Computer Vision

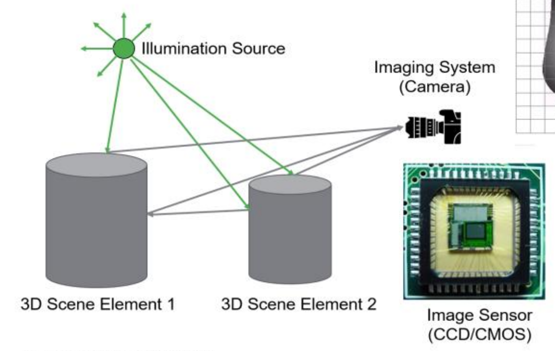
Lecture 3: Digital Image Processing

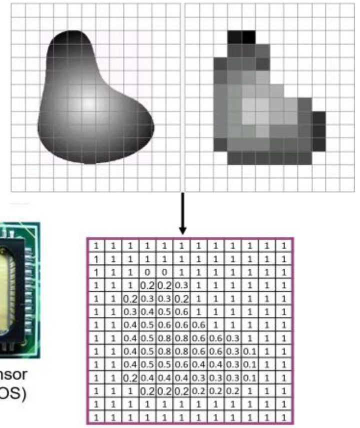
Oliver Bimber

JOHANNES KEPLER UNIVERSITY LINZ

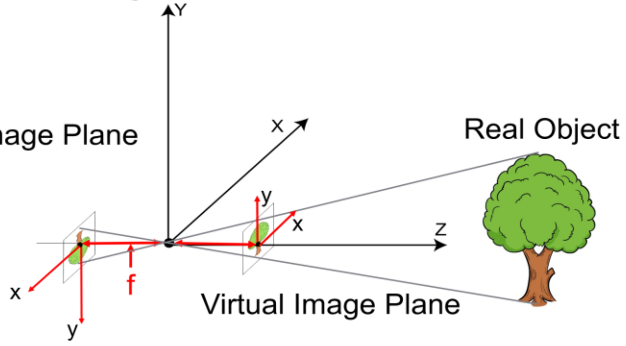
Last Week: Capturing Digital Images

Formation of Digital Images

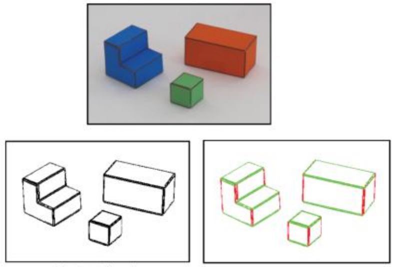




Perspective Projection



On a lower Level: Detect Features (e.g., Edges)



Using E(x,y) Using θ(x,y)

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

Approximation image derivative:

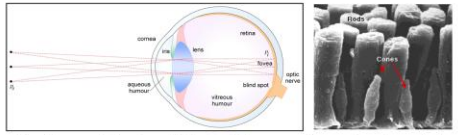
$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

Edge strength $E(x, y) = |\nabla \mathbf{I}(x, y)|$

Edge orientation: $\theta(x, y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I}/\partial y}{\partial \mathbf{I}/\partial x}$

Edge normal: $\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$

How Humans do it?



The Human Eye is a biological Camera (Lens, Aperture=Iris, Sensor=Retina) and is used for Measurement=Sensing.

The Perception of the Visual Cortex is really what we want to mimic with Computer Vision!

Course Overview

CW	Topic	Date	Place	Lab
41	Introduction and Course Overview	07.10.2025	Zoom	Lab 1
42	Capturing Digital Images	14.10.2025	Zoom	Lab 2
43	Digital Image Processing	21.10.2025	Zoom	Assignment 1
44	Machine Learning	28.10.2025	Zoom	
45	Feature Extraction	04.11.2025	Zoom	Open Lab 1
46	Segmentation	11.11.2025	Zoom	Assignment 2
47	Optical Flow	18.11.2025	Zoom	Open Lab 2
48	Object Detection	25.11.2025	Zoom	Assignment 3
49	Multi-View Geometry	02.12.2025	Zoom	Open Lab 3
50	3D Vision	09.12.2025	Zoom	Assignment 4
3	Trends in Computer Vision	13.01.2026	Zoom	
4	Q&A	20.01.2026	Zoom	Open Lab 4
5	Exam	27.01.2026	HS1 (Linz), S1/S3 (Vienna), S5 (Bregenz)	
9	Retry Exam	24.02.2026	tba	

Recap: Detect Features (e.g., Edges)

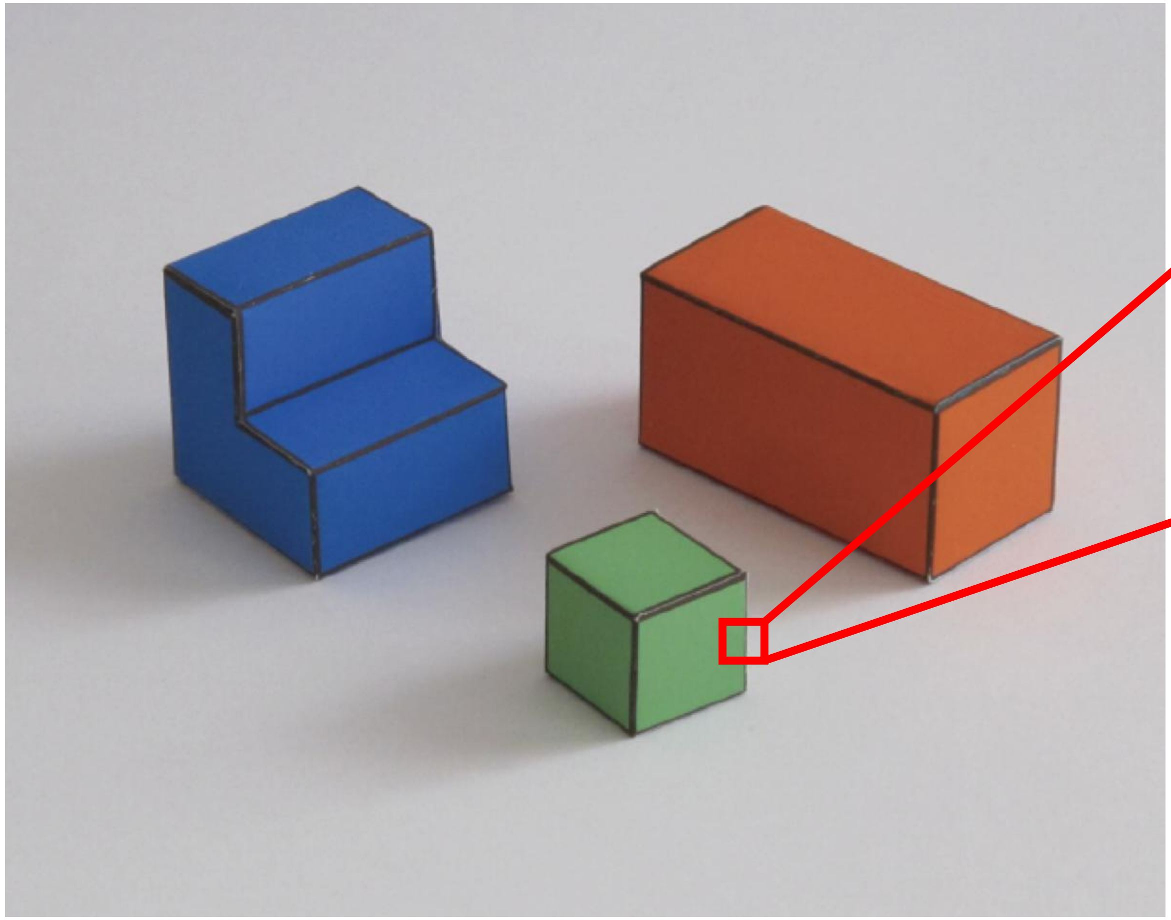
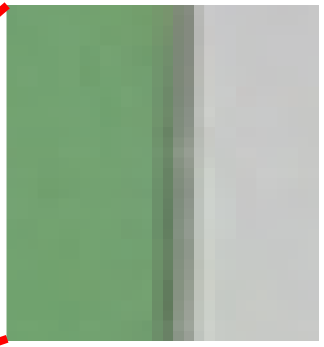


Image Patch



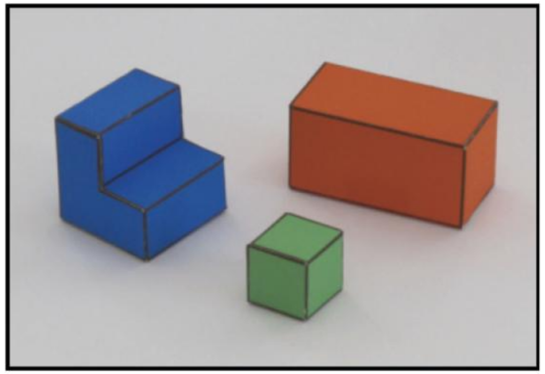
… 125, 126, 50, 10, 223, 223, …
…,124, 126, 50, 10, 223, 224, …
…,125, 127, 51, 9, 223, 224, …
…

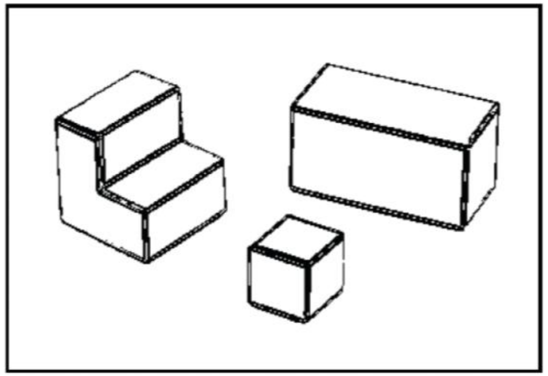
What Measure can tell us that there is an Edge here?

Image gradient:

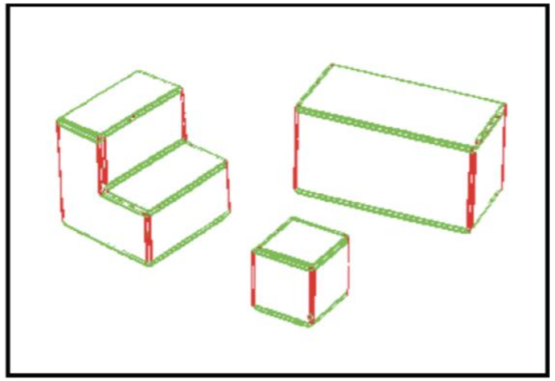
$$\nabla \mathrm{I} = \left(\frac{\partial \mathrm{I}}{\partial x}, \frac{\partial \mathrm{I}}{\partial y} \right)$$

Recap: Detect Features (e.g., Edges)





Using E(x,y)



Using θ(x,y)

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

Approximation image derivative:

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

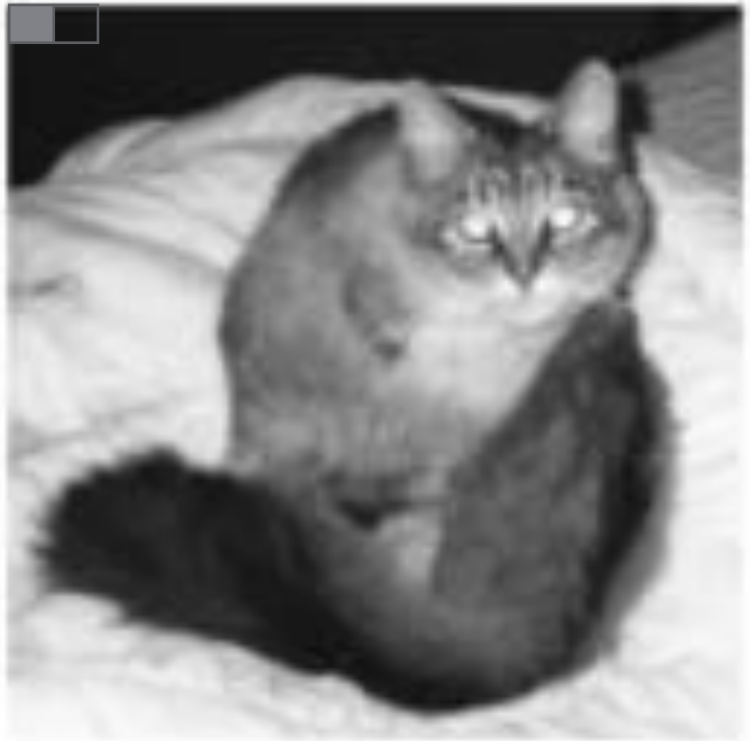
Edge strength $E(x, y) = |\nabla \mathbf{I}(x, y)|$

Edge orientation: $\theta(x, y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I} / \partial y}{\partial \mathbf{I} / \partial x}$

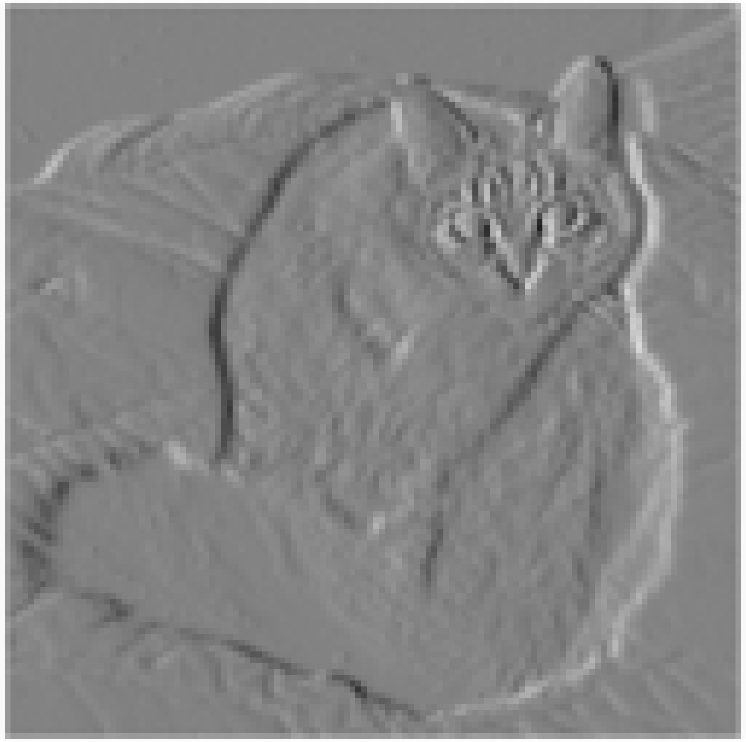
Edge normal: $\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$

Image Gradients

What happens if we apply this to every Pixel?



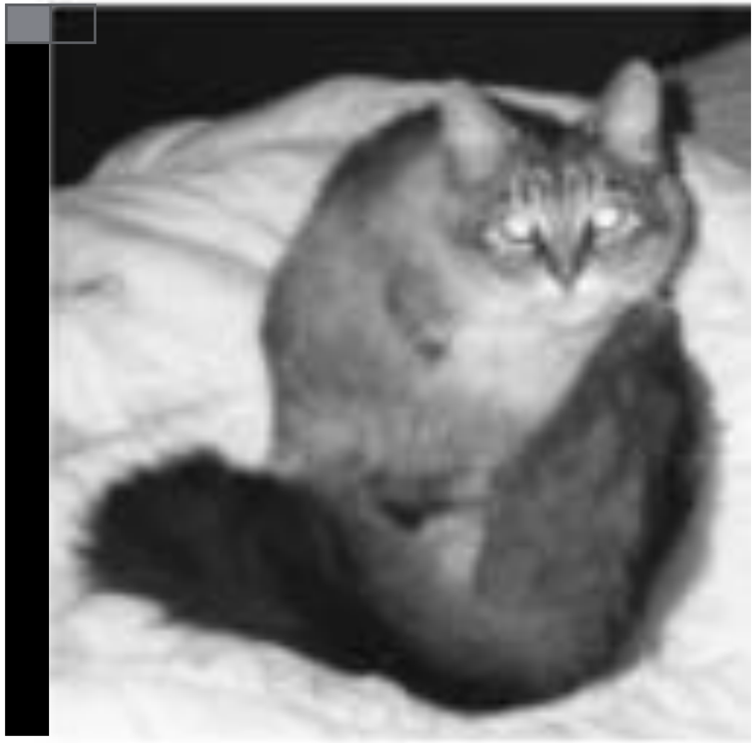
$$\mathbf{I}(x, y)$$



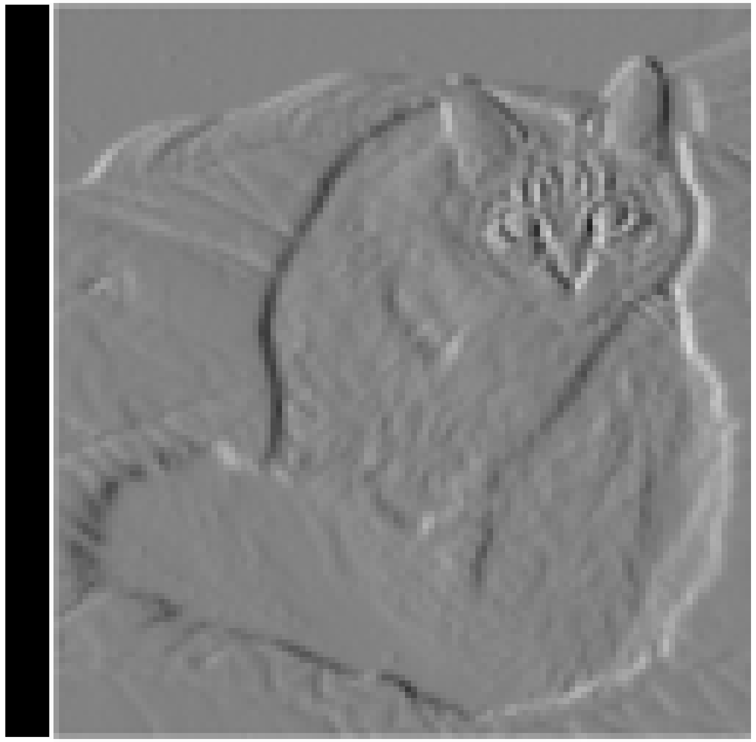
$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

Image Gradients

And what happens if we shift the Image?



$\mathbf{I}(x,y)$



$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x,y) - \mathbf{I}(x-1,y)$$

This is called Shift-Invariance.

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3						

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3						

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3	-4					

Convolution

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3	-4	-4				

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3	-4	-4	-4			

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3	-4	-4	-4	-4		

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3	-4	-4	-4	-4	-3	

Convolution

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3	-4	-4	-4	-4	-3	
	-3						

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

=

y

	-3	-4	-4	-4	-4	-3	
	-3	-4	-4	-3	-1	0	
	0	0	0	0	0	0	
	2	1	0	1	3	3	
	2	1	0	1	3	3	
	1	3	4	3	1	0	

Convolution

x

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

Kernel

1	2	1
0	0	0
-1	-2	-1

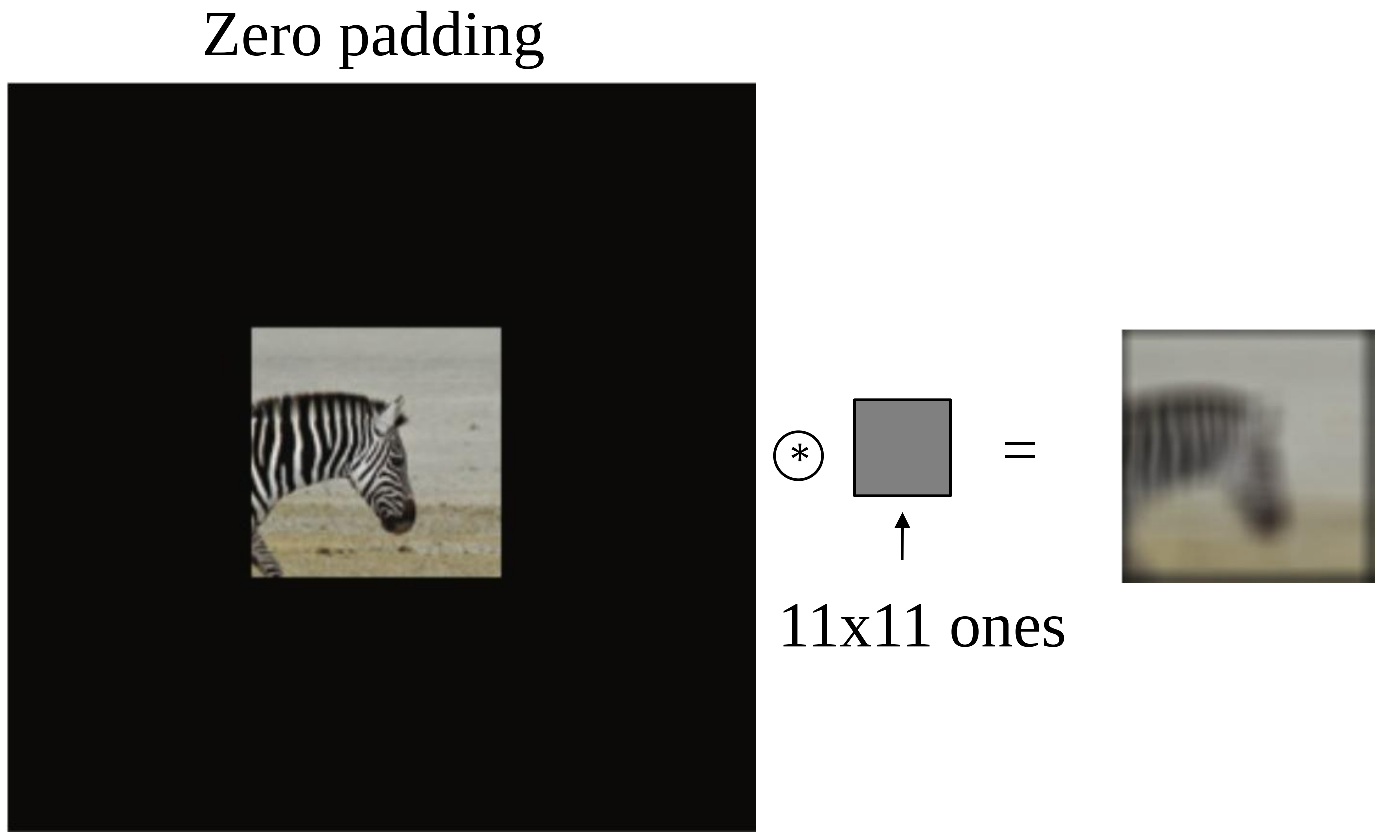
=

y

?							
	-3	-4	-4	-4	-4	-3	
	-3	-4	-4	-3	-1	0	
	0	0	0	0	0	0	
	2	1	0	1	3	3	
	2	1	0	1	3	3	
	1	3	4	3	1	0	

You have to handle the Border Cases: e.g. Zero-Padding.

Example: Zero Padding



Convolution

$$y[m,n] = x * h = \sum_{k,l=-N}^{N} x[m+k, n+l]\, h[-k,-l]$$

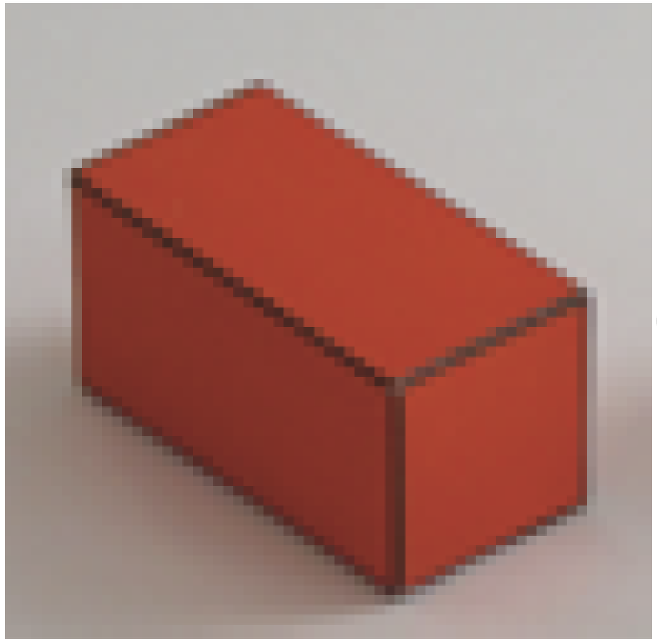
0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0

l \ k	-1	0	1
1	-1	-2	-1
0	0	0	0
-1	1	2	1

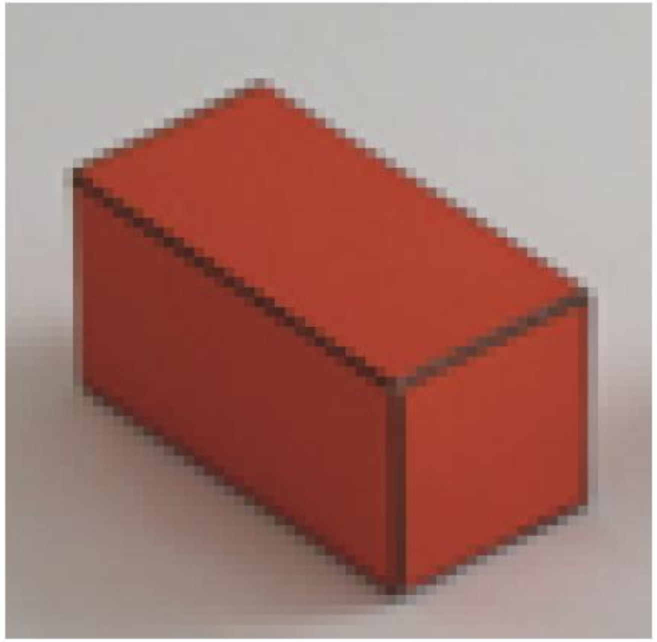
In this case, N = 1.

Convolution is a shift-invariant linear Operation.

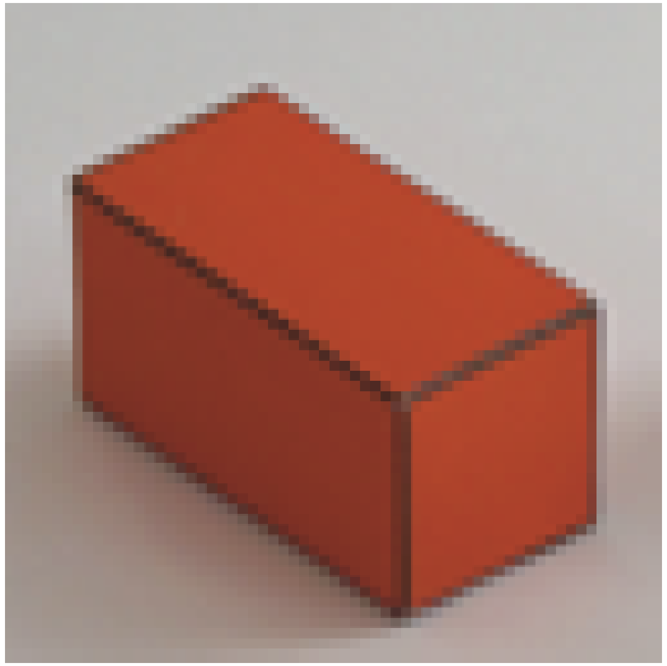
Example: Impulse



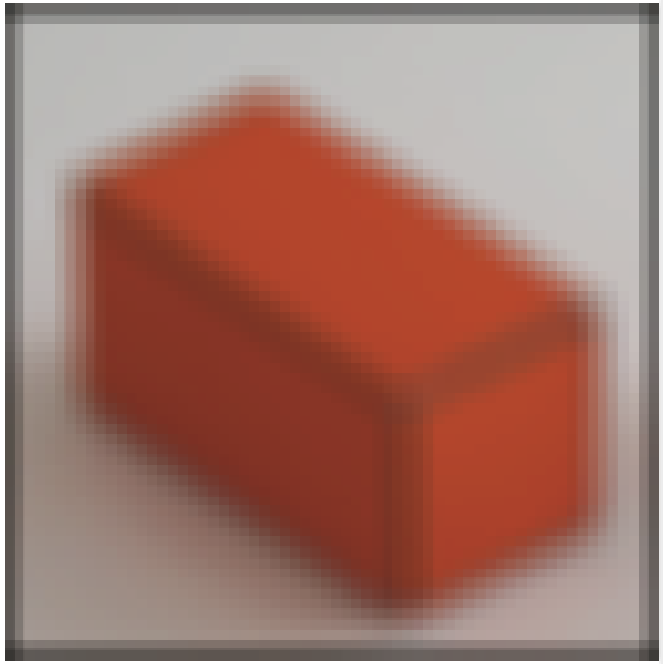
$$\circledast \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} =$$



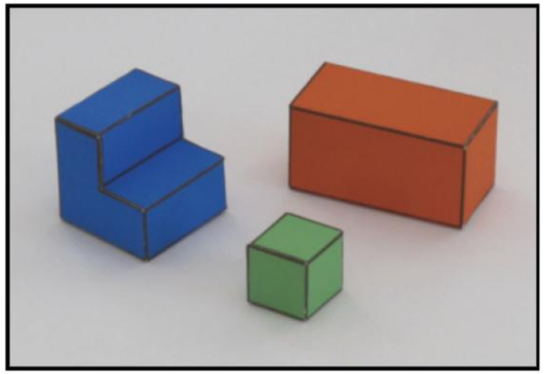
Example: Blurring (Box Filter)



$$\circledast \quad \begin{array}{c} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\ /25 \end{array} \quad =$$

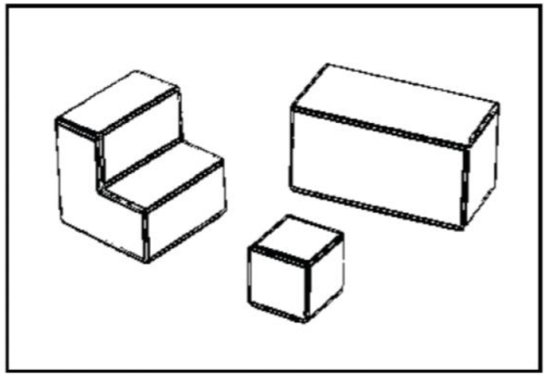


Recap: Gradients

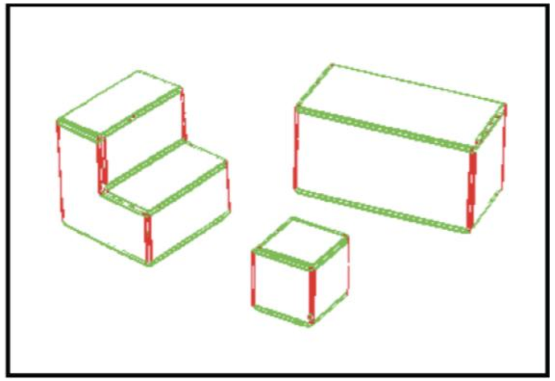


$$K_{Gx} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$K_{Gy} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



Using E(x,y)



Using θ(x,y)

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

Approximation image derivative:

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x,y) - \mathbf{I}(x-1,y)$$

Edge strength $E(x,y) = |\nabla \mathbf{I}(x,y)|$

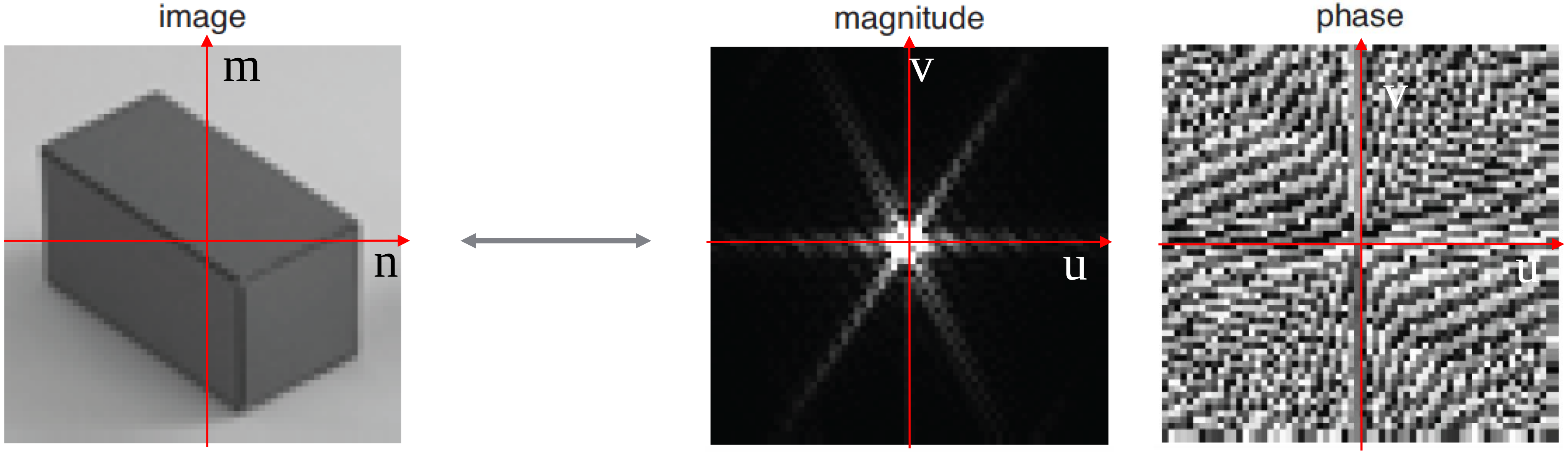
Edge orientation: $\theta(x,y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I}/\partial y}{\partial \mathbf{I}/\partial x}$

Edge normal: $\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$

JOHANNES KEPLER UNIVERSITY LINZ

Spatial vs. Frequency Domain

$f[n,m]$ $F[u,v]$



Fourier Transformation /
Inverse Fourier Transformation

Spectra of Natural Images

Convolution Theorem

The Fourier transform of the convolution is the product of Fourier transforms

$$f[n,m] = h * g \qquad \Longleftrightarrow \qquad F[u,v] = G[u,v]H[u,v]$$

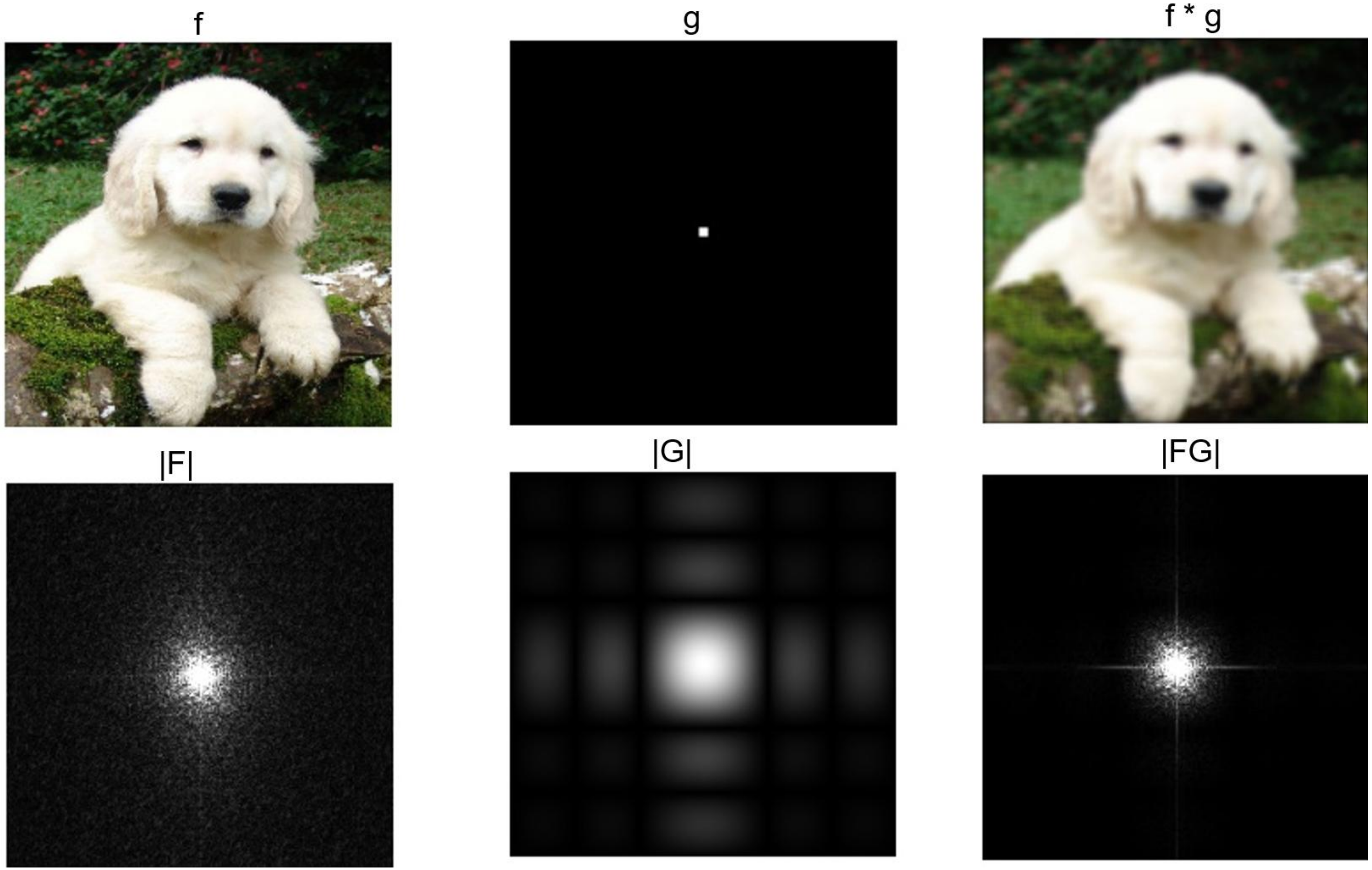
The Fourier transform of the product is the convolution of Fourier transforms

$$f[n,m] = g[n,m]h[n,m] \Longleftrightarrow F[u,v] = \frac{1}{NM} G[u,v] * H[u,v]$$

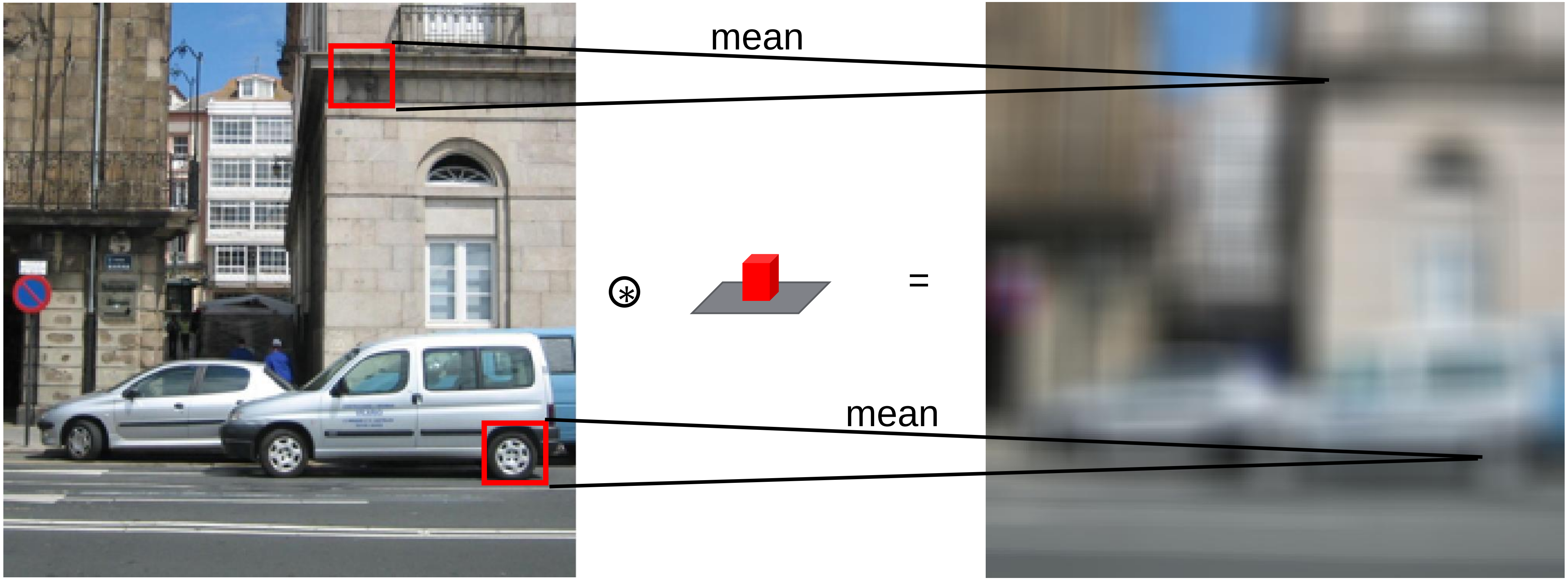
Convolution Properties

	Spatial Domain	Fourier Domain
Commutative	$f * g = g * f$	$FG = GF$
Associative	$(f * g) * h = f * (g * h)$	$(FG)H = F(GH)$
Distributive	$(f + g) * h = f * h + g * h$	$(F + G)H = FH + GH$
Linear	$(af + bg) * h = af * h + bg * h$	$(aF + bG)H = aFH + bGH$
Shift Invariant	$f(x + t) * h = (f * h)(x + t)$	$(e^{i\omega t}F)H = e^{i\omega t}(FH)$

Example: Box Filter



Low-Pass Filter (Box Filter)



256X256

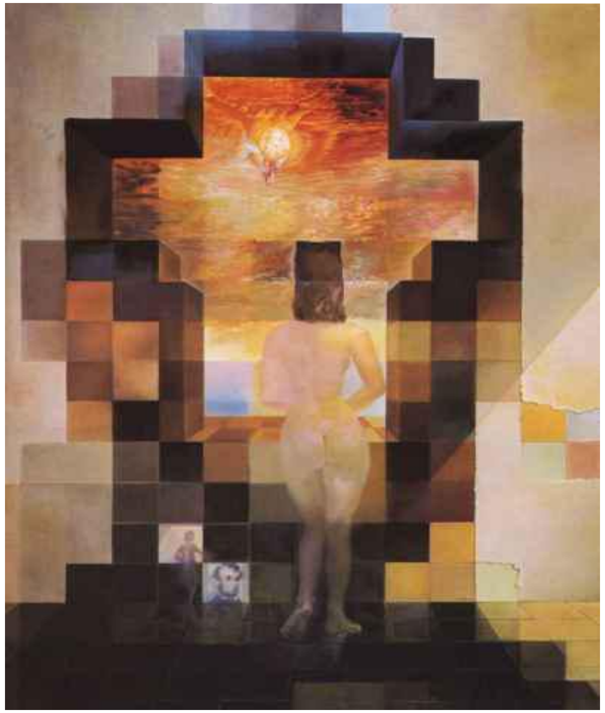
256X256

- Replaces each pixel with an Average of its Neighborhood.
- Achieves smoothing Effect (remove sharp Features, but also Noise).

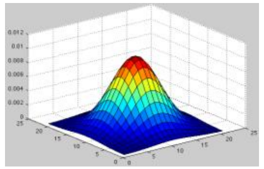
JOHANNES KEPLER
UNIVERSITY LINZ

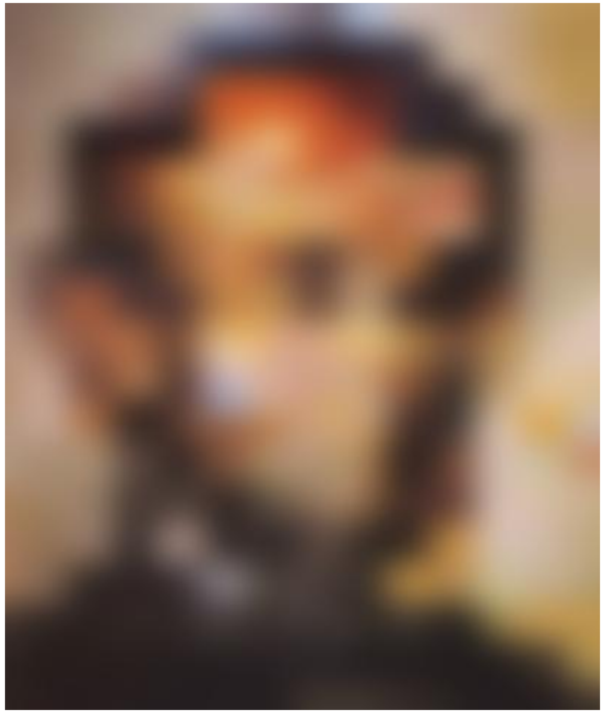
Low-Pass Filter (Gaussian Filter)

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp - \frac{x^2 + y^2}{2\sigma^2}$$



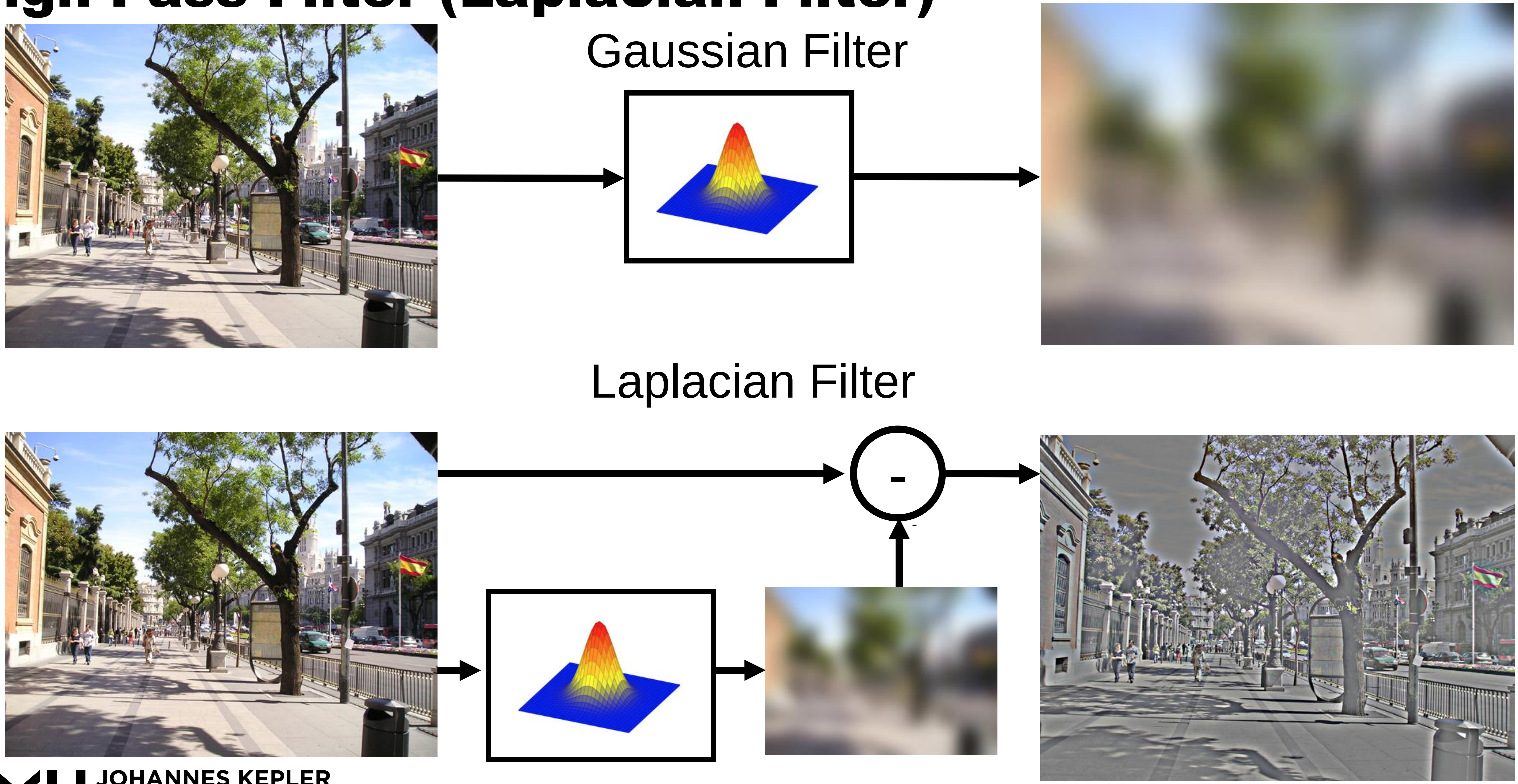
*



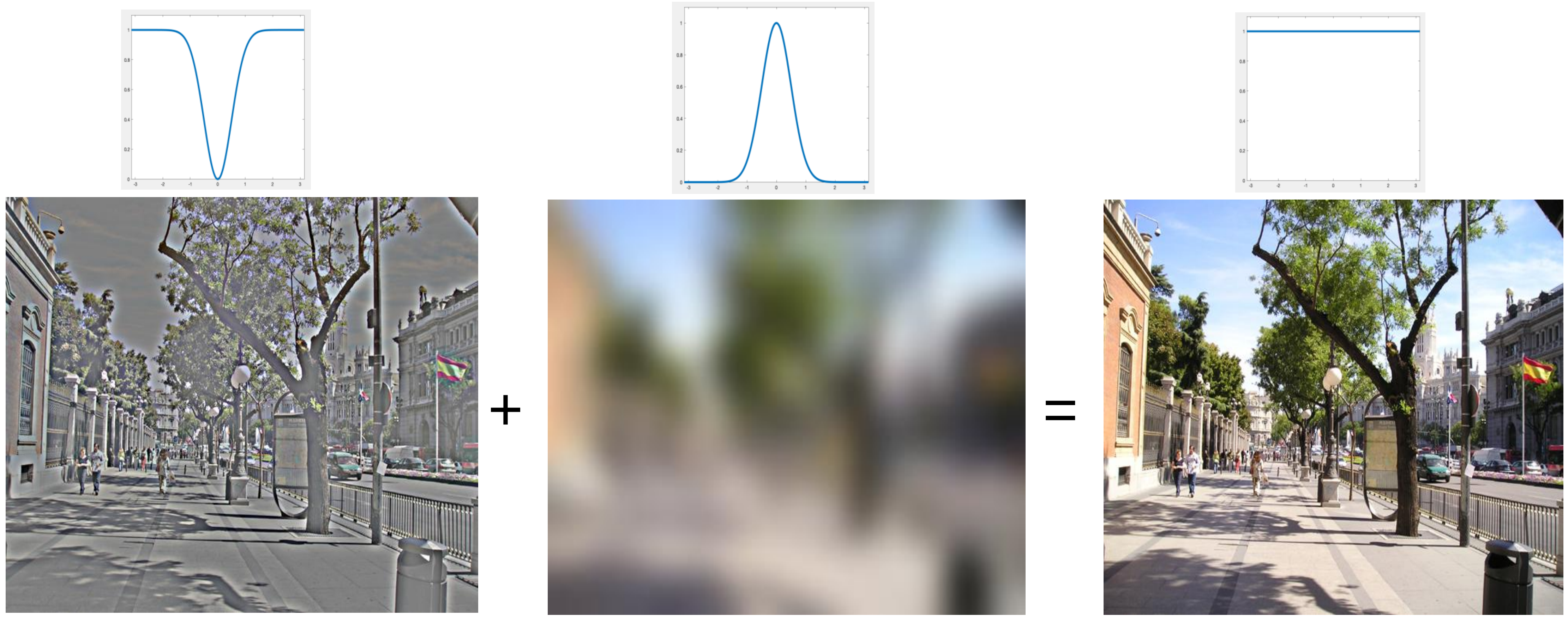


- Convolution of two Gaussians is a Gaussian.
- The (continuous) Fourier Transform of a Gaussian is another Gaussian.
- Gaussians are separatable (2x1D Convolution instead of 1x2D).

High-Pass Filter (Laplacian Filter)



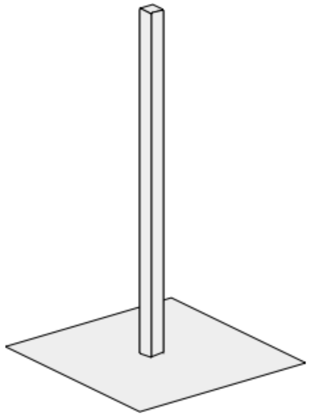
High-Pass Filter (Laplacian Filter)



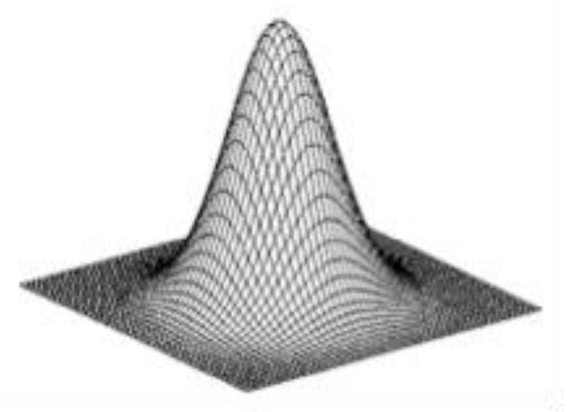
High-Pass Filter (Laplacian Filter)

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

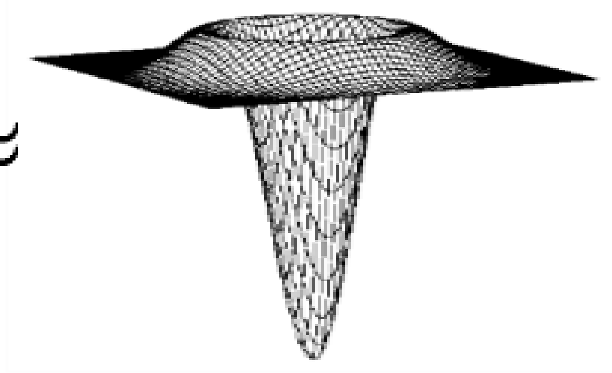
$$K_{\nabla^2} = -1 \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



-



$\approx$



unit **Gaussian** **Laplacian**

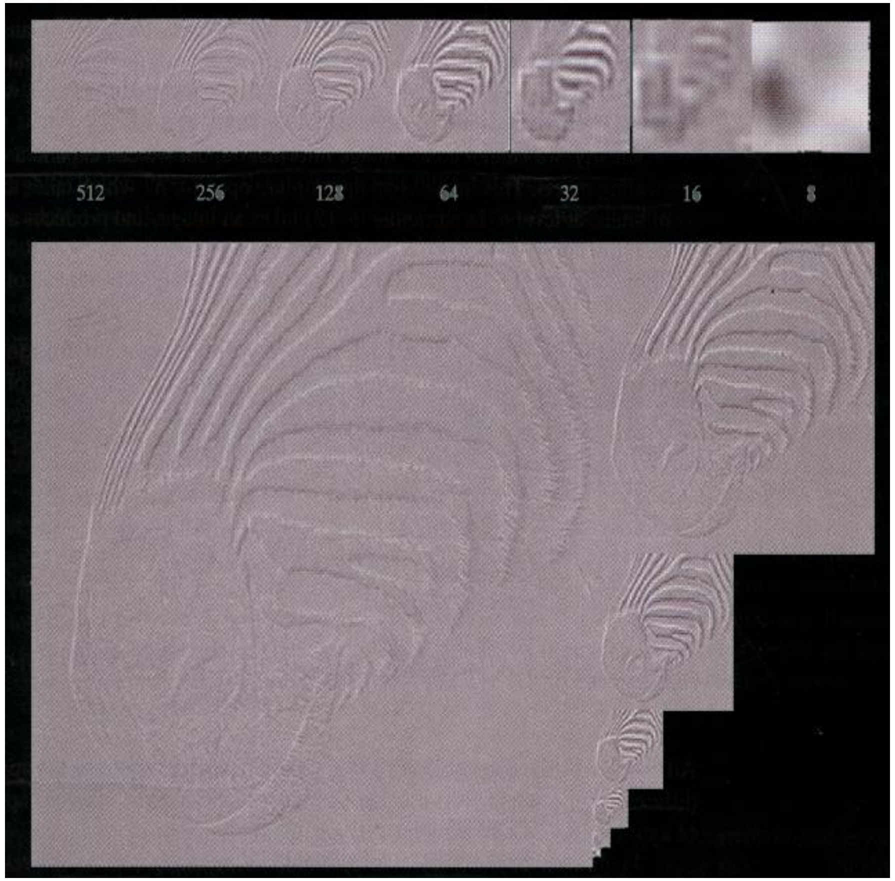
Image Pyramids (Scale Invariance)

Gaussian Pyramid

$$P_G^{n+1}(I) = \downarrow (G_s \otimes P_G^n(I))$$

$$P_G^1(I) = I$$

Laplacian Pyramid



$$P_L^n(I) = P_G^n(I) - \uparrow P_G^{n+1}(I)$$

$$P_L^n(I) = P_L^n(I) + \uparrow P_L^{n+1}(I) \rightarrow I$$

Deconvolution / Inverse Filtering

$$I(x,y) * K_s(x,y) = I'(x,y) \qquad \hat{I}(f_x,f_y) \cdot \hat{K}_s(f_x,f_y) = \hat{I}'(f_x,f_y)$$

Convolution in Spatial Domain…….is……Multiplication in Frequency Domain

Does that mean that
Deconvolution in Spatial Domain
is Division in Frequency Domain?

$$\hat{I}(f_x,f_y) = \frac{\hat{I}'(f_x,f_y)}{\hat{K}_s(f_x,f_y)}$$

Deconvolution / Inverse Filtering

This is simplified and does not consider Noise.

Division by small Values in Frequency Domain leads to Ringing Artefacts in Spatial Domain.

Better is to apply regularized Techniques: Wiener Filter, Regularized Filter, Lucy-Richardson Algorithm, Blind Deconvolution

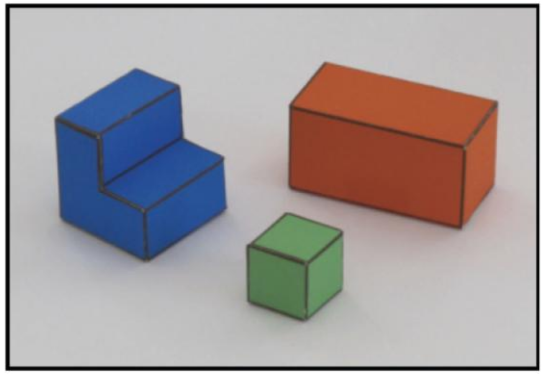
Inverse Filtering

Original Image

After Convolution
(Motion Blur)

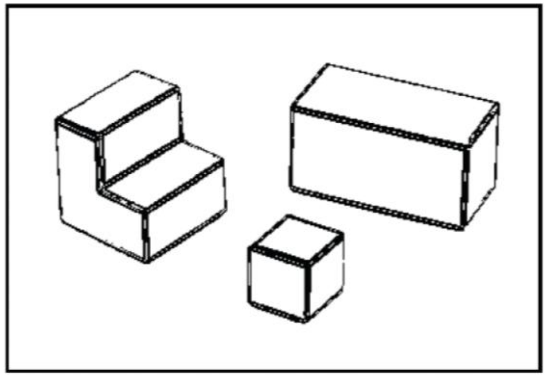
After Inverse Filtering

Recap: Gradients

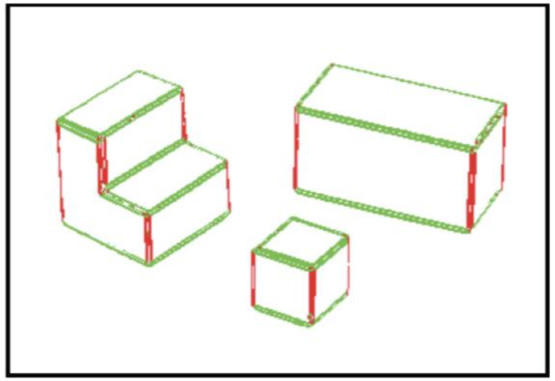


$$K_{Gx} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$K_{Gy} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



Using E(x,y)



Using θ(x,y)

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

Approximation image derivative:

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

Edge strength $\quad E(x,y) = |\nabla \mathbf{I}(x,y)|$

Edge orientation: $\quad \theta(x,y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I}/\partial y}{\partial \mathbf{I}/\partial x}$

Edge normal: $\quad \mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$

Spatial vs. Gradient Domain

Laplacian!

$$\mathrm{div}(\nabla I) = \frac{\partial I_x}{\partial x} + \frac{\partial I_y}{\partial y} = I_{xx} + I_{yy} = \nabla^2 I$$

Gradient Domain

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

$$\mathrm{curl}(\nabla I) = \frac{\partial I_y}{\partial x} - \frac{\partial I_x}{\partial y} = I_{yx} - I_{xy}$$

I_x I_y We can consider this as Coefficients of a Vectorfield.

Spatial vs. Gradient Domain

Laplacian!

$$\mathrm{div}(\nabla \mathrm{I}) = \frac{\partial \mathrm{I}_x}{\partial x} + \frac{\partial \mathrm{I}_y}{\partial y} = \mathrm{I}_{xx} + \mathrm{I}_{yy} = \nabla^2 \mathrm{I}$$

$$\mathrm{curl}(\nabla \mathrm{I}) = \frac{\partial \mathrm{I}_y}{\partial x} - \frac{\partial \mathrm{I}_x}{\partial y} = \mathrm{I}_{yx} - \mathrm{I}_{xy}$$

Gradient Domain

Image gradient:

$$\nabla \mathbf{I} = \left(\underset{\mathrm{I}_x}{\frac{\partial \mathbf{I}}{\partial x}}, \underset{\mathrm{I}_y}{\frac{\partial \mathbf{I}}{\partial y}} \right)$$

We can consider this as Coefficients of a Vectorfield.

Derivative / Integral

Spatial Domain

$$\mathrm{I} = \int \nabla \mathrm{I}$$

Integrating this Vectorfield results in the original Image as long as the Vectorfield has zero Curl.

Spatial vs. Gradient Domain

Laplacian!

$$\mathrm{div}(\nabla I) = \frac{\partial I_x}{\partial x} + \frac{\partial I_y}{\partial y} = I_{xx} + I_{yy} = \nabla^2 I$$

$$\mathrm{curl}(\nabla I) = \frac{\partial I_y}{\partial x} - \frac{\partial I_x}{\partial y} = I_{yx} - I_{xy}$$

Gradient Domain

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

I_x I_y We can consider this as Coefficients of a Vectorfield.

Derivative / Integral

Processing the Vectorfield is called Gradient Domain Processing.

Spatial Domain

$$I = \int \nabla I$$

Integrating this Vectorfield results in the original Image as long as the Vectorfield has zero Curl. **In Practice: that does not work as the Vectorfields is no longer conservative anymore after processing it (it has non-zero Curl).**

Spatial vs. Gradient Domain

Laplacian!

$$\mathrm{div}(\nabla I) = \frac{\partial I_x}{\partial x} + \frac{\partial I_y}{\partial y} = I_{xx} + I_{yy} = \nabla^2 I$$

$$\mathrm{curl}(\nabla I) = \frac{\partial I_y}{\partial x} - \frac{\partial I_x}{\partial y} = I_{yx} - I_{xy}$$

Gradient Domain

Image gradient:

$$\nabla I = \left(\underbrace{\frac{\partial I}{\partial x}}_{I_x}, \underbrace{\frac{\partial I}{\partial y}}_{I_y} \right)$$

We can consider this as Coefficients of a Vectorfield.

Derivative / Integral

Processing the Vectorfield is called Gradient Domain Processing.

Spatial Domain

$$I = \int \nabla I$$

Integrating this Vectorfield results in the original Image as long as the Vectorfield has zero Curl. **In Practice: that does not work as the Vectorfields is no longer conservative anymore after processing it (it has non-zero Curl).**

Instead: solve 2D Poisson Equation: $\nabla^2 I = \mathrm{div}(G)$

Examples

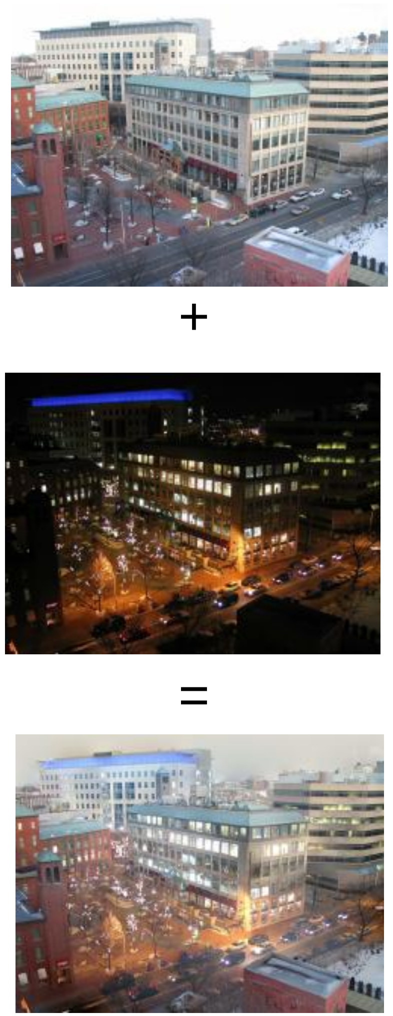
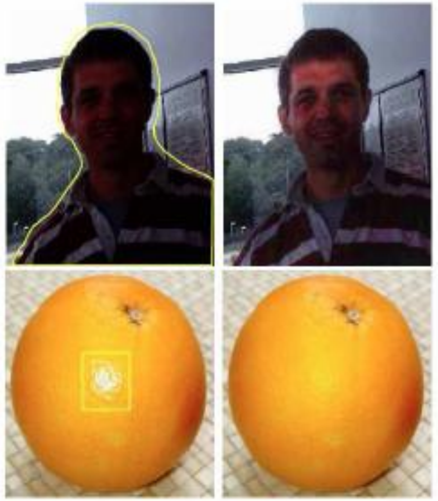


Image Fusion



Removing Reflections

HDR Compression

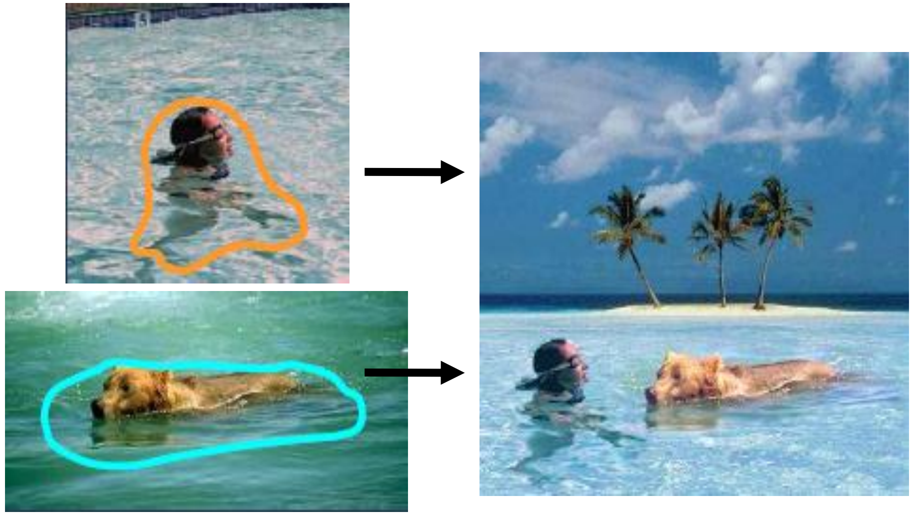
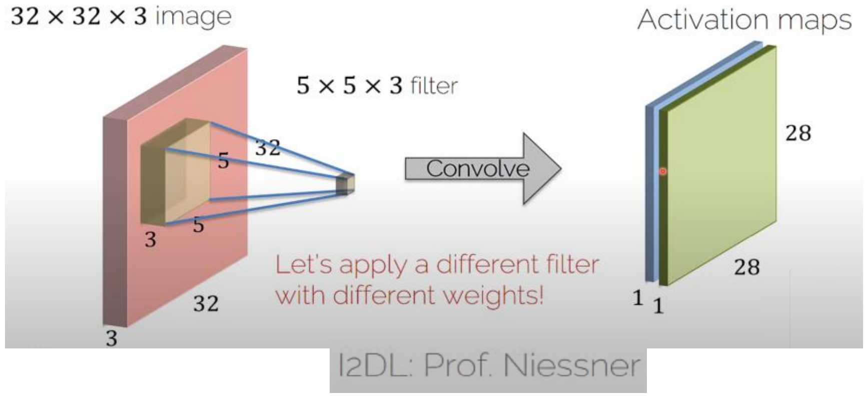
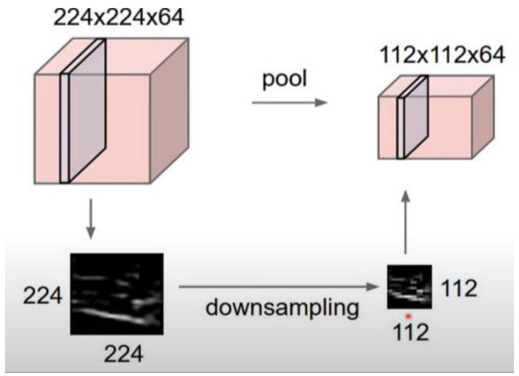
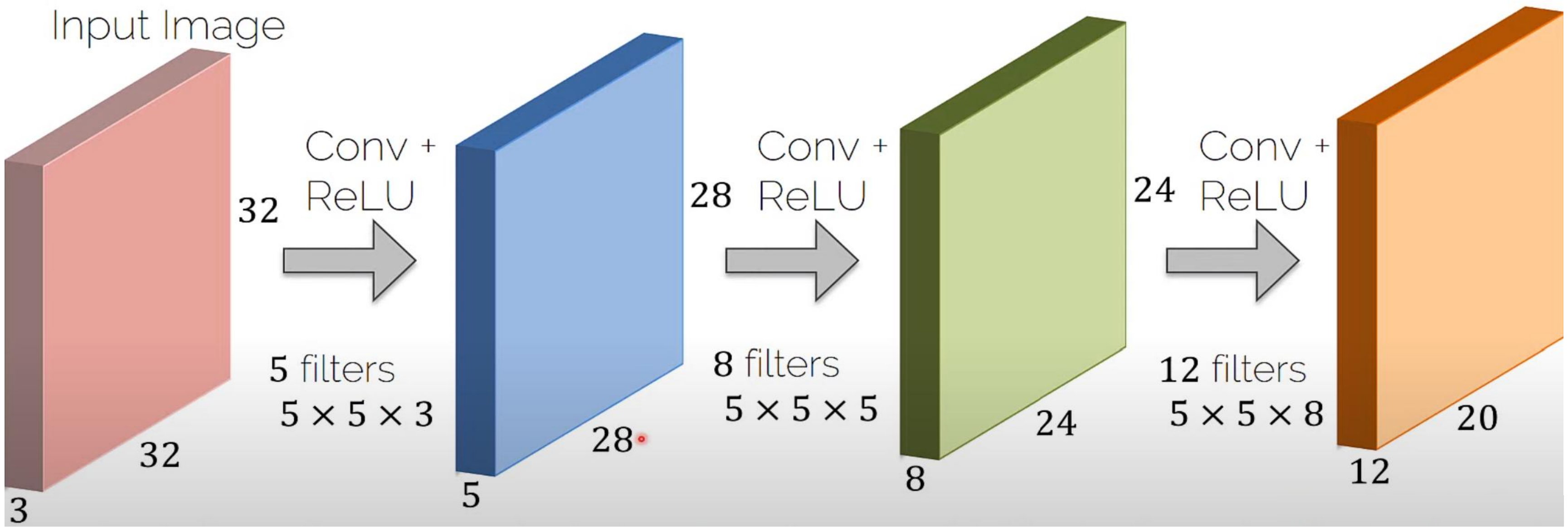


Image Stitching / Editing

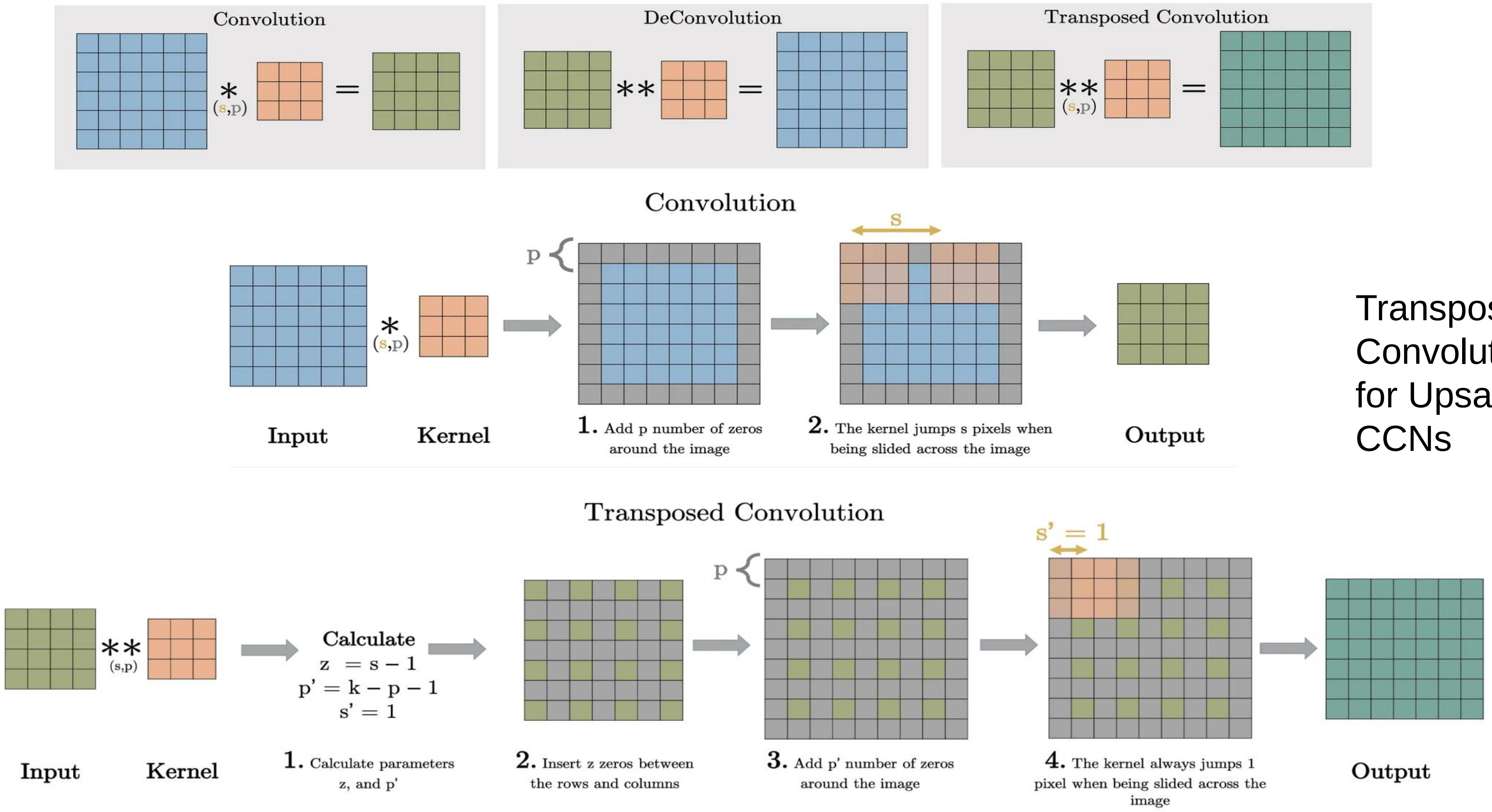
Convolutional Neural Networks (CNNs)







Transposed Convolution



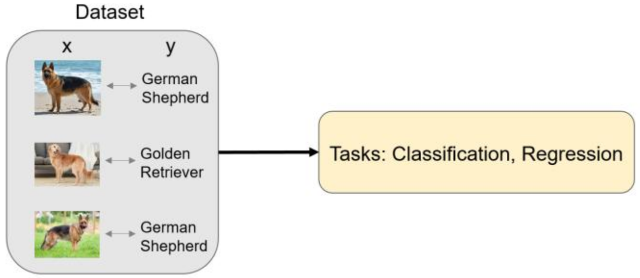
Transposed Convolution used for Upsampling in CCNs

Course Overview

CW	Topic	Date	Place	Lab
41	Introduction and Course Overview	07.10.2025	Zoom	Lab 1
42	Capturing Digital Images	14.10.2025	Zoom	Lab 2
43	Digital Image Processing	21.10.2025	Zoom	Assignment 1
44	Machine Learning	28.10.2025	Zoom	
45	Feature Extraction	04.11.2025	Zoom	Open Lab 1
46	Segmentation	11.11.2025	Zoom	Assignment 2
47	Optical Flow	18.11.2025	Zoom	Open Lab 2
48	Object Detection	25.11.2025	Zoom	Assignment 3
49	Multi-View Geometry	02.12.2025	Zoom	Open Lab 3
50	3D Vision	09.12.2025	Zoom	Assignment 4
3	Trends in Computer Vision	13.01.2026	Zoom	
4	Q&A	20.01.2026	Zoom	Open Lab 4
5	Exam	27.01.2026	HS1 (Linz), S1/S3 (Vienna), S5 (Bregenz)	
9	Retry Exam	24.02.2026	tba	

Next Week: Machine Learning

Supervised Learning

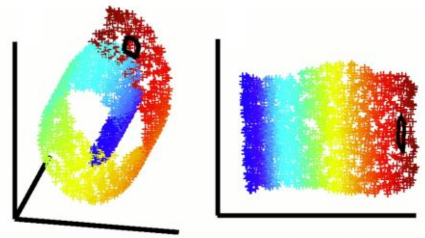


Use labeled Data for a Prediction Task.

JOHANNES KEPLER UNIVERSITY LINZ

Dimensionality Reduction

Given Data Points in *d* Dimensions, convert them to Data Points in *k<d* Dimensions with minimal loss of Information.

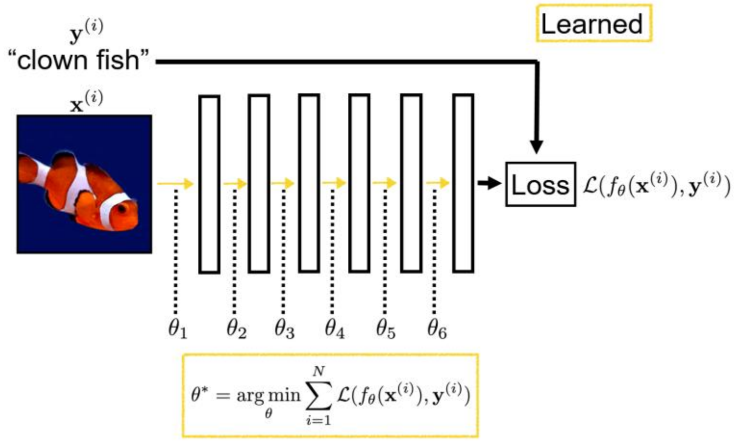


Original Data in 3D

Same data in 2D

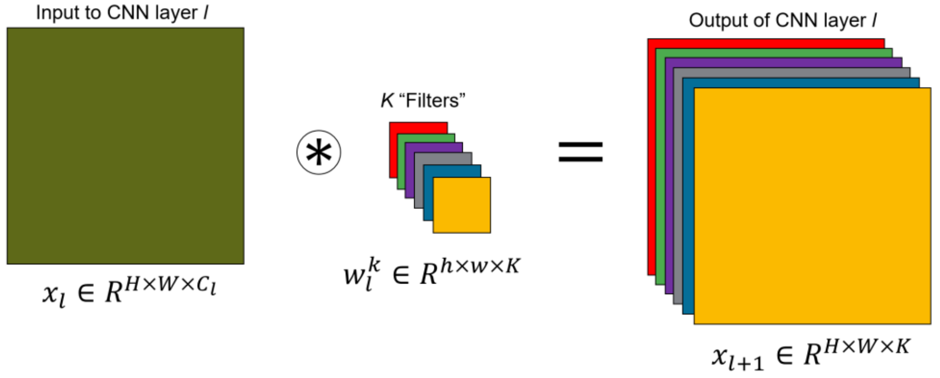
JOHANNES KEPLER UNIVERSITY LINZ

Deep Learning



JOHANNES KEPLER UNIVERSITY LINZ

Convolutional Neural Networks (CNNs)



JOHANNES KEPLER UNIVERSITY LINZ

JOHANNES KEPLER UNIVERSITY LINZ

Thank You

JOHANNES KEPLER
UNIVERSITY LINZ