



Marburg  
University

# Lecture Computer Vision Chapter 2 – Linear Filters

**Prof. Dr. Ralph Ewerth**

Research Group AI – Multimodal Modelling and Machine Learning

Department of Mathematics and CHessian.AI

# Literature for this Lecture

- W. Burger, M. J. Burge:  
„Digital Image Processing –  
An Algorithmic Introduction Using Java“  
Springer-Verlag, Second Edition, 2016  
Chapter 5 & Chapter 6

# Image Representation - Introduction

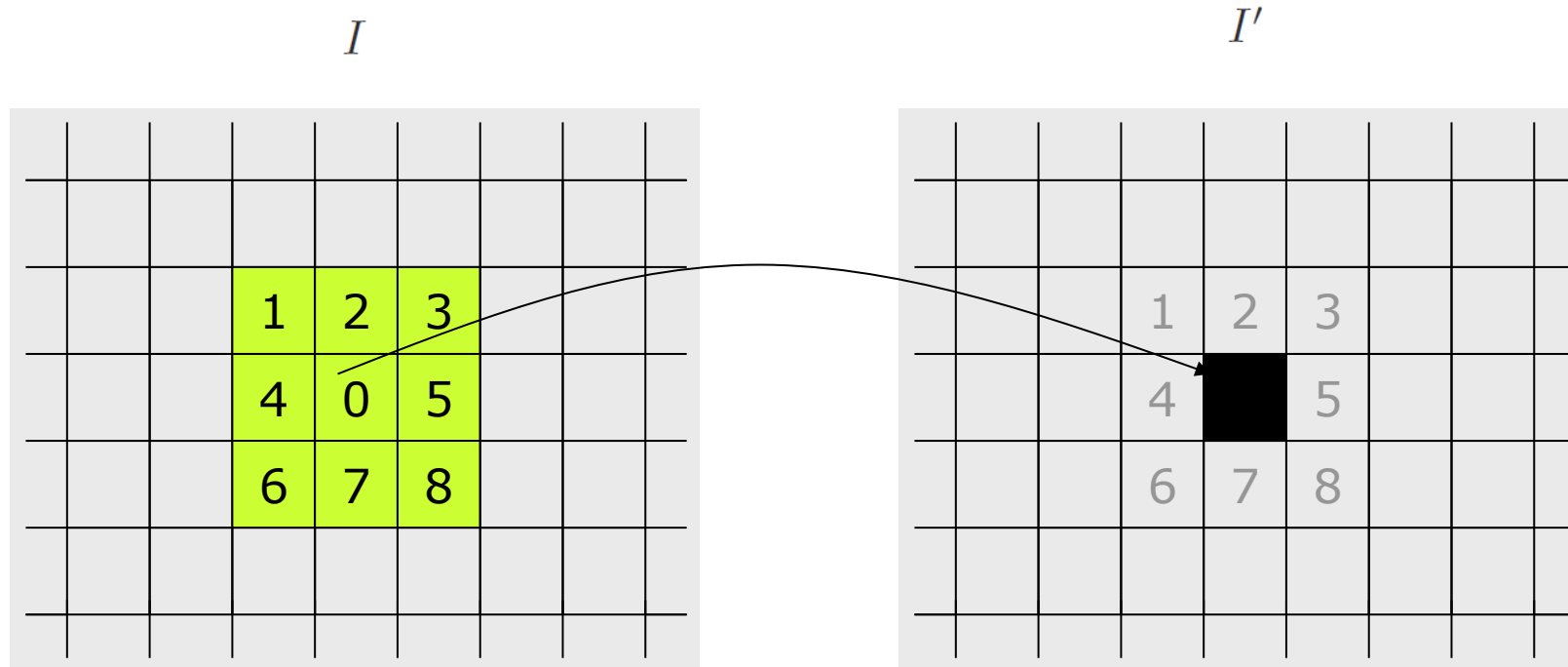
- Representation at pixel level does not represent well visual content
- Task: Find appropriate representations for
  - color
  - shape
  - texture
  - objects
  - Regions
- Today we talk about linear/non-linear filtering methods, useful for describing shape or texture (other lecture)
  - Linear filter for blurring/smoothing images
  - Linear filters for blurring/smoothing images

## Example Filter: Smooth / Blur



- Can not be realized via point operators
- Brightness, contrast do not change
- Geometry of image remains unchanged

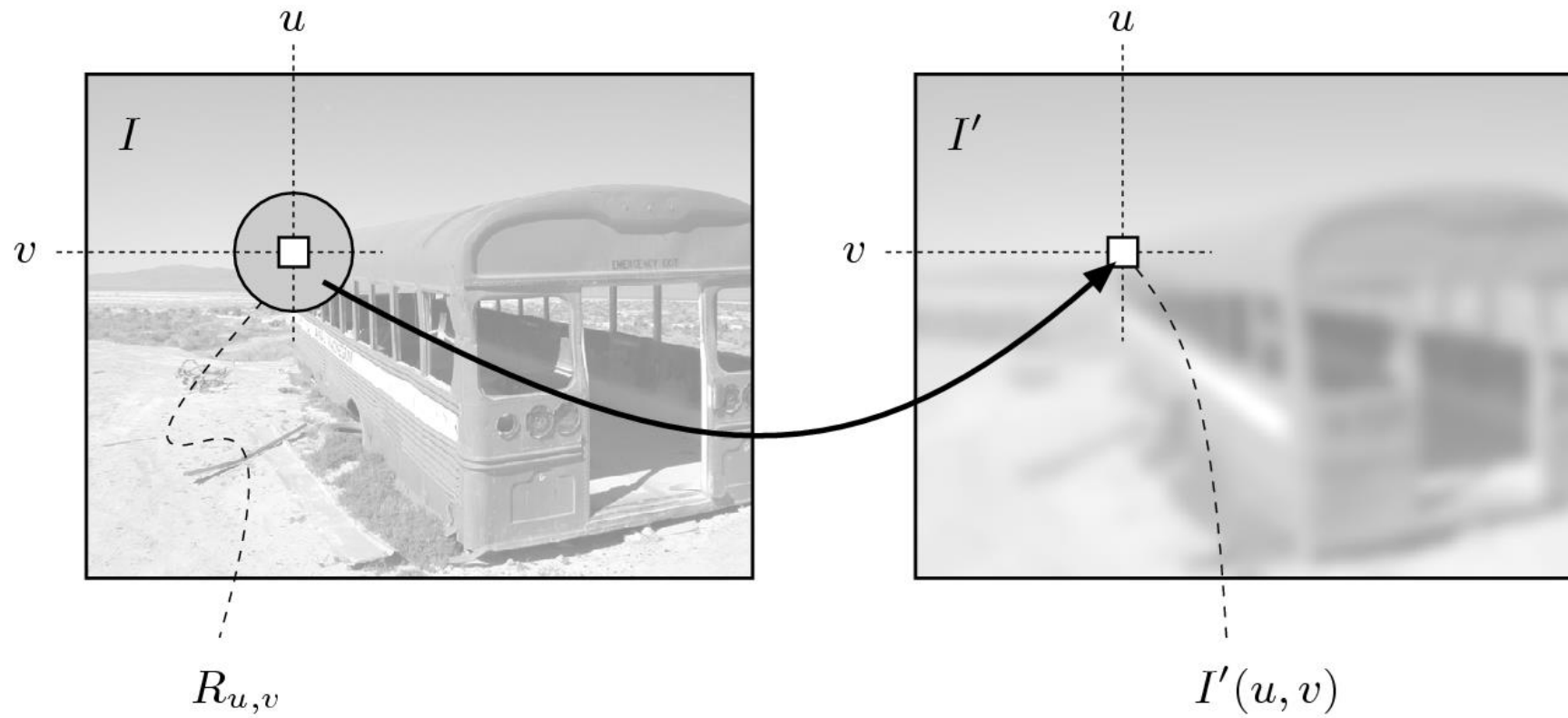
# Simple Smoothing Filter by Averaging



Neighbor pixels  $p_0, \dots, p_8$

$$I'(u, v) \leftarrow \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9}$$

# Principle of Filtering

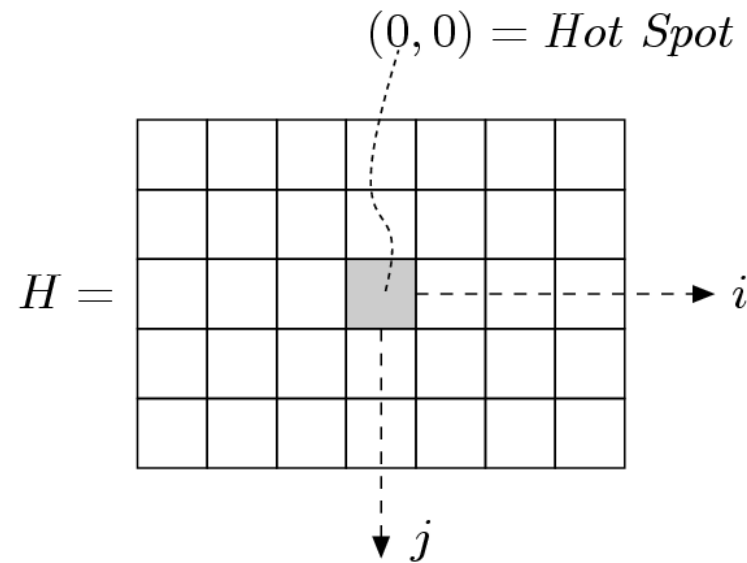


# Filter in 2D

1. Size of region  $R$ ?
2. Same weight for all pixels?
3. Can we do more than smoothing?
4. How can we describe filter operations systematically?

# Linear Filter – Filter Matrix

- New pixel value is **weighted sum** (linear combination) of pixel values
- Weights = **filter coefficients**
- Simple formal properties (spatial/frequency space)



- **Filter matrix** = 2D function of weights (coefficients), determined by region  $R$
- **Filter function is infinite**, all values outside matrix are zero
- „**Hot Spot**“ = Coordinate origin of the filter function (typically centered)

## Local average – revisited

$$I'(u, v) \leftarrow \frac{1}{9} \left[ \begin{array}{l} I(u-1, v-1) + I(u, v-1) + I(u+1, v-1) + \\ I(u-1, v) + I(u, v) + I(u+1, v) + \\ I(u-1, v+1) + I(u, v+1) + I(u+1, v+1) \end{array} \right]$$

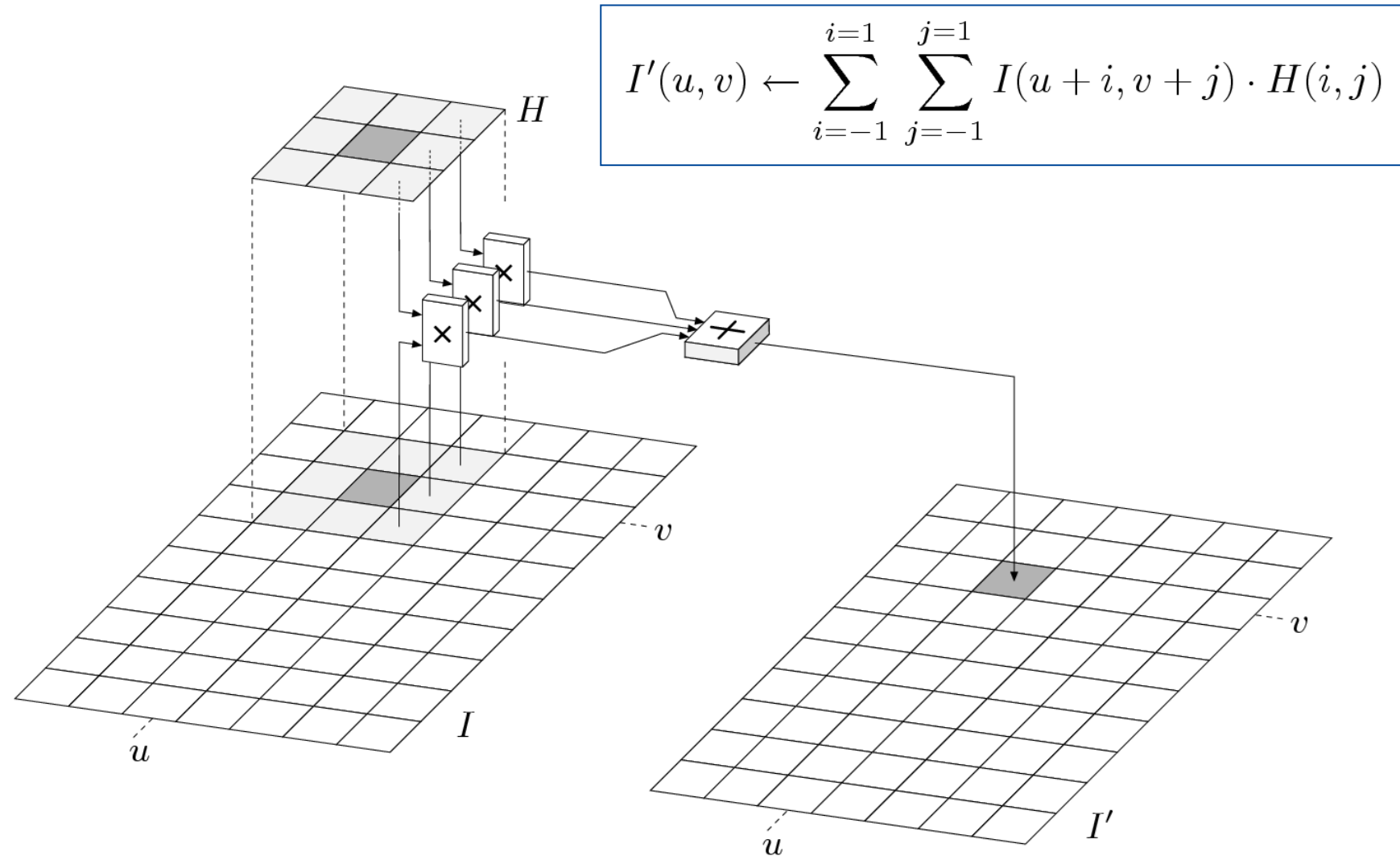
Associated filter function (filter matrix):

$$H(i, j) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

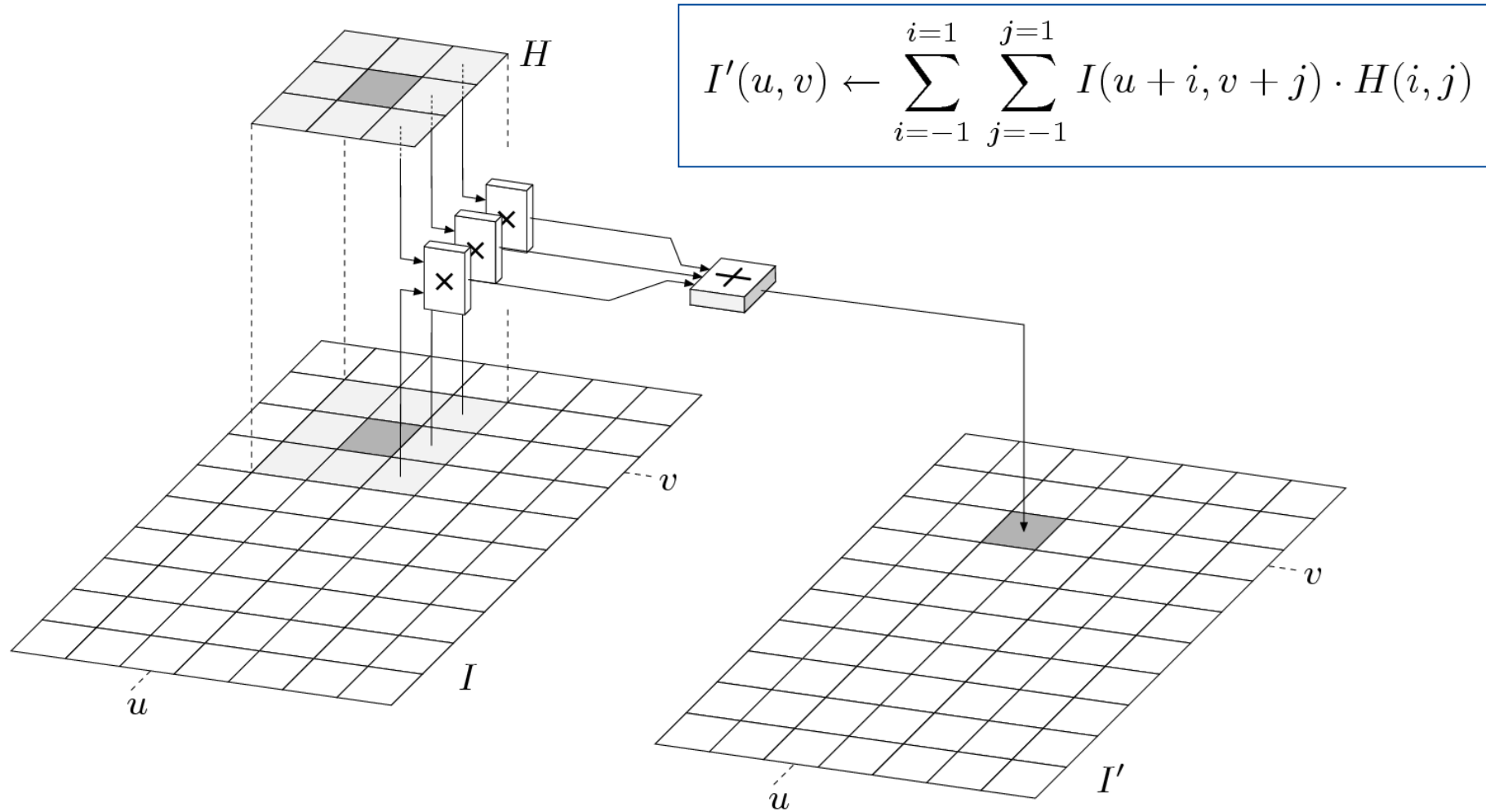
Filter operation:

$$I'(u, v) \leftarrow \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u+i, v+j) \cdot H(i, j) \quad \text{for each pixel } (u, v)$$

# Linear 3x3 Filter



# General, Linear 3x3 Filter



# Another Blur Filter

**Gaussian blur filter:** The weighting of the pixels depends on the distance to the hot spot.

$$H(i, j) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & \underline{0.200} & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix}$$

Radial Gaussian function

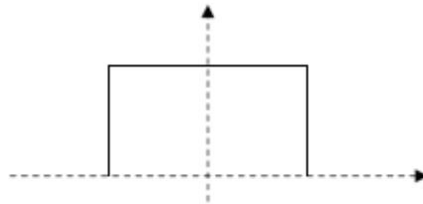
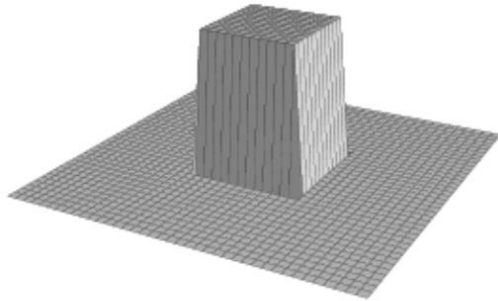
$$G_{\sigma}(r) = e^{-\frac{r^2}{2\sigma^2}}$$

- Filter coefficients do not have to be integers!
- With smoothing filters, usually the **sum of the coefficients is normalized to 1**.
- Efficient calculation with integer coefficients and a shared scaling factor, e.g.:

$$H(i, j) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & \underline{0.200} & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix} = \frac{1}{40} \begin{bmatrix} 3 & 5 & 3 \\ 5 & \underline{8} & 5 \\ 3 & 5 & 3 \end{bmatrix}$$

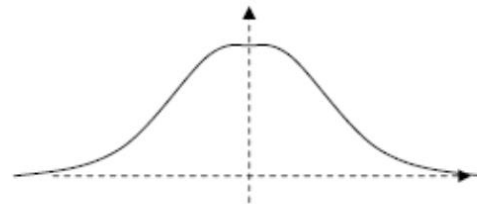
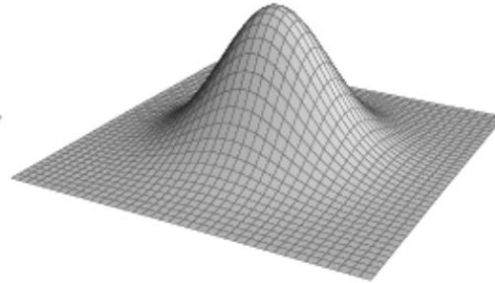
# Examples for Linear 2D Filters

„Box“ Filter



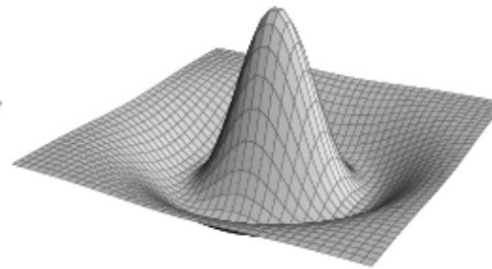
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Gaussian Filter



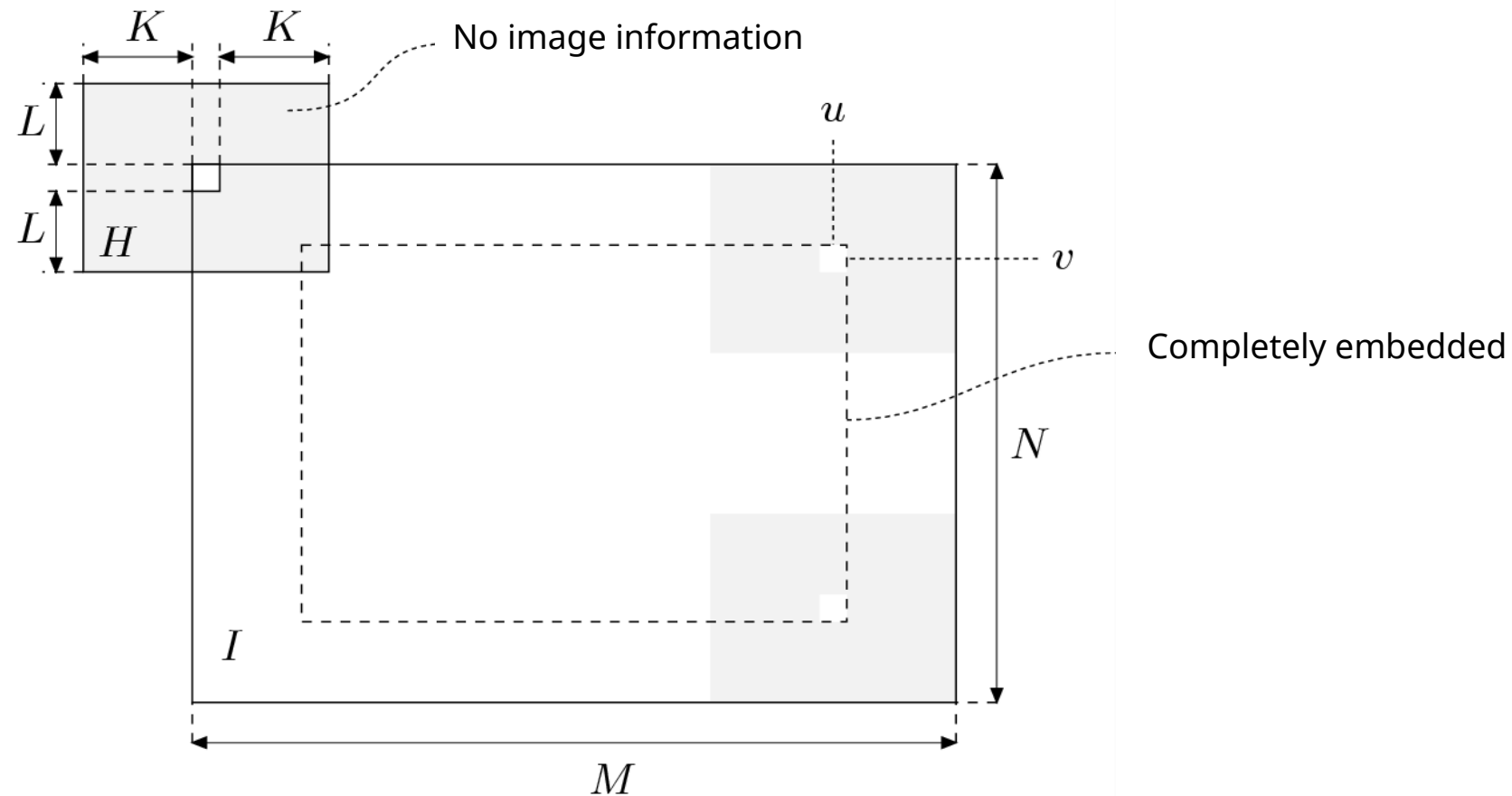
0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

Laplace (Mexican Hat)



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

# Calculation of Filters – Boundary Problem



# Boundary Problem – Possible Solutions

- Do not compute boundary pixels in new image (e.g., set to black)
  - Keep boundary pixels unchanged (without filtering)
- 
- Filter until boundary and process the pixels outside the image area by...

...assuming constant value (e.g. 128)



(a)

...filling with neighboring pixel's value



(b)

...assuming a cyclic image (in both dimensions)



(c)

# Filter calculation – value ranges

- What values can occur (max/min)?

Example: 8-bit grayscale image (0...255), no clamping!

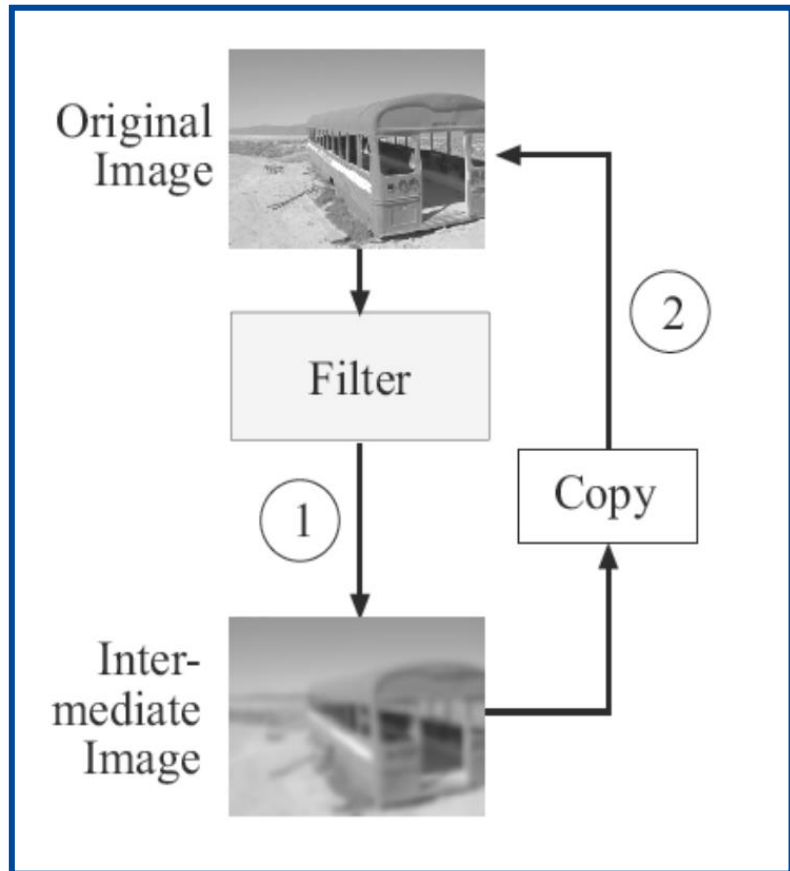
$$H(i, j) = \begin{bmatrix} -1 & -2 & 0 \\ -2 & 0 & 2 \\ 0 & 2 & 1 \end{bmatrix} \quad \begin{array}{l} \text{min} = \underline{\hspace{2cm}} \text{ ?} \\ \text{max} = \underline{\hspace{2cm}} \text{ ?} \end{array}$$

- What to do with results that are too large (truncate?)
- Filter coefficients can be negative!
- What to do with negative results?
- Data format for intermediate results (floating point number, integer values)?

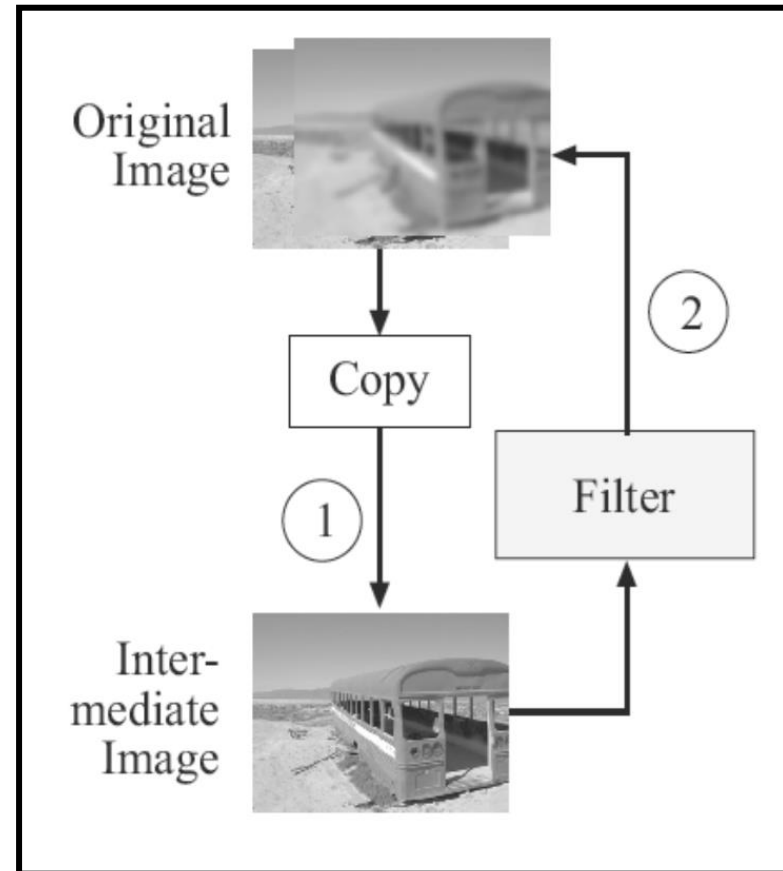
# Practical Aspects

- „In-Place“ computation not possible!
- Additional image copy is necessary

Variante A



Variante B



# Linear Filters and Convolution

- All linear filters are **fully specified** by their associated filter matrix ***H***!
- In general, linear filters correspond to a linear convolution operation

$$I'(u, v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u-i, v-j) \cdot H(i, j)$$

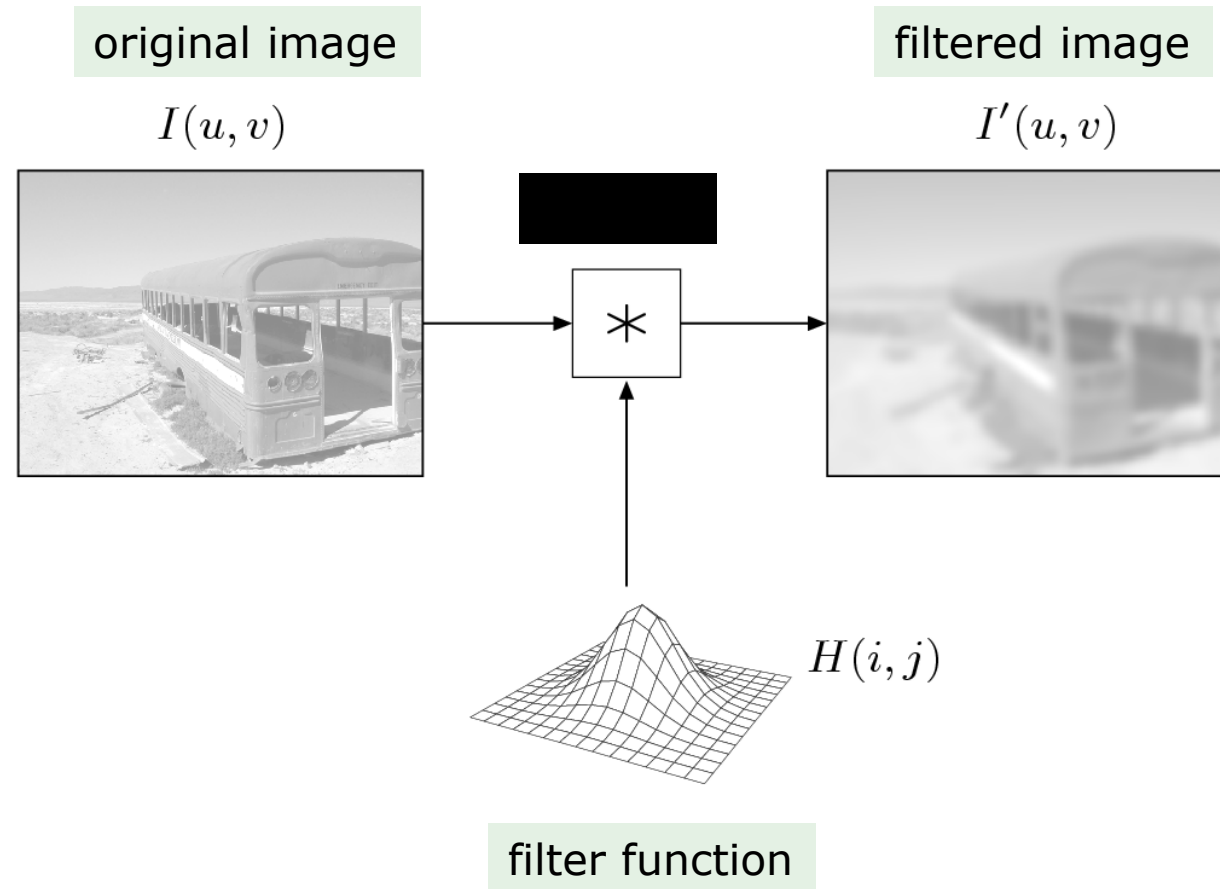
function  $I$       function  $H$

$$I' = I * H$$

convolution operator

```
graph TD; subgraph FormulaBox [ ]; I_prime_u_v["I'(u, v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u-i, v-j) \cdot H(i, j)"]; end; subgraph OperatorBox [ ]; I_prime_eq_I_H["I' = I * H"]; end; F1["function I"] --> I_prime_eq_I_H; F2["function H"] --> I_prime_eq_I_H; C1["convolution operator"] --> Star["*"];
```

# Convolution Operation



# Convolution: Important Formal Characteristics

Commutative

$$I * H = H * I$$

Associativ

$$A * (B * C) = (A * B) * C$$

Linear

$$(a \cdot I) * H = I * (a \cdot H) = a \cdot (I * H) \quad (\text{scalar } a)$$

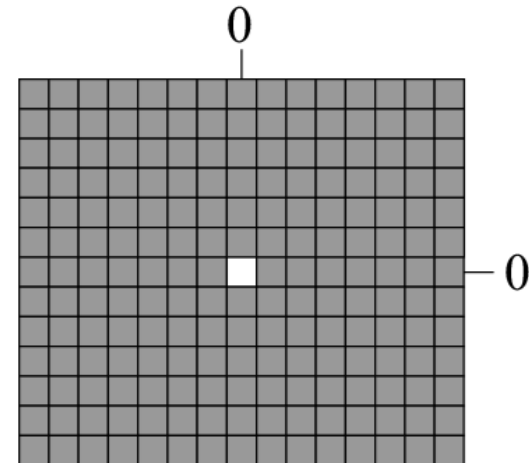
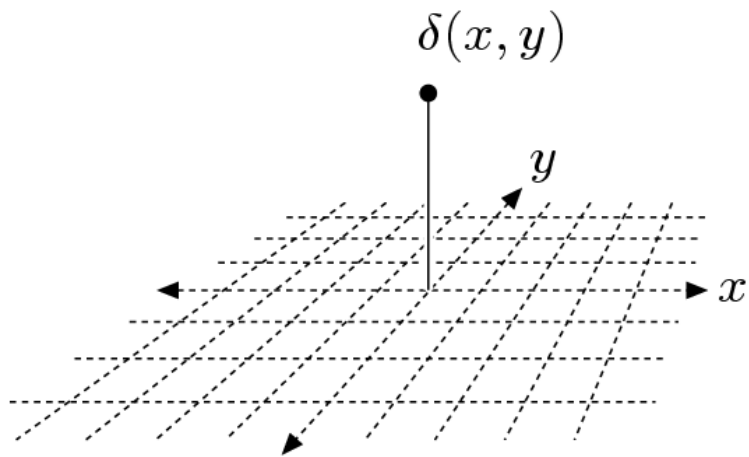
$$(I_1 + I_2) * H = (I_1 * H) + (I_2 * H)$$

# Neutral Function – Dirac Function

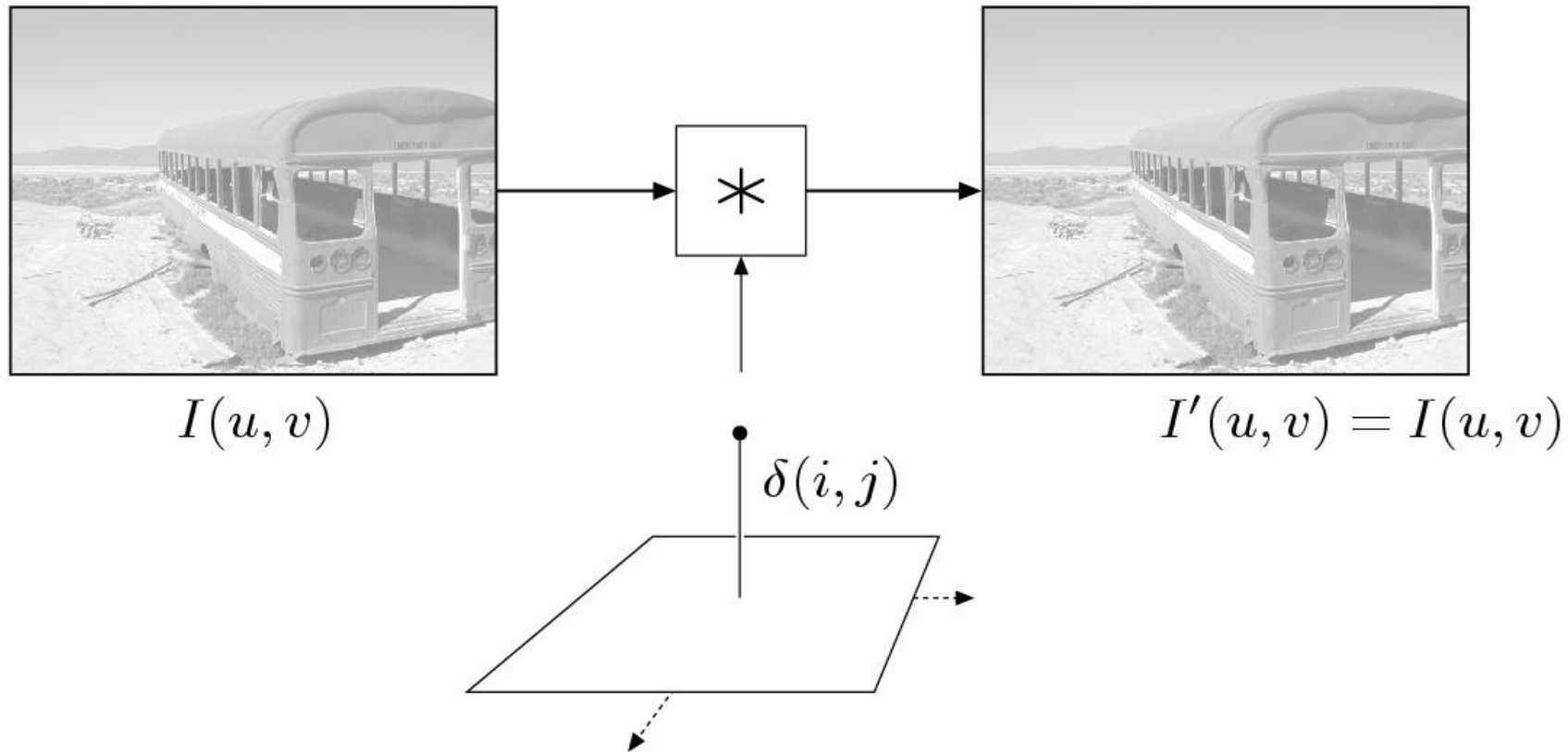
The Dirac or impulse function is the “neutral element” of the linear convolution operation, i.e.

$$I * \delta = I$$

$$\delta(i, j) = \begin{cases} 1 & \text{für } i = j = 0 \\ 0 & \text{sonst.} \end{cases}$$



# Convolution with Dirac Function



# X/Y Separability

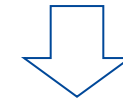
1. Decomposition of two-dimensional filters into a sequence of one-dimensional filters
2. Efficient implementation of large filters possible

Example:

Two 1D filters:  $H_x = \begin{bmatrix} 1 & 1 & \underline{1} & 1 & 1 \end{bmatrix}$        $H_y = \begin{bmatrix} 1 \\ 1 \\ \underline{1} \\ 1 \\ 1 \end{bmatrix}$

Applying both filters:

$$I' \leftarrow (I * H_x) * H_y = I * \underbrace{(H_x * H_y)}_{H_{xy}}$$



Result:

A larger filter can be created from two smaller filters.

What computational savings does this result in?

$$H_{xy} = H_x * H_y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \underline{1} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

# X/Y Separability – Another Example

$$H = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$H_1 \qquad H_2 \qquad H$

There are two ways to calculate a filter operation with H:

Variant 1:  $I * H$       9 operations per pixel

Variant 2:  $(I * H_1) * H_2$     3+3=6 operations

The advantage is  
significant  
for larger filters:

5x5	25	10
7x7	49	14
25x25	625	50

# Linear, Homogeneous Filters

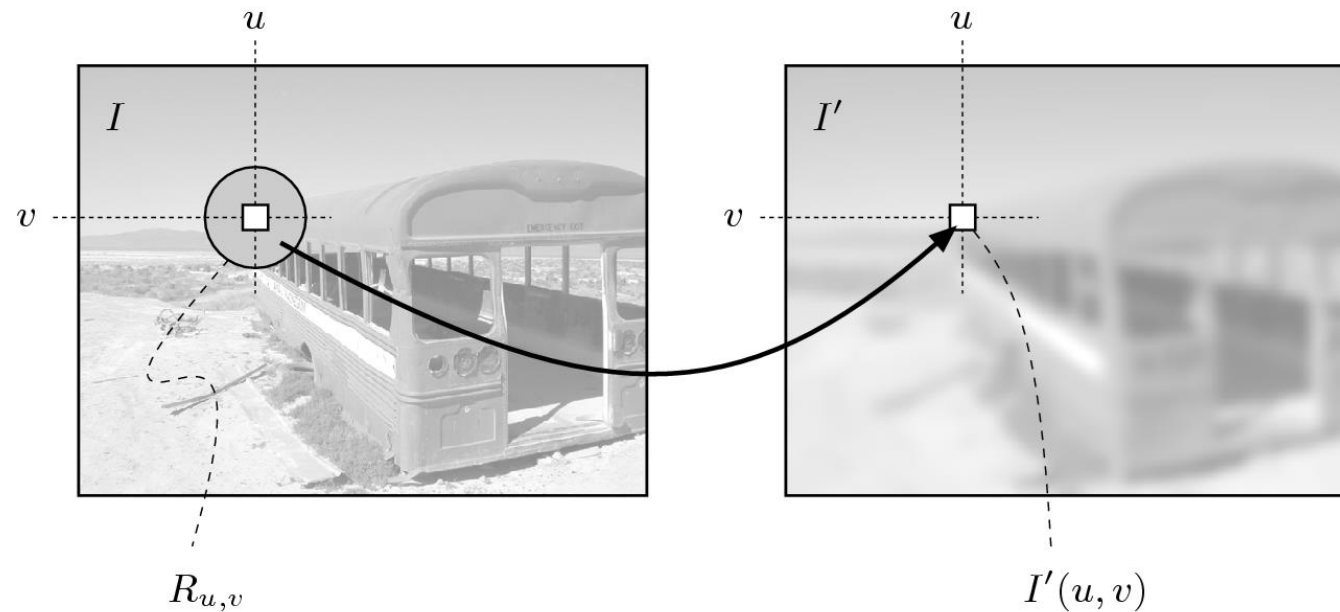
- Simple description
- Comprehensive mathematical theory
- Practical application:
  - Smoothing, contour filter (later)
  - Linearity not guaranteed due to limited value range
  - Often used in combination with nonlinear operations

# Simple Non-linear Filters

The new pixel value is the minimum/maximum within the specified filter region  $R$ :

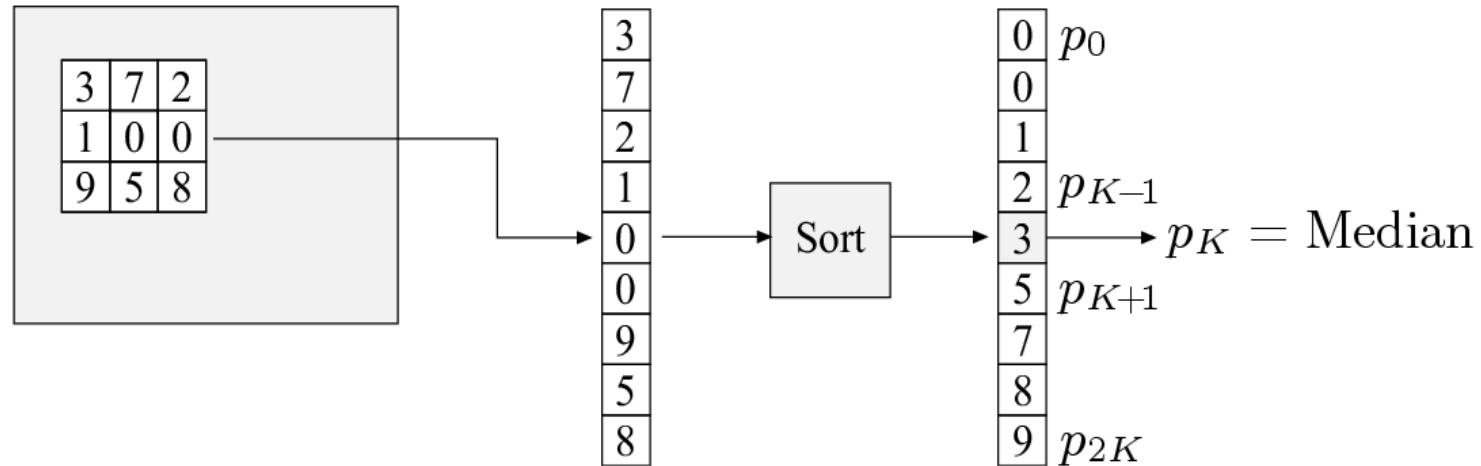
$$I'(u, v) \leftarrow \min \{I(u + i, v + j) \mid (i, j) \in R\} = \min(R_{u,v})$$

$$I'(u, v) \leftarrow \max \{I(u + i, v + j) \mid (i, j) \in R\} = \max(R_{u,v})$$

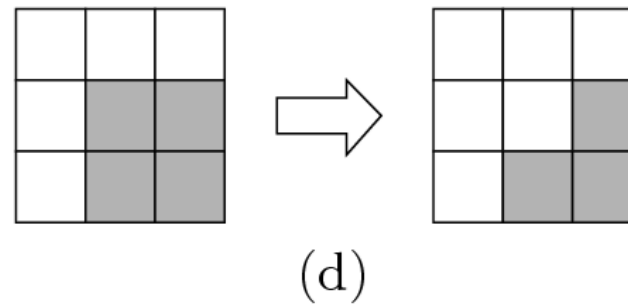
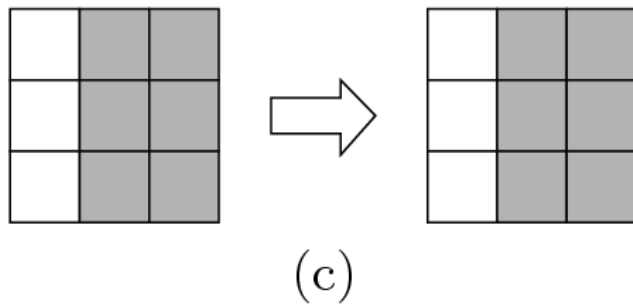
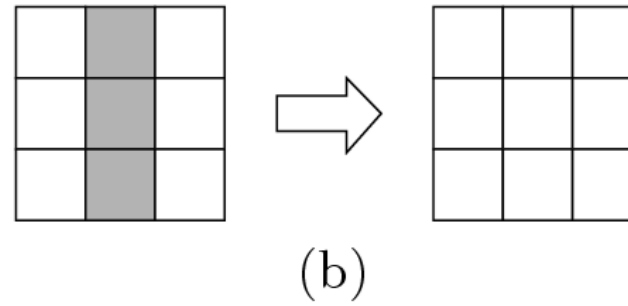
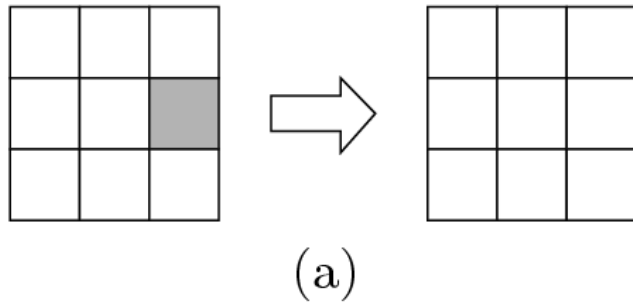


# Median Filter

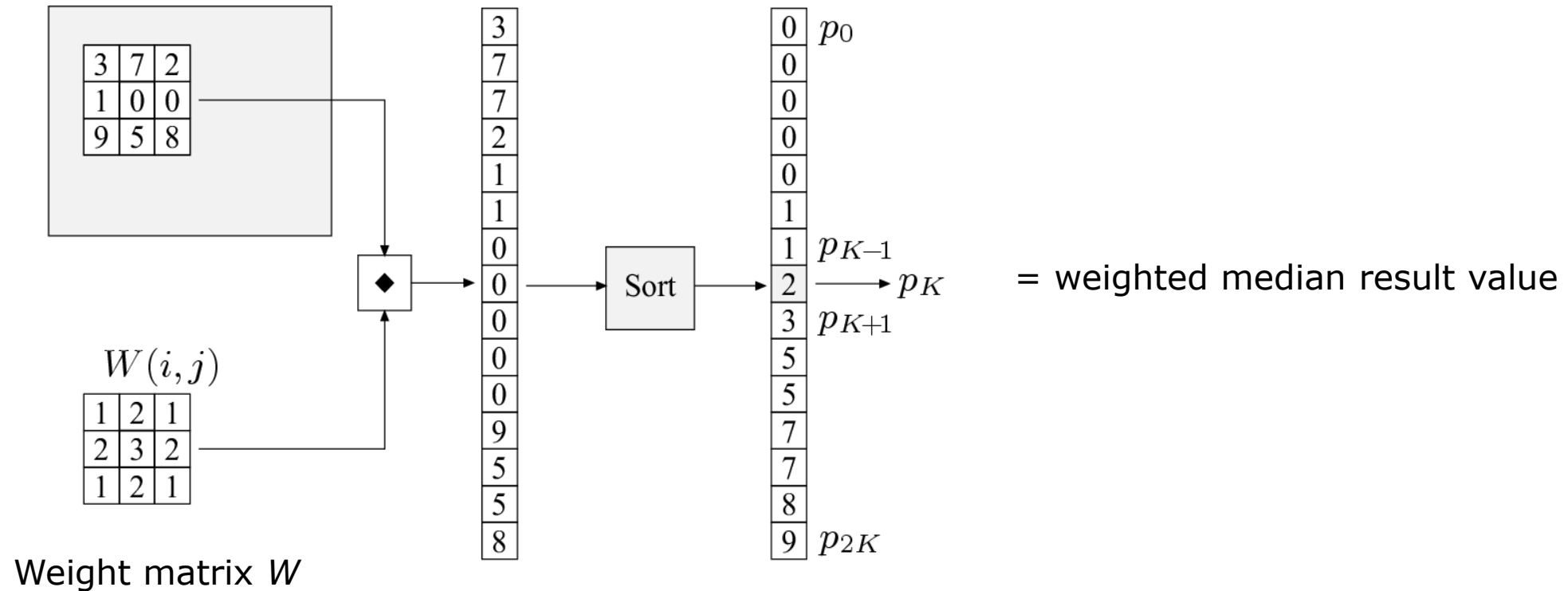
$$I'(u, v) \leftarrow \text{median}(R_{u,v}) \quad \text{median}(p_0, p_1, \dots, p_K, \dots, p_{2K}) = p_K$$



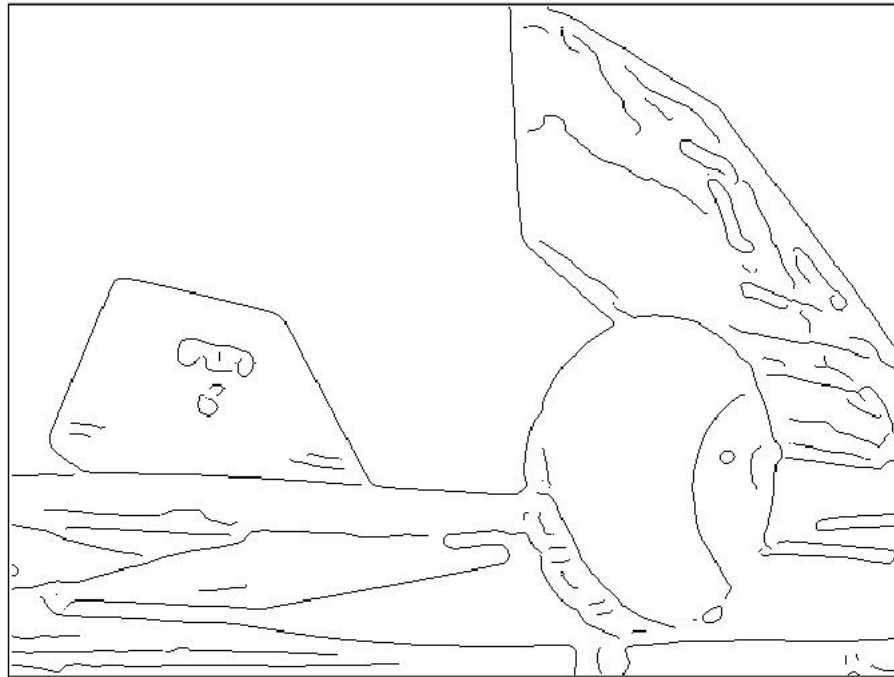
# Effects of a Median Filter



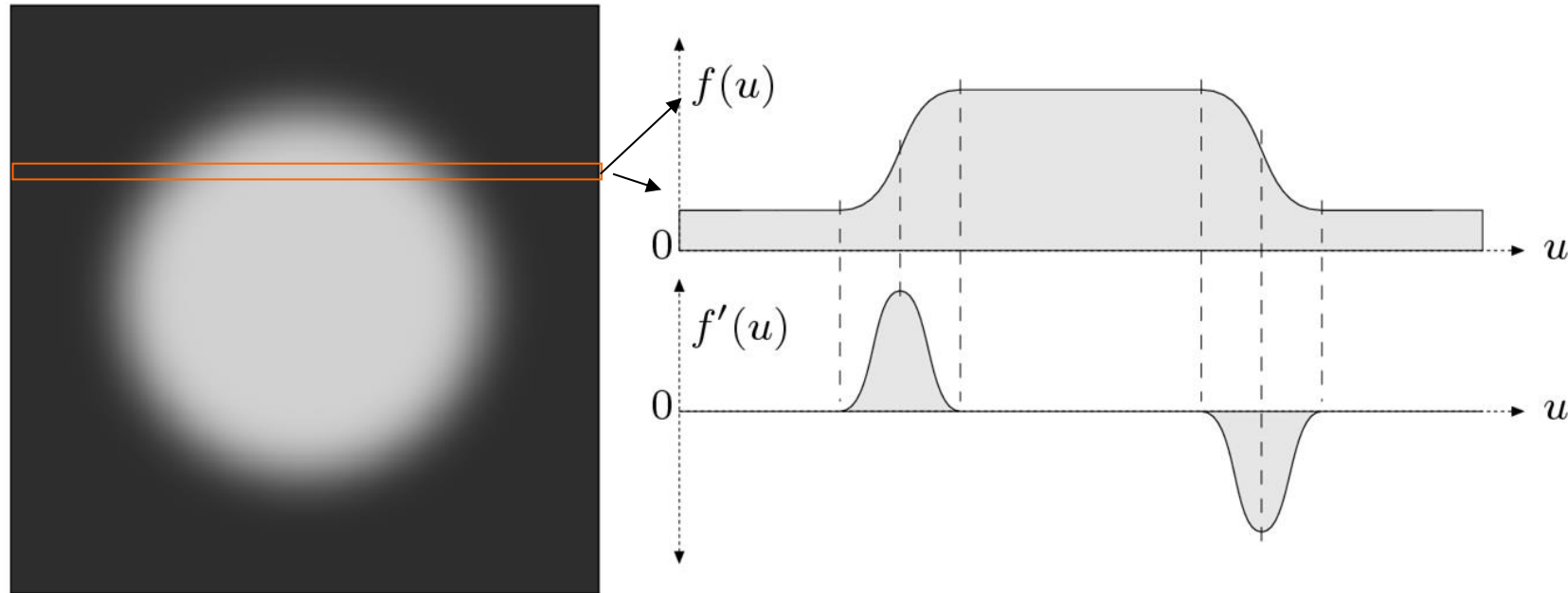
# Weighted Median Filter



# Edges

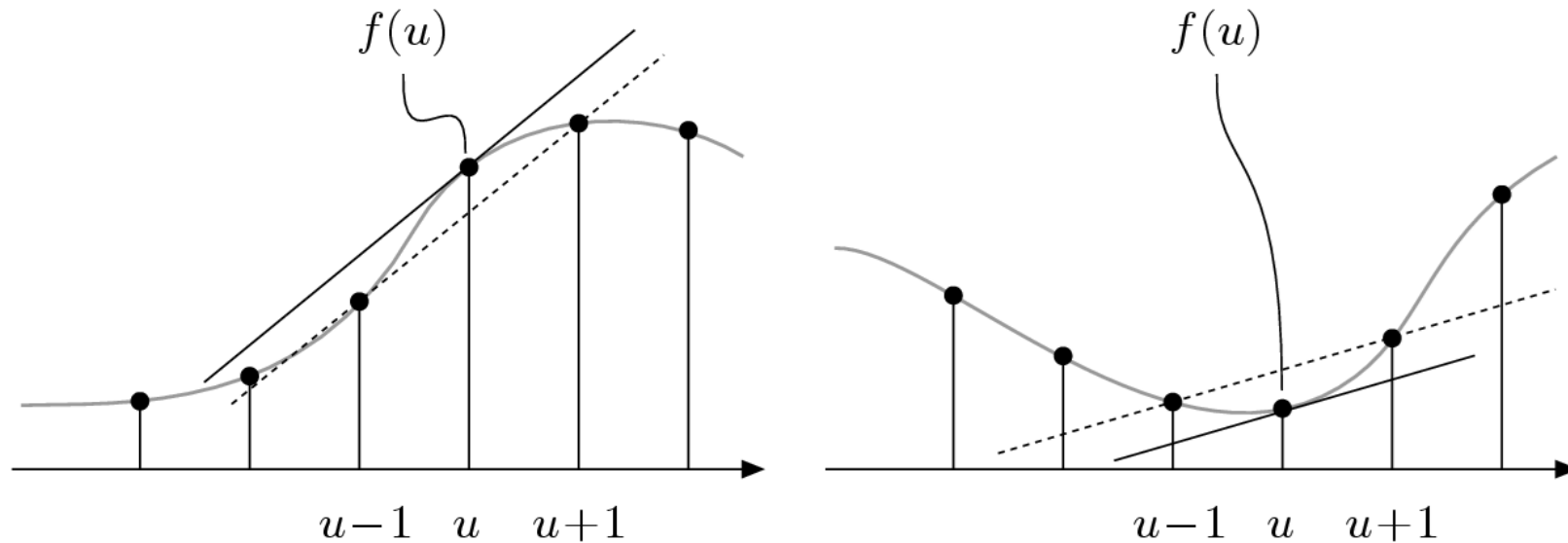


# Edges as Derivation of the Image “Function”



$$f'(u) = \frac{df}{du}(u)$$

# Approximation of 1. Derivation (Tangent's Gradient)



$$\begin{aligned}\frac{df}{du}(u) &\approx \frac{f(u+1) - f(u-1)}{2} \\ &= 0.5 \cdot (f(u+1) - f(u-1))\end{aligned}$$

# Simple Edge Filter (Based on Approximated Derivation)

$$H_x^D = \begin{bmatrix} -0.5 & \underline{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \underline{0} & 1 \end{bmatrix} \quad \frac{\partial I}{\partial u}(u, v)$$

$$H_y^D = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \frac{\partial I}{\partial v}(u, v)$$

Gradient (= vector)

$$\nabla I(u, v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u, v) \\ \frac{\partial I}{\partial v}(u, v) \end{bmatrix}$$

Magnitude of gradient

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial u}\right)^2 + \left(\frac{\partial I}{\partial v}\right)^2}$$

# Example: Prewitt Edge Filter

Prewitt filter in horizontal direction (x)



\*

-1	0	1
-1	0	1
-1	0	1

=



# Example: Prewitt Edge Filter

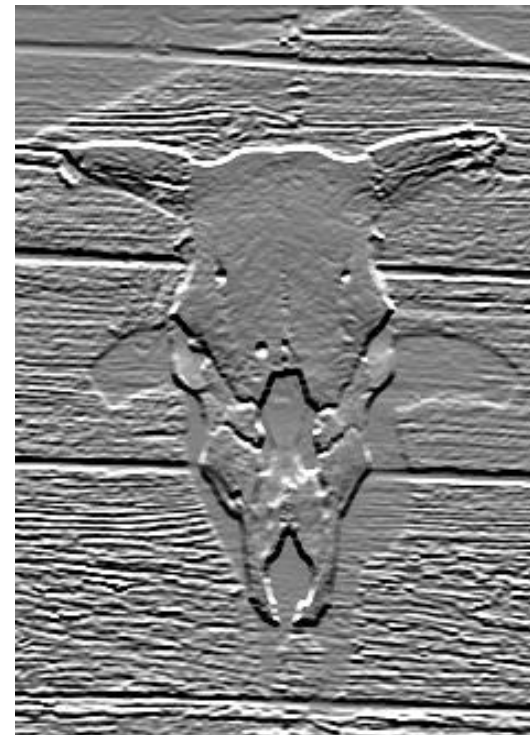
Prewitt filter in vertical direction (y)



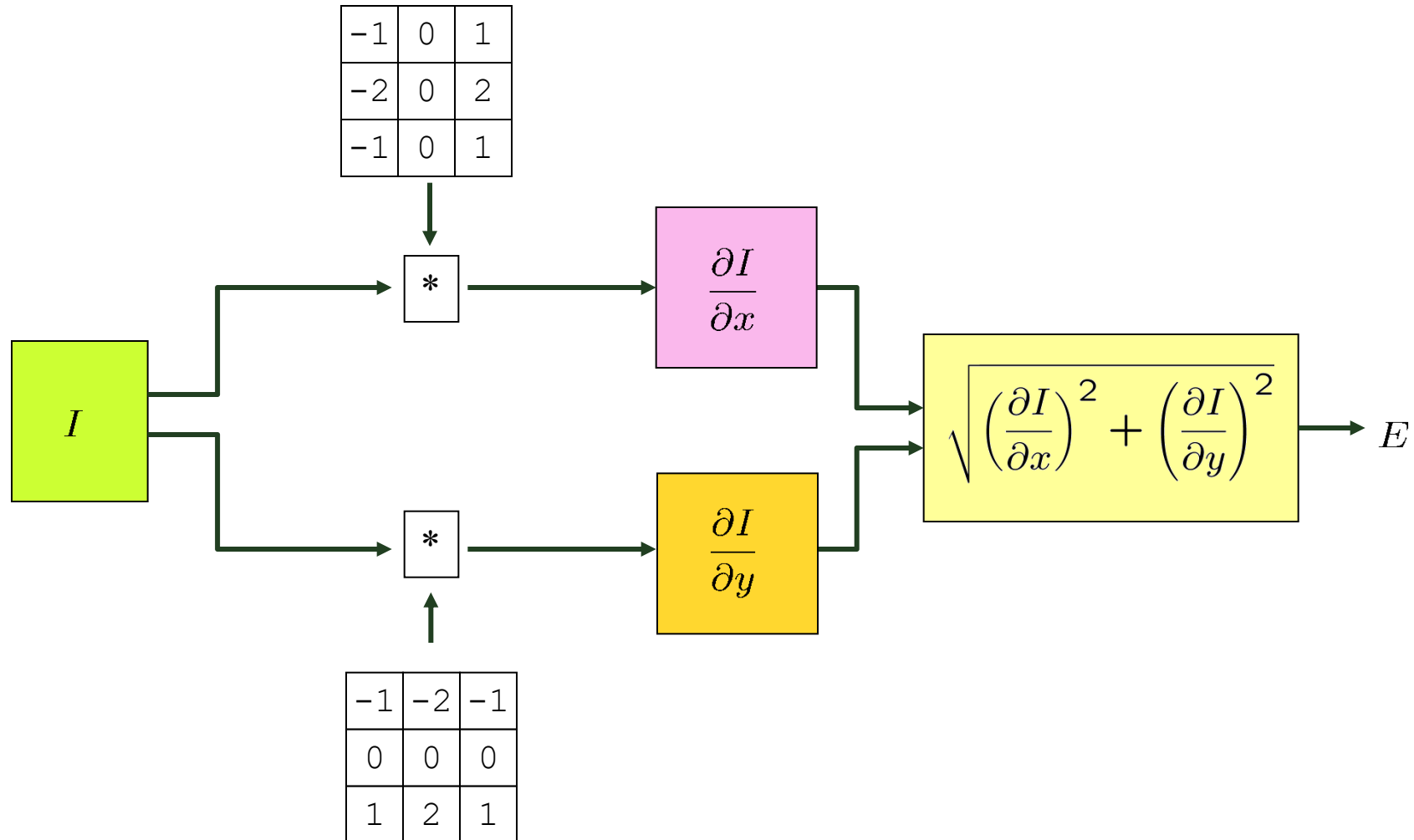
\*

-1	-1	-1
0	0	0
1	1	1

=



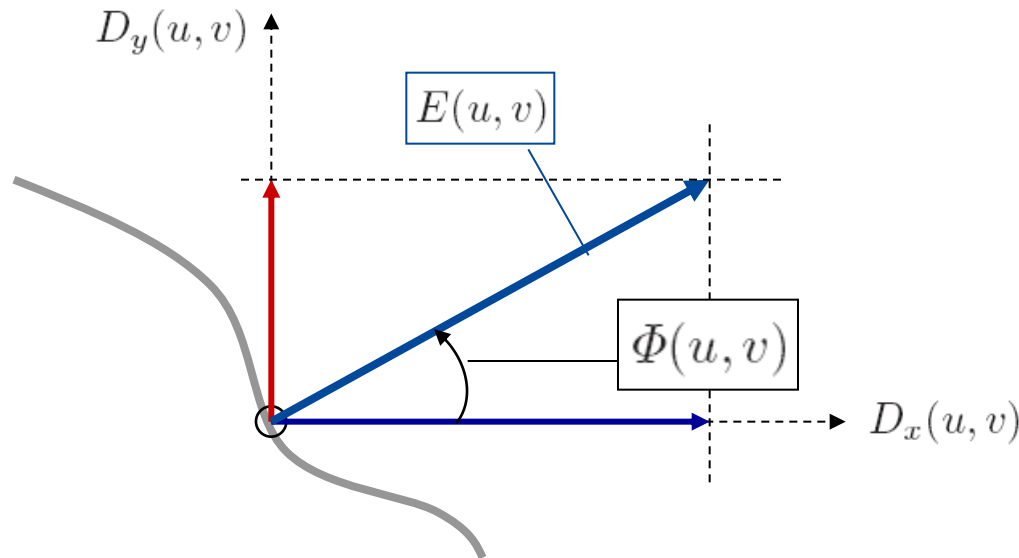
# Sobel Edge Operator



# Edge Strength (Magnitude) and Orientation

$$D_x(u, v) = H_x * I$$

$$D_y(u, v) = H_y * I$$



Local edge strength (magnitude)

$$E(u, v) = \sqrt{(D_x(u, v))^2 + (D_y(u, v))^2}$$

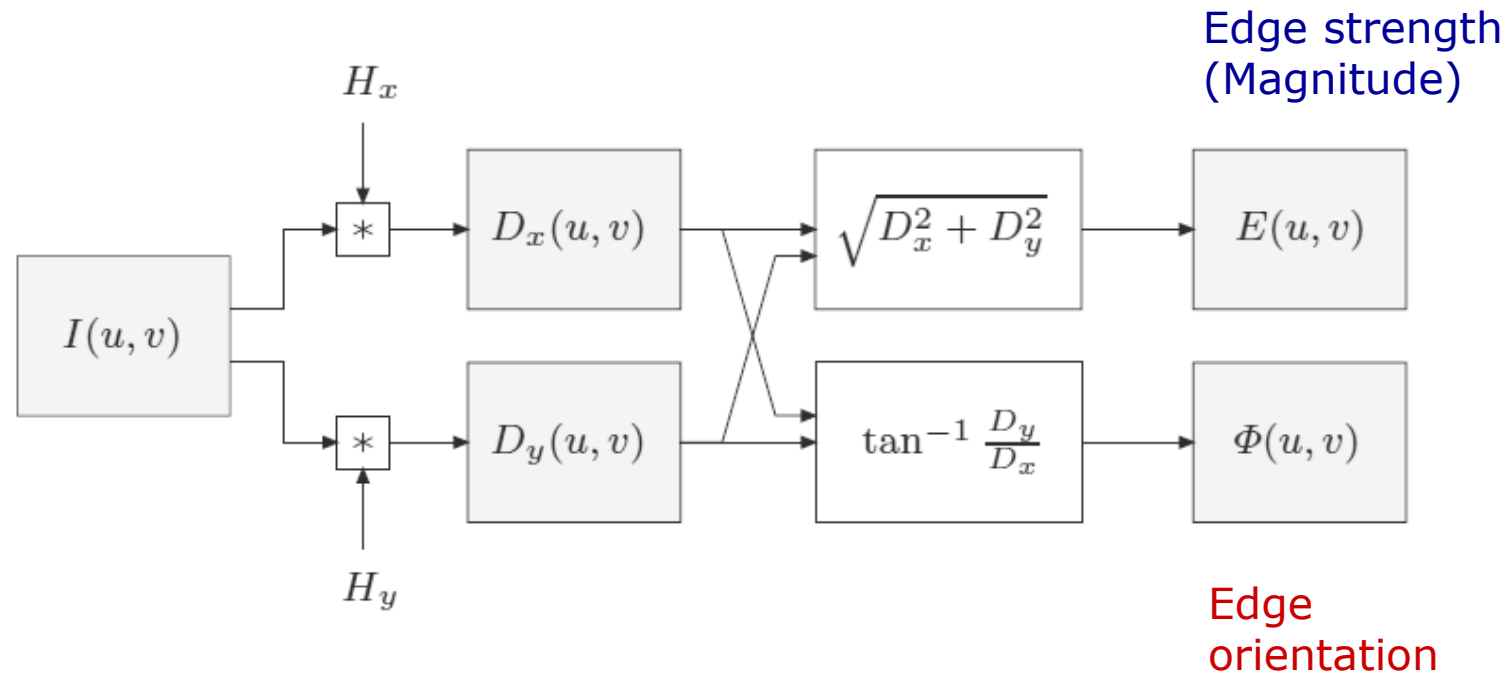
Local edge orientation

$$\Phi(u, v) = \tan^{-1} \left( \frac{D_y(u, v)}{D_x(u, v)} \right)$$

calculation of arc tangent, i.e. the inverse function of tangent function

# Typical Use of Gradient Filters

Computation of edge strength and orientation

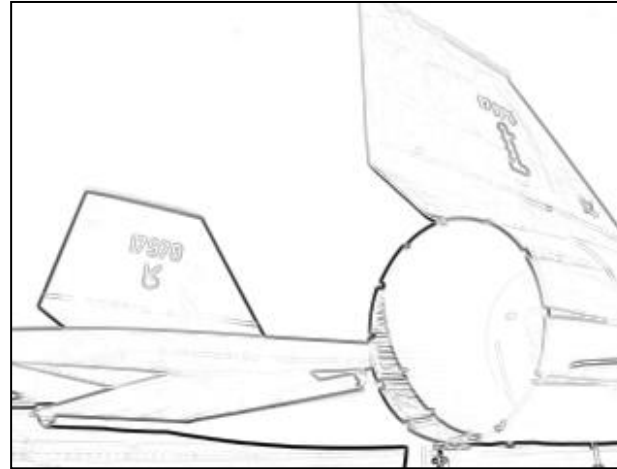


# Edge Magnitude + Orientation

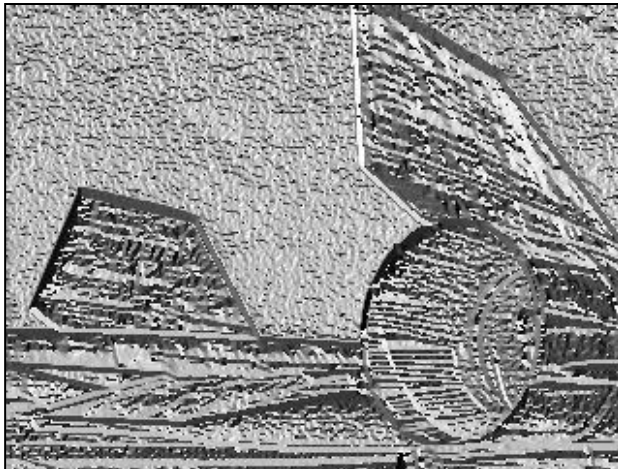
Original



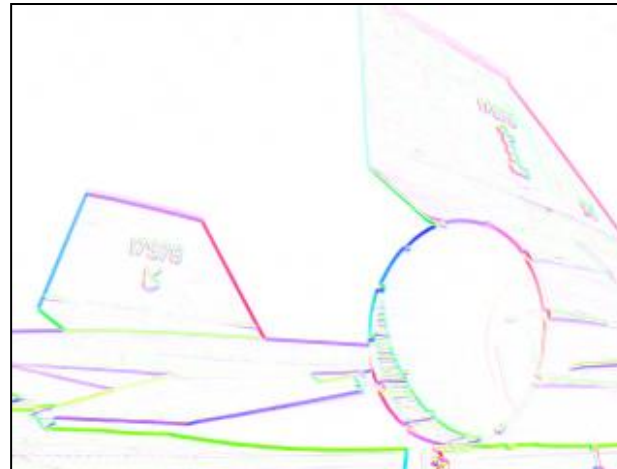
Edge  
Magnitude



Edge  
Orientation



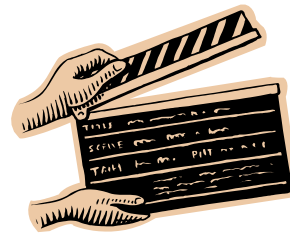
Magnitude  
plus  
Orientation



# Lessons Learned So Far

- Difference between local linear filters and point operations
- What is a linear filter?
- What are the properties of linear filters?
- What are non-linear filters?
- How does the weighted median filter work?
- How can we approximate first derivation of image function?
- How does the Sobel operator work?

# The End.



Thanks for your attention!