Photogrammetry & Robotics Lab

Local Operators Through Convolutions – Part 1 Smoothing Filters

Cyrill Stachniss

The slides have been created by Cyrill Stachniss.

Three Types of Operators









Point Operators

- Cannot deal well with noise
- Cannot deal well with local structures ("one intensity value is not enough")





Today: Local Operators

- Local operators are also called neighborhood operators
- Convolutions as a framework for specifying such local operators
- We will look in several local operators
 - Noise reduction
 - Binomial filter
 - Gradients

• ..

Box Filter

"Replace an intensity value by the mean intensity value of the neighborhood."

$$g(i) = \frac{1}{K} \sum_{k} f(i-k)$$

$$g(i,j) = \frac{1}{KL} \sum_{k,l} f(i-k,j-l)$$

DE: "Rechteckfilter (oder gleitende Mittelbildung)"

"Replace an intensity value by the mean intensity value of the neighborhood."



"Replace an intensity value by the mean intensity value of the neighborhood."



$$g(i,j) = \frac{1}{9} \sum_{k=\{-1,0,1\}} \sum_{l=\{-1,0,1\}} f(i-k,j-l)$$

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$$g(i,j) = \frac{1}{9} \sum_{k=\{-1,0,1\}} \sum_{l=\{-1,0,1\}} f(i-k,j-l)$$

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 0 | 10 | | | | |
|---|----|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

$$g(i,j) = \frac{1}{9} \sum_{k=\{-1,0,1\}} \sum_{l=\{-1,0,1\}} f(i-k,j-l)$$

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 0 | 10 | 20 | | | |
|---|----|----|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

$$g(i,j) = \frac{1}{9} \sum_{k=\{-1,0,1\}} \sum_{l=\{-1,0,1\}} f(i-k,j-l)$$

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 0 | 10 | 20 | 30 | | | |
|---|----|----|----|--|--|--|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

$$g(i,j) = \frac{1}{9} \sum_{k=\{-1,0,1\}} \sum_{l=\{-1,0,1\}} f(i-k,j-l)$$

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 0 | 10 | 20 | 30 | 30 | 30 | 20 | 10 | |
|----|----|----|----|----|----|----|----|--|
| 0 | 20 | 40 | 60 | 60 | 60 | 40 | 20 | |
| 0 | 30 | 60 | 90 | 90 | 90 | 60 | 30 | |
| 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| 0 | 20 | 30 | 50 | 50 | 60 | 40 | 20 | |
| 10 | 20 | 30 | 30 | 30 | 30 | 20 | 10 | |
| 10 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | |
| | | | | | | | | |

Kernel

 We can formulate the box filter by using a weighting function w

$$g(i,j) = \sum_{k,l} w(k,l) f(i-k,j-l)$$

- This weighting function is called kernel (or kernel function)
- Often, these filtering operators involve weighted combinations of intensity values in a neighborhood

Linear Shift Invariant Filters

A filter L that transforms

g(i,j) = L(f(i,j))

is called linear and shift invariant if $L(\alpha_1 f_1 + \alpha_2 f_2) = \alpha_1 g_1 + \alpha_2 g_2$ and

$$L(f(i-k, j-l)) = g(i-k, j-l)$$

Convolution (DE: Faltung)

Filters of the form

$$g(i,j) = \sum_{k,l} w(k,l) f(i-k,j-l)$$

- are convolutions of the function f and a kernel function $\ w$

$$g = w * f$$

Box Filter as a Convolution

The box filter (R) is a convolution

 $g = {\pmb{R}}_{3}^{(2)} { \overleftarrow{} \hspace{-.5em} } {}^{\text{dimensionality}}_{\text{neighborhood}}$

of the image function f and a box kernel



$$g(i) = \sum_{k=-1}^{k=1} w(k) f(i-k)$$

$$g(i) = \sum_{k=-1}^{k=1} w(k) f(i-k)$$

= $w(-1)f(i-(-1)) + w(0)f(i-0) + w(1)f(i-1)$

$$g(i) = \sum_{k=-1}^{k=1} w(k) f(i-k)$$

= $w(-1)f(i-(-1)) + w(0)f(i-0) + w(1)f(i-1)$
= $\frac{1}{3}f(i+1) + \frac{1}{3}f(i) + \frac{1}{3}f(i-1)$

we set
$$w(i) = \frac{1}{3}$$

$$g(i) = \sum_{k=-1}^{k=1} w(k) f(i-k)$$

= $w(-1)f(i-(-1)) + w(0)f(i-0) + w(1)f(i-1)$
= $\frac{1}{3}f(i+1) + \frac{1}{3}f(i) + \frac{1}{3}f(i-1)$
= $\frac{1}{3}(f(i+1) + f(i) + f(i-1))$

The box filter takes the average of the neighborhood pixels (here 3) and can be expressed as a convolution

Box Filter Example Using Different Neighborhoods

Box Kernel

 The sum of all weights of the kernel yields 1. As a result, the mean of the image function does not change

$$\boldsymbol{R}_{3}^{(2)} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

 The underlined number is the reference pixel with index (0,0)

Noise Reduction

- Convolutions with such a kernel changes the noise of the input signal
- For the box filter, we have

$$\sigma_{n_g}\left(\boldsymbol{R}_m^{(1)}\right) = \frac{1}{\sqrt{m}}\sigma_{n_f} \qquad \sigma_{n_g}\left(\boldsymbol{R}_m^{(2)}\right) = \frac{1}{m}\sigma_{n_f}$$

- The box filter reduces the noise
- In general, we have

$$\sigma_{n_g}^2 = \sum_i (w(i))^2 \ \sigma_{n_f}^2$$

Median Filter

- As an alternative to the box filter, we can compute the median within the neighborhood
- Robust against outliers
- Not a linear filter anymore

How to Deal with the Borders?

- A pixel at the border of an image has no fully defined neighborhood
- Padding options
 - constant value: for all outside pixels
 - cyclic wrap: loop "around" the image
 - clamp: repeat edge pixels indefinitely
 - mirror: reflect pixels across the image edge

Padding Options

- constant zero: 0 for all outside pixels
- cyclic wrap: loop "around" the image
- clamp: repeat edge pixels indefinitely
- mirror: reflect pixels across the image edge

Image courtesy: Szelinsky 27

Binomial Filter

DE: "Binomial-Filter"

Binomial Filter

- Performs a smoothing
- Smoothing using an approximation of a Gaussian as the kernel function
- Discrete approximation due to pixels
- The decay of the weights approximates a Gaussian using the coefficients of a Binomial distribution B(0.5, n)
- Elements of the Pascal's triangle

Binomial Kernel in 1D

$$\boldsymbol{B}_{2}^{(1)} = \frac{1}{4} \begin{bmatrix} 1\\ 2\\ 1 \end{bmatrix}$$
$$\boldsymbol{B}_{4}^{(1)} = \frac{1}{16} \begin{bmatrix} 1\\ 4\\ \frac{6}{4}\\ 1 \end{bmatrix}$$

Connection to Pascal's Triangle

Binomial Kernel in 1D and 2D

$$\boldsymbol{B}_{2}^{(1)} = \frac{1}{4} \begin{bmatrix} 1\\ \frac{2}{1} \end{bmatrix} \qquad \boldsymbol{B}_{2}^{(2)} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1 \end{bmatrix}$$
$$\boldsymbol{B}_{4}^{(1)} = \frac{1}{16} \begin{bmatrix} 1\\ 4\\ \frac{6}{4}\\ 1 \end{bmatrix} \qquad \boldsymbol{B}_{4}^{(2)} = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

33

Noise Reduction

 For the Binomial filter, we obtain for the input-output noise relation

$$\sigma_{n_g} \left(\boldsymbol{B}_m^{(1)} \right) = \frac{1}{\sqrt[4]{\pi m}} \sigma_{n_f} \quad \sigma_{n_g} \left(\boldsymbol{B}_m^{(2)} \right) = \frac{1}{\sqrt{\pi m}} \sigma_{n_f}$$

- Similar to the box filter, also the Binomial filter reduces the noise
- Less aggressive smoothing than the box filter (for the same neighborhood)

Degree of Smoothing • **Degree of smoothing** $r = \frac{\sigma_{n_g}}{\sigma_{n_f}}$

- We obtain for the kernel size $m_R(r) = \frac{1}{r}$ $m_B(r) = \frac{1}{\pi r^2}$
- Example for r=1/5: $m_R(0.2) = 5$ $m_B(0.2) = \frac{25}{\pi} \approx 8$
- Box filter offers a stronger smoothing than the Binomial filter

Convolution

DE: "Faltung"

Convolution

- We have seen that linear filters can be expressed as convolutions
- Therefore, we now analyze the properties of the convolution

$$g = w * f$$

output kernel input image image

Definition

The discrete convolution of the functions a(i) and b(i) is defined as

$$c(i) = \sum_{k=-\infty}^{+\infty} a(k) b(i-k)$$

and in 2D as

$$c(i,j) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} a(k,l) b(i-k,j-l)$$

• and is written as c = a * b

Commutative Property

The convolution is commutative

$$a * b = b * a$$

because

$$c(i) = \sum_{k=-\infty}^{+\infty} a(k) \ b(i-k)$$

• equals – after replacement j = i - kk = i - j

$$c(i) = \sum_{j=-\infty}^{+\infty} a(i-j) b(j)$$

Further Properties

Associative property

$$a * b * c = (a * b) * c = a * (b * c)$$

Distributive property

$$(a + b) * c = a * c + b * c$$

Scalar multiplication

$$\lambda(\boldsymbol{a} \ast \boldsymbol{b}) = (\lambda \boldsymbol{a}) \ast \boldsymbol{b} = \boldsymbol{a} \ast (\lambda \boldsymbol{b})$$

Neutral Element / Unit Impulse

Unit impulse

$$\boldsymbol{\delta} = \begin{bmatrix} \vdots \\ 0 \\ \frac{1}{0} \\ \vdots \end{bmatrix} \quad \boldsymbol{\delta} = \begin{bmatrix} \cdots & \vdots & \cdots \\ \cdots & \frac{1}{2} & \cdots \\ \cdots & \vdots & \cdots \end{bmatrix}$$

• The convolution with δ yields the input function, i.e., $a * \delta = a$

Translation/Shift Through Convolution

• A convolution with δ can be used to shift the function f by (x, y)

$$f(i-x, j-y) = f(i, j) * \delta(i-x, j-y)$$

Multiplication and Convolution

- Multiplication and convolution can be seen as similar
- Difference: Multiplication uses carry (DE: Übertrag)
- Example:

corresponds to extended signal

De-Convolution

• If the inverse $a^{-1}(i)$ of a(i) exists and $\sum_{i} a(i) \neq 0$

we can de-convolute a function

• Given c(x) = a(x) * b(x)we can recover b(i) given $a^{-1}(i)$ by $b(x) = a^{-1}(x) * c(x) = c(x) * a^{-1}(x)$

Separable Kernels

- A multi-dimensional kernel that can be split into the individual dimensions, is called separable.
- Example:

$$\boldsymbol{B}_{2}^{(2)} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \boldsymbol{B}_{2}^{(1)} * (\boldsymbol{B}_{2}^{(1)})^{\mathsf{T}}$$

Separable Kernels Allow for Efficient Computations

Two 1D convolutions are more efficient to compute than one 2D convolution

$$\boldsymbol{g} = \boldsymbol{R}_n^{(2)} * \boldsymbol{f} = \boldsymbol{R}_n^{(1)} * \left((\boldsymbol{R}_n^{(1)})^\mathsf{T} * \boldsymbol{f} \right)$$

 $\mathcal{O}(n^2)$ operations $\mathcal{O}(2n) = \mathcal{O}(n)$ operations

Multiple Convolutions

Smoothing filters have the property

 $\sum_{i} w(i) = 1$

 The concatenation of smoothing filters is again a smoothing filter

 $w = w_1 * \ldots * w_n$

For the Binomial filter holds:

$$B_n = \underbrace{B_1 * \ldots * B_1}_{n \text{ times}}$$

Integral Image

An integrate image is an image in which each pixel stores a sum of intensity values of the form

$$s(i,j) = \sum_{i'=1}^{i} \sum_{j'=1}^{j} f(i',j')$$

The integral image can used to efficiently execute a box filter

Integral Image

 Allow for effective computing the sum over intensities in any rectangle

 $S([i_0, i_1] \times [j_0, j_1]) = s(i_1, j_1) + s(i_0 - 1, j_0 - 1)$ $-s(i_0 - 1, j_1) - s(i_1, j_0 - 1)$

| 3 | 2 | 7 | 2 | 3 |
|---|---|---|---|---|
| 1 | 5 | 1 | 3 | 4 |
| 5 | 1 | 3 | 5 | 1 |
| 4 | 3 | 2 | 1 | 6 |
| 2 | 4 | 1 | 4 | 8 |

| 3 | 5 | 12 | 14 | 17 |
|----|----|----|----|----|
| 4 | 11 | 19 | 24 | 31 |
| 9 | 17 | 28 | 38 | 46 |
| 13 | 24 | 37 | 48 | 62 |
| 15 | 30 | 44 | 59 | 81 |

$$S = 48 + 3 - 14 - 13 = 24$$

Summary

- Linear filters as local operators
- Convolution as a framework
- Introduction of important filters
- Box filter
- Binomial/Gaussian filter
- 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