Computer Vision

Lecture 6: Spatial Analysis

October 21, 2021

Prof. Sébastien Lefèvre
sebastien.lefevre@univ-ubs.fr

Spatial vs point-based operators

[REMINDER] Point-based operators DO NOT take into account the spatial context: They produce the same result if the pixels are randomly shuffled!

Spatial vs point-based operators

[REMINDER] Point-based operators DO NOT take into account the spatial context: They produce the same result if the pixels are randomly shuffled!

However, spatial information is (in general) of high importance in image analysis and processing tasks.

For instance, see lectures on Mathematical Morphology.

Spatial vs point-based operators

[REMINDER] Point-based operators DO NOT take into account the spatial context: They produce the same result if the pixels are randomly shuffled!

However, spatial information is (in general) of high importance in image analysis and processing tasks.

For instance, see lectures on Mathematical Morphology.

Question

> Examples of spatial operators?

Spatial operation

The result of a spatial operator g applied on an image f depends on each pixel $p = (x, y)$.

More precisely, it relies on the value of the pixel $f(p)$ AND the values of its neighbours.

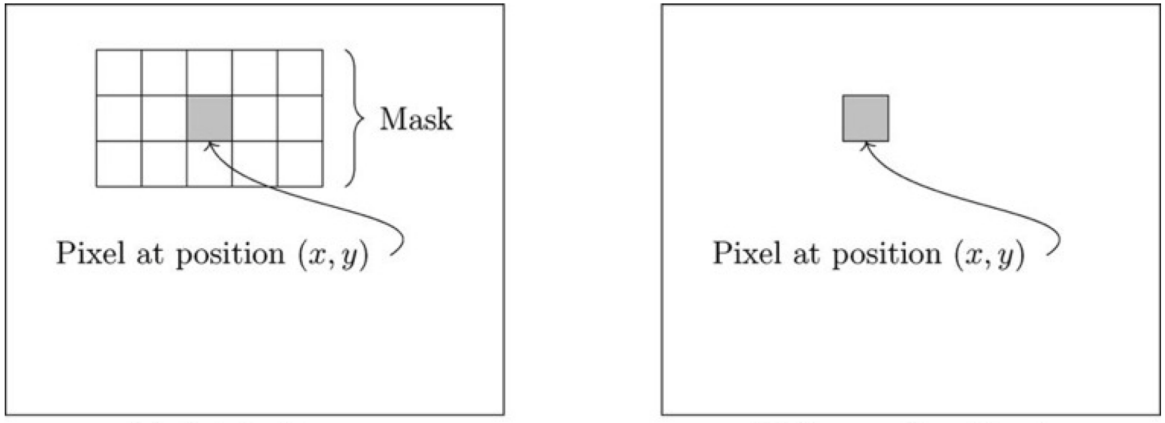
Spatial operation

The result of a spatial operator g applied on an image f depends on each pixel $p = (x, y)$.

More precisely, it relies on the value of the pixel $f(p)$ AND the values of its neighbours.

The neighbourhood $\mathcal{N}(p)$ of a pixel p is made of pixels "surrounding" p, i.e. $p' = (x', y')$ s.t. $d(p, p') \leq \varepsilon$.

Figure 5.1: Using a spatial mask on an image



(a) Original image (b) Image after filtering

Neighbourhood

A neighbourhood (a.k.a. mask, window, kernel, template) can be defined through a connectivity rule (number of neighbours):

- 4-connectivity:

$$\mathcal{N}_4(\mathrm{p}) = \{\mathrm{p}' = (\mathrm{x}', \mathrm{y}') : |\mathrm{x} - \mathrm{x}'| + |\mathrm{y} - \mathrm{y}'| = 1\}$$

- 8-connectivity:

$$\mathcal{N}_8(\mathrm{p}) = \{\mathrm{p}' = (\mathrm{x}', \mathrm{y}') : \max(|\mathrm{x} - \mathrm{x}'|, |\mathrm{y} - \mathrm{y}'|) = 1\}$$

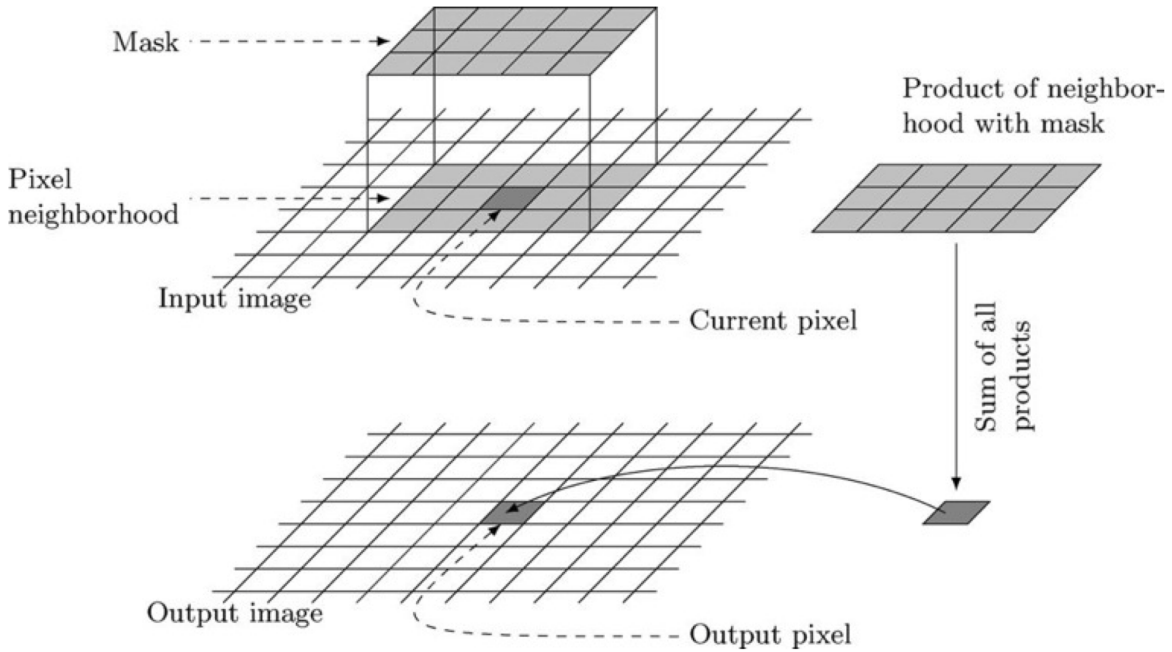
- larger neighbourhoods can be considered, i.e. $(2\mathrm{n} + 1)^2$ with $\mathrm{n} > 1$.

Processing the neighbouring pixels

All neighbours count (to some extent) in the output for a given pixel.

This is the main idea of linear spatial filtering.

Figure 5.2: Performing linear spatial filtering



Correlation & convolution

For a given mask w of size $(2a+1) \times (2b+1)$ with center in $(0,0)$, we have

- Correlation:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

- Convolution:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Note that the correlation with a mask w is equivalent to convolution with a mask w' obtained by rotating w by π.

Usually the masks are symmetric w.r.t. their origin, so correlation and convolution lead to the same results (and so both terms are used).

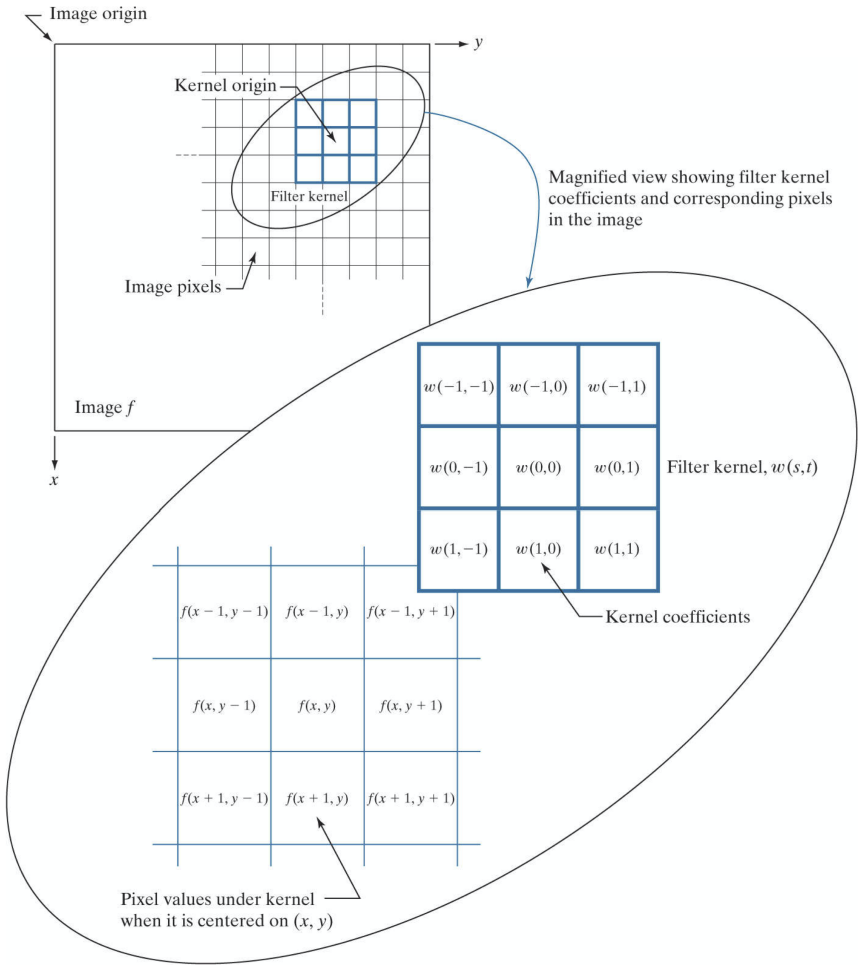


FIGURE 3.34
The mechanics of linear spatial filtering using a 3×3 kernel. The pixels are shown as squares to simplify the graphics. Note that the origin of the image is at the top left, but the origin of the kernel is at its center. Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.

Dealing with borders

Dealing with borders

- ignoring the edges
- padding with zeros
- repeating the images
- reflecting the images
- (ignoring the pixels outside the image)

Dealing with borders

Dealing with borders

- ignoring the edges
- padding with zeros
- repeating the images
- reflecting the images
- (ignoring the pixels outside the image)

Output values can be outside the range [0,255]

- make negative values positive
- clip values
- scale values in [0,255]
- (if no need to display, keep the original values)

Interactive kernel understanding

Discussion

How to set the kernel coefficients?

- $\sum \mathrm{w}$ lower/greater/equal to 1
- $\sum \mathrm{w}$ equal to 0 (with some $\mathrm{w_i}$ negative)

Interactive kernel understanding

Discussion

How to set the kernel coefficients?

- $\sum \mathrm{w}$ lower/greater/equal to 1
- $\sum \mathrm{w}$ equal to 0 (with some $\mathrm{w_i}$ negative)

Labs

1. Code the correlation/convolution operator in Python
   - with loops, without relying on Python convolution function
   - using functions available in Python (e.g. NumPy, SciPy)
   - pay attention to borders and value range!
2. Apply correlation/convolution with basic kernels
   - $\mathrm{W} = 0, \mathrm{W} = \mathrm{Id}, \mathrm{W} = 1$
   - $|\mathrm{W}|$ lower/greater/equal to 1, equal to 0 (with some $\mathrm{w_i}$ negative)

Labs

Convolution in Python

- `numpy.convolve` for 1D signals
- `scipy.signal.convolve2d` for 2D signals

Labs

Convolution in Python

- `numpy.convolve` for 1D signals
- `scipy.signal.convolve2d` for 2D signals

Examples of kernels to experiment

$$\mathrm{A}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathrm{A}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathrm{A}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

$$\mathrm{A}_4 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \mathrm{A}_5 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \mathrm{A}_6 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

$$\mathrm{A}_7 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \mathrm{A}_8 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \mathrm{A}_9 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}.$$

Images as 2D signals

An image can be seen as a sum of sinusoids at different frequencies and amplitudes.

Regions with slow variations are characterized by sinusoids of low frequencies.

Edges and sharp intensity variations are characterized by high frequency.

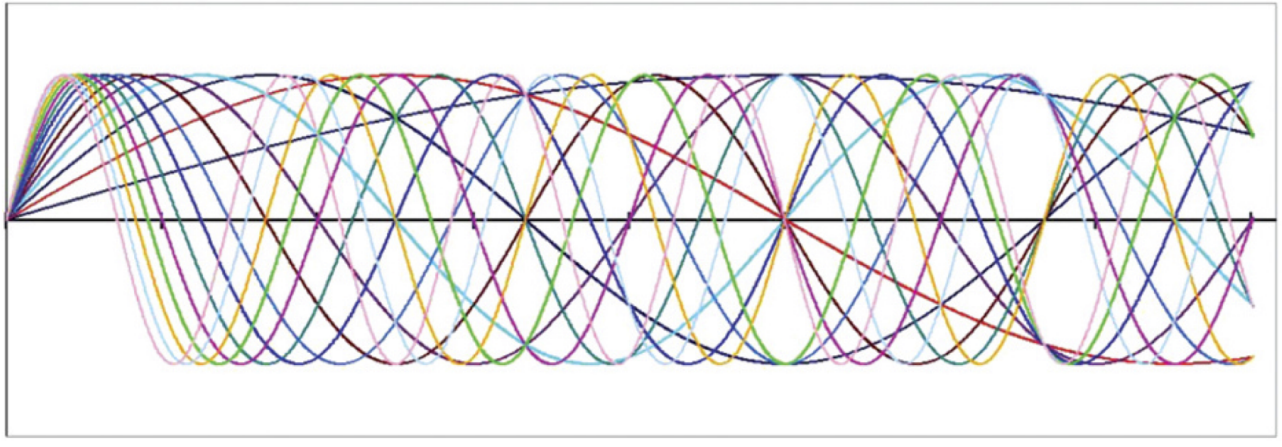


Fig. 4.1 An image can be considered as an assembly of spatial information at various frequencies.

Image smoothing

Smooth the image aims to reduce sharp transitions in intensity.

Image smoothing is achieved with a low-pass filter, done by pixel averaging/integration.

Various applications:

- image blurring,
- noise reduction,
- reduce aliasing (if prior to image resampling),
- smoothing the false contours if too strong quantization.

Low-pass spatial filters

Popular smoothing operators:

- mean filtering (a.k.a. moving average filter),
  through a box kernel is a constant kernel (e.g. $w_{i,j} = 1$).
  It requires a normalization (division by $|W|$).

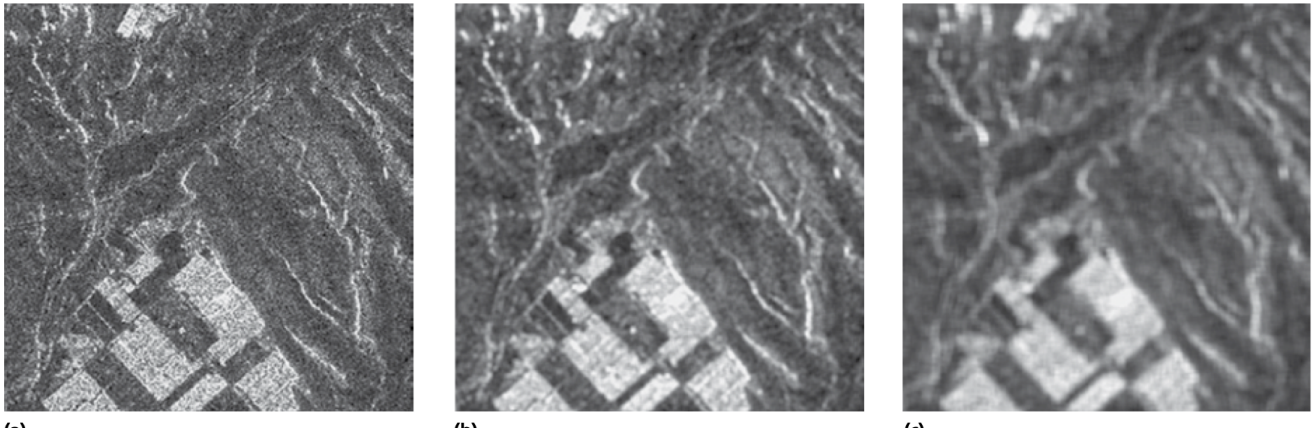
Question: effect of the kernel size?

Low-pass spatial filters

Popular smoothing operators:

- mean filtering (a.k.a. moving average filter),
  through a box kernel is a constant kernel (e.g. $w_{i,j} = 1$).
  It requires a normalization (division by $|W|$).

Question: effect of the kernel size?



(a) (b) (c)

Fig. 4.5 (a) Original image; (b) 5 × 5 mean filter result; and (c) 9 × 9 mean filter result.

Low-pass spatial filters (cont.)

The furthest pixels might contribute less to the output:

- weighted mean filtering;

Low-pass spatial filters (cont.)

The furthest pixels might contribute less to the output:

- weighted mean filtering;
- Gaussian filtering $w_{i,j} = Ke^{-\frac{i^2+j^2}{2\sigma^2}}$, i.e. for a 3×3 filter:

$$\frac{1}{4.8976} \times \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix}$$

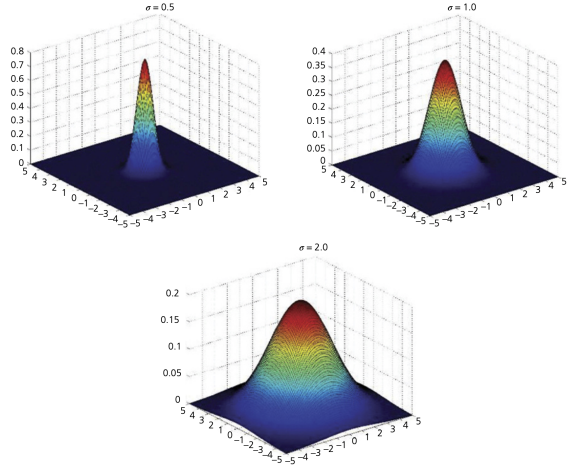


Fig. 4.6 Three-dimensional illustrations of PSFs of 2D Gaussian filters with different σ values.

Image sharpening

Sharp the image aims to highlights transitions in intensity.

Image sharpening is achieved with a high-pass filtering, done by pixel differentiation.

Image sharpening

Sharp the image aims to highlights transitions in intensity.

Image sharpening is achieved with a high-pass filtering, done by pixel differentiation.

- First derivative

$$\frac{\partial \mathrm{f}}{\partial \mathrm{x}} = \mathrm{f}(\mathrm{x}+1) - \mathrm{f}(\mathrm{x})$$

  - must be zero in areas of constant intensity
  - must be nonzero at the onset of an intensity step or ramp
  - must be nonzero along intensity ramps

Image sharpening

Sharp the image aims to highlights transitions in intensity.

Image sharpening is achieved with a high-pass filtering, done by pixel differentiation.

- First derivative

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

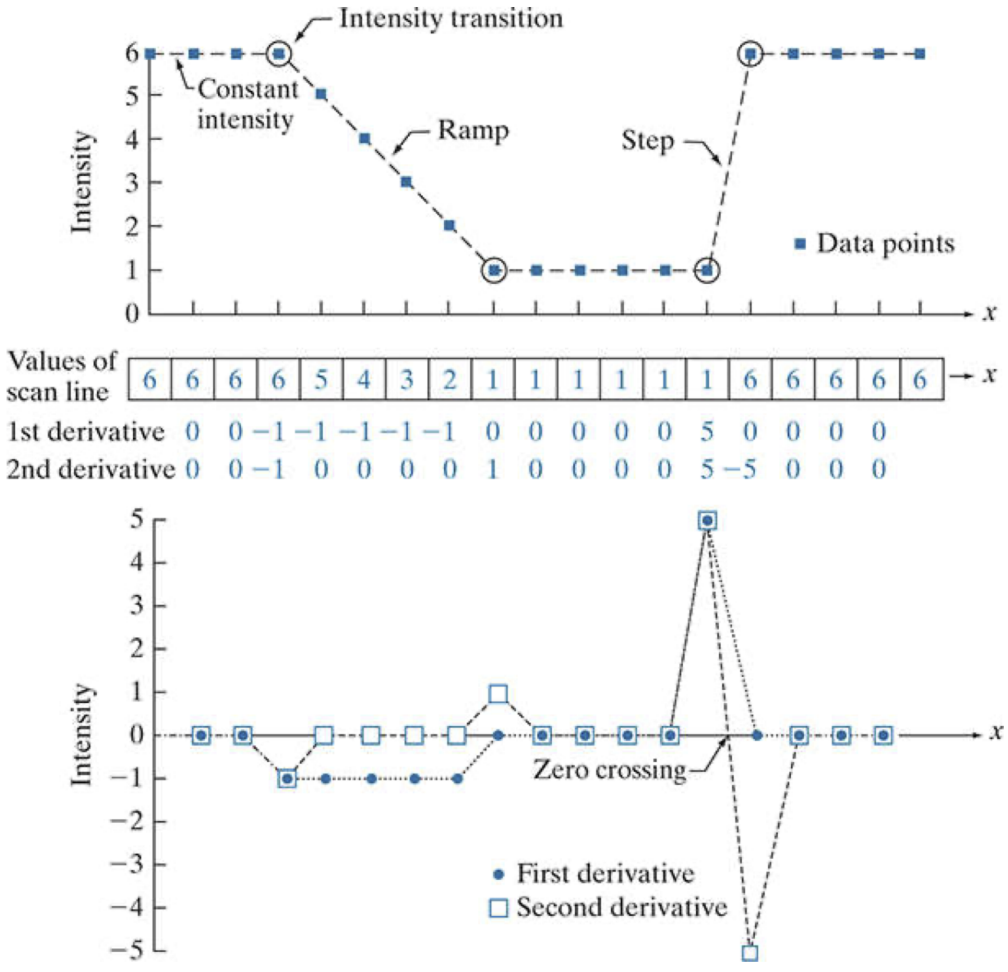
  - must be zero in areas of constant intensity
  - must be nonzero at the onset of an intensity step or ramp
  - must be nonzero along intensity ramps

- Second derivative

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

  - must be zero in areas of constant intensity
  - must be nonzero at the onset and end of an intensity step or ramp
  - must be zero along intensity ramps

Image derivatives



Second derivatives

The simplest isotropic derivative operator is the Laplacian defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Its discrete definition can be written with

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

so we have

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

using the convolution kernel $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ or even $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

Using the second derivative

Detection of isolated points

Thresholding the laplacian identifies isolated points:

$$g(x,y) = \begin{cases} 1 & \text{if } |\nabla^2 f(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$

Using the second derivative

Detection of isolated points

Thresholding the laplacian identifies isolated points:

$$g(x,y) = \begin{cases} 1 & \text{if } |\nabla^2 f(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$

Image sharpening

Laplacian highlights sharp intensity images, so can be added to the original image to emphasize edges:

$$g(x,y) = f(x,y) + |\nabla^2 f(x,y)|$$

using the convolution kernel $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ or $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$.

Using the second derivative

Detection of lines

It is also possible to rely on non-isotropic kernels if more importance has to be given to some specific directions:

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$

Then,

- orientation is given by the maximal response,
- lines are kepts with thresholding.

First derivatives

The gradient of an image is defined by

$$\nabla \mathrm{f} = \begin{bmatrix} \mathrm{g_x} \\ \mathrm{g_y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathrm{f}}{\partial \mathrm{x}} \\ \\ \frac{\partial \mathrm{f}}{\partial \mathrm{y}} \end{bmatrix}$$

and it points in the direction of the greatest rate of change of f at location (x, y).

First derivatives

The gradient of an image is defined by

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

and it points in the direction of the greatest rate of change of f at location (x, y).

The magnitude (a.k.a. norm or length) of ∇f indicates the rate of change, and is used to build the gradient image:

$$||\nabla f|| = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y|$$

while the direction of ∇f indicates the orientation of the change:

$$\alpha(\nabla f) = \tan^{-1} \begin{bmatrix} \frac{g_y}{g_x} \end{bmatrix}$$

Note: (first and second) derivatives are very sensitive to noise.
So prior smoothing is usually applied.

Edge detection

Various pairs of kernels exist to compute the gradient:

- $\mathrm{A}_{\text{vertical}} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $\mathrm{A}_{\text{horizontal}} = \begin{bmatrix} -1 & 1 \end{bmatrix}$;
- Roberts: $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$;
- Prewitt: $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$;
- Sobel (more robust to noise): $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$.

Edge detection

Kirsch compass kernels

$$\mathrm{A_N} = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}, \mathrm{A_{NW}} = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix},$$

$$\mathrm{A_W} = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \mathrm{A_{SW}} = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix},$$

$$\mathrm{A_S} = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & 3 \\ 5 & -3 & -3 \end{bmatrix}, \mathrm{A_{SE}} = \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix},$$

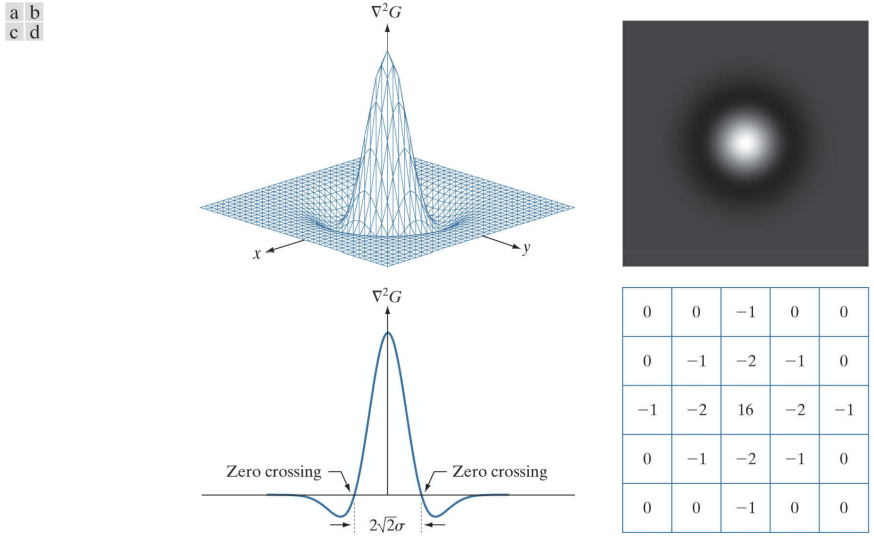
$$\mathrm{A_E} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}, \mathrm{A_{NE}} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}.$$

Edge Detection

Laplacian of a Gaussian

The Laplacian of a Gaussian (LoG) $\nabla^2 G(x, y) = \left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$

is a differential operator able to approximate derivative at every point in the image, and capable of being tuned to act at any desired scale, i.e. large operator to detect blurry edges vs small one to detect sharply focused fine detail.



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

FIGURE 10.21
(a) 3-D plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5×5 kernel approximation to the shape in (a). The negative of this kernel would be used in practice.

Marr-Hildreth edge detector

The Marr-Hildreth algorithms consist of convolving (noted $\star$) the LoG kernel with the input image

$$g(x, y) = [\nabla^2 G(x, y)] \star f(x, y) = \nabla^2 [G(x, y) \star f(x, y)]$$

The algorithm is made of 3 steps:

1. filter the input image with an $n \times n$ Gaussian lowpass kernel
2. compute the Laplacian of the filtered image using e.g. a 3×3 kernel
3. find the zero crossings of the Laplacian image

Marr-Hildreth edge detector

The Marr-Hildreth algorithms consist of convolving (noted $\star$) the LoG kernel with the input image

$$g(x, y) = [\nabla^2 G(x, y)] \star f(x, y) = \nabla^2 [G(x, y) \star f(x, y)]$$

The algorithm is made of 3 steps:

1. filter the input image with an $n \times n$ Gaussian lowpass kernel
2. compute the Laplacian of the filtered image using e.g. a 3×3 kernel
3. find the zero crossings of the Laplacian image

How to find zero crossings?

- consider a 3×3 neighbourhood
- at least two opposing neighbours should have different signs
- (absolute difference between these neighbours should also be higher than a given threshold)

Difference of Gaussians

The LoG can be approximated by Difference of Gaussians (DoG) (with $\sigma_1 > \sigma_2$)

$$D_G(x, y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

Canny edge detector

3 objectives:

- low error rate (no missing edges, few false detections)
- edge points well localized
- single edge point responses

4 steps:

1. smooth the input image with a Gaussian filter
2. compute the gradient magnitude and angle images
3. apply nonmaxima suppression to the gradient magnitude image
4. use double thresholding and connectivity analysis to detect and link edges

Canny edge detector

a b
c d

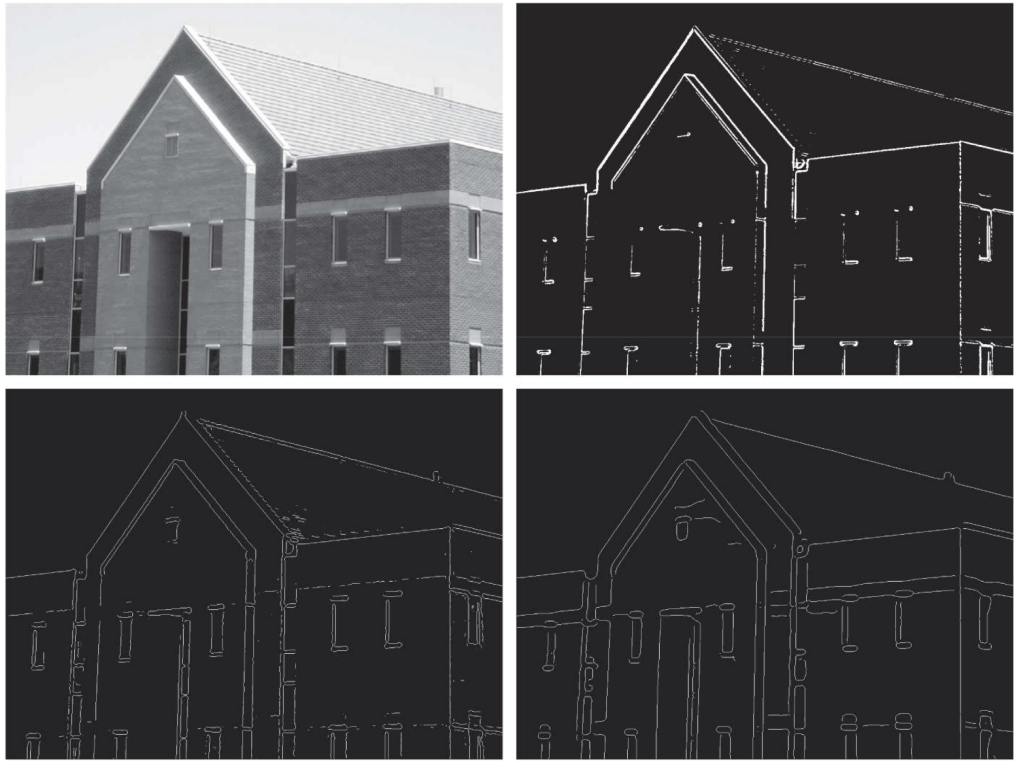


FIGURE 10.25
(a) Original image of size 834×1114 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of the smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

Edge sharpening

Unsharp masking

Substracting an unsharp (blurred) image from the original leads to a so-called mask that emphasizes sharp areas.

$$g_{mask}(x, y) = f(x, y) - \bar{f}(x, y)$$

$$g(x, y) = f(x, y) + g_{mask}(x, y)$$

Edge sharpening

Unsharp masking

Substracting an unsharp (blurred) image from the original leads to a so-called mask that emphasizes sharp areas.

$$g_{mask}(x, y) = f(x, y) - \bar{f}(x, y)$$

$$g(x, y) = f(x, y) + g_{mask}(x, y)$$

High Boost filtering

The previous equation can be tuned by a parameter $k > 1$ to increase the sharpening effect:

$$g(x, y) = f(x, y) + kg_{mask}(x, y)$$

Conversely, $k < 1$ reduces the filtering effect.

Separability

Remark: all spatial operators relying on a $m \times n$ kernel require mn products (+ 1 sum and possible normalization) per pixel.

The cost is directly proportional to the size of the kernel.

Separability

Remark: all spatial operators relying on a $m \times n$ kernel require mn products (+ 1 sum and possible normalization) per pixel.

The cost is directly proportional to the size of the kernel. An appealing strategy is to reduce the size of kernel:

- through iterative applications of smaller kernels
- through separation of the 2D kernel into two 1D kernels that are applied sequentially on the image.

Separability

Remark: all spatial operators relying on a $m \times n$ kernel require mn products (+ 1 sum and possible normalization) per pixel.

The cost is directly proportional to the size of the kernel. An appealing strategy is to reduce the size of kernel:

- through iterative applications of smaller kernels
- through separation of the 2D kernel into two 1D kernels that are applied sequentially on the image.

Discussion

- How to rewrite popular 2D kernels into pairs of 1D kernels?
- What is the resulting complexity?

Nonlinear filtering

Convolution/correlation are linear operations.

Spatial filtering can also be achieved using nonlinear operators:

- min or max (more details in the lecture on Mathematical Morphology),
- median,
- rank,
- mode (a.k.a. majority) filter: useful for label images.

Nonlinear filtering

Convolution/correlation are linear operations.

Spatial filtering can also be achieved using nonlinear operators:

- min or max (more details in the lecture on Mathematical Morphology),
- median,
- rank,
- mode (a.k.a. majority) filter: useful for label images.

Question

> Can we separate these nonlinear filters for efficient implementation?

Edge-preserving smoothing filters

Some filters are specifically designed to smooth the images while preserving the edges.

Nonlinear filters:

- adaptive median filter, i.e. median of three median-filtered images using the following masks (ignored pixels correspond to zeros):

$$\mathrm{A}_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \mathrm{A}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \mathrm{A}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix};$$

- K nearest median filter, i.e. median of the K-nearest neighbors;
- conditional smoothing filters (e.g. clean pixels, lines, columns), e.g.

```
if (condition):
  filter1
else
  filter2
```

Clean pixel filter

$$AVE1 = \frac{1}{4}\left(x_{i-1,j-1} + x_{i+1,j+1} + x_{i+1,j-1} + x_{i-1,j+1}\right)$$

$$AVE2 = \frac{1}{4}\left(x_{i-1,j} + x_{i+1,j} + x_{i,j+1} + x_{i,j+1}\right)$$

$$DIF = \left|AVE1 - AVE2\right|$$

$$\text{If}: \left|x_{i,j} - AVE1\right| > DIF \quad \text{and} \quad \left|x_{i,j} - AVE2\right| > DIF,$$

$$\text{then}: y_{i,j} = AVE2$$

$$\text{otherwise}: y_{i,j} = x_{i,j}$$

Clean line filter

$$AVE1 = \frac{1}{3}\left(x_{i-1,j-1} + x_{i-1,j} + x_{i-1,j+1}\right)$$

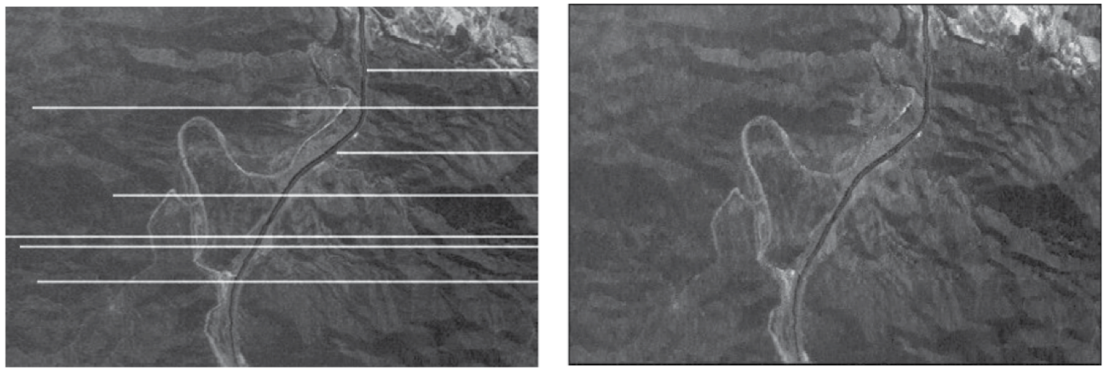
$$AVE2 = \frac{1}{3}\left(x_{i+1,j-1} + x_{i+1,j} + x_{i+1,j+1}\right)$$

$$DIF = \left|AVE1 - AVE2\right|$$

$$\text{If}: \left|x_{i,j} - AVE1\right| > DIF \quad \text{and} \quad \left|x_{i,j} - AVE2\right| > DIF,$$

$$\text{then}: y_{i,j} = \left(x_{i-1,j} + x_{i+1,j}\right)/2$$

$$\text{otherwise}: y_{i,j} = x_{i,j}$$



(a) (b)

Fig. 4.8 (a) An Airborne Thematic Mapper (ATM) image is degraded by several bad lines as the result of occasional sensor failure; and (b) the bad lines are successfully removed by the clean lines filter.

Kuwahara filters

For a given pixel (x, y):

- consider 4 neighbourhoods with the top-left / top-right / bottom-left / bottom-right pixels of (x, y)
- compute the variance of each neighbourhood
- output the mean associated with the lowest variance

Kuwahara filters

For a given pixel (x, y):

- consider 4 neighbourhoods with the top-left / top-right / bottom-left / bottom-right pixels of (x, y)
- compute the variance of each neighbourhood
- output the mean associated with the lowest variance

Bilateral filters

Each neighbour $p' \in \mathcal{N}_p$ is weighted by both its spatial distance $d(p, p')$ but also its intensity/radiometric difference $d(f(p), f(p'))$.

Both spatial and intensity kernels can be Gaussian kernels, leading to

$$g(x, y) = \frac{\sum_{(x',y') \in \mathcal{N}(p)} f(x, y) w(x, y, x', y')}{\sum_{(x',y') \in \mathcal{N}(p)} w(x, y, x', y')}$$

$$w(x, y, x', y') = \exp\left(-\frac{(x - x')^2 + (y - y')^2}{2\sigma_d^2} - \frac{||f(x, y) - f(x', y')||^2}{2\sigma_r^2} \right)$$

Labs

All the following questions can be answered with either predefined convolution python functions or direct loop implementations.

Exercices:

1. For a given spatial operator, compare the filtering effect and CPU time when increasing the kernel size.
2. For a given spatial operator, compare the filtering effect and CPU time for a 2D separable kernel and its 1D alternatives.
3. Compare the results of different smoothing operators.
4. Compare the results of different sharpening operators.
5. Compare the results of different edge detection operators.
6. Are smoothing/sharpening operators invertible?
7. Compare the linear and nonlinear filtering operators.
8. Propose separable implementations of nonlinear operators.
9. Verify the ability of edge-preserving smoothing filters.