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)

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

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

$\xrightarrow{H}$

(b) Observation y

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’

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

$\xrightarrow{\psi}$

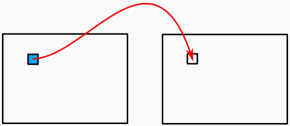
(b) Estimate $\hat{x}$

Basics of filtering

Action of filters

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

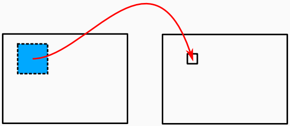
Punctual:



New pixel value depends on the input one only

e.g., change of contrast

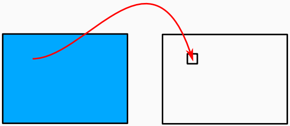
Local:



New pixel value depends on the surrounding input pixels

e.g., averaging/convolutions

Global:



New pixel value depends on the whole input image

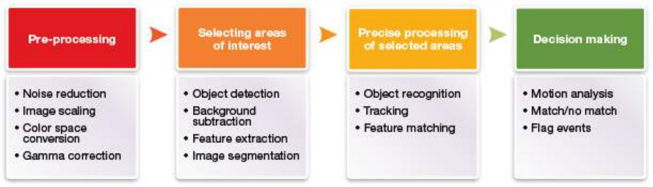
e.g., sigma filter

Basics of filtering

Filters

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

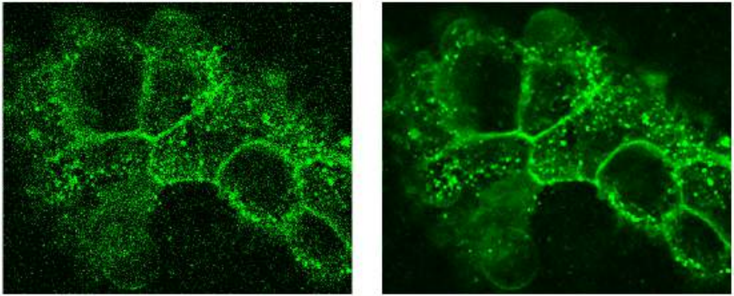
CV pipeline



Source: Mike Thompson

Improve/denoise/detect

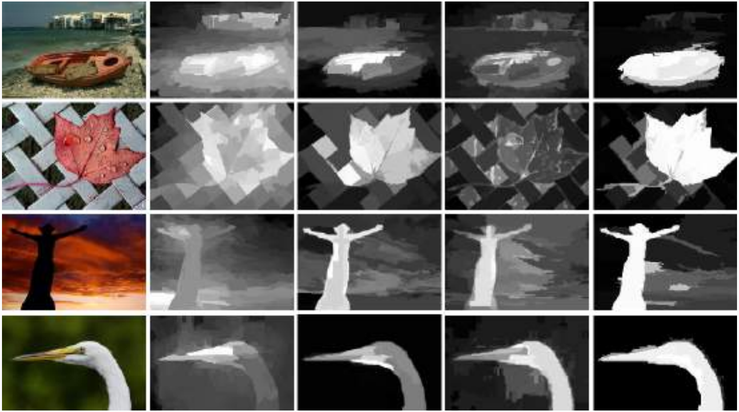
Improve/**denoise**/detect



Fibroblast cells and microbreads (fluorescence microscopy)

Source: F. Luisier & C. Vonesch

Improve/denoise/**detect**



Foreground/Background separation

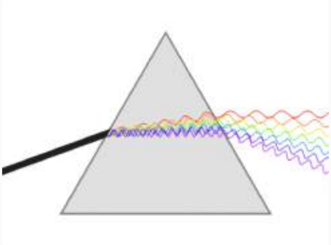
Source: H. Jiang, et al.

Basics of filtering

Standard filters

Two main approaches:

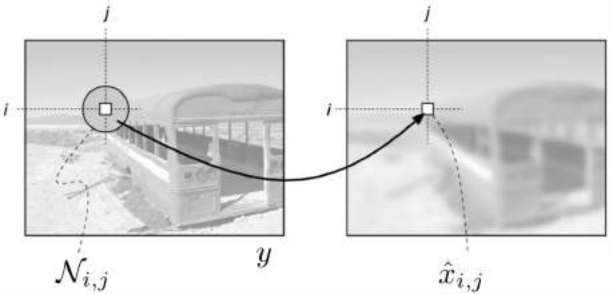
- **Spatial domain:** use the pixel grid / spatial neighborhoods
- **Spectral domain:** use Fourier transform, cosine transform, ...

Spatial  Spectral

Spatial filtering

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



Spatial filtering – Moving average

Moving average

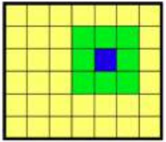
$$\hat{x}_{i,j} = \frac{1}{\text{Card}(\mathcal{N})} \sum_{(k,l)\in\mathcal{N}_{i,j}} y_{k,l}$$

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

Spatial filtering – Moving average

Moving average

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

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

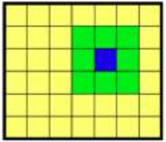
- Boxcar filter: $\mathcal{N}_{i,j} = \{(k,l) \; ; \; |i-k| \leqslant \tau \quad \text{and} \quad |j-l| \leqslant \tau\}$
- Diskcar filter: $\mathcal{N}_{i,j} = \left\{(k,l) \; ; \; |i-k|^2 + |j-l|^2 \leqslant \tau^2\right\}$

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

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



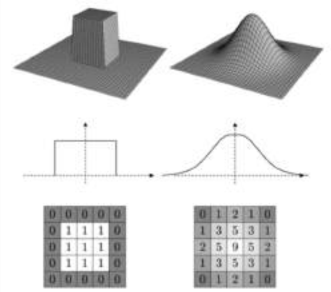
Parameters:

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

Spatial filtering – Moving average

Moving weighted average

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

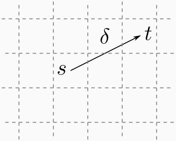


- Neighboring filter: $w_{i,j} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathcal{N} \\ 0 & \text{otherwise} \end{cases}$
- Gaussian kernel: $w_{i,j} = \exp\left(-\frac{i^2+j^2}{2\tau^2}\right)$
- Exponential kernel: $w_{i,j} = \exp\left(-\frac{\sqrt{i^2+j^2}}{\tau}\right)$

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

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

- $\varphi : \mathbb{R}^+ \to \mathbb{R}$: kernel function (⚠ $\neq$ convolution kernel)

- Often, φ satisfies
  - $\varphi(0) = 1$,
  - $\displaystyle\lim_{\alpha \to \infty} \varphi(\alpha) = 0$,
  - φ non-increasing: $\alpha > \beta \Rightarrow \varphi(\alpha) \geqslant \varphi(\beta)$.

Example

- Box filter

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

- Disk filter

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

- Gaussian filter

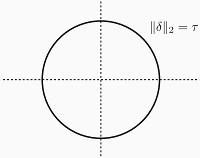
$$\varphi(\alpha) = \exp\left(-\frac{\alpha^2}{2\tau^2}\right) \quad \text{and} \quad \mathrm{length}(\delta) = \|\delta\|_2$$

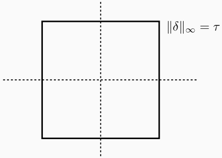
- Exponential filter

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

Reminder:

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





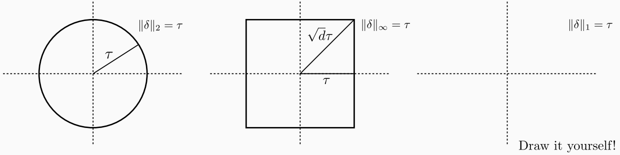
Spatial filtering – Moving average

- 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)

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)

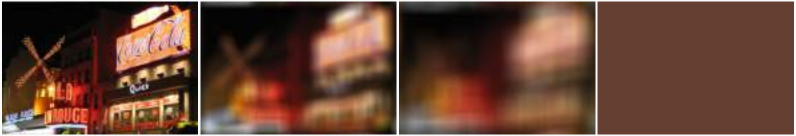
What would provide $\varphi(\alpha) = \begin{cases} 1 & \textbf{if} \quad \alpha \leqslant \tau \\ 0 & \textbf{otherwise} \end{cases}$ **and** $\mathrm{length}(\delta) = \|\delta\|_1$**?**



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

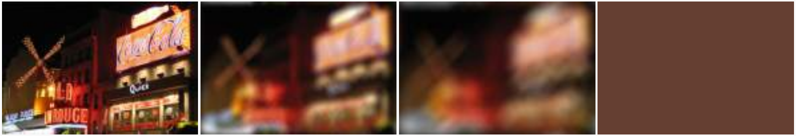
Spatial filtering – Moving average

Box filter



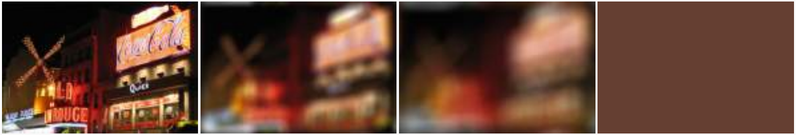
(a) $\tau = 1$ (b) $\tau = 20$ (c) $\tau = 40$ (d) $\tau = 10^3$

Diamond filter



(e) $\tau = 1$ (f) $\tau = 20$ (g) $\tau = 40$ (h) $\tau = 10^3$

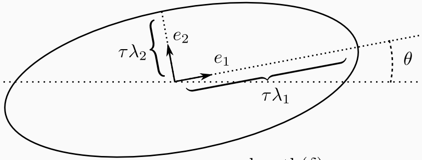
Disk filter



(i) $\tau = 1$ (j) $\tau = 20$ (k) $\tau = 40$ (l) $\tau = 10^3$

Spatial filtering

How to express anisotropy?



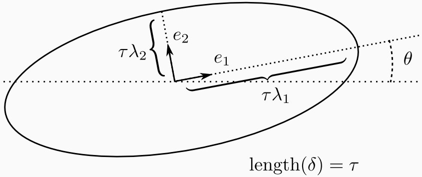
length$(\delta) = \tau$

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

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

Spatial filtering

How to express anisotropy?

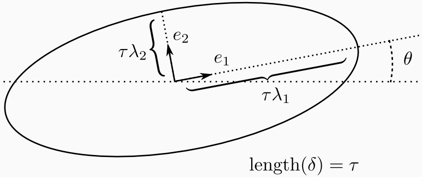


- $\|e_1\|_2 = 1$
- $\|e_2\|_2 = 1$
- $\langle e_1,\, e_2 \rangle = 0$

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

Spatial filtering

How to express anisotropy?



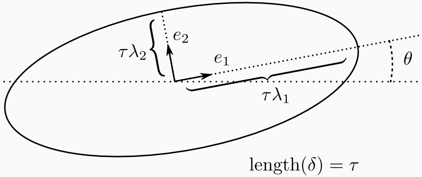
- $\|e_1\|_2 = 1$
- $\|e_2\|_2 = 1$
- $\langle e_1,\, e_2 \rangle = 0$

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

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

Spatial filtering

How to express anisotropy?



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

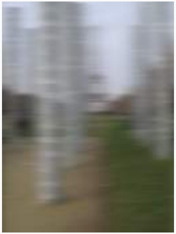
$$\mathrm{length}(\delta) = \sqrt{\delta^T \Sigma^{-1} \delta} \quad \text{where} \quad \Sigma = \underbrace{\begin{pmatrix} e_1 & e_2 \end{pmatrix} \begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix} \begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix}}_{\text{eigen-decomposition}}$$

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

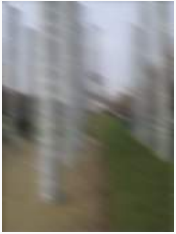
$$\text{indeed,} \quad e_1 = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}, e_2 = \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix} \quad \text{i.e.} \quad R = \underbrace{\begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}}_{\text{rotation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

Spatial filtering – Moving average

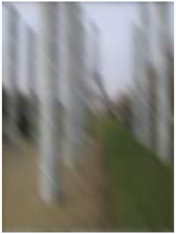
(a) y



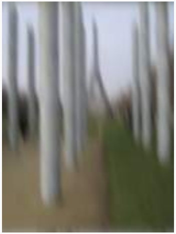
(b) $\hat{x}$, $\theta = 0°$



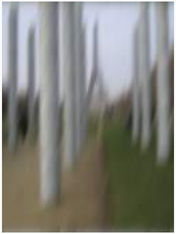
(c) $\theta = 26°$



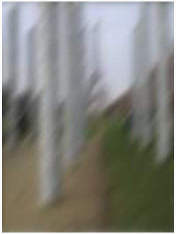
(d) $\theta = 51°$



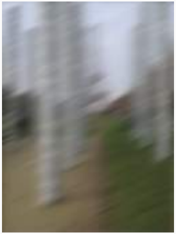
(e) $\theta = 77°$



(f) $\theta = 103°$



(g) $\theta = 129°$



(h) $\theta = 154°$

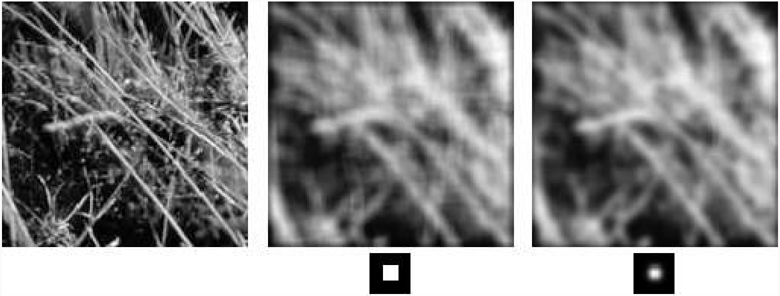
Moving average for denoising?

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: reduce noise ☺,
  but loss of resolution, blurry aspect, remove edges ☹

Spatial filtering – Moving average for denoising

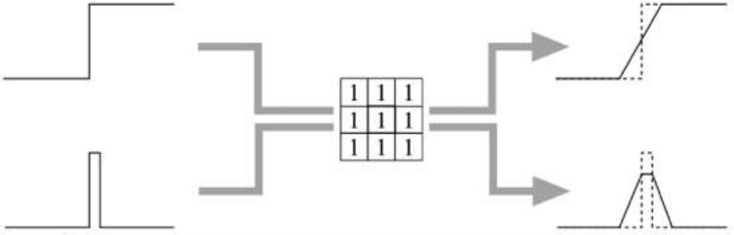


Image blur $\Rightarrow$ No more edges $\Rightarrow$ Structure destruction
$\Rightarrow$ Reduction of image quality

Spatial filtering – Moving average for denoising

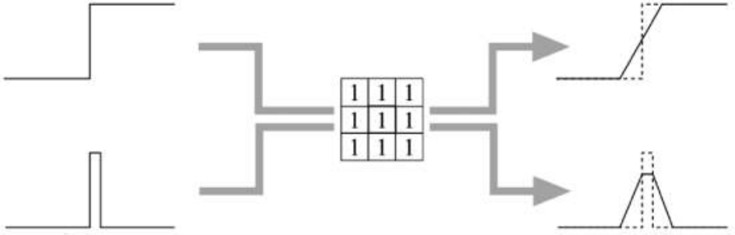
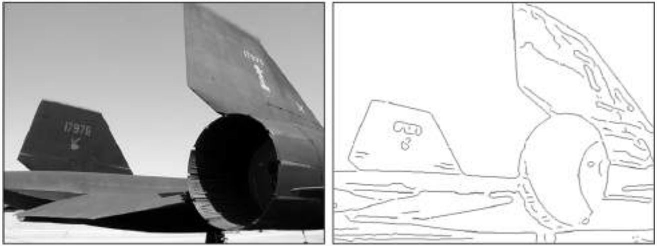


Image blur $\Rightarrow$ No more edges $\Rightarrow$ Structure destruction
$\Rightarrow$ Reduction of image quality

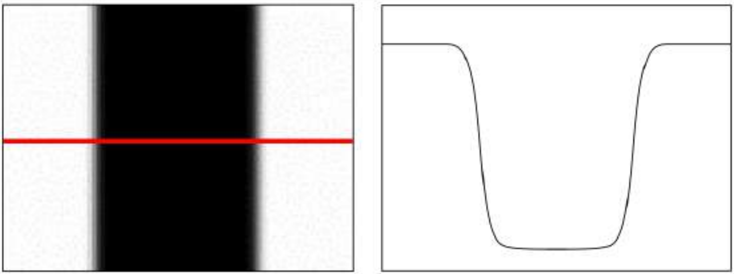
What is an edge?

Edges?

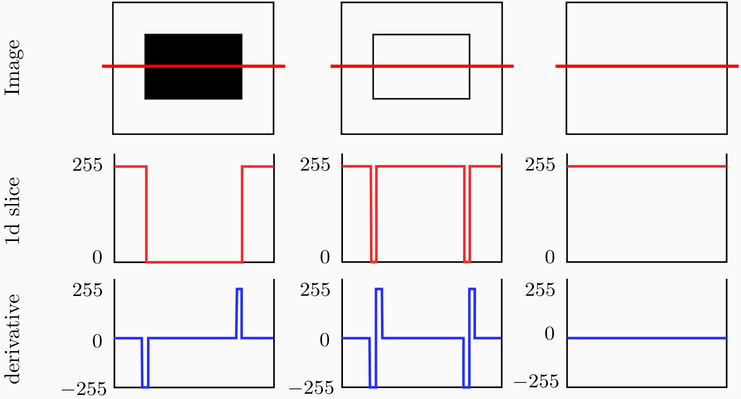
- Separation between objects, important parts of the image
- Necessary for vision in order to reconstruct objects



Spatial filtering – Edges



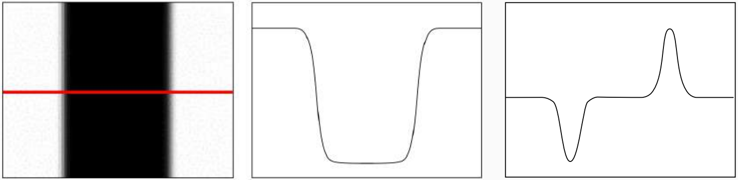
Edge: More or less brutal change of intensity



- 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 build 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

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

Spatial filtering – Derivative filters

Derivative of 1d signals

- Can be written as a filter

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

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

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

Spatial filtering – Derivative filters

Derivative of 2d signals

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

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

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

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

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

Forward | Backward | Centered

Spatial filtering – Derivative filters

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

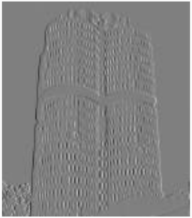
Laplacian of 2d signals

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

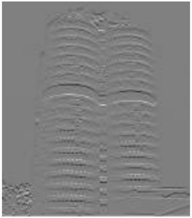
$$\Delta x = \frac{\partial^2 x}{\partial s_1^2} + \frac{\partial^2 x}{\partial s_2^2}$$

- 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: $h = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$

(a) x



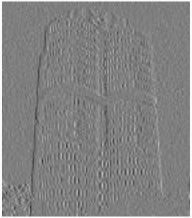
(b) $\nabla_1 x$



(c) $\nabla_2 x$

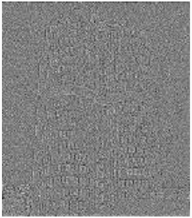
(d) Δx

(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)

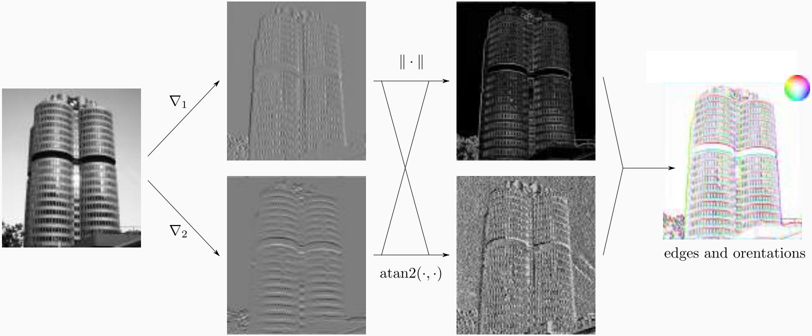
$$\kappa_1 = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \kappa_2 = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}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

Spatial filtering – Derivative filters

Edge detection

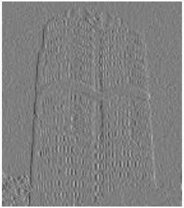
Based on the norm (and angle) of the discrete approximation of the gradient

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



Spatial filtering – Derivative filters

(a) x

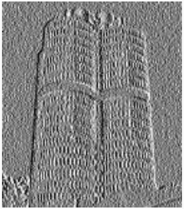


(b) $\nabla_1 x$

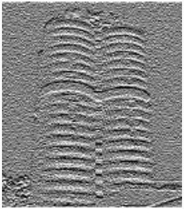
(c) $\nabla_2 x$

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

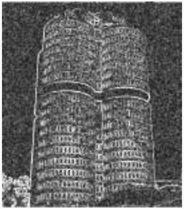
(e) x



(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?

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

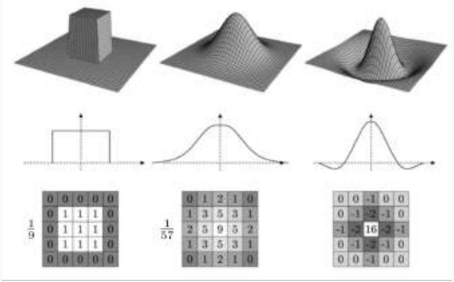
$$\psi(y)_{i,j} = \sum_{(k,l)\in\mathbb{Z}^2} \kappa_{k,l} y_{i+k,j+l}$$

The reciprocal holds true

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

Linear translation-invariant filters

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



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, ...

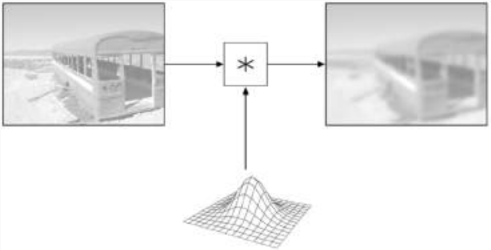
Spatial filtering – Linear translation-invariant filters

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

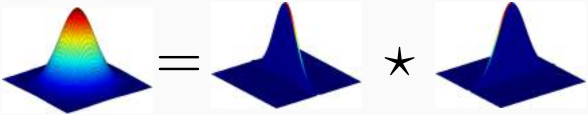
Properties of the convolution product

- **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 $O(s^2 n_1 n_2)$)

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

(Complexity $O(\qquad)$)

$\square$

Directional separability of Gaussians.

$$
\begin{aligned}
(y * \mathcal{G}_\tau^{\text{2d}})_{i,j} &= \frac{1}{Z} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \exp\left(-\frac{k^2+l^2}{2\tau^2}\right) y_{i-k,j-l} \\
&\approx \underbrace{\frac{1}{Z} \sum_{k=-q}^{q} \sum_{l=-q}^{q} \exp\left(-\frac{k^2+l^2}{2\tau^2}\right) y_{i-k,j-l}}_{\text{Restriction to a } s \times s \text{ window, } s = 2q+1}
\end{aligned}
$$

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

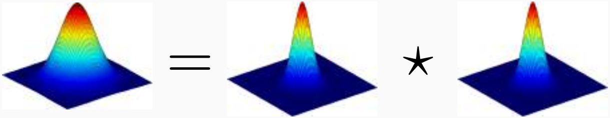
$$
\approx \frac{1}{Z} \underbrace{\sum_{k=-q}^{q} \exp\left(-\frac{k^2}{2\tau^2}\right) \underbrace{\sum_{l=-q}^{q} \exp\left(-\frac{l^2}{2\tau^2}\right) y_{i-k,j-l}}_{\propto (y * \mathcal{G}^{\text{1d horizontal}})_{i-k,j}}}_{\propto (y * \mathcal{G}^{\text{1d horizontal}}) * \mathcal{G}^{\text{1d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

(Complexity $O(s n_1 n_2)$)

$\square$

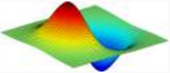
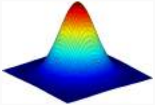
Spatial filtering – LTI filters and convolution

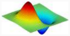
- **Multi-scale separability of Gaussians:** (Continuous case)

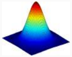


$$\mathcal{G}_{\tau_1^2} * \mathcal{G}_{\tau_2^2} = \mathcal{G}_{\tau_1^2+\tau_2^2}$$

- **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		-

Spatial filtering – LTI filters and convolution

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	$\surd$ $(\downarrow * \rightarrow)$	-

Spatial filtering – LTI and linear algebra

LTI filters can be written as a matrix vector product

Functional representation

$$\hat{x}(s) = \sum_{\delta \in \mathbb{Z}^2} \kappa^*(\delta) y(s + \delta) = \sum_{\delta \in \mathbb{Z}^2} \nu(\delta) y(s - \delta)$$

Vector representation

$$\hat{x} = Hy \quad \text{with} \quad h_{i,j} = \kappa^*(s_j - s_i) = \nu(s_i - s_j)$$

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

Proof in the periodical case.

- Let $\delta = s_j - s_i$. Assuming periodical boundary conditions, we get

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

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

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

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

$\square$

Spatial filtering – LTI and linear algebra

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

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

What does H look like?

2d periodical case

- In 2d, H is a **doubly block circulant matrix** given by

$$
\begin{array}{cccc}
\overbrace{\hphantom{\nu_{0,0}\quad\nu_{0,-1}\quad\nu_{0,-1}\quad\nu_{0,1}\quad\nu_{0,1}}}^{\text{First line}} &
\overbrace{\hphantom{\nu_{0,0}\quad\nu_{0,-1}\quad\nu_{0,-1}\quad\nu_{0,1}\quad\nu_{0,1}}}^{\text{Second line}} &
\cdots &
\overbrace{\hphantom{\nu_{0,0}\quad\nu_{0,-1}\quad\nu_{0,-1}\quad\nu_{0,1}\quad\nu_{0,1}}}^{\text{Last line}}
\end{array}
$$

$$
Hx = \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_{1,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)
$$

Spatial filtering – Properties of circulant matrices

Properties of circulant matrices

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

  $\Rightarrow$ Idem for (doubly block) circulant matrices: $H_1 H_2 = H_2 H_1$

- 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**

(to be continued later...)

Spatial filtering – Properties of circulant matrices

Theorem (Proof in exercise)

- *The n eigenvectors, with unit norm, of any circulant matrix 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 $n - 1$. *(NB: here i is the imaginary number)*

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

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

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 ⇒ 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 ⇒ Non-linear filter

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

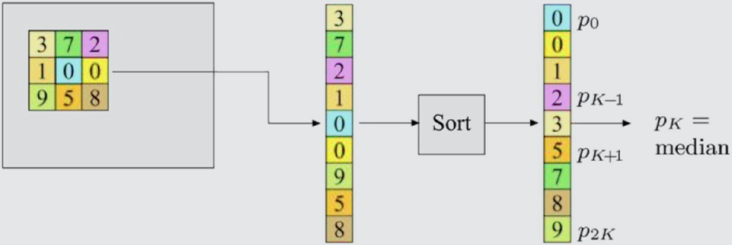
$$\psi(\alpha x + \beta y) \neq \alpha\psi(x) + \beta\psi(y)$$

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

Median filters

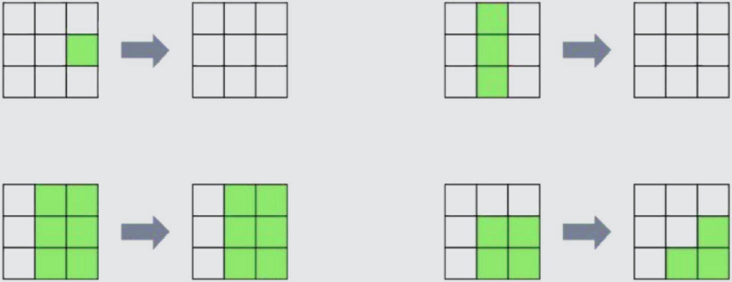
- Try to denoise while respecting main structures

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



Behavior of median filters

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



Spatial filtering – Median filter

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

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

- Dilatation

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

- $\mathcal{N}$ called structural element

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

Spatial filtering – Morphological operators

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

- Closing: dilatation and next erosion
- Opening: erosion and next dilatation

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

Spatial filtering – Global average filters

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)

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

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

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

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

Spatial filtering – Sigma filter

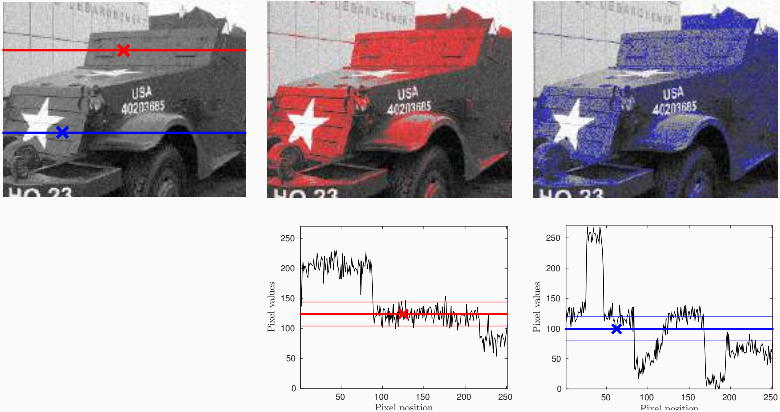


Figure 8 – Selection of pixel candidates in the sigma filter

Spatial filtering – Sigma filter

(a) Noisy image $\sigma = 10$

(b) Sigma filter $\tau = 50$

(c) $\tau = 100$

(d) $\tau = 150$

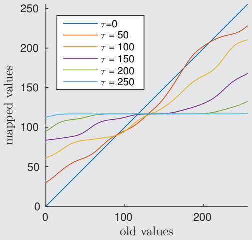
(e) $\tau = 200$

(f) $\tau = 250$

Spatial filtering – Sigma filter

Limitations of Sigma filter

- ☺ Respect edges
- ☹ Produce a loss of contrast: dull effect
- ☹ Do 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

Spatial filtering – Bilateral filter

Bilateral filter [Tomasi & Manduchi, 1998]

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

Weights depend on both the distance

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

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

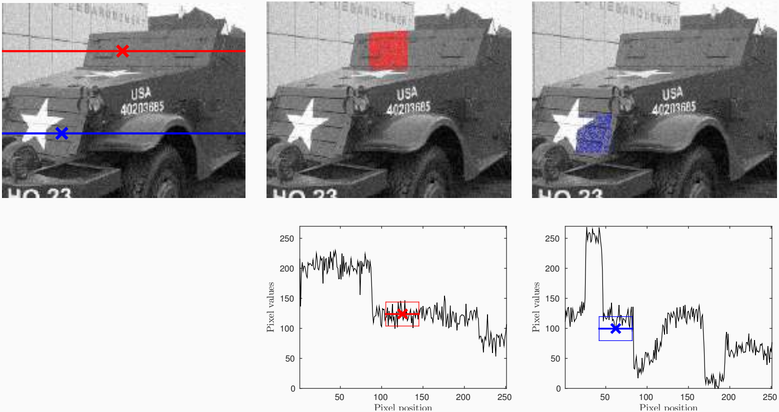


Figure 9 – Selection of pixel candidates in the bilateral filter

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

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

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

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

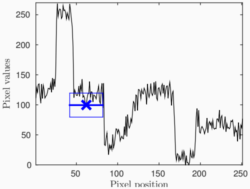
Spatial filtering – Bilateral vs moving average

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

Bilateral filter

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

Spatial filtering – Bilateral vs moving average



Why is there remaining noises?

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

Why is 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?