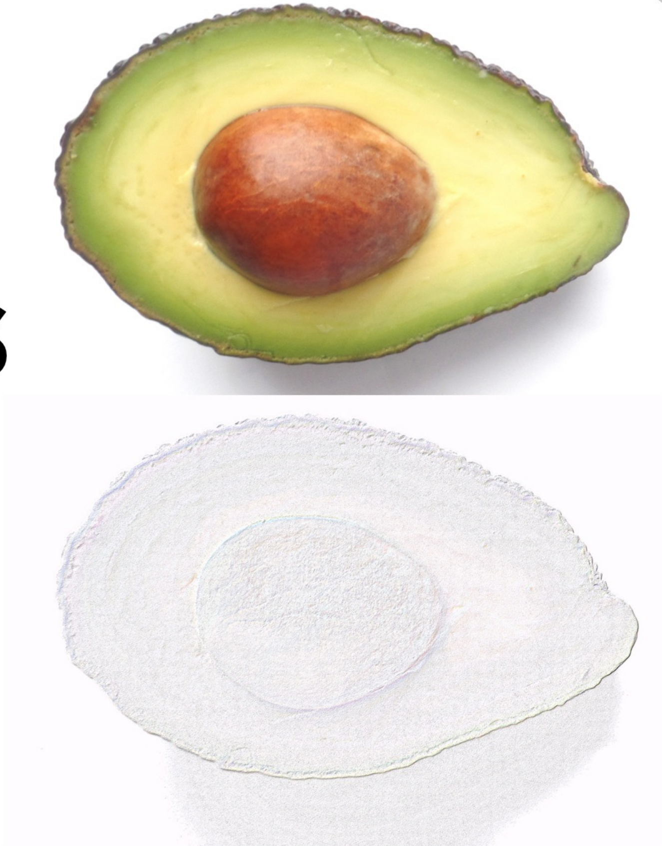
Operators Through Convolutions

#2 Gradient Filters

Lecture



Cyrill Stachniss

Summer term 2024 – Cyrill Stachniss

Photogrammetry & Robotics Lab

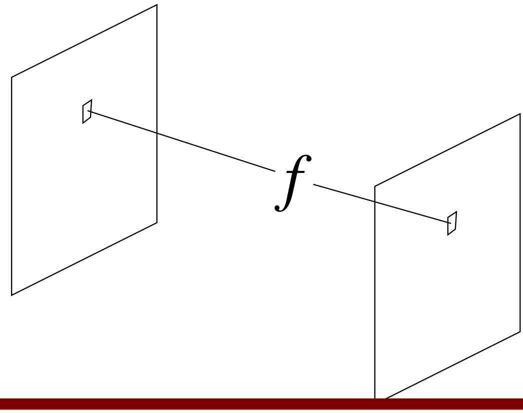
Local Operators Through Convolutions – Part 2 Gradient Filters

Cyrill Stachniss

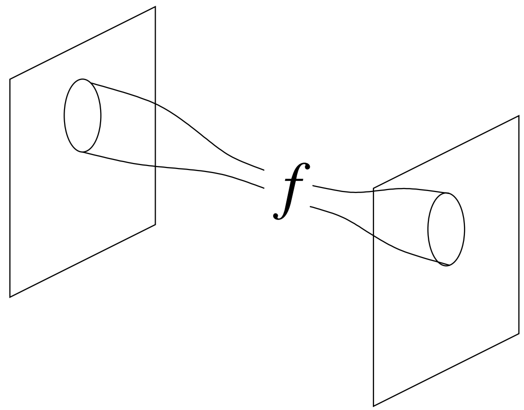
The slides have been created by Cyrill Stachniss.

Three Types of Operators

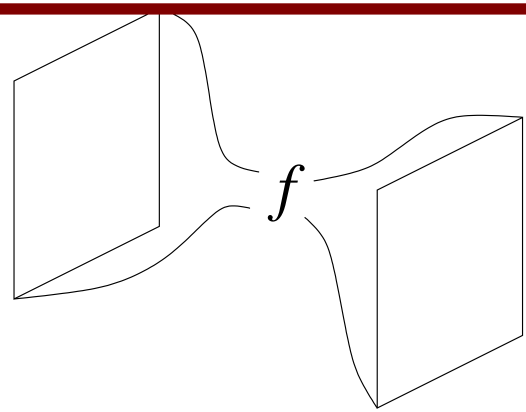
- Point operator



- Local operator



- Global operator



Gradient Filters

Images are Functions

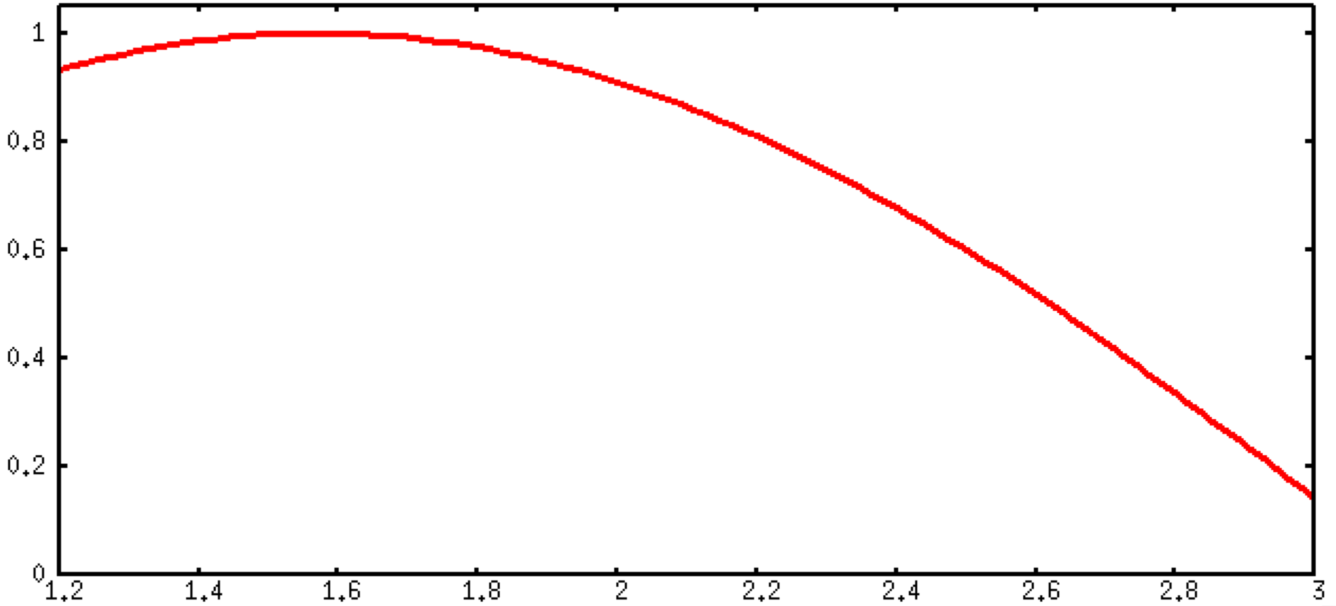
- An image is nothing else than a function $g(i,j) : \mathcal{B} \mapsto \mathcal{G}$
- with a 2-dimentional input in $\mathcal{B}$
- mapped to a 1-dimensional value in $\mathcal{G}$

Images are Functions

- An image is nothing else than a function $g(i, j) : \mathcal{B} \mapsto \mathcal{G}$
- with a 2-dimentional input in $\mathcal{B}$
- mapped to a 1-dimensional value in $\mathcal{G}$
- Maps 2D locations on the image plane to photon counts or intensities values
- Real world: $\mathcal{B} = \mathbb{R} \times \mathbb{R}, \mathcal{G} = \mathbb{N} \approx \mathbb{R}_{\geq 0}$
- Image domain: $\mathcal{B} = \mathbb{N} \times \mathbb{N}, \mathcal{G} = \mathbb{N}$
- Image files: $\mathcal{B} = \mathbb{N} \times \mathbb{N}, \mathcal{G} = [0..255]$

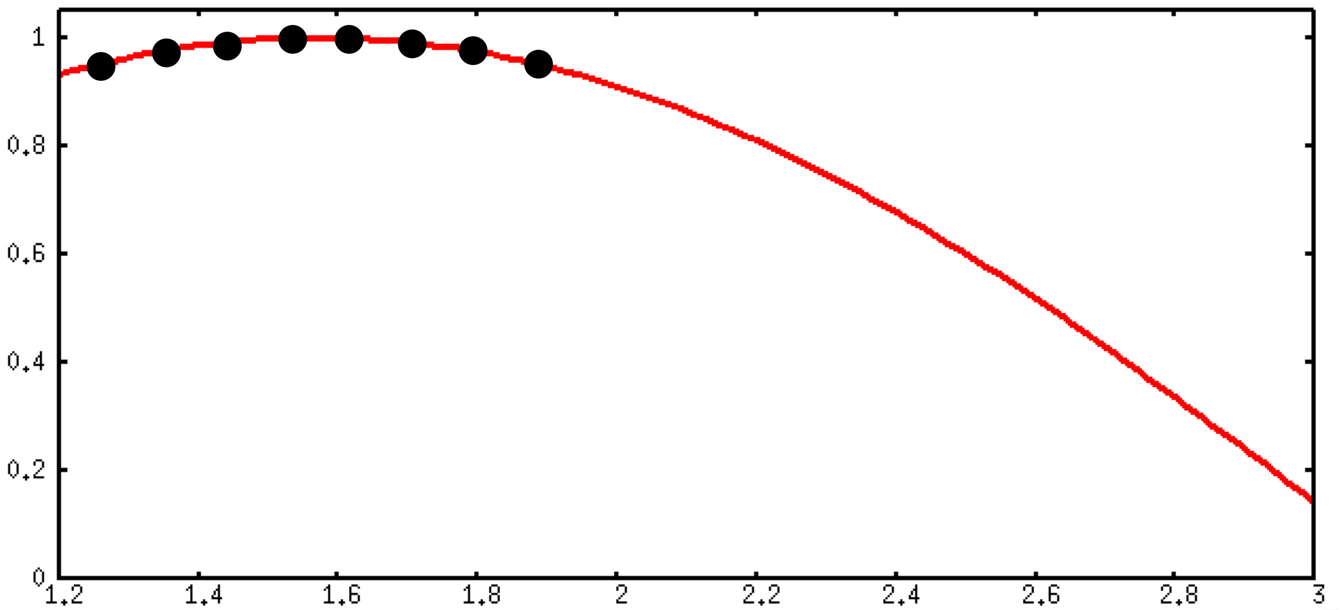
Gradient Filter

- Approximating the first derivative of a function



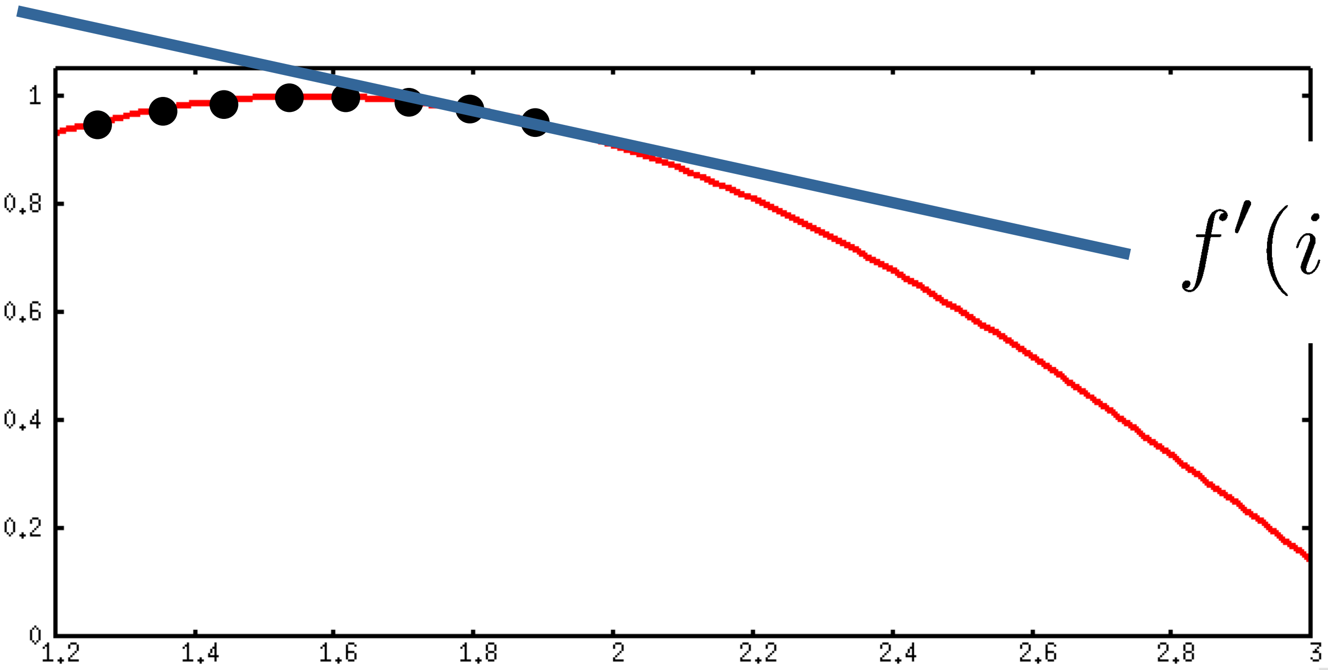
Gradient Filter

- Approximating the first derivative of a function sampled in discrete steps



Gradient Filter

- Approximating the first derivative of a function sampled in discrete steps



$$f'(i) \approx \frac{\Delta f}{\Delta x}$$

Gradient Filter

- First derivative (1-dim) is given by

$$f'(i) \approx \frac{\Delta f}{\Delta x} = \frac{f(i+1) - f(i)}{i+1-i}$$

Gradient Filter

- First derivative (1-dim) is given by

$$f'(i) \approx \frac{\Delta f}{\Delta x} = \frac{f(i+1) - f(i)}{i+1-i}$$

- Thus, $\Delta f(i) = f(i+1) - f(i)$

Gradient Filter

- First derivative (1-dim) is given by

$$f'(i) \approx \frac{\Delta f}{\Delta x} = \frac{f(i+1) - f(i)}{1}$$

- We can define the vector $\Delta = \left[\begin{array}{c} 1 \\ \underline{-1} \end{array} \right]$

- so that $f'(i) \approx \Delta * \boldsymbol{f}$

$$\begin{aligned} &= \sum_{k=-1}^{0} \Delta(k) f(i-k) \\ &= f(i+1) - f(i) \end{aligned}$$

Gradient Filter

- We could also smooth the function by considering the left and right point
- Then, the gradient turns into

$$f'(i) \approx \frac{\Delta f}{\Delta x} = \frac{f(i+1) - f(i-1)}{i+1-i+1}$$

$$= \frac{f(i+1) - f(i-1)}{2}$$

Gradient Filter

- We have $f'(i) \approx \frac{\Delta f}{\Delta x} = \frac{f(i+1) - f(i-1)}{2}$

Gradient Filter

- We have $f'(i) \approx \frac{\Delta f}{\Delta x} = \frac{f(i+1) - f(i-1)}{2}$
- Define analogously the weight vector

$$\Delta = \frac{1}{2} \begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix}$$

- such that

$$f'(i) \approx \Delta * \boldsymbol{f} = \sum_{k=-1}^{1} \Delta(k) f(i-k)$$

Gradient Filter

- The weight vector $\Delta = \frac{1}{2}\begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix}$

- is a smoothed variant of our original weight vector $\Delta = \begin{bmatrix} 1 \\ \underline{-1} \end{bmatrix}$

- This can be seen by

$$\frac{1}{2}\begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \underbrace{\boldsymbol{B}_1^{(1)}}_{\text{Binomial}} * \underbrace{\begin{bmatrix} 1 \\ -1 \end{bmatrix}}_{\text{gradient}}$$

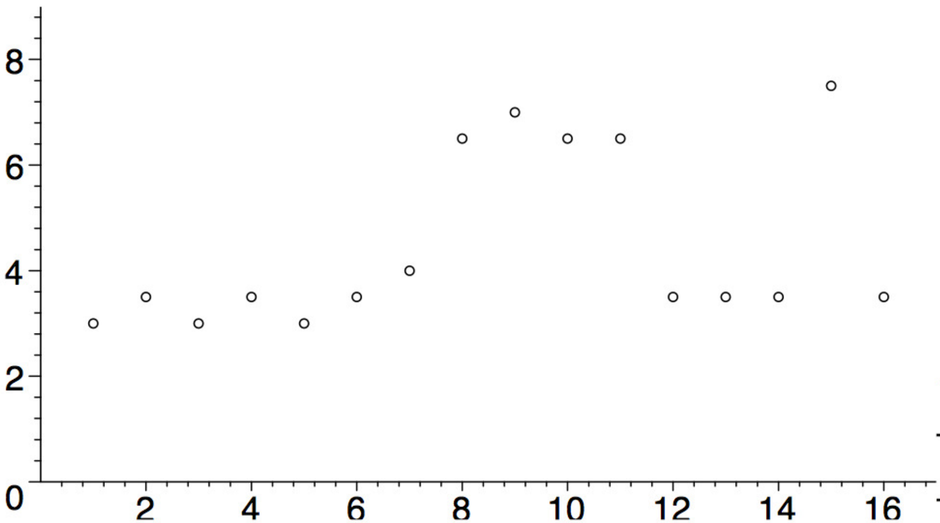
Gradient Filter

- We define the first derivative of the image function as

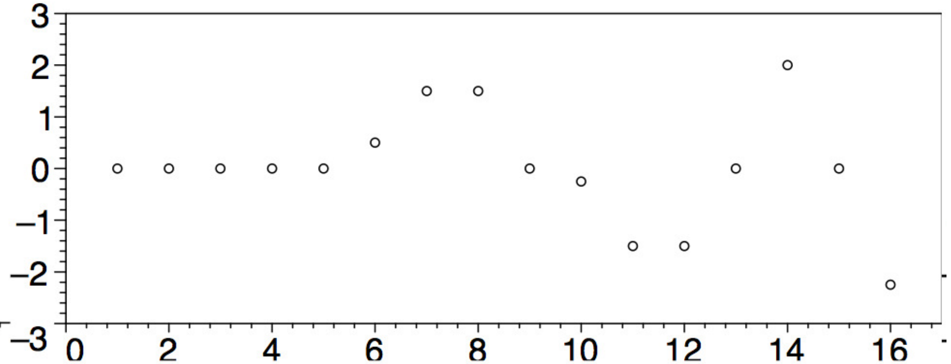
$$f' = \frac{\mathrm{d}f}{\mathrm{d}x} \approx \Delta * f = \frac{1}{2}\begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix} * f$$

- In contrast to smoothing kernels used before, the weight vector contains **negative** weights and sums up to 0
- First derivative of a constant signal equals to zero

Example



f



$\Delta * f$

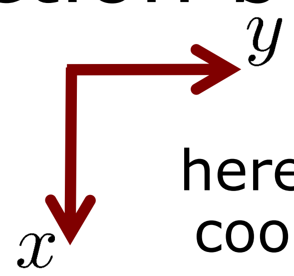
Gradient in Multiple Dimensions

- Gradient operator ∇ ("Nabla") is a vector consisting of the partial derivatives

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \approx \begin{bmatrix} \Delta \\ \Delta^{\mathsf{T}} \end{bmatrix}$$

- Thus, we can compute the 2D gradient images from the image function by

$$\nabla g = \nabla * g = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$



here: matrix coordinates

Gradient of the Image Function

- Gradient vector of the image function

$$\nabla g = \nabla * g = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

these are both 2D gradient images

- with the magnitude of the gradient

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

- and the direction

$$\alpha = \arctan\left(\frac{g_y}{g_x}\right) = \operatorname{atan2}(g_y, g_x)$$

Gradient Vector

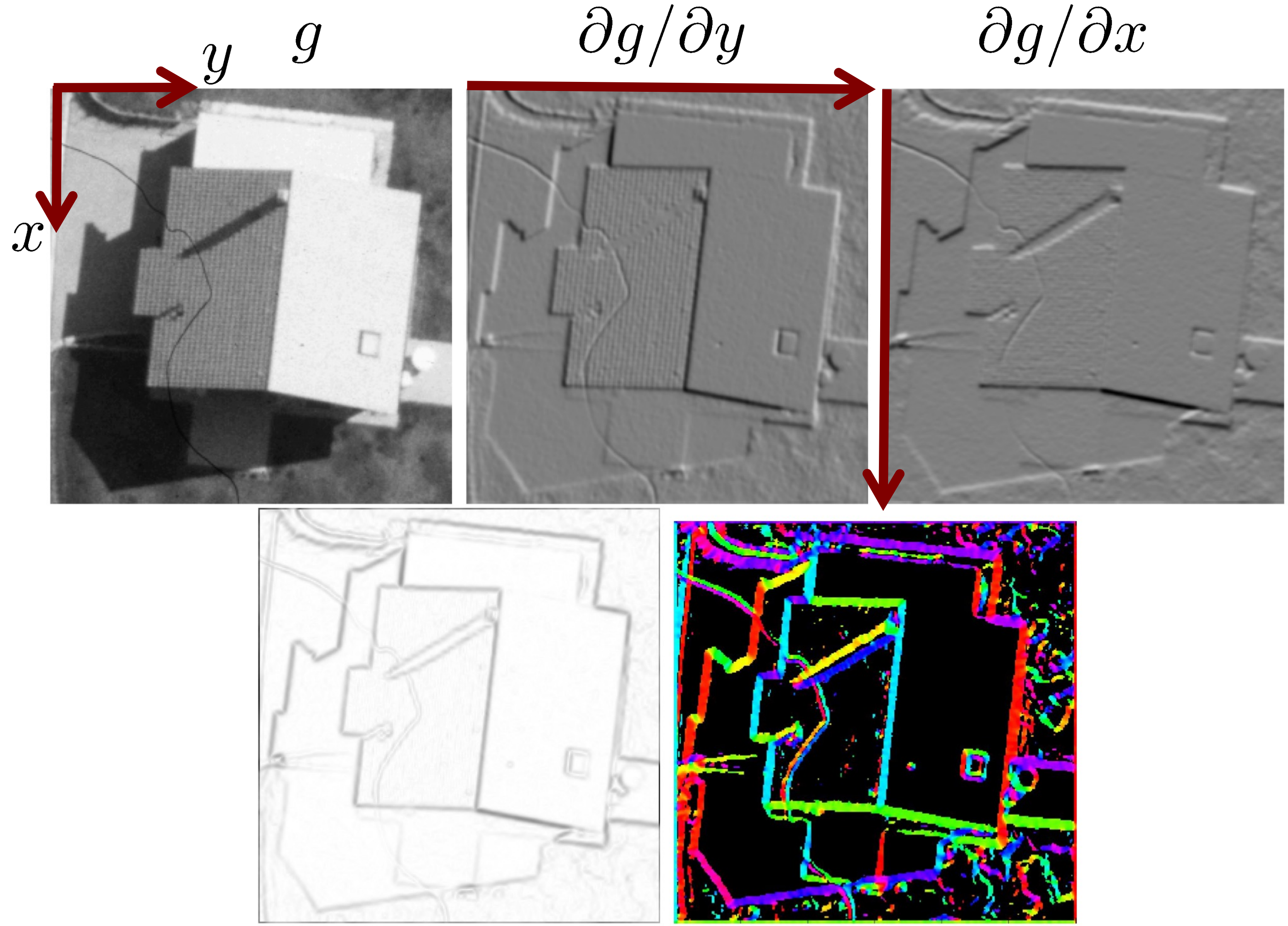
Thus, the 2D gradient vector of the image function can be written as

$$\nabla g = \nabla * g = \left[\begin{array}{c} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{array} \right] * g = \left[\begin{array}{c} g_x \\ g_y \end{array} \right] = \left| \nabla_g \right| \left[\begin{array}{c} \sin\alpha \\ \cos\alpha \end{array} \right]$$

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

$$\alpha = \operatorname{atan2}\left(g_y, g_x\right)$$

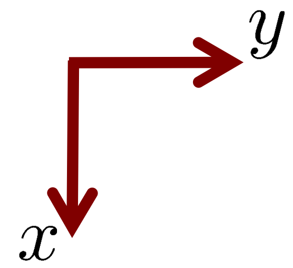
Example



$$|\nabla g| = \sqrt{g_x^2 + g_y^2} \qquad \alpha = \operatorname{atan2}\left(g_y, g_x\right)$$

Image courtesy: Förstner

Sobel Operator



- The Sobel operator is the standard operator for computing gradients using a 3x3 window
- It is a combination of a Binomial filter and the gradient

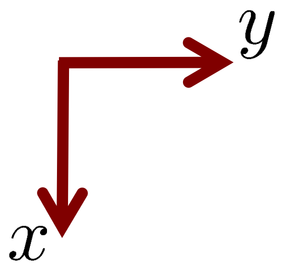
smoothing

gradient

$$\Delta_x = \left(\boldsymbol{B}_2^{(2)}\right)^{\mathsf{T}} * \Delta$$

$$= \frac{1}{4}[1\ \underline{2}\ 1] * \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & \underline{0} & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel Operator



- The Sobel operator for a 3x3 window

$$\Delta_x = \left(\boldsymbol{B}_2^{(2)}\right)^{\mathsf{T}} * \frac{1}{2}\begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & \underline{0} & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

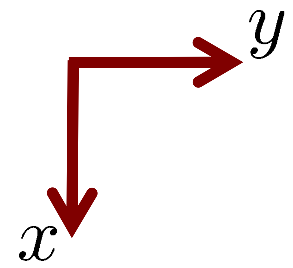
gradient

$$\Delta_y = \boldsymbol{B}_2^{(2)} * \frac{1}{2}[1 \;\; \underline{0} \;\; -1] = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & \underline{0} & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

gradient

Sobel-Based Edge Detection

Scharr Operator



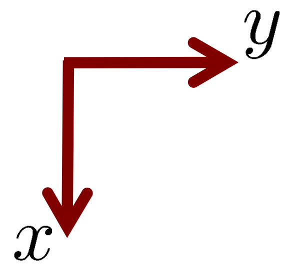
- Improved Sobel operator

$$\Delta_x = \frac{1}{4}[1\ \underline{2}\ 1] * \frac{1}{2}\begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix}$$

$$\Delta_x^{\mathrm{Scharr}} = \frac{1}{16}[3\ \underline{10}\ 3] * \frac{1}{2}\begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix}$$

- Uses a different smoothing kernel
- Better suited for computing the direction of the first derivative

Scharr Operator



- Improved Sobel operator

$$\Delta_x^{\text{Scharr}} = \frac{1}{32}\begin{bmatrix} 3 & 10 & 3 \\ 0 & \underline{0} & 0 \\ -3 & -10 & -3 \end{bmatrix} = \frac{1}{16}[3\ \underline{10}\ 3] * \frac{1}{2}\begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix}$$

$$\Delta_y^{\text{Scharr}} = \frac{1}{32}\begin{bmatrix} 3 & 0 & -3 \\ 10 & \underline{0} & -10 \\ 3 & 0 & -3 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 3 \\ \underline{10} \\ 3 \end{bmatrix} * \frac{1}{2}[1\ \ \underline{0}\ \ -1]$$

- 10-times **more accurate** than Sobel in determining the **gradient direction**

Scharr Operator

- Improved Sobel operator
- 10-times more accurate than Sobel (only for the direction)
- Errors stay typically below 0.5 deg

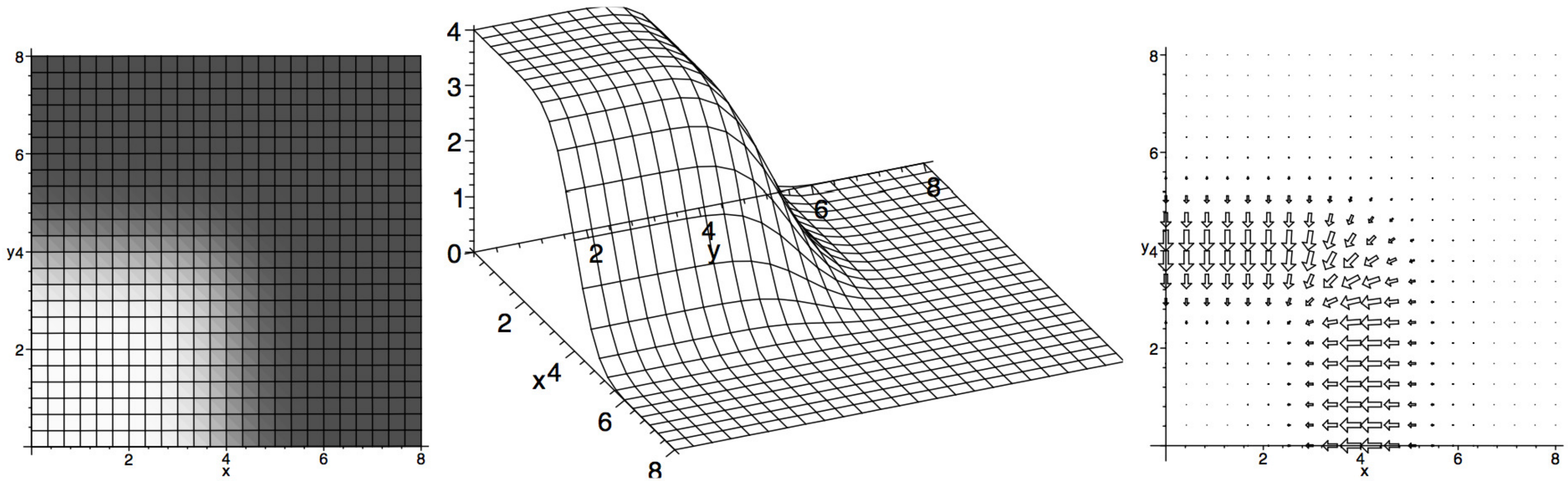


Image courtesy: Förstner

2^{nd} Derivatives

2nd Derivative – 1 Dimensional

- We can also express the second derivative of a function f

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial^2}{\partial x^2} * f = \left(\frac{\partial}{\partial x} * \frac{\partial}{\partial x}\right) * f$$

$$\begin{bmatrix} 1 \\ \underline{-1} \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ \underline{-1} \end{bmatrix} * \begin{bmatrix} 1 \\ \underline{-1} \end{bmatrix} = \begin{bmatrix} 1 \\ \underline{-2} \\ 1 \end{bmatrix}$$

2^{nd} Derivative – 1 Dimensional

- We can also express the second derivative of a function f

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial^2}{\partial x^2} * f = \left(\frac{\partial}{\partial x} * \frac{\partial}{\partial x}\right) * f$$

- with the kernel

$$\left[\begin{array}{c} 1 \\ \underline{-1} \end{array}\right]$$

$$\left[\begin{array}{c} 1 \\ \underline{-1} \end{array}\right] * \left[\begin{array}{c} 1 \\ \underline{-1} \end{array}\right] = \left[\begin{array}{c} 1 \\ \underline{-2} \\ 1 \end{array}\right]$$

$$\frac{\partial^2}{\partial x^2} = \left[\begin{array}{c} 1 \\ \underline{-2} \\ 1 \end{array}\right]$$

2^{nd} Derivative – 1 Dimensional

- The second derivative can again be computed via a single convolution
- Kernel

$$\frac{\partial^2}{\partial x^2} = \left[\begin{array}{c} 1 \\ -2 \\ \hline 1 \end{array} \right]$$

- Thus, the second derivative is given by

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial^2}{\partial x^2} * f = \left[\begin{array}{c} 1 \\ -2 \\ \hline 1 \end{array} \right] * f$$

2nd Derivative – 2 Dimensional

- The second derivative are given through the Hessian matrix

$$H(f) = [h(f)_{ij}] = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial y \partial x} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

- Consists of the individual partial derivatives

2nd Derivative Kernels in 2D

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

2nd derivative smoothing

$$\frac{\partial^2}{\partial x^2} = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ -2 & \underline{-4} & -2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \underline{-2} \\ 1 \end{bmatrix} * \frac{1}{4} [1 \ \underline{2} \ 1]$$

1st derivative 1st derivative

$$\frac{\partial^2}{\partial x \, \partial y} = \frac{1}{4} \begin{bmatrix} 1 & 0 & -1 \\ 0 & \underline{0} & 0 \\ -1 & 0 & 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 \\ \underline{0} \\ -1 \end{bmatrix} * \frac{1}{2} [1 \ \underline{0} \ -1]$$

smoothing 2nd derivative

$$\frac{\partial^2}{\partial y^2} = \frac{1}{4} \begin{bmatrix} 1 & -2 & 1 \\ 2 & \underline{-4} & 2 \\ 1 & -2 & 1 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 \\ \underline{2} \\ 1 \end{bmatrix} * [1 \ \underline{-2} \ 1]$$

Further Derivatives

- We can easily extend this concepts to higher-order derivatives
- Image processing often uses the first derivate, and sometime the second

Laplace Operator

The **Laplace operator** can be used for **edge detection** and is defined as

$$\Delta_L = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} = \frac{1}{2}\begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial^2}{\partial x^2} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ -2 & -4 & -2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{\partial^2}{\partial y^2} = \frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ 2 & -4 & 2 \\ 1 & -2 & 1 \end{bmatrix}$$

Laplace Operator

A smoother variant of the Laplace operator is

$$\Delta_L = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & \underline{-12} & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Laplace Operator Example

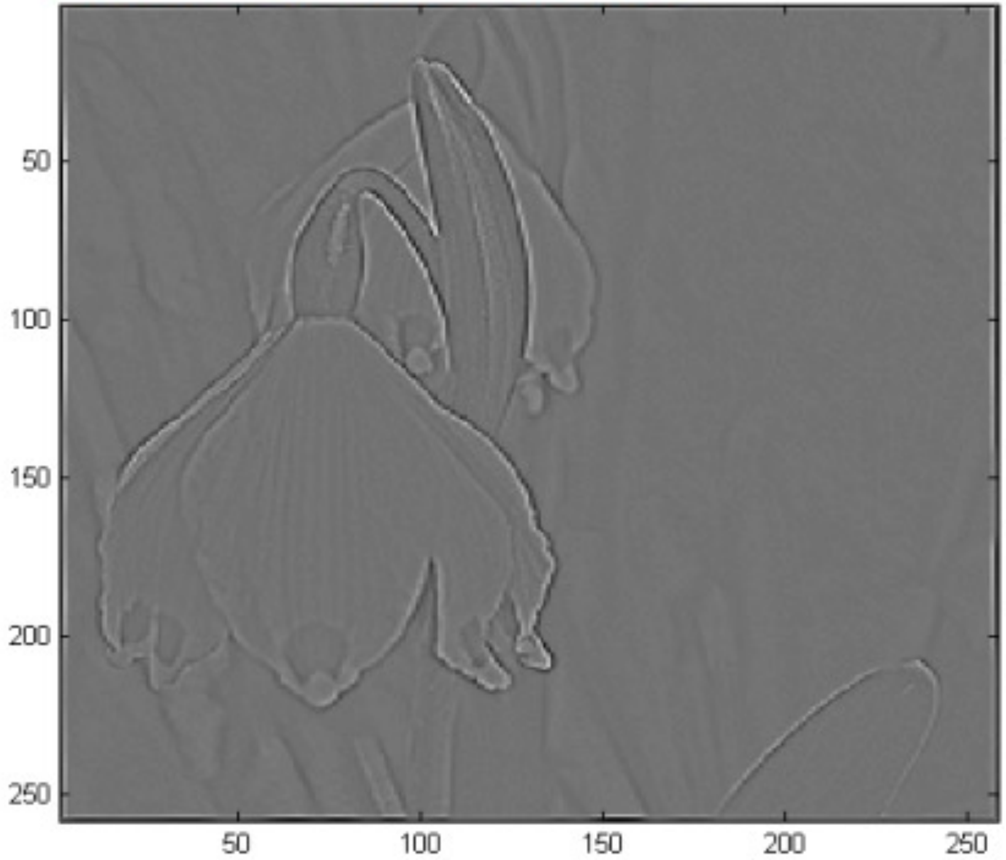


Image courtesy: J. Krieger 46

Summary

- Linear filters as local operators
- Convolution as a defining framework
- Introduction of important filters
- Part 1: Box filter & Binomial filter
- Part 2: Gradient filters, 1^{st} and 2^{nd} derivatives, Sobel and Scharr operator
- There are several other operators

Literature

- Szeliski, Computer Vision: Algorithms and Applications, Chapter 3
- Förstner, Scriptum Photogrammetrie I, Chapter “Lokale Operatoren”

Slide Information

- The slides have been created by Cyrill Stachniss as part of the photogrammetry and robotics courses.
- **I tried to acknowledge all people from whom I used images or videos. In case I made a mistake or missed someone, please let me know.**
- The photogrammetry material heavily relies on the very well written lecture notes by Wolfgang Förstner and the Photogrammetric Computer Vision book by Förstner & Wrobel.
- Parts of the robotics material stems from the great Probabilistic Robotics book by Thrun, Burgard and Fox.
- If you are a university lecturer, feel free to use the course material. If you adapt the course material, please make sure that you keep the acknowledgements to others and please acknowledge me as well. To satisfy my own curiosity, please send me email notice if you use my slides.

Cyrill Stachniss, cyrill.stachniss@igg.uni-bonn.de