 image)
- ☺ Or FFT: $O(n|\mathcal{N}| \log |\mathcal{N}|)$



More elaborate schemes mostly rely on patches and use more sophisticated estimators than the average

But we will need to study some more of the basics first...

Questions?

Next class: basics of filtering II

Sources, images courtesy and acknowledgment

L. Condat
L. Denis
J.Y. Gil
A. Horodniceanu
H. Jiang

I. Kokkinos
R. Kimmel
F. Luisier
S. Seitz

M. Thompson
V.-T. Ta
C. Vonesh
Wikipedia