# Introduction to Image Processing

Lecturer:
Dr. mat. nat. Christian Kehl

Week 3 – Colour Image Processing and Derivative Filters

November 20, 2024

# Colour Image Processing

# Colour Image Processing

- *Pseudocolour image processing*: map colours to intensity values
  → 2D Visualisation (colour-mapping)

# Colour Image Processing

- *Pseudocolour image processing*: colour mapping
- *Full-colour image processing*: processing of colour images:

$$\mathbf{c}(x,y) = \begin{pmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{pmatrix}, \qquad x \in \{0 \ldots M-1\}, y \in \{0 \ldots N-1\}$$

# Colour Image Processing

- *Pseudocolour image processing: map colours to intensity values*
  *→ 2D Visualisation (colour-mapping)*
- *Full-colour image processing*: processing of colour images
- *Intensity-extracted processing*: extract *intensity* from trichromatic image, then just process the intensity → image simplification

# Colour Image Processing

- *Pseudocolour image processing*: 2D Visualisation (colour-mapping) → just added labour ...
- **Full-colour image processing**: processing of colour images
- *Intensity-extracted processing*: image simplification → methods discussed last time

# (Full-)Colour Image Processing

Three major approaches:

- Colour composition of seperately-processed channels:

$$\mathbf{c}(x,y) = \begin{pmatrix} f(R(x,y)) \\ f(G(x,y)) \\ f(B(x,y)) \end{pmatrix}$$

# (Full-)Colour Image Processing

Three major approaches:

- Colour composition of seperately-processed channels:

$$\mathbf{c}(x,y) = \begin{pmatrix} f(R(x,y)) \\ f(G(x,y)) \\ f(B(x,y)) \end{pmatrix}$$

$\rightarrow$ Per-channel processing

# (Full-)Colour Image Processing

Three major approaches:

- Colour composition of seperately-processed channels:

$$\mathbf{c}(x,y) = \begin{pmatrix} f(R(x,y)) \\ f(G(x,y)) \\ f(B(x,y)) \end{pmatrix}$$

$\rightarrow$ Per-channel processing

- Intensity decomposition, intensity processing, and colour recombination:

$$\mathbf{c} = f \begin{pmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{pmatrix} \rightarrow T \begin{pmatrix} f_1(H(x,y)) \\ f_2(S(x,y)) \\ f_3(I(x,y)) \end{pmatrix}$$

# (Full-)Colour Image Processing

Three major approaches:

- Colour composition of seperately-processed channels:

$$\mathbf{c}(x,y) = \begin{pmatrix} f(R(x,y)) \\ f(G(x,y)) \\ f(B(x,y)) \end{pmatrix}$$

  $\rightarrow$ Per-channel processing

- Intensity decomposition, intensity processing, and colour recombination:

$$\mathbf{c} = f \begin{pmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{pmatrix} \rightarrow T \begin{pmatrix} f_1(H(x,y)) \\ f_2(S(x,y)) \\ f_3(I(x,y)) \end{pmatrix}$$

  $\rightarrow$ Luminance processing

# (Full-)Colour Image Processing

Three major approaches:

- Colour composition of seperately-processed channels:

$$\mathbf{c}(x,y) = \begin{pmatrix} f(R(x,y)) \\ f(G(x,y)) \\ f(B(x,y)) \end{pmatrix}$$

$\rightarrow$ Per-channel processing

- Intensity decomposition, intensity processing, and colour recombination:

$$\mathbf{c} = f \begin{pmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{pmatrix} \rightarrow T \begin{pmatrix} f_1(H(x,y)) \\ f_2(S(x,y)) \\ f_3(I(x,y)) \end{pmatrix}$$

$\rightarrow$ Luminance processing

- Colour image as discrete samples in vector space:

$$\begin{pmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{pmatrix} = \mathbf{c}(x,y) \in C$$

$$r = (1,0,0), g = (0,1,0), b = (0,0,1)$$

$$C \in \mathcal{R}^3, C = r \times g$$

# (Full-)Colour Image Processing

Three major approaches:

- Colour composition of seperately-processed channels:

$$\mathbf{c}(x,y) = \begin{pmatrix} f(R(x,y)) \\ f(G(x,y)) \\ f(B(x,y)) \end{pmatrix}$$

$\rightarrow$ Per-channel processing

- Intensity decomposition, intensity processing, and colour recombination:

$$\mathbf{c} = f \begin{pmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{pmatrix} \rightarrow T \begin{pmatrix} f_1(H(x,y)) \\ f_2(S(x,y)) \\ f_3(I(x,y)) \end{pmatrix}$$

$\rightarrow$ Luminance processing

- Colour image as discrete samples in vector space:

$$\begin{pmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{pmatrix} = \mathbf{c}(x,y) \in C$$

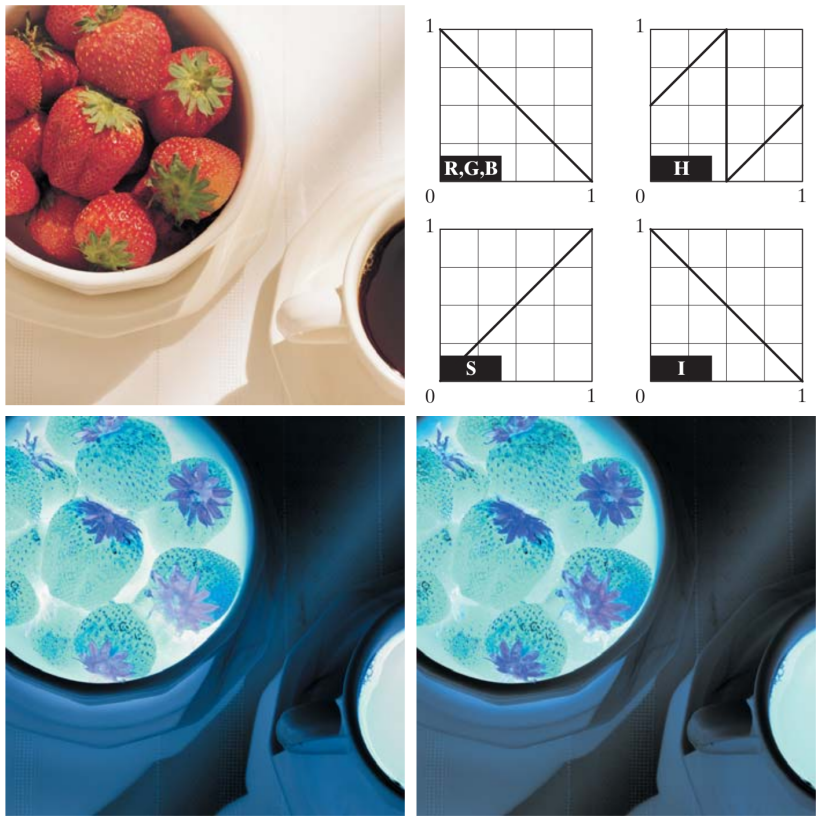
$$r = (1,0,0), g = (0,1,0), b = (0,0,1)$$

$$C \in \mathcal{R}^3, C = r \times g$$

$\rightarrow$ Colour-space processing

# Per-channel processing - Examples

Inversion & complement colours



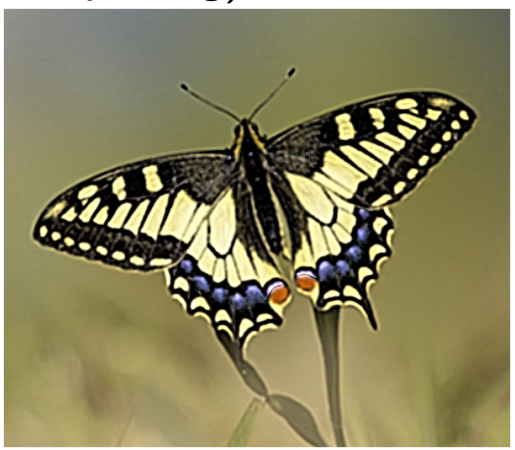

Inversion of colour image, via per-channel- and hue-luminance processing

$$\mathbf{c}(x,y) = \begin{pmatrix} 1-R(x,y) \\ 1-G(x,y) \\ 1-B(x,y) \end{pmatrix} = T \begin{pmatrix} (H(x,y)+180)\%360 \\ S(x,y) \\ 1-I(x,y) \end{pmatrix}$$

# Per-channel processing - Examples

Derivative filters (here: Sharpening)

Per RGB channel.



Just intensity (HSI).

Difference.

Image sharpening via Laplacian: $\nabla^2 \mathbf{c}(x,y) = \begin{pmatrix} \nabla^2 R(x,y) \\ \nabla^2 G(x,y) \\ \nabla^2 B(x,y) \end{pmatrix}$

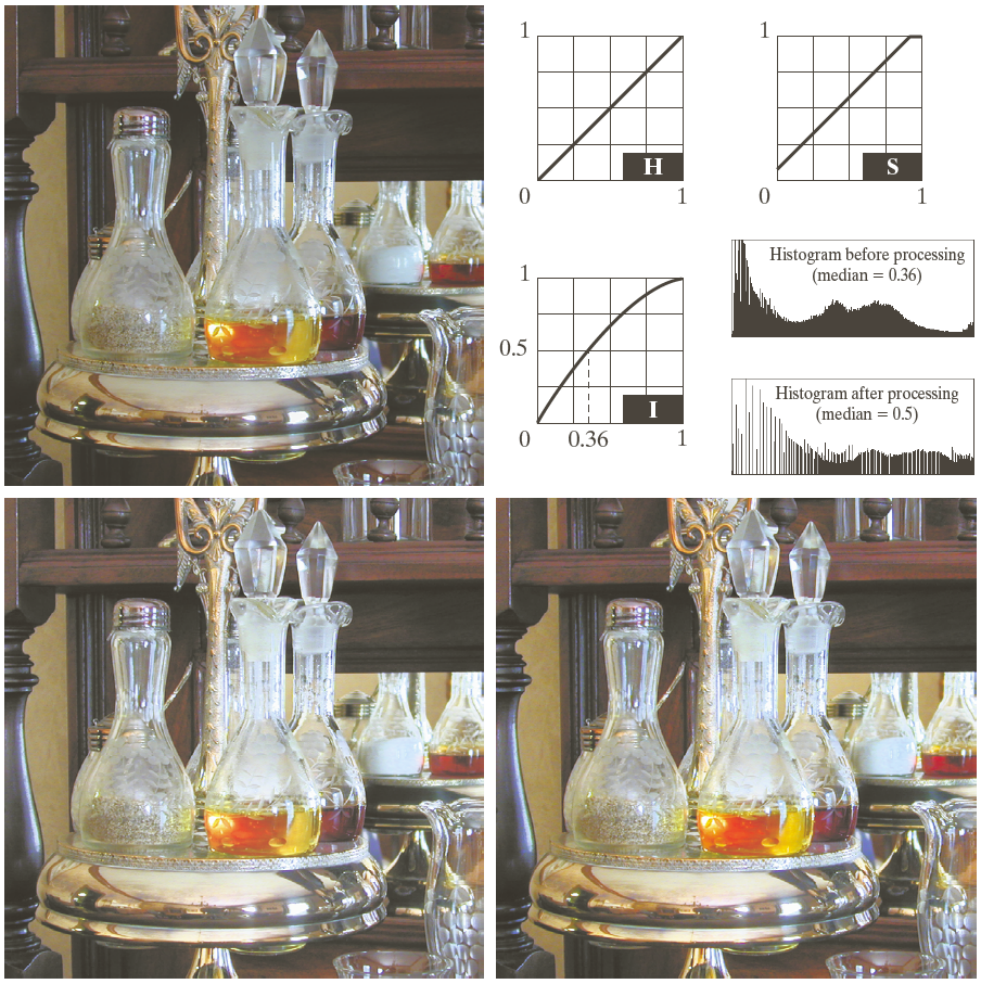
# Luminance processing - Examples

$\gamma$ adjustment


$\gamma$-adjusted imaged ($\gamma = 0.5$), applied per-channel on RGB vs. just on luminance (HSV). Note the preservation of hues (right).

# Luminance processing - Examples

Colour histogram equalization




Equalization based on the histogram of the intensity in HSI space, followed by saturation adjustment

# Colour-space processing - Examples

some operations are not that easily separable

- rank-order (i.e. morphology) filters
- histogram filters
- segmentation

# Colour-space processing - Examples

some operations are not that easily separable

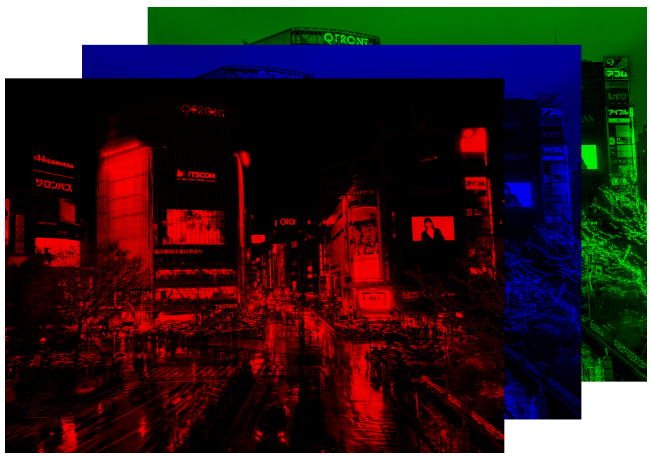
- rank-order (i.e. morphology) filters
- histogram filters
- segmentation

Here: switch from processing 2D image space

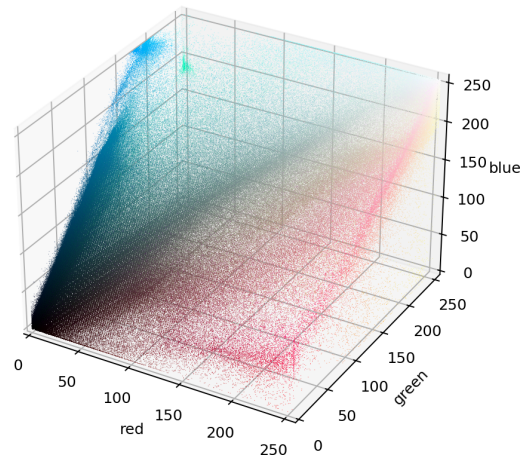
$$\mathrm{I} = f(x, y) \in \mathcal{Z}^2 \rightarrow \mathcal{Z}^3$$

to 3D value space

$$x_i = \{x_1, x_2, \ldots, x_N\}, N = \text{rows} \cdot \text{columns}, x_i \in \mathcal{Z}^3$$



2D image space layers

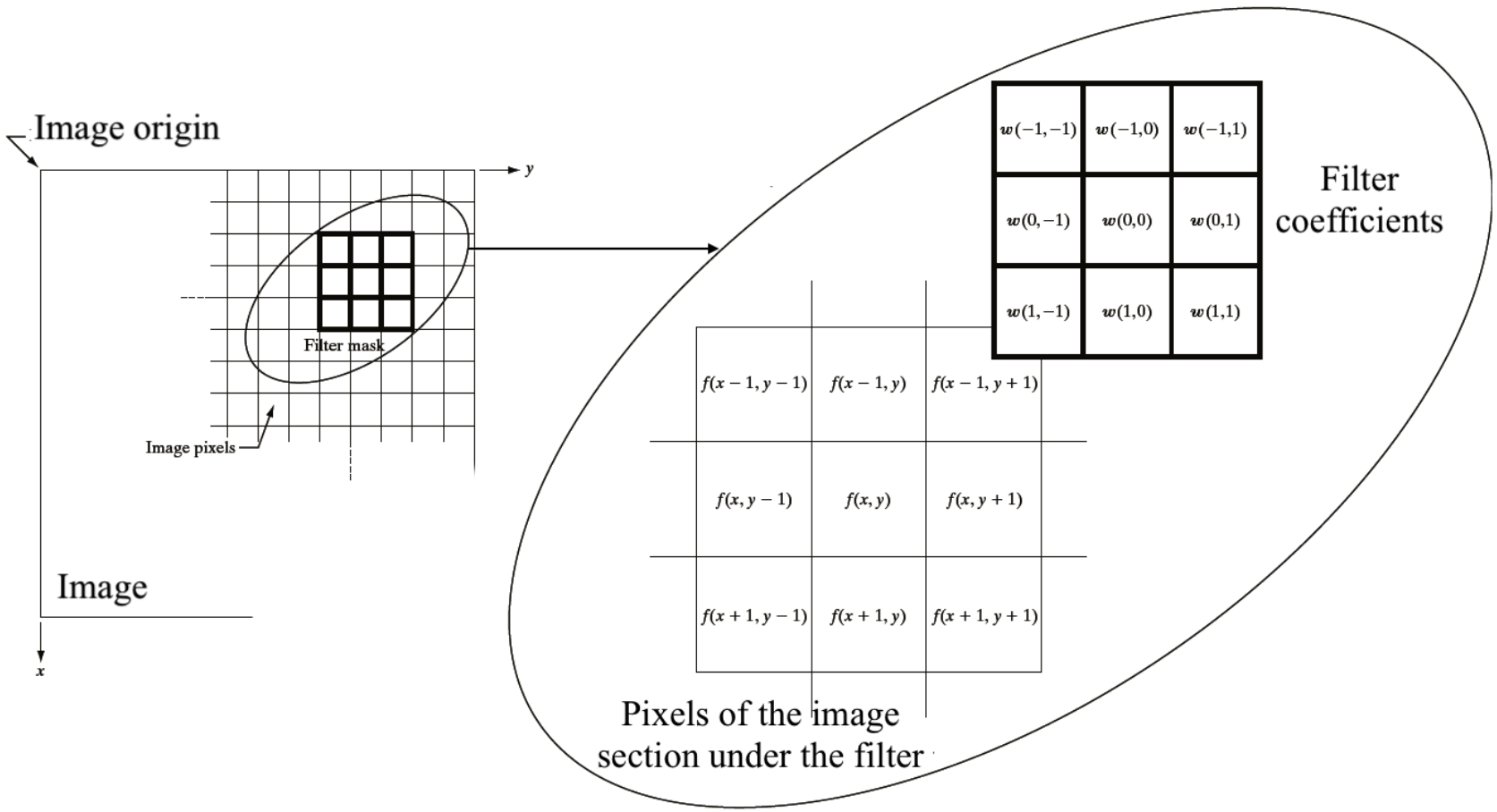



3D value space colour-cloud.

# Interlude: Spatial Filtering

# Spatial filtering

*Spatial filtering* modifies an image by replacing the value of each pixel by a function of the value of the pixel and its neighbours.



# Spatial filtering: Convolution

- A *filter kernel* or *filter mask* is a set of coefficients $w(s,t)$ where $(s,t)$ runs over a small neighborhood $\mathcal{N}$ around the origin: $\mathcal{N} = \{(s,t) : -a \leqslant s \leqslant a, -b \leqslant t \leqslant b\}$. Usually $\sum_{(s,t)\in\mathcal{N}} w(s,t) = 1$.

# Spatial filtering: Convolution

- A *filter kernel* or *filter mask* is a set of coefficients $w(s,t)$ where $(s,t)$ runs over a small neighborhood $\mathcal{N}$ around the origin: $\mathcal{N} = \{(s,t) : -a \leqslant s \leqslant a, -b \leqslant t \leqslant b\}$. Usually $\sum_{(s,t)\in\mathcal{N}} w(s,t) = 1$.
- Spatial filtering transforms an input image $f$ to an output image $g$ by moving the mask to each pixel $(x,y)$ and summing the pixel values in the neighborhood multiplied by the corresponding filter coefficient:

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

This is called the *convolution* of $w$ and $f$, written $w \star f$.

# Spatial filtering: Convolution and Correlation

- The *correlation* of $w$ and $f$ is defined as

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

- The *convolution* of $w$ and $f$ is defined as

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

# Spatial filtering: Convolution and Correlation

- The *correlation* of $w$ and $f$ is defined as

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

- The *convolution* of $w$ and $f$ is defined as

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

- Note that these operations can be easily mapped to one another:

$$g(x,y) = (w \circledast f)(x,y) = (\tilde{w} \star f)(x,y)$$

  where $\tilde{w}(s,t) = w(-s,-t)$ is the *mirrored* version of $w$.

- For symmetric masks (i.e. $w = \tilde{w}$) correlation and convolution are the same operation.

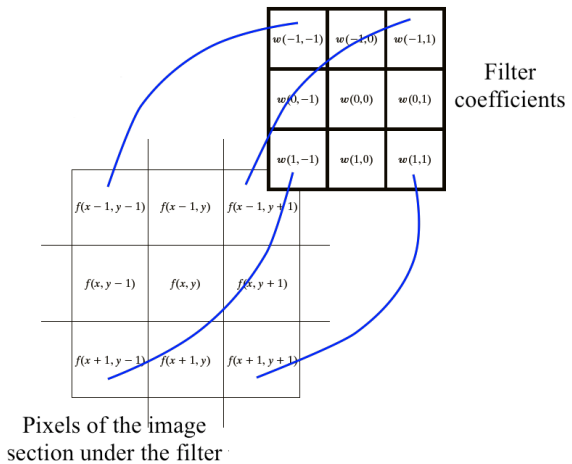
# Spatial filtering: Convolution and Correlation

- The *correlation* of $w$ and $f$ is defined as

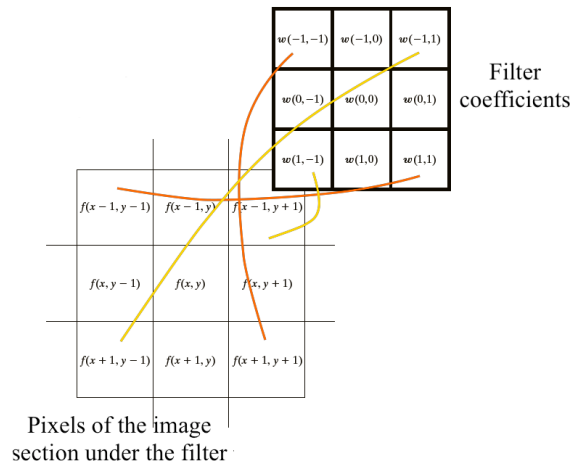
$$g(x,y) = (w \circledast f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)$$

- The *convolution* of $w$ and $f$ is defined as

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




Correlation




Convolution

# Filterbank types

- *Weighted window*: Weighting individual pixel contributions using a small *mask* surrounding each pixel.
- *Order statistic (percentile) filters*: rank the pixel values within the mask surrounding the center pixel (e.g. median filtering)
- *Morphological filters*: general class of nonlinear filters.

# Separable filters

Consider:

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

Assuming (approximately) squared images, the computational complexity $\mathcal{O}(N)$ for $g(x,y)$: $\mathcal{O}(N^2 \cdot n^2) = \mathcal{O}((N \cdot n)^2)$, for $n = 2a \rightarrow$ computationally expensive

Can this be circumvented ?

# Separable filters

Consider:

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

Assuming (approximately) squared images, the computational complexity $\mathcal{O}(N)$ for $g(x,y)$: $\mathcal{O}(N^2 \cdot n^2) = \mathcal{O}((N \cdot n)^2)$, for $n = 2a \rightarrow$ computationally expensive

Can this be circumvented ? consider following example matrices:

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

Which of those *could be split* ?

# Separable filters

A *separable filter* mask can be written as the convolution of lower-rank partial filters (i.e. vectors).

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

# Separable filters

A *separable filter* mask can be written as the convolution of lower-rank partial filters (i.e. vectors).

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

Result 1: reduced computational costs for an $N \times M$ image with a $m \times n$ mask, from

$$\mathcal{O}(M \cdot N \cdot m \cdot n)$$

down to

$$\mathcal{O}(M \cdot N \cdot (m + n))$$

.

# Separable filters

A *separable filter* mask can be written as the convolution of lower-rank partial filters (i.e. vectors).

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

Result 1: reduced computational costs for an $N \times M$ image with a $m \times n$ mask, from

$$\mathcal{O}(M \cdot N \cdot m \cdot n)$$

down to

$$\mathcal{O}(M \cdot N \cdot (m + n))$$

.

Result 2: Parallelization via shared-memory processing (SMP) & vectorization.

# Separable filters

A *separable filter* mask can be written as the convolution of lower-rank partial filters (i.e. vectors).

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

Result 1: reduced computational costs for an $N \times M$ image with a $m \times n$ mask, from

$$\mathcal{O}(M \cdot N \cdot m \cdot n)$$

down to

$$\mathcal{O}(M \cdot N \cdot (m + n))$$

.

Result 2: Parallelization via shared-memory processing (SMP) & vectorization.
Procedure:

- Perform 1D convolution per column independently
- Perform 1D convolution per row independently (on previous result)

# Separable filters

A *separable filter* mask **can be written** as the convolution of lower-rank partial filters (i.e. vectors).
→ **conditions** for replacing kernel matrix $w$ with vectors $v_x$, $v_y$

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

# Separable filters

A *separable filter* mask **can be written** as the convolution of lower-rank partial filters (i.e. vectors).
→ **conditions** for replacing kernel matrix $w$ with vectors $v_x, v_y$

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

- Assure $rank(w) = 1$
- $w$ is linear combination of $v_x \cdot v_y^T$
- all row-vectors $w_x(s)$ and column-vectors $w_y(t)$ of $w(s,t)$ need to be linearly-dependent

→ remember: *singular value decompostion (SVD)* from Linear Algebra

# Filtering: Derivative- and Integral Filters

# Weighted integration & local averages

Starting point: integration

$$g(x,y) = \frac{1}{\lambda}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(\tau,\kappa) f(x+\tau, y+\kappa) d\tau d\kappa$$

# Weighted integration & local averages

Starting point: integration

$$g(x,y) = \frac{1}{\lambda}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(\tau,\kappa) f(x+\tau, y+\kappa) d\tau d\kappa$$

$$g(x,y) = \frac{1}{\lambda}\sum_{i=-a}^{a}\sum_{j=-b}^{b} w_{ij} f(x+i, y+j),\ \lambda = \sum_{i=-a}^{a}\sum_{j=-b}^{b} w_{ij}$$

# Weighted integration & local averages

Starting point: integration

$$g(x,y) = \frac{1}{\lambda}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(\tau,\kappa) f(x+\tau, y+\kappa) d\tau d\kappa$$

$$g(x,y) = \frac{1}{\lambda}\sum_{i=-a}^{a}\sum_{j=-b}^{b} w_{ij} f(x+i, y+j),\ \lambda = \sum_{i=-a}^{a}\sum_{j=-b}^{b} w_{ij}$$

$\frac{1}{9} \times$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$\frac{1}{16} \times$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

(left) uniform, scalar $w$. (right) non-uniform, symmetric $w$.

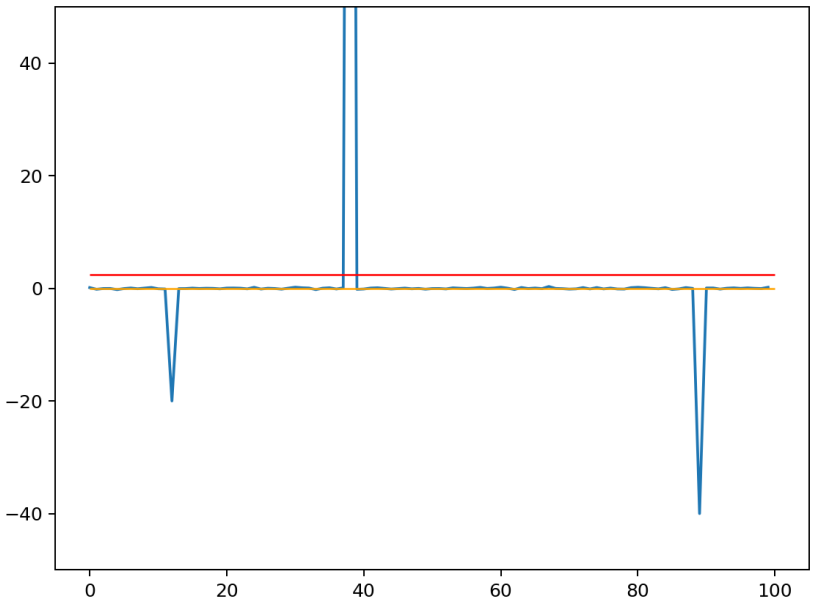
# Weighted average filters



(left): Original. (middle): uniform filter. (right): non-uniform filter.

# Weighted average drawback

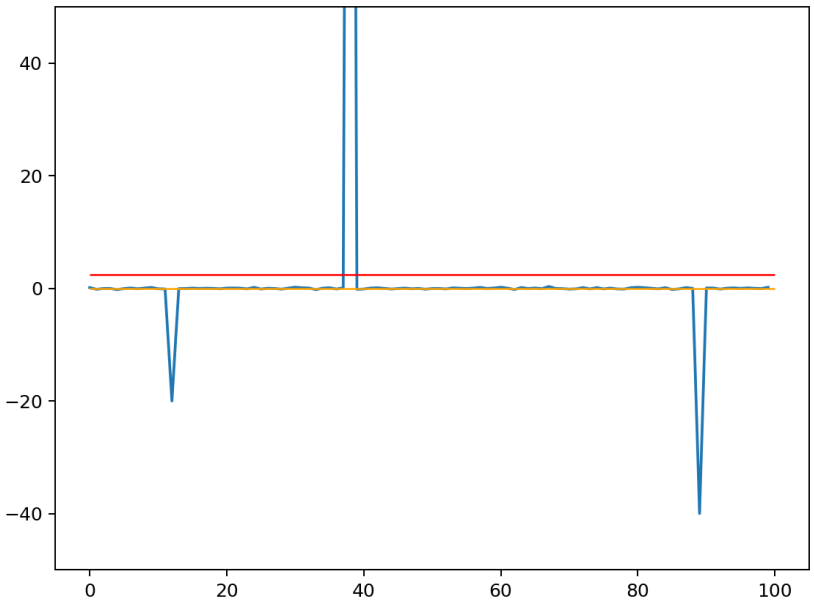
Issue of weighted-average integration: outliers




(left) Outliers mean deviation in 1D function. (right) spot-noise with uniform average-filter as 2D outlier analogue.

# Weighted average drawback

Issue of weighted-average integration: outliers

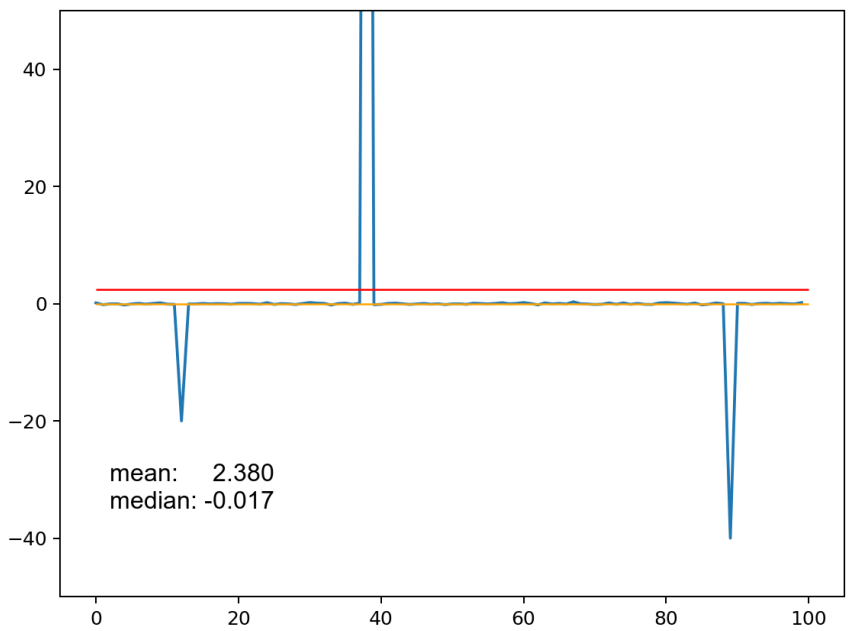



(left) Outliers mean deviation in 1D function. (right) spot-noise with uniform average-filter as 2D outlier analogue.

statistical tool to supress outliers: *median* computation

# Weighted average drawback

Issue of weighted-average integration: outliers




(left) $\Delta$ of mean & median in 1D function. (right) Difference of 2D mean-filter (top) & median-filter (bottom) result.

# Rank-order filters & median filtering

Meaning of 'median': the centred sample of a *ordered* distribution

Consequence: ordering of masked neighbourhood $w$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

# Rank-order filters & median filtering

Meaning of 'median': the centred sample of a *ordered* distribution

Consequence: ordering of masked neighbourhood $w$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$$\rightarrow \tilde{w} = \text{sort}((w \cdot f)(x, y))$$

# Rank-order filters & median filtering

Meaning of 'median': the centred sample of a *ordered* distribution

Consequence: ordering of masked neighbourhood $w$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$$\rightarrow \tilde{w} = \text{sort}((w \cdot f)(x, y)) \rightarrow g(x, y) = \tilde{w}(c)$$

, with $c = \frac{n \cdot m}{2}$ (for *median*) and $\{n, m\}$ being the kernel size(s)

# Rank-order filters & median filtering

Meaning of 'median': the centred sample of a *ordered* distribution

Consequence: ordering of masked neighbourhood $w$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$$\rightarrow \tilde{w} = \text{sort}((w \cdot f)(x, y)) \rightarrow g(x, y) = \tilde{w}(c)$$

, with $c = \frac{n \cdot m}{2}$ (for *median*) and $\{n, m\}$ being the kernel size(s)

Other rank-orders: minimum ($c = 0$) & maximum ($c = (n \cdot m) - 1$)
$\rightarrow$ coming back to them later.

# Discrete derivatives (1D) - recap

- Differentiation goal: get slope $f'(x)$ of function $f(x)$ at $x$:

$$f'(x) = \partial f = \frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

# Discrete derivatives (1D) - recap

- Differentiation goal: get slope $f'(x)$ of function $f(x)$ at $x$:

$$f'(x) = \partial f = \frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- in digital world: $h = \Delta x$, $x$ is *discrete* sample
- Hence: $Df(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}$

# Discrete derivatives (1D) - recap

- Differentiation goal: get slope $f'(x)$ of function $f(x)$ at $x$:

$$f'(x) = \partial f = \frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- in digital world: $h = \Delta x$, $x$ is *discrete* sample
- Hence: $Df(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}$
- in normalized domain: $\Delta x = 1$, resulting in:
  $Df(x) = f(x+1) - f(x) \rightarrow$ *central* derivative

# Discrete derivatives (1D) - recap

- Differentiation goal: get slope $f'(x)$ of function $f(x)$ at $x$:

$$f'(x) = \partial f = \frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

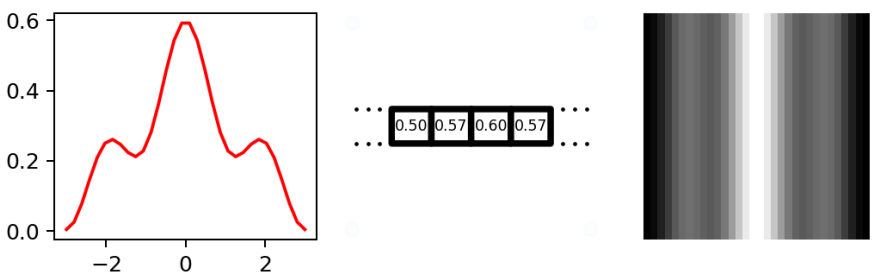
- in digital world: $h = \Delta x$, $x$ is *discrete* sample
- Hence: $Df(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}$
- in normalized domain: $\Delta x = 1$, resulting in:
  $Df(x) = f(x+1) - f(x) \rightarrow$ *central* derivative
- two-point derivative:
  $Df(x) = f(x+1) - f(x-1) = -f(x-1) + f(x+1)$

# Discrete derivatives (1D) - recap

- Differentiation goal: get slope $f'(x)$ of function $f(x)$ at $x$:

$$f'(x) = \partial f = \frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- in digital world: $h = \Delta x$, $x$ is *discrete* sample
- Hence: $Df(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}$
- in normalized domain: $\Delta x = 1$, resulting in:
  $Df(x) = f(x+1) - f(x) \rightarrow$ *central* derivative
- two-point derivative:
  $Df(x) = f(x+1) - f(x-1) = -f(x-1) + f(x+1)$
- if $\{x, f(x)\}$ discrete: $Df(x) = w \star f, w^T = (-1, 0, 1)$



# 2D Derivatives - $1^{st}$ order

- image $\mathrm{I}(x, y) = f(x, y)$: discrete spatial- ($\{x, y\}$) and value domain ($f(x, y)$)
- thus:

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

- filter kernel $w$: $w(s, t) = w_x(s) \cdot w_y(s)^T$
- **Audience Q**: what is the layout of $w_x$ and $w_y$ ?

# 2D Directional derivatives

- derivative filters are *directional*
  - horizontal: $D_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
  - vertical: $D_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

# 2D Directional derivatives

- derivative filters are *directional*
  - horizontal: $D_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
  - vertical: $D_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- $1^{st}$ order derivative filters are separable

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [-1\,0\,1] = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} [1\,1\,1] = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

# 2D Directional derivatives

- derivative filters are *directional*
  - horizontal: $D_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
  - vertical: $D_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- $1^{st}$ order derivative filters are separable

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [-1\,0\,1] = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} [1\,1\,1] = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- issue: double edges $\rightarrow$ illustration on blackboard

# Derivative filters & edges

- Integration = smoothing → Differentiation = sharpening ? (audience Q)

# Derivative filters & edges

- Integration = smoothing → Differentiation = sharpening ?
  (audience Q) No, not exactly.

# Derivative filters & edges

- Integration = smoothing $\rightarrow$ Differentiation = sharpening ? (audience Q) No, not exactly.
- Derivatives extract (directional) edges and corners
- Type of edge depends derivative order: $f'(x, y), f''(x, y)$
- Applications in object extraction, corner detection, etc.

# Prewitt operator

The Prewitt operator calculates the gradient magnitude of the image intensity at each point of an image (i.e. $f'(x, y)$). The result of the Prewitt operator at a pixel which is in a region of constant image intensity is zero, while it is non-zero at edges of regions.

# Prewitt operator

The Prewitt operator calculates the gradient magnitude of the image intensity at each point of an image (i.e. $f'(x, y)$). The result of the Prewitt operator at a pixel which is in a region of constant image intensity is zero, while it is non-zero at edges of regions.

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

$D_y = \frac{\partial f}{\partial y}$ (discrete)

| | | |
|---|---|---|
| +1 | 0 | -1 |
| +1 | 0 | -1 |
| +1 | 0 | -1 |

$D_x = \frac{\partial f}{\partial x}$ (discrete)

Kernels for the Prewitt operator.

# Prewitt operator

The Prewitt operator calculates the gradient magnitude of the image intensity at each point of an image (i.e. $f'(x, y)$). The result of the Prewitt operator at a pixel which is in a region of constant image intensity is zero, while it is non-zero at edges of regions.

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

$D_y = \frac{\partial f}{\partial y}$ (discrete)

| | | |
|---|---|---|
| +1 | 0 | -1 |
| +1 | 0 | -1 |
| +1 | 0 | -1 |

$D_x = \frac{\partial f}{\partial x}$ (discrete)

Kernels for the Prewitt operator.

We define

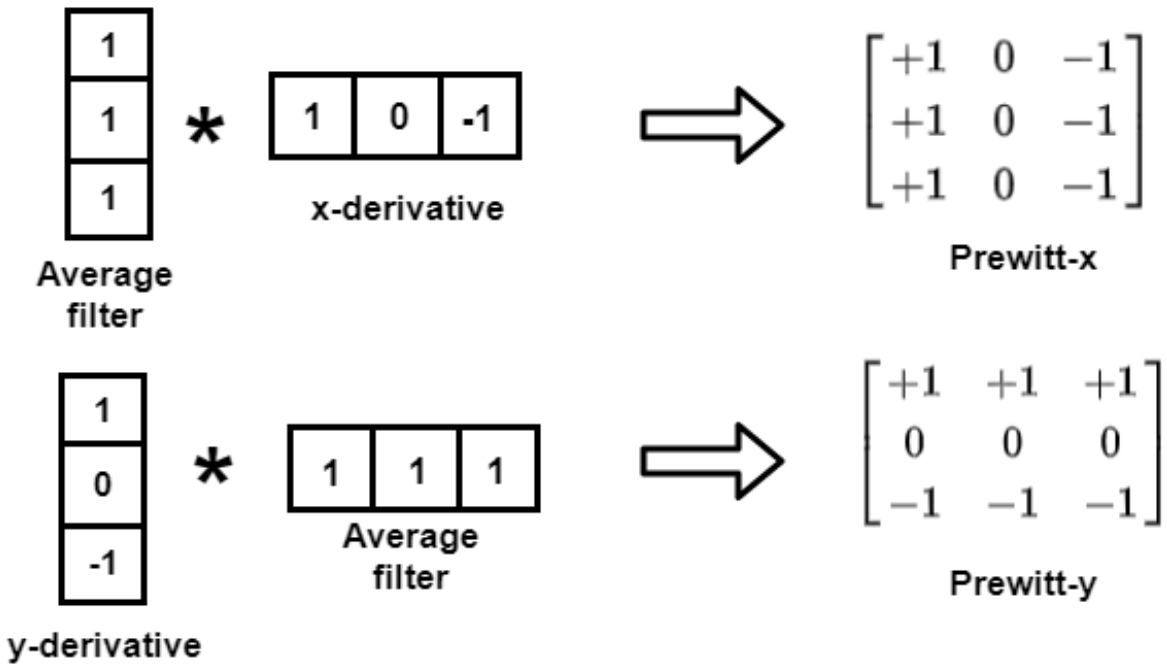
$$P_x(I) = D_x \star I \quad \text{and} \quad P_y(I) = D_y \star I$$

The Prewitt operator is then defined as

$$P(I) = \sqrt{(P_x(I))^2 + (P_y(I))^2}$$

# Prewitt operator

# Prewitt operator as a separable filter



# Sobel operator

- Each point is convolved with two kernels, one for detecting horizontal edges and the other for vertical edges.
- Output of the operator is the maximum or square root of the two convolutions.

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

Kernels for the Sobel operator.

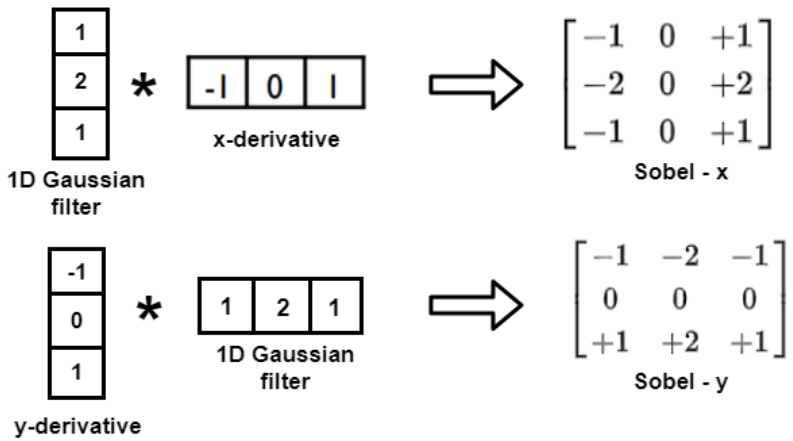
# Sobel operator

- Each point is convolved with two kernels, one for detecting horizontal edges and the other for vertical edges.
- Output of the operator is the maximum or square root of the two convolutions.

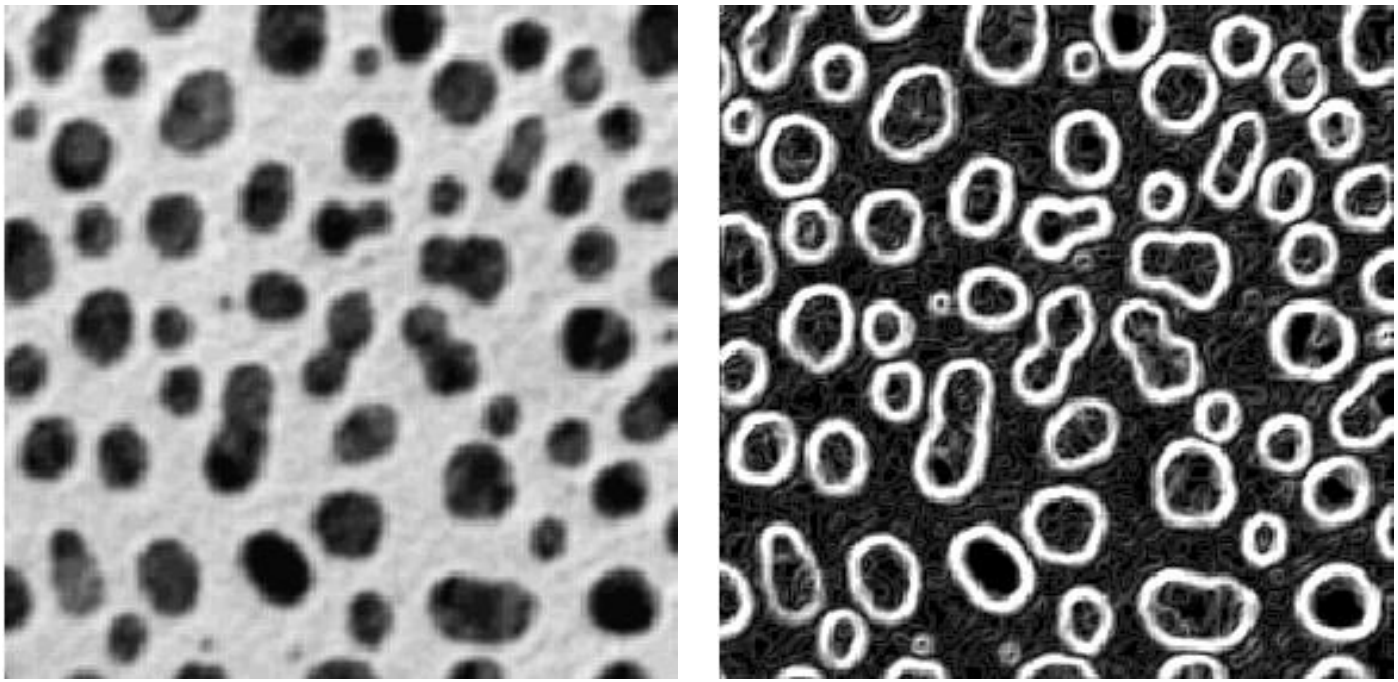
| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

Kernels for the Sobel operator.



# Sobel operator



(left): Original. (right): Sobel filter.

# 2D Derivatives - $2^{nd}$ order

- function analysis:
  $1^{st}$ order $\rightarrow$ slopes, $2^{nd}$ order $\rightarrow$ curvature

# 2D Derivatives - $2^{nd}$ order

- function analysis:
  $1^{st}$ order $\rightarrow$ slopes, $2^{nd}$ order $\rightarrow$ curvature
- formulation for $2^{nd}$ order derivative:

$$f''(x) = \partial^2 f = \frac{\partial^2 f}{\partial x^2} = \lim_{h \to 0} \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

- digital, discrete form: $D^2 f(x) = f(x-1) - 2f(x) + f(x+1)$,
  $D^2 f(x) = w \star f, w^T = (1, -2, 1)$

# 2D Derivatives - $2^{nd}$ order

- function analysis:
  $1^{st}$ order $\rightarrow$ slopes, $2^{nd}$ order $\rightarrow$ curvature
- formulation for $2^{nd}$ order derivative:

$$f''(x) = \partial^2 f = \frac{\partial^2 f}{\partial x^2} = \lim_{h \to 0} \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

- digital, discrete form: $D^2 f(x) = f(x-1) - 2f(x) + f(x+1)$,
  $D^2 f(x) = w \star f, w^T = (1, -2, 1)$
- goal: *extremal* point detection $\rightarrow$ *maximum*

# 2D Derivatives - $2^{nd}$ order

- function analysis:
  $1^{st}$ order $\rightarrow$ slopes, $2^{nd}$ order $\rightarrow$ curvature
- formulation for $2^{nd}$ order derivative:

$$f''(x) = \partial^2 f = \frac{\partial^2 f}{\partial x^2} = \lim_{h \to 0} \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

- digital, discrete form: $D^2 f(x) = f(x-1) - 2f(x) + f(x+1)$,
  $D^2 f(x) = w \star f, w^T = (1, -2, 1)$
- goal: *extremal* point detection $\rightarrow$ *maximum*
  - $1^{st}$ order filter $\rightarrow$ *stationary* point: $\frac{\partial f}{\partial x} = 0$
  - stationary point: either maximum or minimum

# 2D Derivatives - $2^{nd}$ order

- function analysis:
  $1^{st}$ order $\rightarrow$ slopes, $2^{nd}$ order $\rightarrow$ curvature
- formulation for $2^{nd}$ order derivative:

$$f''(x) = \partial^2 f = \frac{\partial^2 f}{\partial x^2} = \lim_{h \to 0} \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

- digital, discrete form: $D^2 f(x) = f(x-1) - 2f(x) + f(x+1)$,
  $D^2 f(x) = w \star f, w^T = (1, -2, 1)$
- goal: *extremal* point detection $\rightarrow$ *maximum*
  - $1^{st}$ order filter $\rightarrow$ *stationary* point: $\frac{\partial f}{\partial x} = 0$
  - stationary point: either maximum or minimum
  - $2^{nd}$ order test:

$$x = \begin{cases} max(f(x)) \text{ if } \frac{\partial^2 f}{\partial x^2} < 0 \\ min(f(x)) \text{ if } \frac{\partial^2 f}{\partial x^2} > 0 \\ \text{inflection point if } \frac{\partial^2 f}{\partial x^2} = 0 \end{cases}$$

# Laplacian Operator

- observed in $1^{st}$ order: derivative vs. operator
  - derivative: directional (i.e. *anisotropic*); multiple filter kernels
  - operator: absolute (i.e. *isotropic*); single filter kernel
- same procedure in $2^{nd}$ order filtering: Laplacian $\rightarrow$ isotropic
- Laplacian is the sum of the $2^{nd}$ order derivatives $D_{xx}$ and $D_{yy}$

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

# Laplacian Operator

- observed in $1^{st}$ order: derivative vs. operator
  - derivative: directional (i.e. *anisotropic*); multiple filter kernels
  - operator: absolute (i.e. *isotropic*); single filter kernel
- same procedure in $2^{nd}$ order filtering: Laplacian $\rightarrow$ isotropic
- Laplacian is the sum of the $2^{nd}$ order derivatives $D_{xx}$ and $D_{yy}$

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

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

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

# Laplacian Operator

- observed in $1^{st}$ order: derivative vs. operator
  - derivative: directional (i.e. *anisotropic*); multiple filter kernels
  - operator: absolute (i.e. *isotropic*); single filter kernel
- same procedure in $2^{nd}$ order filtering: Laplacian $\rightarrow$ isotropic
- Laplacian is the sum of the $2^{nd}$ order derivatives $D_{xx}$ and $D_{yy}$

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

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

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

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

# Laplacian Operator

Laplacian is the sum of the $2^{nd}$ order derivatives $D_{xx}$ and $D_{yy}$

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

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

# Laplacian Operator

Laplacian is the sum of the $2^{nd}$ order derivatives $D_{xx}$ and $D_{yy}$

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

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

**Audience Q**: How would about a Laplacian that includes diagonal edges ?

# Laplacian Operator

Laplacian is the sum of the $2^{nd}$ order derivatives $D_{xx}$ and $D_{yy}$

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

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

**Audience Q**: How would about a Laplacian that includes diagonal edges ?

# 2D Derivatives - $2^{nd}$ order directional

- until now: univariate derivatives per spatial dimension $\{x, y\}$
- alternative interpretation: two successive derivations along same dimension

# 2D Derivatives - $2^{nd}$ order directional

- until now: univariate derivatives per spatial dimension $\{x, y\}$
- alternative interpretation: two successive derivations along same dimension
- actual interest: multivariate derivation $\rightarrow$ directional derivatives
  - $1^{st}$ order (central differences): $D_x$, $D_y$

$$D_x f(x, y) = -f(x, y) + f(x + 1, y)$$

$$D_y f(x, y) = -f(x, y) + f(x, y + 1)$$

# 2D Derivatives - $2^{nd}$ order directional

- until now: univariate derivatives per spatial dimension $\{x, y\}$
- alternative interpretation: two successive derivations along same dimension
- actual interest: multivariate derivation $\rightarrow$ directional derivatives
  - $1^{st}$ order (central differences): $D_x$, $D_y$

$$D_x f(x, y) = -f(x, y) + f(x + 1, y)$$

$$D_y f(x, y) = -f(x, y) + f(x, y + 1)$$

  - $2^{nd}$ order: $D_{xx}$, $D_{yy}$, $D_{yx} = D_{xy}$

$$D_{xx} f(x, y) = -D_x(x - 1, y) + D_x(x, y)$$

$$D_{xy} f(x, y) = -D_x(x, y - 1) + D_x(x, y)$$

$$D_{yy} f(x, y) = -D_y(x, y - 1) + D_y(x, y)$$

# 2D Derivatives - $2^{nd}$ order directional

- actual interest: multivariate derivation $\rightarrow$ directional derivatives
  - $1^{st}$ order: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$
  - $2^{nd}$ order: $\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2}, \frac{\partial^2 f}{\partial y \partial x} = \frac{\partial^2 f}{\partial x \partial y}$
- combiniation of dir. derivatives in *Hessian matrix* (2D)

$$H\ f(x,y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

# 2D Derivatives - $2^{nd}$ order directional

- actual interest: multivariate derivation $\rightarrow$ directional derivatives
  - $1^{st}$ order: $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$
  - $2^{nd}$ order: $\frac{\partial^2 f}{\partial x^2}$, $\frac{\partial^2 f}{\partial y^2}$, $\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial^2 f}{\partial x \partial y}$
- combiniation of dir. derivatives in *Hessian matrix* (2D)

$$H\ f(x,y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

- $H\ f(x,y)$ simplifies maximum test via its *determinant*:

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

$$(x,y) = \begin{cases} max(f(x,y)) \text{ if } D_{xx} f(x,y) < 0, Det\ f(x,y) > 0 \\ max(f(x,y)) \text{ if } D_{xx} f(x,y) > 0, Det\ f(x,y) > 0 \\ \text{inflection point if } Det\ f(x,y) < 0 \end{cases}$$

# Derivative Filters - Edges & Sharpening

- Edges → local intensity maxima → highpass- and derivative filters → $f'(x, y)/D_x\ D_y$ or $f''(x, y)$ and similar

# Derivative Filters - Edges & Sharpening

- Edges → local intensity maxima → highpass- and derivative filters → $f'(x, y)/D_x\ D_y$ or $f''(x, y)$ and similar
- Sharpening goal: enhance *fine details* in distinctive image areas

# Derivative Filters - Edges & Sharpening

- Edges → local intensity maxima → highpass- and derivative filters → $f'(x, y)/D_x\ D_y$ or $f''(x, y)$ and similar
- Sharpening goal: enhance *fine details* in distinctive image areas
- Sharpeing filters via edges: $\mathrm{I}(x, y) + c \cdot f'(x, y)$
  → enhance borders, edges and corners

# Derivative Filters - Edges & Sharpening

- Edges → local intensity maxima → highpass- and derivative filters → $f'(x, y)/D_x\ D_y$ or $f''(x, y)$ and similar
- Sharpening goal: enhance *fine details* in distinctive image areas
- Sharpeing filters via edges: $\mathrm{I}(x, y) + c \cdot f'(x, y)$
  → enhance borders, edges and corners
- Drawback: noise pixels are "corners" too (i.e. local intensity maxima)

# Sharpening filters

- Prewitt:

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

| | | |
|---|---|---|
| +1 | 0 | -1 |
| +1 | 0 | -1 |
| +1 | 0 | -1 |

# Sharpening filters

- Prewitt:

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

| | | |
|---|---|---|
| +1 | 0 | -1 |
| +1 | 0 | -1 |
| +1 | 0 | -1 |

- Sobel:

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

# Sharpening filters

- Prewitt:

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

| | | |
|---|---|---|
| +1 | 0 | -1 |
| +1 | 0 | -1 |
| +1 | 0 | -1 |

- Sobel:

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

- Laplacian:

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

# Sharpening filters

- Prewitt:

| | | |
|---|---|---|
| +1 | +1 | +1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

| | | |
|---|---|---|
| +1 | 0 | -1 |
| +1 | 0 | -1 |
| +1 | 0 | -1 |

- Sobel:

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

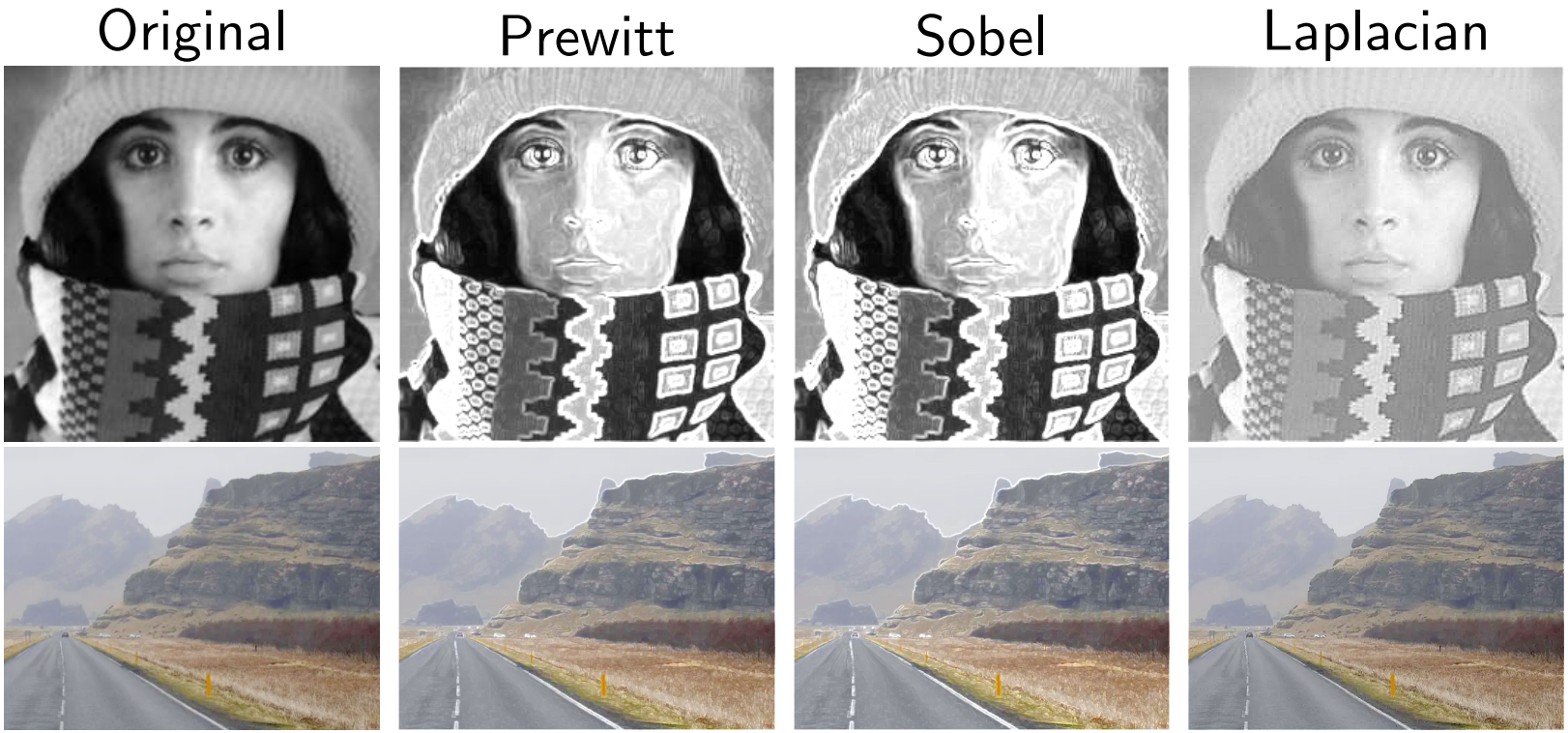
| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

- Laplacian:

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

- hint 1: *subtract* $\nabla^2 f$ from I ($\rightarrow$ book, sec. 3.6.3)
- hint 2: use luminance processing- and image composition

# Sharpening filters - Comparison



# That's it for this week!