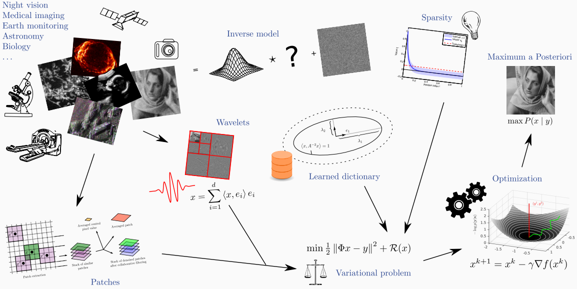
Formation école doctorale :
Introduction au traitement d'images

Introduction to image processing

Course II – Contrast change and Spatial Filtering

Bruno Galerne

Mardi 21 juin 2022



Credits

Several slides from **Charles Deledalle's** course "UCSD ECE285 Image and video restoration" (30 $\times$ 50 minutes course) given at UCSD (University of California, San Diego) and Julie Delon's course "Perception, acquisition et analyse d'images" at Université de Paris.

www.charles-deledalle.fr/

https://delon.wp.imt.fr/

Today's program

Content:

- Contrast change in images.
- Spatial filters, linear (= spatial convolution) and non-linear.

Image Contrast

What is a contrast?

Definition (Cambridge dictionary)

contrast, *noun*: an obvious difference between two or more things.

Low / High contrast

(source wikipedia)

What is a contrast?

Definition (Cambridge dictionary)

contrast, *noun*: an obvious difference between two or more things.

Low / High contrast

(source wikipedia)

- Human perception is robust to contrast change.

Image contrast

- We will limit the discussion to grayscale images.
- Human perception is robust to contrast change: Our perception is nearly the same when applying an increasing function to the gray-levels of the image.
- Examples: Sun glasses, contrast/luminosity of a display screen...

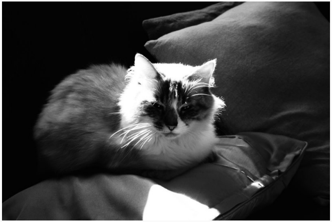


Image contrast

- This is false if the function is not increasing.
- Example: Negative image: Apply $g : x \mapsto 1 - x$:

Who is this?

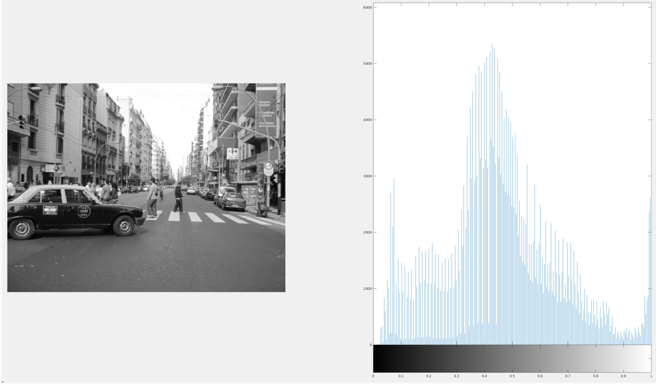
Image contrast

- This is false if the function is not increasing.
- Example: Negative image: Apply $g : x \mapsto 1 - x$:

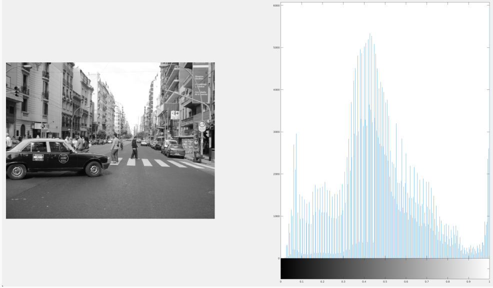
Who is this?

Image Histogram

- The histogram of an image counts the number of times a gray-level is used.



Normalized Histogram



- One often normalize the histogram to get a probability distribution.
- If the pixel grid is Ω and the gray-level values are $\mathcal{Y} = \{y_0, \ldots, y_{n-1}\}$ then the normalized histogram of u is:

$$h_u = \sum_{i=0}^{n-1} h_i \delta_{y_i} \quad \text{where} \quad h_i = \frac{|\{x \in \Omega,\ \text{s.t. } u(x) = y_i\}|}{|\Omega|}$$

- $h_i =$ proportion of pixels having gray-level y_i.

Contrast change

- A **contrast change** is an increasing function $g : \mathbb{R} \to \mathbb{R}$.
- Applying the contrast change consists in applying g to all pixel values:

$$g(u)(x) = g(u(x)), \quad x \in \Omega.$$

- The normalized histogram of $g(u)$ is obtained by shifting the pics of the histogram:

$$h_{g(u)} = \sum_{i=0}^{n-1} h_i \delta_{g(y_i)}$$

- Since g is increasing, there is no pic left-right inversion (order is preserved):

$$y_0 < y_1 < \cdots < y_{n-1} \quad \Rightarrow \quad g(y_0) \leqslant g(y_1) < \cdots < g(y_{n-1})$$

- If g takes several times the same vaues $g(y_i) = g(y_j)$ the pics are piled up together. Then information is lost.
- **Example:** Image binarization: g is the step function

$$g(y) = \begin{cases} 1 & \text{if } y > \lambda, \\ 0 & \text{if } y \leqslant \lambda. \end{cases}$$

Contrast equalization

- **Contrast equalization** consists in making the gray-level distribution as uniform as possible.
- What does it mean for the histogram?

Contrast equalization

- **Contrast equalization** consists in making the gray-level distribution as uniform as possible.
- What does it mean for the histogram?
- **Make the histogram flat**.

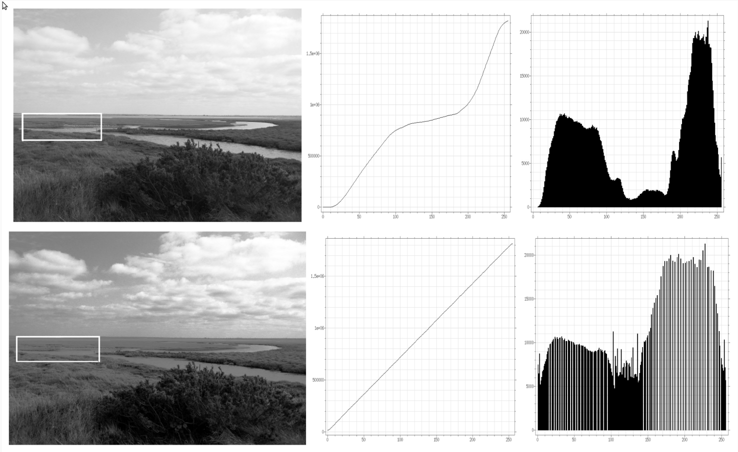
Contrast equalization

- **Contrast equalization** consists in making the gray-level distribution as uniform as possible.
- What does it mean for the histogram?
- **Make the histogram flat**.
- What does it mean for the cumulative histogram?

Contrast equalization

- **Contrast equalization** consists in making the gray-level distribution as uniform as possible.
- What does it mean for the histogram?
- **Make the histogram flat**.
- What does it mean for the cumulative histogram?
- **Make it like the identity line**.

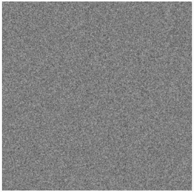
Contrast equalization

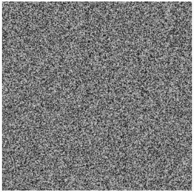


- Dark regions and bright regions have more details.
- But some local contrast decrease.

Histogram matching

- More general: Apply the histogram of a reference image to another image.
- Example of histogram matching: The histogram of a Gaussian white noise is matched with the histogram of the Lena image.

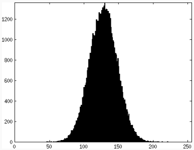


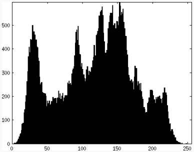


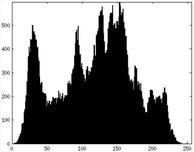
Input image

Reference image

Output image







Histogram matching

Algorithm 1: Histogram matching

Input : Input image u, reference image v (both images have size $M \times N$)

Output: Image u having the same histogram as v (the input u is lost)

Define $L = MN$ and describe the image as arrays of length L (*e.g.* by reading them line by line).

Sort the reference image v:

Determine the permutation τ such that $v_{\tau(1)} \leqslant v_{\tau(2)} \leqslant \cdots \leqslant v_{\tau(L)}$.

Sort the input image u:

Determine the permutation σ such that $u_{\sigma(1)} \leqslant u_{\sigma(2)} \leqslant \cdots \leqslant u_{\sigma(L)}$.

Match the histogram of u:

for rank $k = 1$ **to** L **do**

  $u_{\sigma(k)} \leftarrow v_{\tau(k)}$ (the k-th pixel of u takes the gray-value of the k-th pixel of v).

end

- Other solutions exists based an inversing cumulative histogram: Apply contrast change $g = H_v^{-1} \circ H_u$.

Histogram interpolation

- One may also want to find the "average histogram" between the one of u and the one of v.
- This is called **midway histogram** (Delon, 2004).

Midway histogram equalization

$$u^{\text{midway}}_{\sigma(k)} \leftarrow \frac{u_{\sigma(k)} + v_{\tau(k)}}{2} \quad \text{and} \quad v^{\text{midway}}_{\tau(k)} \leftarrow \frac{u_{\sigma(k)} + v_{\tau(k)}}{2}$$

The k-th pixel of u takes the average gray-value of the k-th pixel of u and the k-th pixel of v, and similarly for v.

Histogram interpolation

- One may also want to find the "average histogram" between the one of u and the one of v.
- This is called **midway histogram** (Delon, 2004).

Midway histogram equalization

$$u^{\text{midway}}_{\sigma(k)} \leftarrow \frac{u_{\sigma(k)} + v_{\tau(k)}}{2} \quad \text{and} \quad v^{\text{midway}}_{\tau(k)} \leftarrow \frac{u_{\sigma(k)} + v_{\tau(k)}}{2}$$

The k-th pixel of u takes the average gray-value of the k-th pixel of u and the k-th pixel of v, and similarly for v.

- It is useful to compare images (e.g. for stereovision).

Midway

Unequalized images:

(credits: Lionel Moisan, Julie Delon)

Midway

Unequalized images:

(credits: Lionel Moisan, Julie Delon)

Midway

Midway

Midway equalized images:

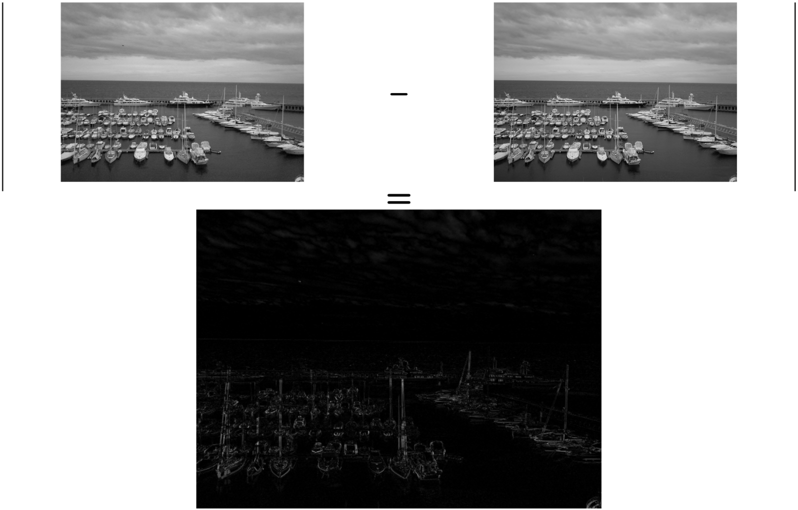
(credits: Lionel Moisan, Julie Delon)

Midway

Midway equalized images:

(credits: Lionel Moisan, Julie Delon)

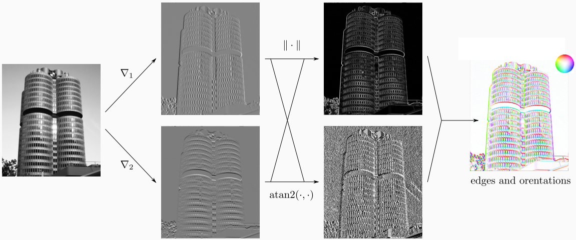
Midway



Conclusion on contrast

- Perception robust by change of contrast.
- Contrast is important for optimal detail visualization.
- For image comparison, equalizing the contrast may be important (depending on the application: image registration, stereovision,...).

Image filters



Basics of filtering

Definition (Collins dictionary)

filter, *noun*: any electronic, optical, or acoustic device that blocks 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 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

$\xrightarrow{\psi}$

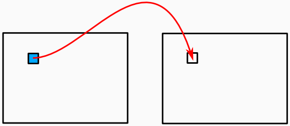
(b) Estimate $\hat{x}$

Basics of filtering

Action of filters

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

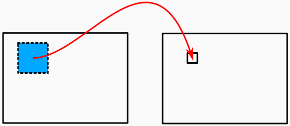
Punctual:



New pixel value depends only on the input one

e.g., change of contrast

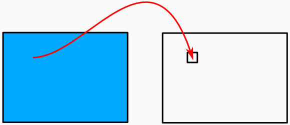
Local:



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. solution of variational problems

Basics of filtering

Filters

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

CV pipeline

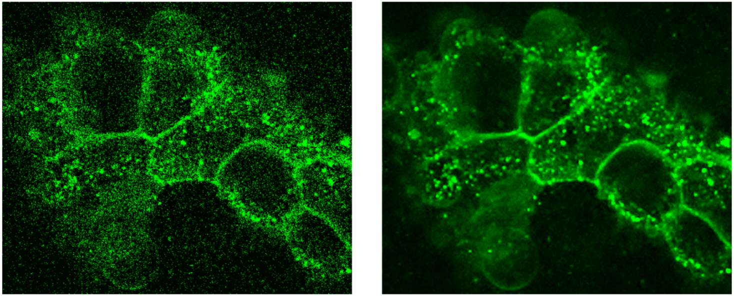
Source: Mike Thompson

Basics of filtering

Improve/denoise/detect

Basics of filtering

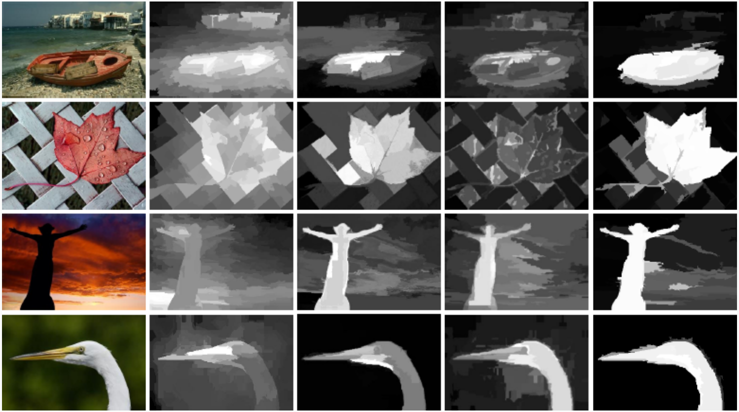
Improve/**denoise**/detect



Fibroblast cells and microbreads (fluorescence microscopy)

Source: F. Luisier & C. Vonesch

Improve/denoise/**detect**



Foreground/Background separation

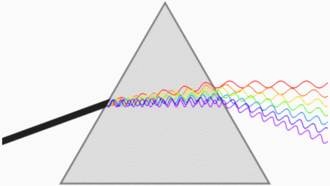
Source: H. Jiang, et al.

Basics of filtering

Standard filters

Two main approaches:

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

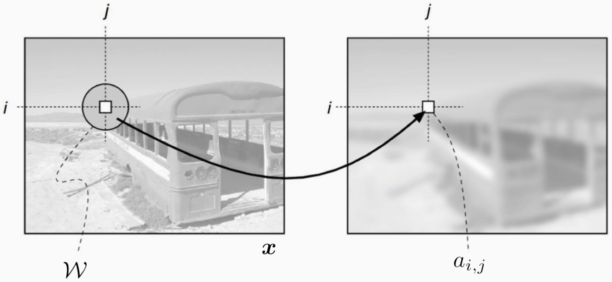
Spatial  Spectral

Spatial filtering

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

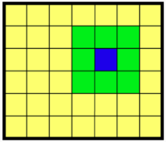
$$\hat{x}_{i,j} = \frac{1}{\mathrm{Card}(\mathcal{N})} \sum_{(k,l)\in\mathcal{N}_{i,j}} y_{k,l}$$

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

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

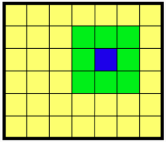
- Boxcar filter: $\mathcal{N}_{i,j} = \{(k,l)\ ;\ |i-k| \leqslant \tau \quad \text{and} \quad |j-l| \leqslant \tau\}$
- Diskcar filter: $\mathcal{N}_{i,j} = \left\{(k,l)\ ;\ |i-k|^2 + |j-l|^2 \leqslant \tau^2\right\}$

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:

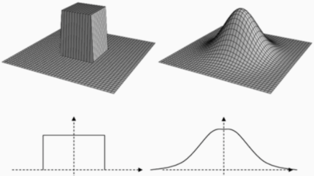
- Size: 3×3, 5×5, ...
- Shape: square, disk
- Centered or not

Spatial filtering – Moving average

Moving weighted average

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

Normalized to preserve constant images!



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

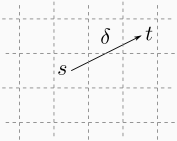
0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

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

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

Spatial filtering – Moving average

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

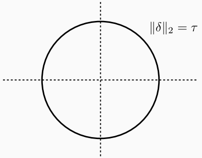
$$\varphi(\alpha) = \begin{cases} 1 & \text{if} \quad \alpha \leqslant \tau \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \text{length}(\delta) = \|\delta\|_2$$

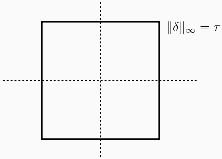
- Gaussian filter

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

Reminder:

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

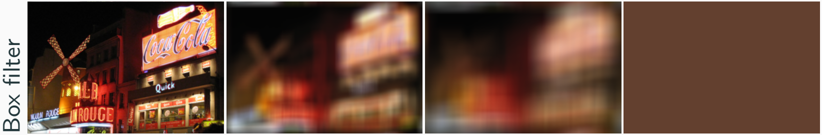




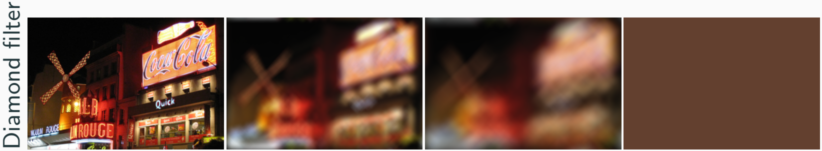
φ often depends on (at least) one parameter τ

- 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)

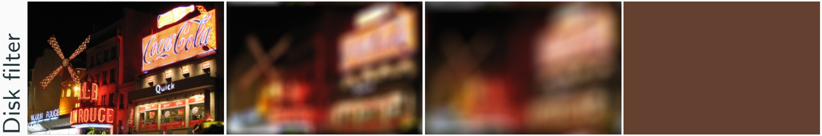
Spatial filtering – Moving average



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



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



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

Moving average for denoising?

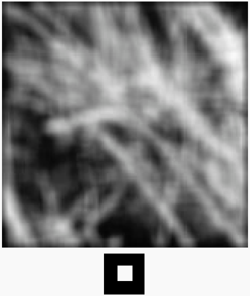
Figure 1 – (left) Gaussian noise $\sigma = 10$. (right) Gaussian filter $\tau = 3$.

Moving average for denoising?

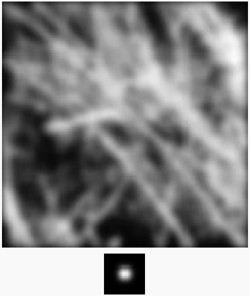
Figure 1 – (left) Gaussian noise $\sigma = 30$. (right) Gaussian filter $\tau = 5$.

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

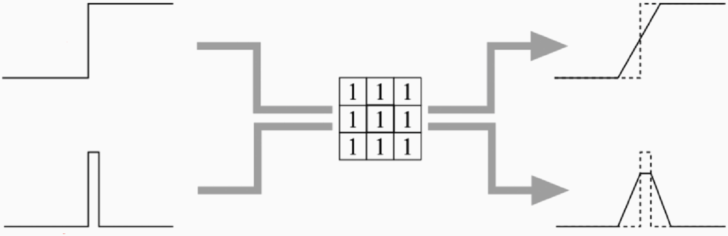


Image blur ⇒ No more edges ⇒ Structure destruction
⇒ Reduction of image quality

Spatial filtering – Moving average for denoising

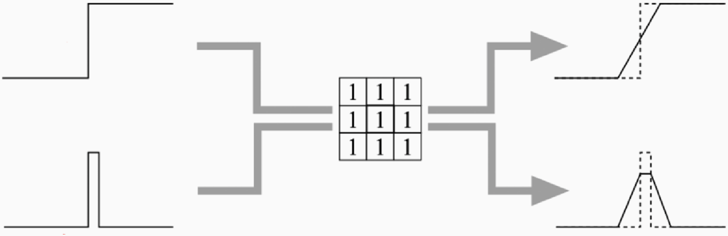


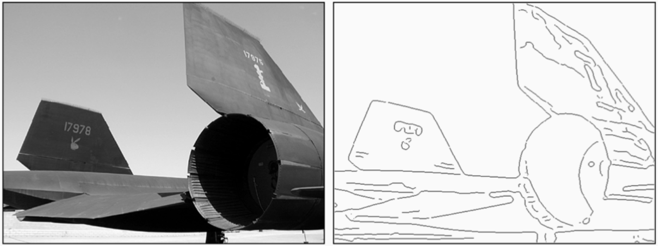
Image blur ⇒ No more edges ⇒ Structure destruction
⇒ Reduction of image quality

What is an edge?

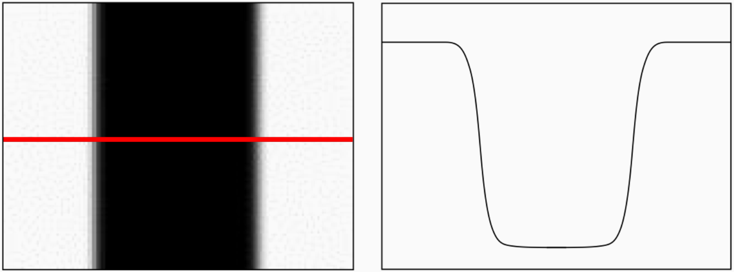
Spatial filtering – Edges

Edges?

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

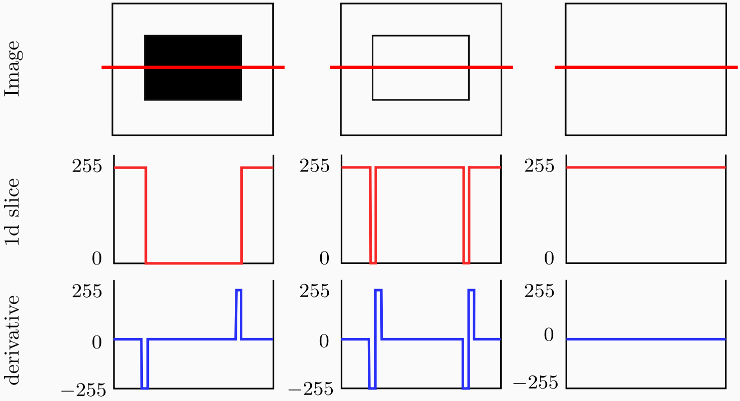


Spatial filtering – Edges



Edge: More or less brutal change of intensity

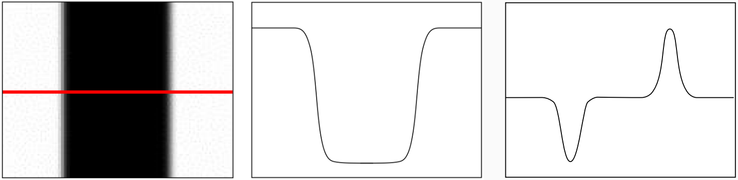
Spatial filtering – Edges



- no edges ≡ no objects in the image
- abrupt change ⇒ gap between intensities ⇒ 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

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

Spatial filtering – Derivative filters

Derivative of 1d signals

- Can be written as a filter

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

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

Derivative of 2d signals

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

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

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

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

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

Forward:

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

Backward:

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

Centered:

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

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

Laplacian of 2d signals

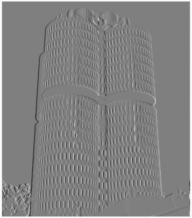
- Laplacian of a function $x : \mathbb{R}^2 \to \mathbb{R}$, if exists, is:

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

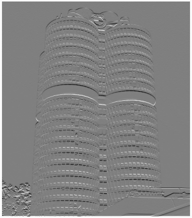
- For a 2d discrete signal: $x''_{i,j} = x_{i-1,j} + x_{i,j-1} - 4x_{i,j} + x_{i+1,j} + x_{i,j+1}$
- Corresponding filter: $h = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$

Spatial filtering – Derivative filters

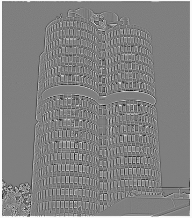
(a) x



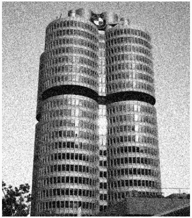
(b) $\nabla_1 x$



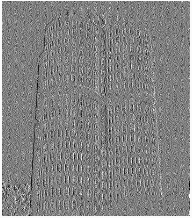
(c) $\nabla_2 x$



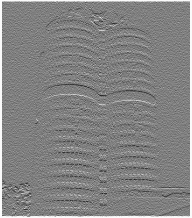
(d) Δx



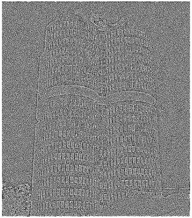
(e) x



(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)

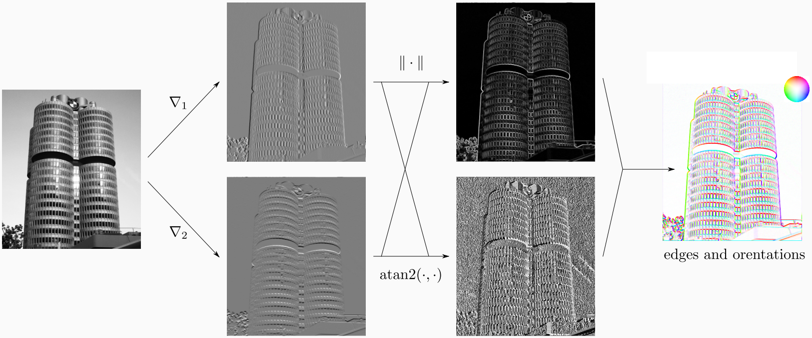
$$\kappa_1 = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \kappa_2 = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}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

Spatial filtering – Derivative filters

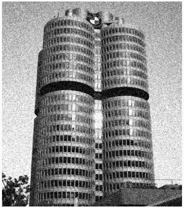
Edge detection

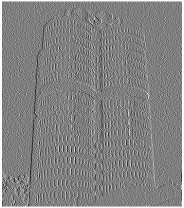
Based on the norm (and angle) of the discrete approximation of the gradient

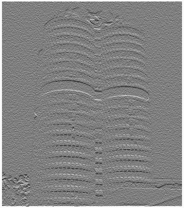
$$\|(\nabla x)_k\| = \sqrt{(\nabla_1 x)_k^2 + (\nabla_2 x)_k^2} \quad \text{and} \quad \angle(\nabla x)_k = \text{atan2}((\nabla_2 x)_k, (\nabla_1 x)_k)$$

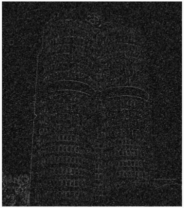


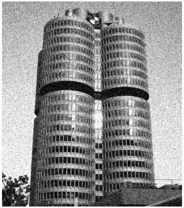
Spatial filtering – Derivative filters

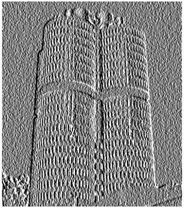

(a) x

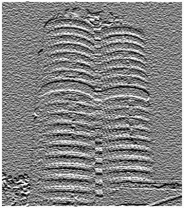

(b) $\nabla_1 x$

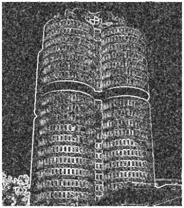

(c) $\nabla_2 x$


(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?

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

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

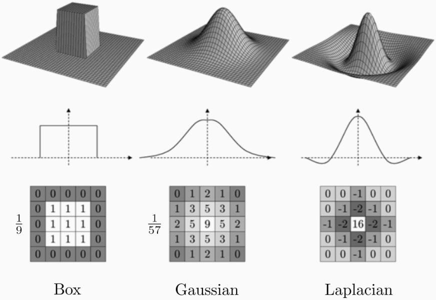
The reciprocal holds true

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

ψ uniquely specified by the coefficients κ

Linear translation-invariant filters

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



- Weighted average filters:

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

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

- Derivative filters:

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

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

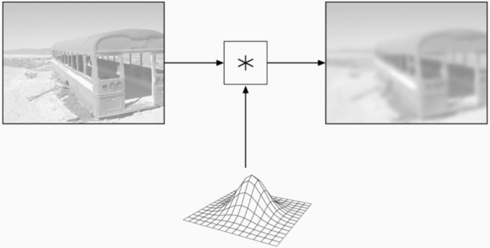
Spatial filtering – Linear translation-invariant filters

LTI filter ≡ Moving weighted sum ≡ Cross-correlation ≡ 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)



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 – 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:

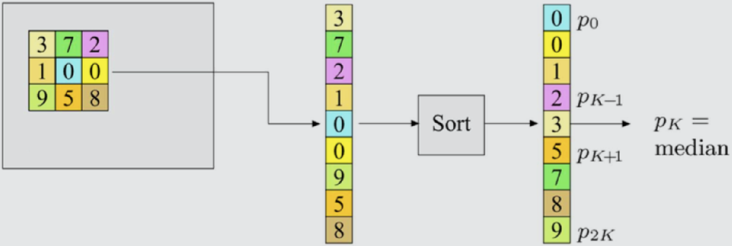
$$\psi(\alpha x + \beta y) \neq \alpha\psi(x) + \beta\psi(y)$$

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

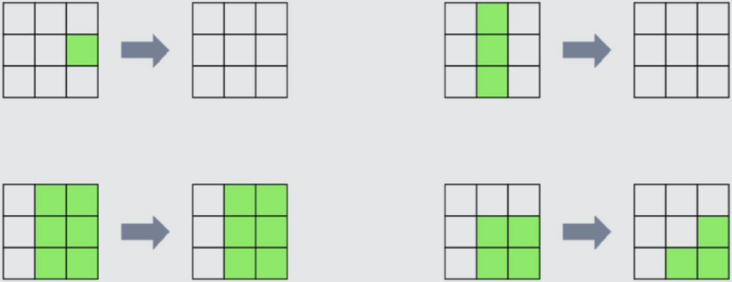
$$\hat{x}_{i,j} = \text{median}(y_{i+k,j+l} \,|\, (k,l) \in \mathcal{N}), \quad \mathcal{N} : \text{neighborhood}$$



Spatial filtering – Median filter

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.

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

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

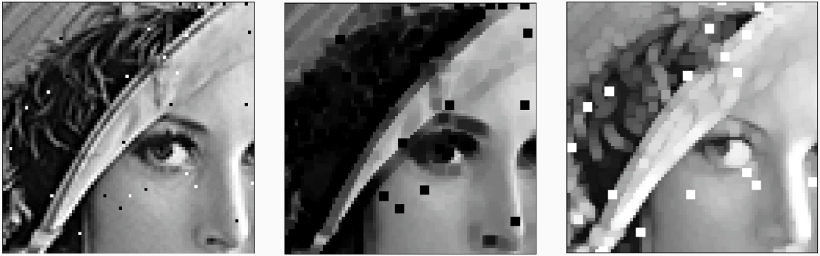


Figure 6 – (left) Salt-and-pepper noise, (center) Erosion, (right) Dilation

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)

Can be used to smooth image segmentations (see next class)

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 – 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 affect more the average,
- Closer intensities affect more the average.

Spatial filtering – Bilateral filter

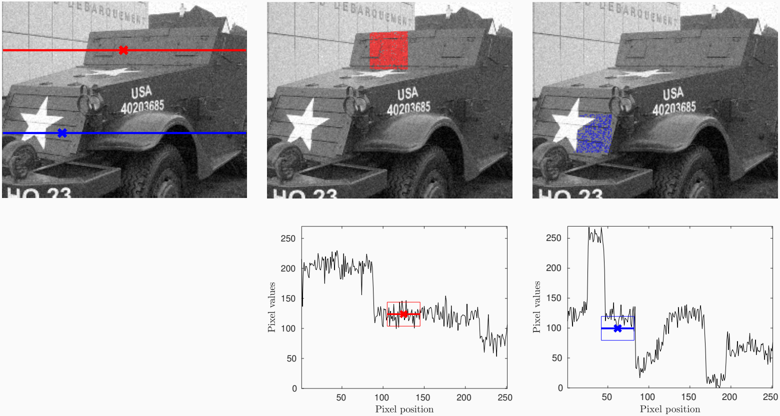


Figure 8 – Selection of pixel candidates in the bilateral filter

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

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

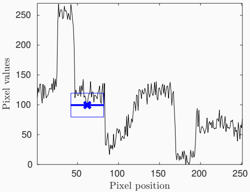
Spatial filtering – Bilateral vs moving average

Figure 9 – (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?

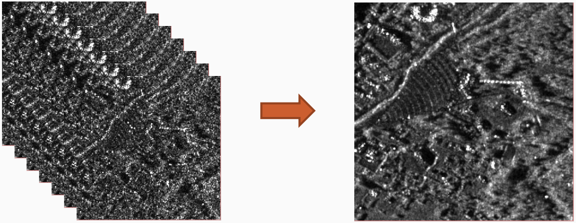
- To counteract the remaining noise effect, τ_{color} should be large

$\Rightarrow$ 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



T noisy observations $y^{(t)}$ — Estimation $\hat{x}$ of the unknown signal x

- Sample averaging of T noisy values:

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

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

(reduce noise)

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

similar = close to being iid

→ How can we select them on a single image?

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

→ most common use:
square regions between
5×5 and 21×21 pixels

→ trade-off:
size ↗ ⇒ more distinctive/informative
size ↘ ⇒ easier to model/learn/match

non-rectangular / deforming shapes:
computational complexity ↗

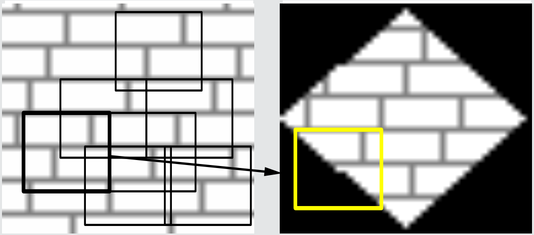
patches capture local context: geometry and texture

Copying/pasting similar patches yields impressive texture synthesis:

Texture synthesis method by Efros and Leung (1999)

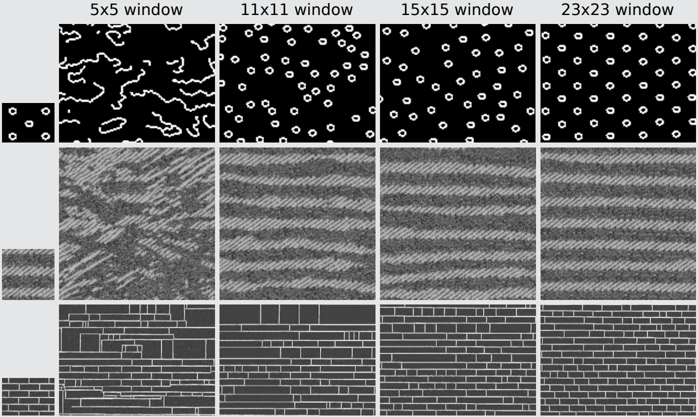
To generate a new pixel value:

- 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



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]

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

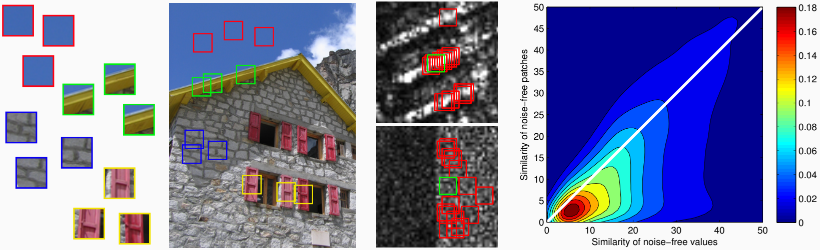
- $\mathcal{N}_i$: large neighborhood of i, called search window (typically 21×21)
- $\mathcal{P}_i$: operator extracting a small window, *patch*, at i (typically 7×7)

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

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)

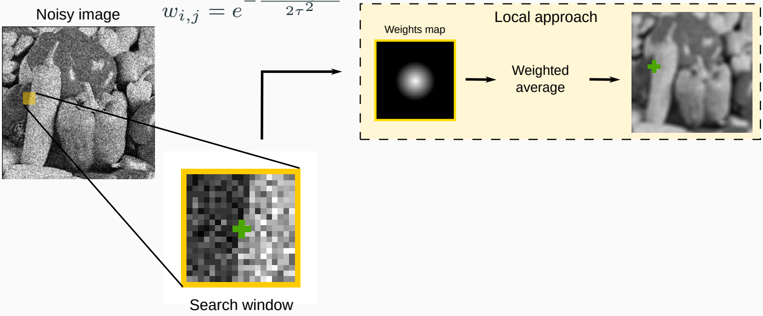
Spatial filtering – Non-local means

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

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

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



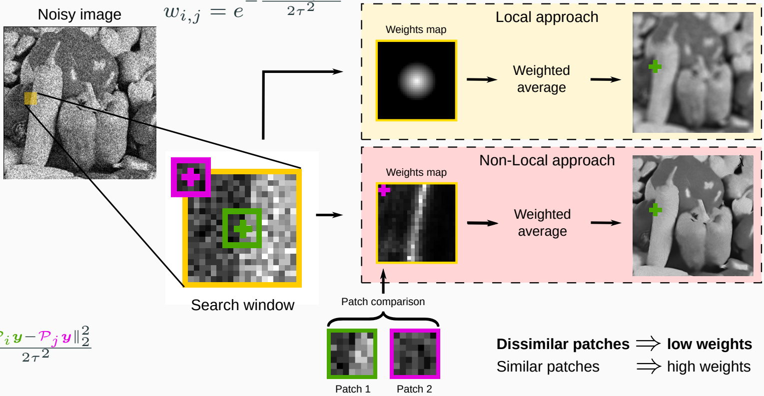
Spatial filtering – Non-local means

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

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

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



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

Dissimilar patches $\Rightarrow$ **low weights**

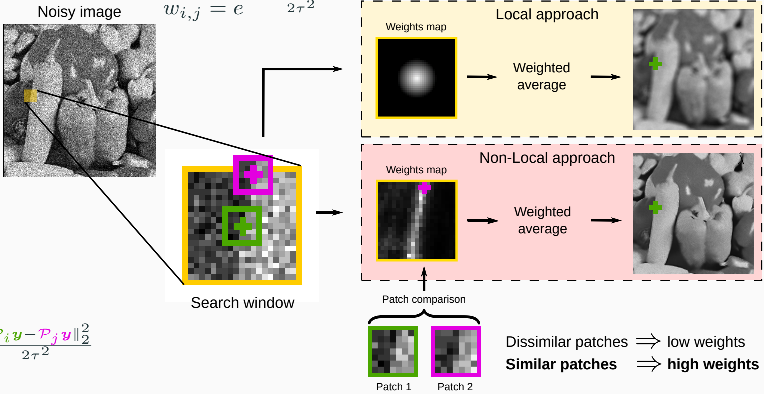
Similar patches $\Rightarrow$ high weights

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)

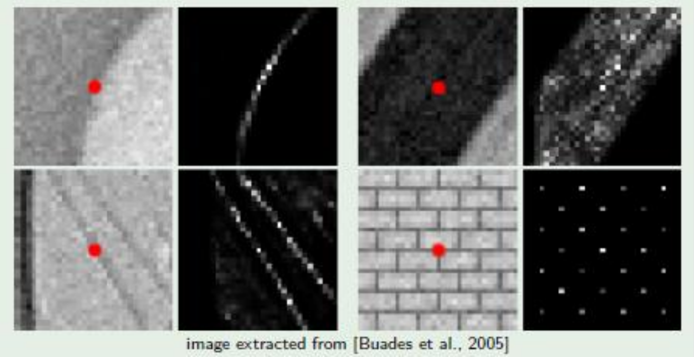
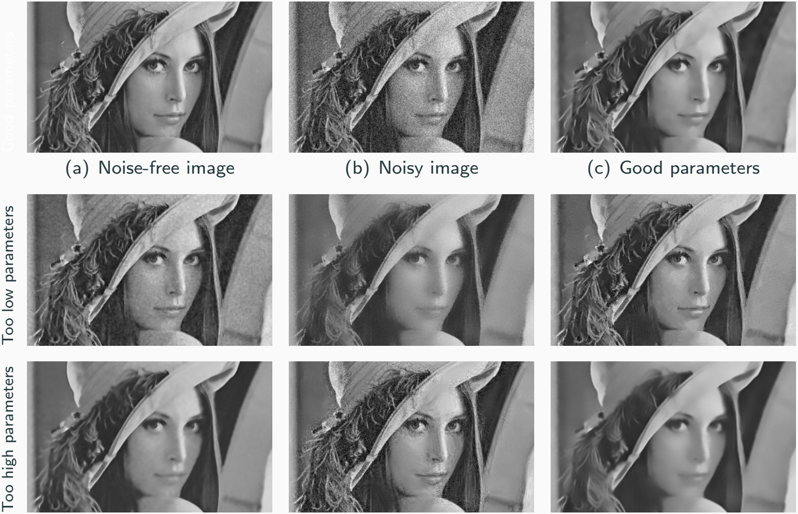


Figure 10 – Image extracted from [Buades et al., 2005]



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

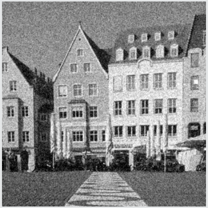
(d) Search window size (e) Patch size (f) Bandwidth τ

Figure 11 – Influence of the three main parameters of the NL means on the solution.

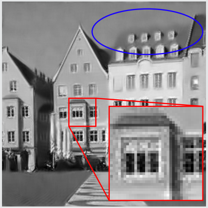
Spatial filtering – Non-local means

Limitations of NL-means

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



(a) Noisy image

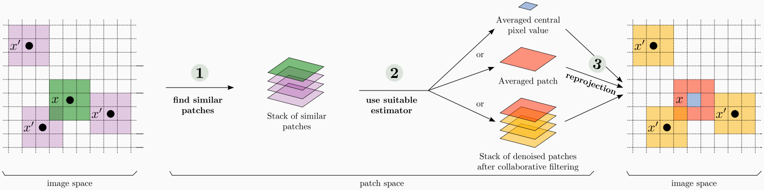


(b) NL-means

(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: $O(n|\mathcal{N}| \log |\mathcal{N}|)$

Spatial filtering – Extensions of non-local means



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

Questions?

Next class: Spectral filtering and segmentation

Slides from Charles Deledalle

Sources, images courtesy and acknowledgment

L. Condat
DLR
DMMD
Dpreview
Y. Gong
A. Horodniceanu
I. Kokkinos
J.-M. Nicolas
A. Newson
D. C. Pearson
S. Seitz
V. Tong Ta
P. Tilakaratna
Wikipedia
R. Willett
Y. Lee