



Digital Images

Spatial Domain  
Filtering

Smoothing Filters

Derivative Filters

# Chapter 1

## Images and Image Filtering

Computer Vision I: Variational Methods

Winter 2016/17

Dr. Yvain QUÉAU  
Chair for Computer Vision and Pattern Recognition  
Departments of Informatics & Mathematics  
Technical University of Munich



## 1 Digital Images

## 2 Spatial Domain Filtering

## 3 Smoothing Filters

## 4 Derivative Filters

Digital Images

Spatial Domain  
Filtering

Smoothing Filters

Derivative Filters



Digital Images

Spatial Domain  
Filtering

Smoothing Filters

Derivative Filters

How to represent and store a digital image?



## 1 Digital Images

## 2 Spatial Domain Filtering

## 3 Smoothing Filters

## 4 Derivative Filters

Digital Images

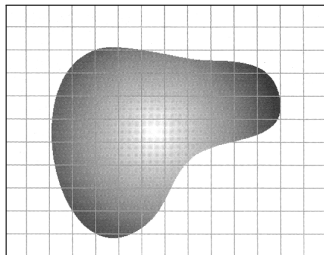
Spatial Domain  
Filtering

Smoothing Filters

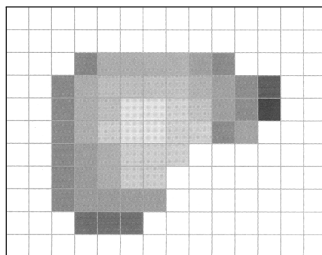
Derivative Filters

## Continuous versus Discrete

Digital images are discrete, both in space and in their values. Nevertheless, one can represent and analyze them in a continuous setting.



continuous



discrete  
(sampling & quantization)





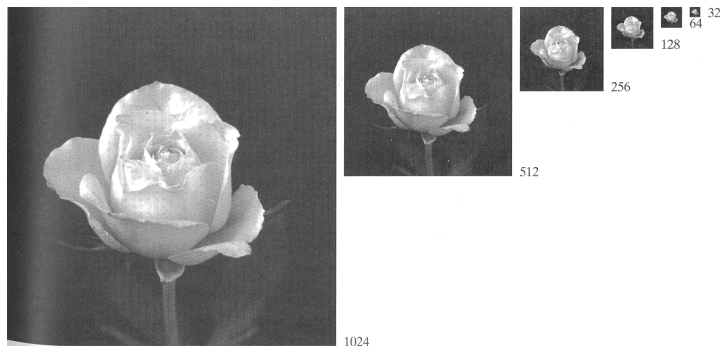
- There are different levels of discretization:
  - Discretization in color or brightness space (=quantization)
  - Discretization in (physical) space
  - Discretization in time (for videos)
- Continuous representation:  $f : (\Omega \subset \mathbb{R}^n) \rightarrow \mathbb{R}^d$
- $n = 2$ : 2-dim. images,  
 $n = 3$ : volumetric images or 2-dim. videos,  
 $n = 4$ : volume + time,...
- $d = 1$ : brightness images,  
 $d = 3$ : color images,  
 $d > 1$ : multispectral images
- Discretization (width  $W$ , height  $H$ ):

$$f(x, y) \longrightarrow \begin{bmatrix} f(1, 1) & f(2, 1) & \dots & f(W, 1) \\ f(1, 2) & f(2, 2) & \dots & f(W, 2) \\ \vdots & \vdots & \ddots & \vdots \\ f(1, H) & f(2, H) & \dots & f(W, H) \end{bmatrix}$$



- Advantages of discrete representations:
  - Digital images *are* discrete, and their processing in a computer will ultimately require a discretization.
  - No numerical approximations in modeling the transition from discrete to continuous.
  - For various problems there exist efficient algorithms from discrete optimization.
  
- Advantages of continuous representations:
  - The world observed through the camera is continuous.
  - There exists abundant mathematical theory for the treatment of continuous functions (functional analysis, differential geometry, partial differential equations, group theory,...).
  - Certain properties (rotational invariance) are easier to model because artefacts of discretization can be ignored.
  - Continuous models correspond to the **limit of infinitely fine discretization**.

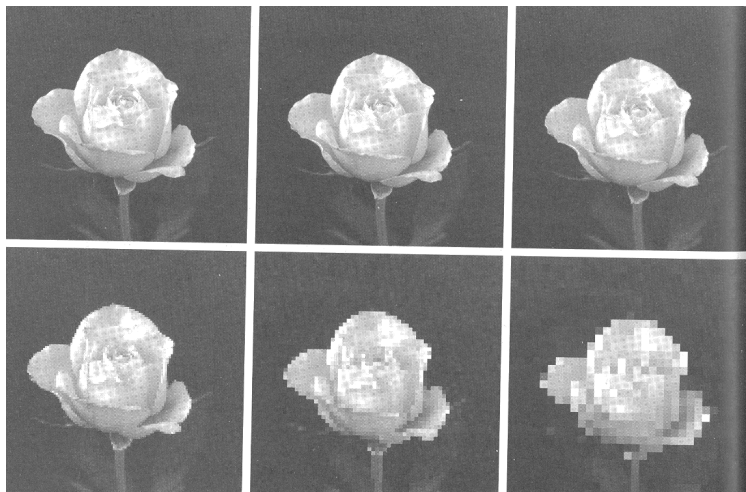
# Spatial Subsampling



Representation of an image with fewer and fewer pixels  
(source: Gonzalez & Woods)



# Spatial Subsampling



Subsampled from  $1024^2$  to  $32^2$  and enlarged  
(Source: Gonzalez & Woods)



Digital Images

Spatial Domain  
Filtering

Smoothing Filters

Derivative Filters

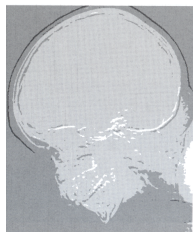
# Brightness Quantization



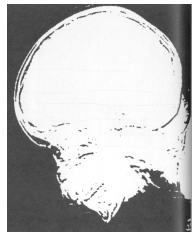
256 levels



16 levels



4 levels

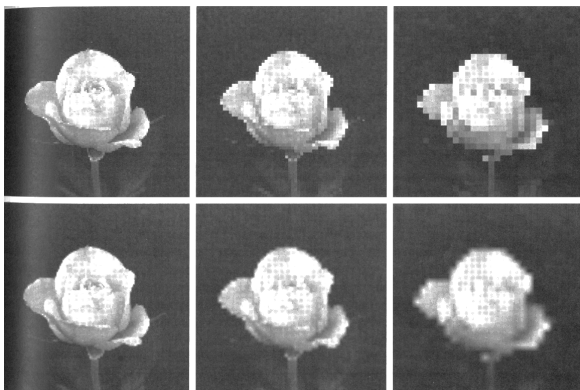


2 levels

(Source: Gonzalez & Woods)

“typical” images:  $256 \times 256$  pixels with 256 brightness values





lower row: Bilinear interpolation of the upper row  
(Source: Gonzalez & Woods)

Bilinear interpolation:  $\hat{f}(x, y) = ax + by + cxy + d$  with coefficients  $a, b, c, d$  determined by fitting to brightness values of 4 neighboring pixels.

Alternatives: nearest neighbor or bi-cubic interpolation,...



# Overview

1 Digital Images

**2 Spatial Domain Filtering**

3 Smoothing Filters

4 Derivative Filters



Digital Images

**Spatial Domain  
Filtering**

Smoothing Filters

Derivative Filters



- Filtering of an image in the spatial domain can be represented by an operator  $T$ :

$$g(x, y) = (Tf)(x, y),$$

where  $f$  denotes the input image and  $g$  the processed image.

- Typically  $T$  acts on a certain spatial neighborhood.
- The simplest form of  $T$  is an operator which simply models a **local** brightness transformation:

$$s = T(r),$$

where  $r$  is the input brightness at a certain location and  $s$  the respective brightness in the transformed image.



- Typically the goal of brightness transforms is to transfer the brightness values into a range that facilitates it for humans to **see the relevant structures**, i.e. the semantically important brightness transitions should be in a range where retinal receptors are particularly sensitive.
- In most cases one considers **monotonically nondecreasing** brightness transforms  $T(r)$ , i.e. transforms which preserve the ordering:

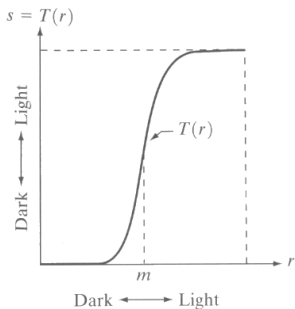
$$r_1 \leq r_2 \Rightarrow T(r_1) \leq T(r_2)$$

In the case that  $r_1 < r_2 \Rightarrow T(r_1) < T(r_2)$  these transforms are called **strictly monotonous**.

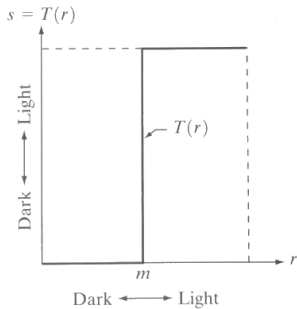
- Strictly monotonous brightness transforms are **invertible**, i.e. the original image data can be recovered from the filtered image.

## Contrast Enhancement

- Two important examples of brightness transforms are **contrast stretching (Kontrastverstärkung)** and **thresholding (Schwellwertbildung)**:



contrast stretching



thresholding

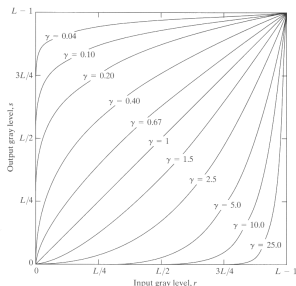
(Source: Gonzalez & Woods)

- Thresholding can be seen as a limiting case of contrast stretching. It provides a binary image as output which is often useful for further processing.



## Log- and Powerlaw-Transform

- Two further examples of brightness transforms are the **logarithm transform**:  $s = c \log(1 + r)$ , and the **powerlaw transform**:  $s = cr^\gamma$ .



Powerlaw transform  $s = cr^\gamma$   
(Source: Gonzalez & Woods)

- Nonlinear brightening ( $\gamma < 1$ ) or darkening ( $\gamma > 1$ ).
- The correction of brightness changes (due to image acquisition and image display) with an inverse powerlaw transform is called **gamma correction**.





## Example: Contrast Enhancement



Digital Images

Spatial Domain  
Filtering

Smoothing Filters

Derivative Filters



Input

$\gamma = 0.6$

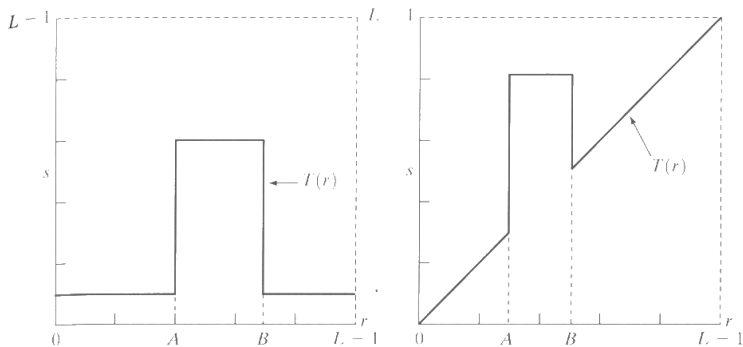
$\gamma = 0.4$

$\gamma = 0.3$

(Source: Gonzalez & Woods)

Through the powerlaw transform certain structures in the input image become more visible.

# Gray Level Slicing



What effects do the above transforms have?

## Linear Filters

- The term **filtering** is derived from frequency space methods where a spatial smoothing of the brightness values corresponds to a signal transform where high-frequency components are **filtered out**.
- An operator  $T$  is called **linear** if the following properties hold:
  - 1  $T(f + g) = T(f) + T(g) \quad \forall \text{ images } f, g.$
  - 2  $T(\alpha f) = \alpha T(f) \quad \forall \text{ images } f, \text{ scalars } \alpha.$
- For linear operators, the output brightness values are linear combinations of the input brightness values. Among the linear transformations is the **convolution (Faltung)**:

$$g(x, y) = \int w(x', y') f(x - x', y - y') dx' dy'.$$

In a spatially discrete setting, this corresponds to a weighted sum:

$$g(i, j) = \sum_{m, n} w(m, n) f(i - m, j - n).$$





- In practice this summation extends over a certain neighborhood, often called **window**. The matrix of weights  $w(m, n)$  is called a **mask**.

$$g(i, j) = \sum_{m, n} w(m, n) f(i - m, j - n)$$

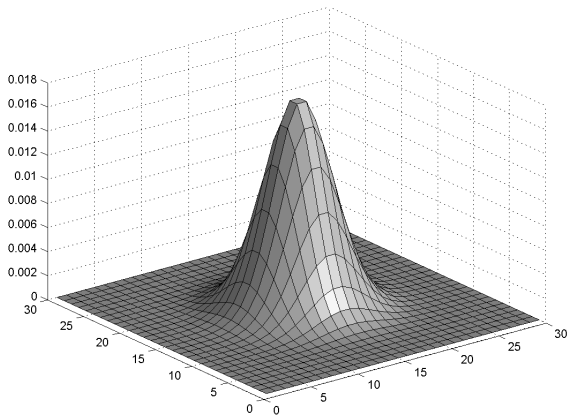
- For example, the  $3 \times 3$  mask:

|             |            |            |
|-------------|------------|------------|
| $w(-1, -1)$ | $w(0, -1)$ | $w(1, -1)$ |
| $w(-1, 0)$  | $w(0, 0)$  | $w(1, 0)$  |
| $w(-1, 1)$  | $w(0, 1)$  | $w(1, 1)$  |

- In the continuous representation the weight function  $w(x', y')$  is called **convolution kernel (Faltungskern)**:

$$g(x, y) = (w * f)(x, y) \equiv \int w(x', y') f(x - x', y - y') dx' dy'$$

# Gaussian Convolution



“Everybody believes in the exponential law of errors: the experimenters, because they think it can be proved by mathematics; and the mathematicians, because they believe it has been established by observation.” (G. Lippman, in a letter to H. Poincaré)



Digital Images

Spatial Domain  
Filtering

Smoothing Filters

Derivative Filters

How to process images using filters?

# Overview

1 Digital Images

2 Spatial Domain Filtering

**3 Smoothing Filters**

4 Derivative Filters



Digital Images

Spatial Domain  
Filtering

**Smoothing Filters**

Derivative Filters



- Smoothing or **low-pass filtering** typically averages the brightness values in a certain spatial neighborhood.
- The most common example of smoothing kernel is the **Gaussian kernel**. It induces a weighted average of brightness values on the scale determined by the standard deviation  $\sigma$ :

$$w(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

- A multitude of alternative convolution kernels (or filter masks) is conceivable, for example **box filters** which are constant within the window:

$$w(i, j) = \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

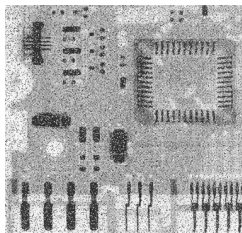
- For pixels at the image boundary, the weight mask must be adapted appropriately.



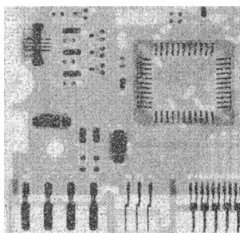


- A specific class of **nonlinear** filters are the **order statistics filters**. For these filters, the brightness of the filtered image at a given pixel depends on the order of brightness values in a certain neighborhood.
- The best known example of an order statistics filter is the **median filter**. For this filter, each pixel is assigned the median value of brightness values in its neighborhood.
- Example: The median of the brightness values  $\{1, 2, 2, 3, 4, 5, 20\}$  is 3, i.e. the central value after sorting.
- Median filters are particularly useful for reduction of **impulse noise**, also called **salt-and-pepper noise**, i.e. noise where some brightness values are randomly replaced by black or white values.
- Median filters typically induce less blurring than Gaussian or other linear smoothing filters.

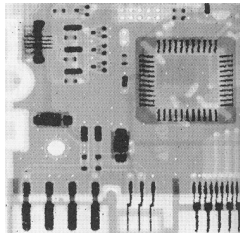
## Median versus Gauss



noisy input



Gauss filtered



median filtered

In contrast to the Gaussian filter (center), the median filter better removes noise without blurring structures. **Nonlinear methods** are often more general and more powerful than linear approaches.





Digital Images

Spatial Domain  
Filtering

Smoothing Filters

Derivative Filters

How to extract information from images using filters?

# Overview

1 Digital Images

2 Spatial Domain Filtering

3 Smoothing Filters

4 **Derivative Filters**



Digital Images

Spatial Domain  
Filtering

Smoothing Filters

**Derivative Filters**

## Derivative Filters

- **Derivative filters** capture the spatial variations of brightness. In particular, they provide information about edges or corners in an image. In a simplified world of black objects on white ground, these brightness edges correspond to object boundaries.
- Mathematically the partial derivatives of the function  $f(x, y)$  with respect to  $x$  is defined as:

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

- The **continuous** derivative can be approximated discretely by (symmetric) finite differences:

$$\partial_x f(x, y) \equiv f_x(x, y) \equiv \frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x - 1, y)}{2}$$

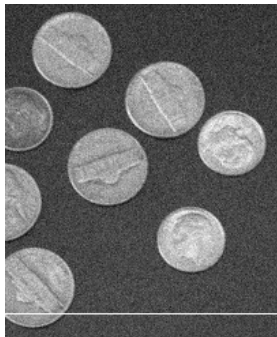
- Alternatives:

$$\partial_x f(x, y) \approx f(x + 1, y) - f(x, y) \quad (\text{forward difference})$$

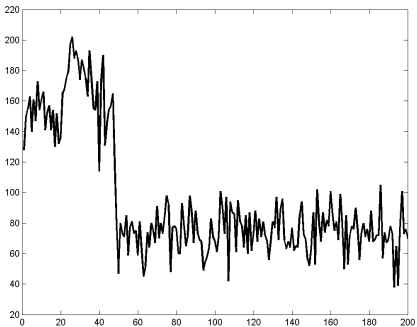
$$\partial_x f(x, y) \approx f(x, y) - f(x - 1, y) \quad (\text{backward difference}).$$



## Example: 1D Brightness Profile

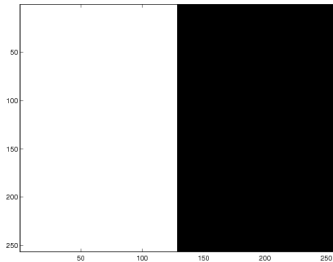


Input

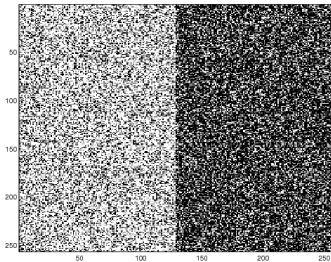


1D brightness profile

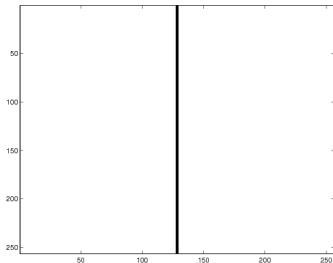
## Example of the First Derivative



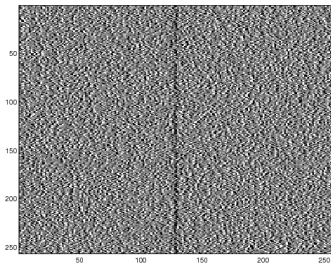
Input image



Input with noise



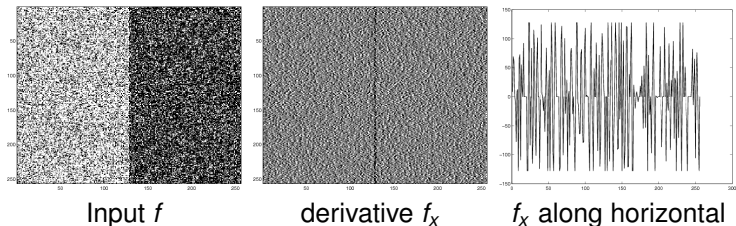
x-derivative



derivative of noisy image



# Noise Sensitivity of the Derivative



Observation:

- **Vertical edges** can be determined as maxima of the norm of the  $x$ -derivative.
- **Horizontal edges** can be determined as maxima of the norm of the  $y$ -derivative.
- This approach only allows to selectively determine horizontal or vertical edges.
- It is very **sensitive to noise**.



## The Image Gradient $\nabla f(x, y)$

- The gradient of a function  $f(x, y)$  is the vector:

$$\nabla f(x, y) = \begin{pmatrix} \partial_x f \\ \partial_y f \end{pmatrix} \equiv \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$

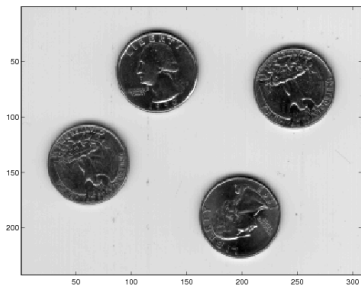
- The **gradient norm** (often also called “gradient”) is given by the Euclidean length of the gradient vector:

$$|\nabla f(x, y)| = \left| \begin{pmatrix} f_x \\ f_y \end{pmatrix} \right| = \sqrt{(f_x)^2 + (f_y)^2}$$

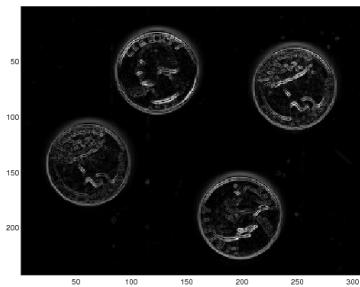
- The gradient norm is a **nonlinear operator** for detection of edges in arbitrary orientation.
- The gradient norm is **rotationally covariant** (sometimes called “rotationally invariant”). This means: The gradient norm of the rotated image is the same as the rotated gradient norm of the unrotated image. This implies that the performance of this operator does not depend on how the input image is rotated.



## Example of the Image Gradient



Input image



Gradient norm

## The Laplace Operator $\Delta f(x, y)$



- The **divergence** of a vector  $v = (v_1, v_2)$  is defined as  $\nabla v = \partial_x v_1 + \partial_y v_2$ .
- The **Laplace operator**  $\Delta$  is given by the concatenation of gradient and divergence:

$$\Delta f(x, y) = \nabla^2 f(x, y) = \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix} \begin{pmatrix} f_x \\ f_y \end{pmatrix} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = f_{xx} + f_{yy}$$

- The Laplace operator is **linear**:

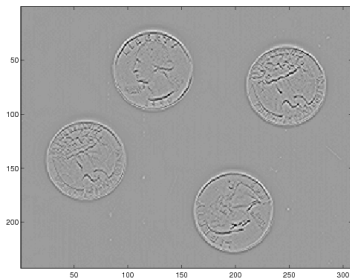
$$\Delta(\alpha_1 f(x) + \alpha_2 g(x)) = \alpha_1 \Delta f(x) + \alpha_2 \Delta g(x) \quad \forall \alpha_1, \alpha_2 \in \mathbb{R}, \forall f, g$$

- **Linearity** has several practical advantages. Linearity implies that it does not matter whether one first sums images and then processes them or vice versa.

## Example of the Laplace Operator



Input image



Laplace operator of the image

## Discretization of Derivatives



There exist different discrete approximations of derivatives. In the following we shall denote width and height of a single pixel by  $h_x$  and  $h_y$ . Then the  $x$ -derivative of a brightness image  $f$  at pixel  $(i, j)$  can be approximated as:

### 1 Symmetric differences:

$$f_x(i, j) \approx \frac{f(i+1, j) - f(i-1, j)}{2h_x}$$

### 2 Forward differences:

$$f_x(i, j) \approx \frac{f(i+1, j) - f(i, j)}{h_x}$$

### 3 Backward differences:

$$f_x(i, j) \approx \frac{f(i, j) - f(i-1, j)}{h_x}$$

How do these masks differ? Which one is better?

## The Taylor Series Expansion

- The key idea of the **Taylor expansion** is to approximate a function in the vicinity of an expansion point by a truncated power series:

$$f(x_0 + \epsilon) = f(x_0) + \epsilon f'(x_0) + \frac{\epsilon^2}{2} f''(x_0) + O[\epsilon^3]$$

- This expansion is easily derived as follows. Assume that the function  $f(x_0 + x)$  can be written as a linear combination of powers of  $x$ :

$$f(x_0 + x) = a_0 + a_1x + a_2x^2 + \dots = \sum_{n=0}^{\infty} a_n x^n$$

By inserting  $x = 0$  into various derivatives of this expression, we get:

$$f(x_0) = a_0, \quad f'(x_0) = a_1, \quad f''(x_0) = 2a_2$$

and in general:

$$f^{(n)}(x_0) = (n!)a_n \quad \Rightarrow \quad a_n = \frac{f^{(n)}(x_0)}{n!}$$



## Application to the Brightness Function

Let  $f(i, j)$  denote the brightness at pixel  $(i, j)$ , and  $h_x$  the width of a pixel. Then we have:

$$f(i + 1, j) = f(i, j) + h_x f_x(i, j) + \frac{h_x^2}{2} f_{xx}(i, j) + O[h_x^3]$$

Similarly:

$$f(i - 1, j) = f(i, j) - h_x f_x(i, j) + \frac{h_x^2}{2} f_{xx}(i, j) + O[h_x^3]$$

Subtracting both equations leads to:

$$f_x(i, j) = \frac{f(i + 1, j) - f(i - 1, j)}{2h_x} + O[h_x^2]$$

Instead, subtracting  $f(i, j)$  from the first equation leads to:

$$f_x(i, j) = \frac{f(i + 1, j) - f(i, j)}{h_x} + O[h_x]$$





- The difference between an analytical expression and its discrete representation is called **discretization error**.
- When discretizing a differential equation, the order of the discretization error is called **order of consistency**.
- The symmetric difference discretization is of order 2 because the discretization error is of order  $h_x^2$ . In contrast, forward or backward differences lead to a consistency order 1.
- In the numerical discretization of differential equations higher orders are typically better because they allow to faster approximate the continuum with finer discretizations, i.e.  $h_x \rightarrow 0$ .
- Using Taylor expansions of higher order one can further improve the consistency order, however at the sake of **larger mask size**.



## Discretization of the 2nd Derivative



As above, let  $f(i, j)$  denote the brightness at pixel  $(i, j)$ , and  $h_x$  the width of each pixel. Then we have:

$$f(i+1, j) = f(i, j) + h_x f_x(i, j) + \frac{h_x^2}{2} f_{xx}(i, j) + \frac{h_x^3}{3!} f_{xxx}(i, j) + O[h_x^4]$$

Similarly:

$$f(i-1, j) = f(i, j) - h_x f_x(i, j) + \frac{h_x^2}{2} f_{xx}(i, j) - \frac{h_x^3}{3!} f_{xxx}(i, j) + O[h_x^4]$$

Summing both equations leads to:

$$f_{xx}(i, j) = \frac{f(i+1, j) + f(i-1, j) - 2f(i, j)}{h_x^2} + O[h_x^2]$$

This discretization of the second derivative is of consistency order 2.

## Discretization of the Laplacian



|   |    |   |   |    |   |
|---|----|---|---|----|---|
| 0 | 1  | 0 | 1 | 1  | 1 |
| 1 | -4 | 1 | 1 | -8 | 1 |
| 0 | 1  | 0 | 1 | 1  | 1 |

Two masks showing discretizations of  $\Delta f = f_{xx} + f_{yy}$ .