

Computer Vision



Lecture 1: Introduction and Course Overview

Oliver Bimber

New Computer Vision Lecture and Labs

• Lecture

- Online only (+ Recordings and Slides)
- No Attendance required (but highly recommended)
- ML Focus
- Application-Oriented rather than Theory-Oriented (reduced Math)

• Labs

- Online only (+ Recordings, Slides, Code of Assignments and Model-Solutions)
- No Attendance required (but highly recommended)
- 2 Onboarding Labs (Basic Image Handling and Machine Learning in Python)
- 4 Individual Assignments related to Lecture Topics
- 2 Open-Labs during Assignments (for Q&A and Problems)

Who We Are

ICG

The Institute
We do Visual Computing!

COMPUTER VISION & GRAPHICS

Computational Imaging
Machine Learning
Intelligent Optics

Oliver Bimber

VISUAL DATA SCIENCE

Visualization
Visual Analytics
Explainable AI

Marc Streit

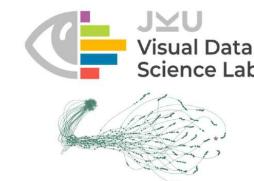
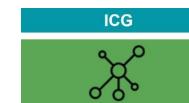
GAME COMPUTING

Games User Research & Analytics
Gameplay Visualization
AI-based Playtesting

Günter Wallner

Website: www.jku.at/cg

Who We Are

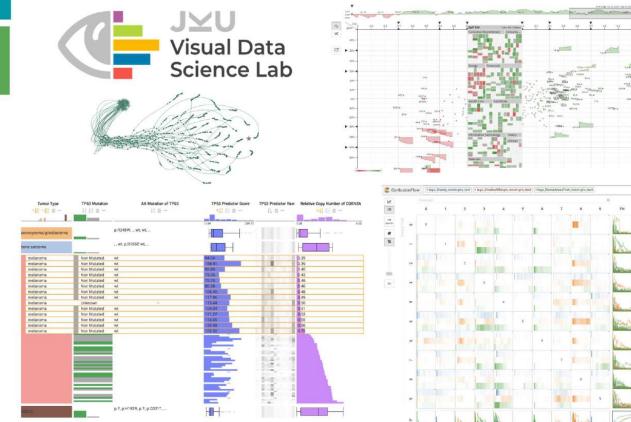


Research Areas:

- Visualization
- Visual Analytics
- Explainable AI
- Biological Data Vis



marc.streit@jku.at



Who We Are



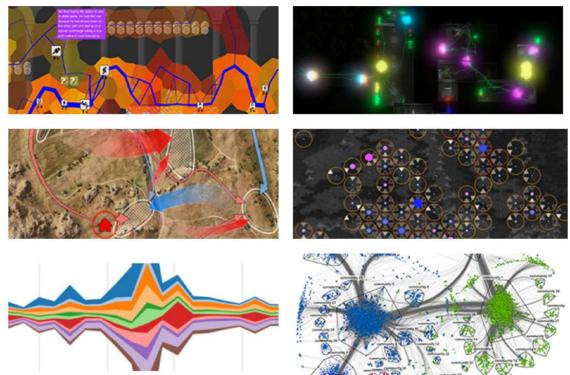
Research Areas:

Games User Research
Games Analytics
Gameplay Visualization
AI-based Playtesting

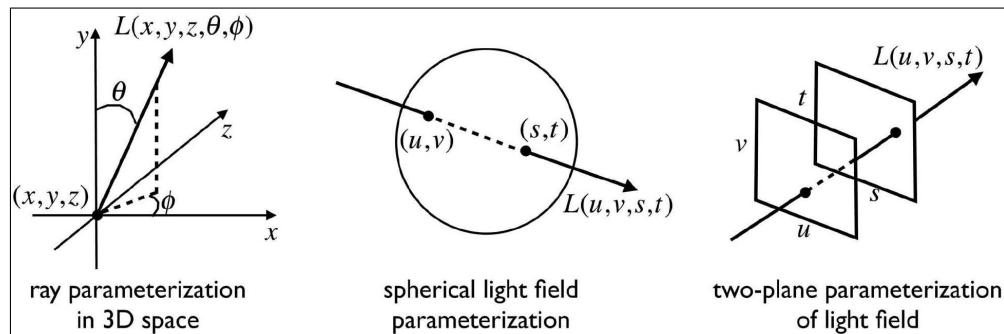


quenter.wallner@jku.at

JYU JOHANNES KEPLER
UNIVERSITY LINZ



Light Fields in a Nutshell



JYU JOHANNES KEPLER
UNIVERSITY LINZ

7

Who We Are



Basic Research: Light Fields
Displays
Computational Imaging & Optics
Computer Vision / Image Processing

Areas:

Aerial Imaging, Microscopy, Thin-Film Sensing, 3D Displays, Autonomous Drones and Swarms

Applications:
Search & Rescue, Wildlife Observation, Wildfire Detection, Forest Ecology, Agriculture, Optogenetics, User Interfaces

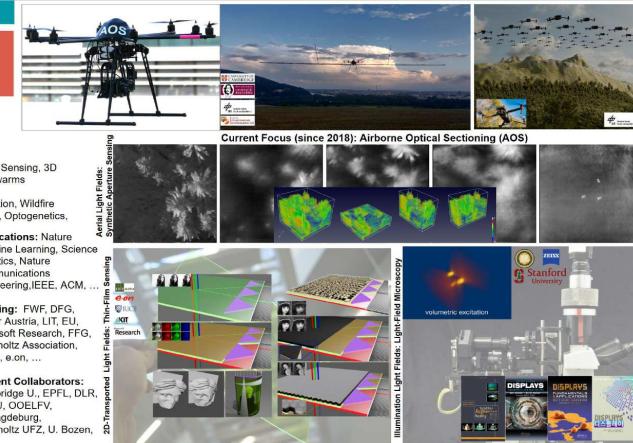
Publications: Nature, Machine Learning, Science, Robotics, Nature Communications, Engineering, IEEE, ACM, ...

Funding: FWF, DFG, Upper Austria, LIT, EU, Microsoft Research, FFG, Helmholtz Association, Nokia, e.on, ...

Current Collaborators:
Cambridge U, EPFL, DLR, BOKU, OÖELFV, U. Magdeburg, Helmholtz UFZ, U. Bozen, ...

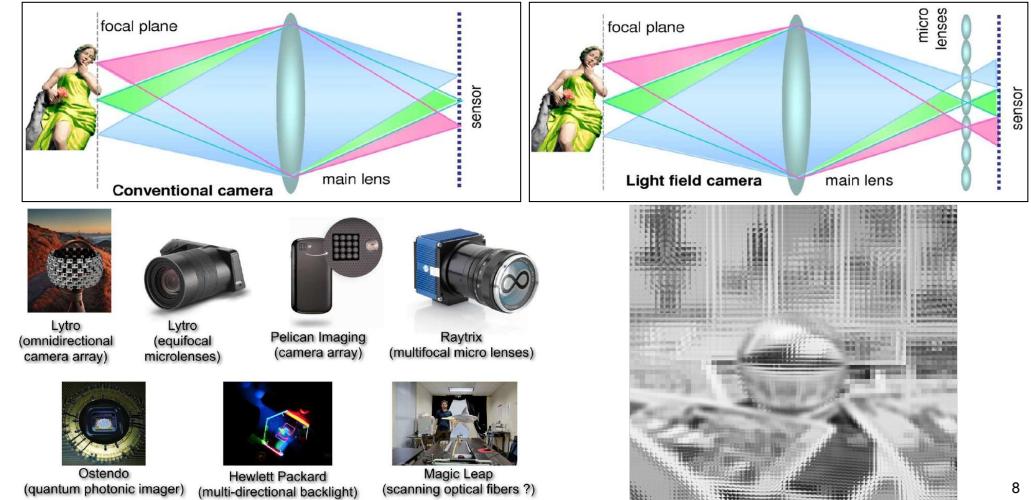
oliver.bimber@jku.at

JYU JOHANNES KEPLER
UNIVERSITY LINZ



6

Light Fields in a Nutshell



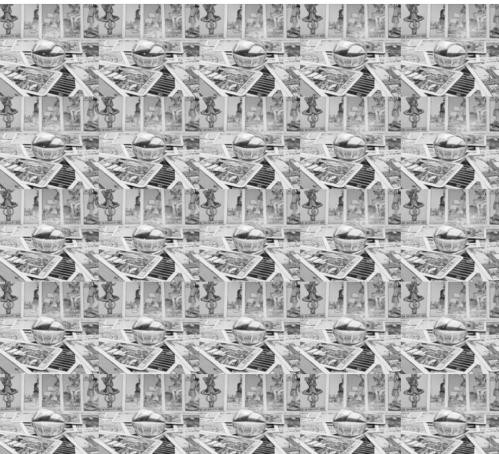
8

Light Fields in a Nutshell

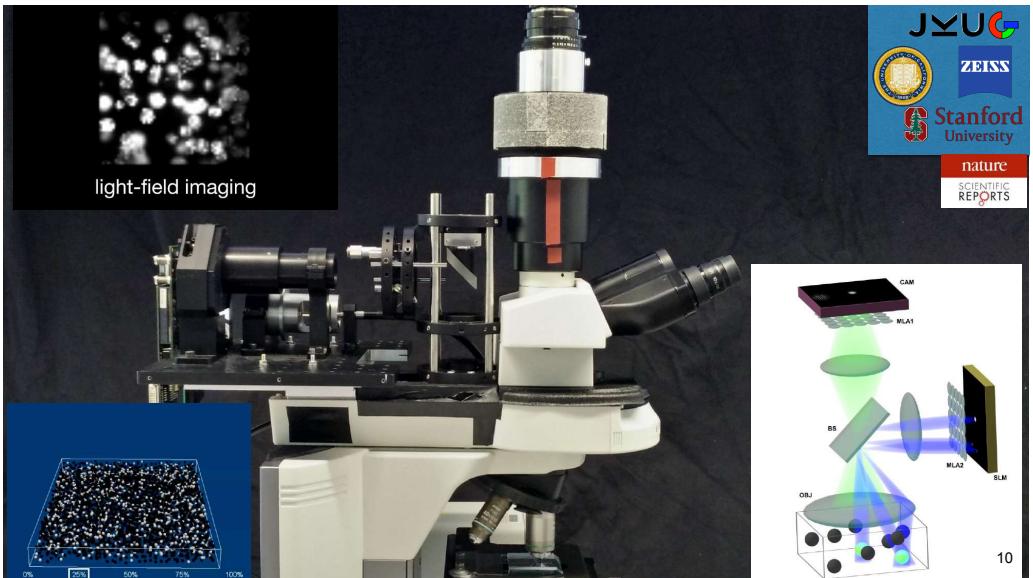


JKU JOHANNES KEPLER
UNIVERSITY LINZ

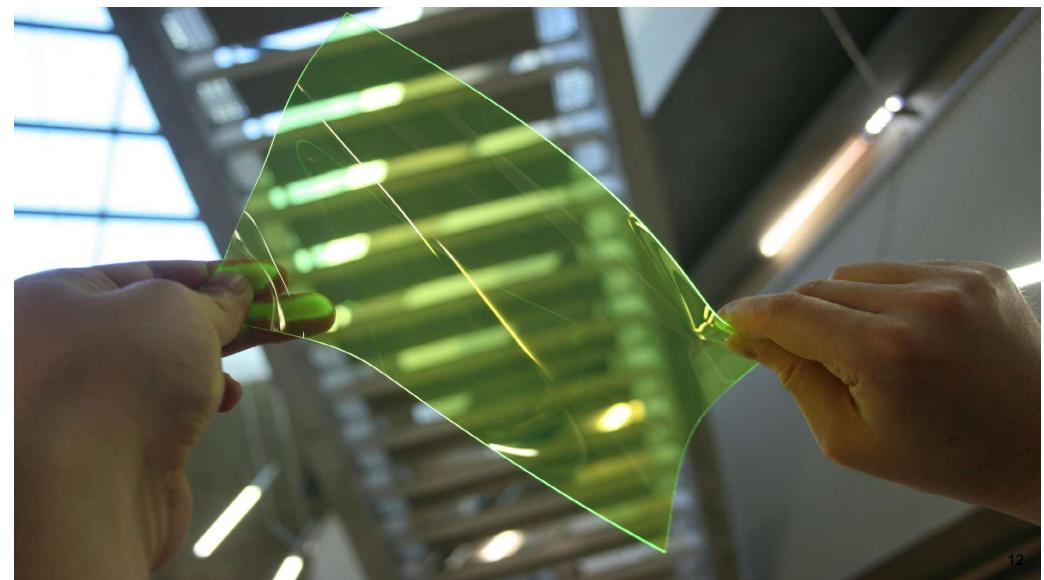
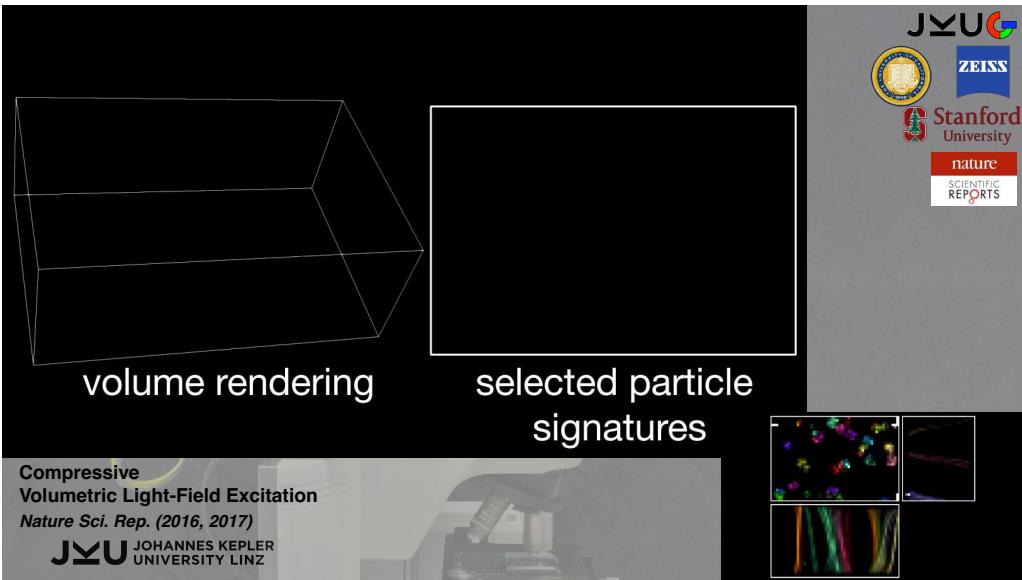
Light field: spatial resolution outside,
angular resolution insight.



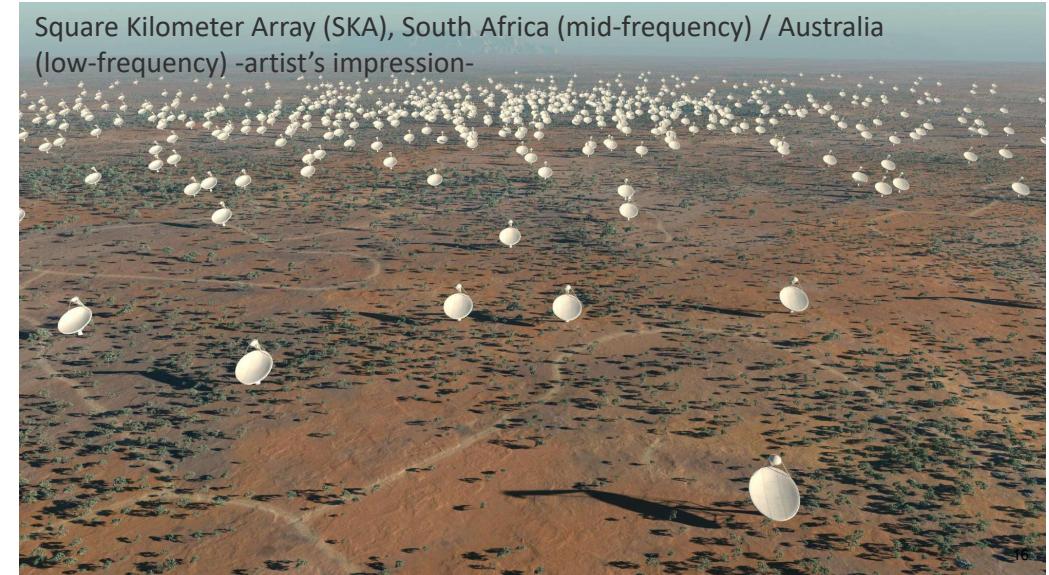
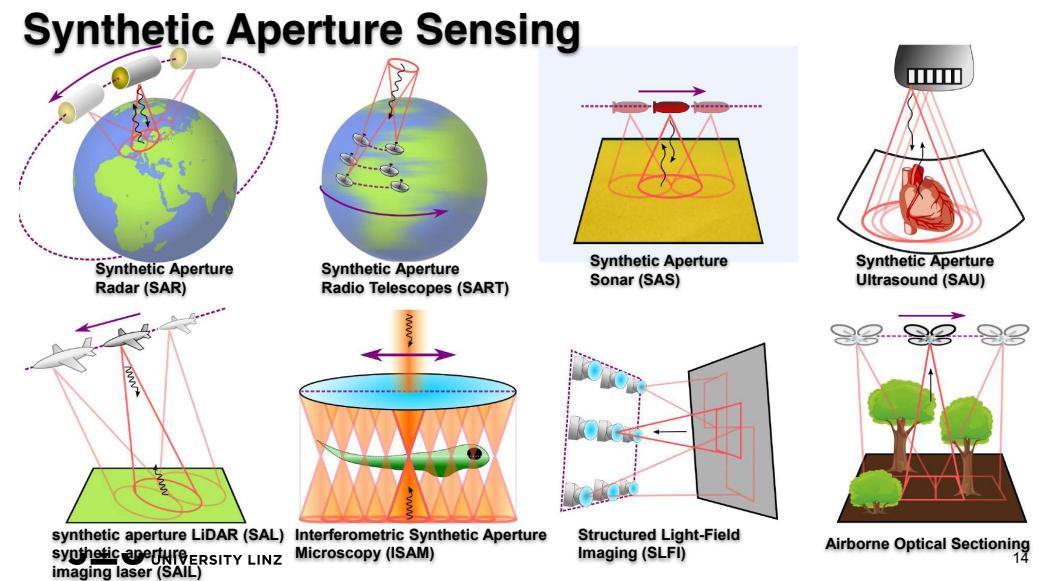
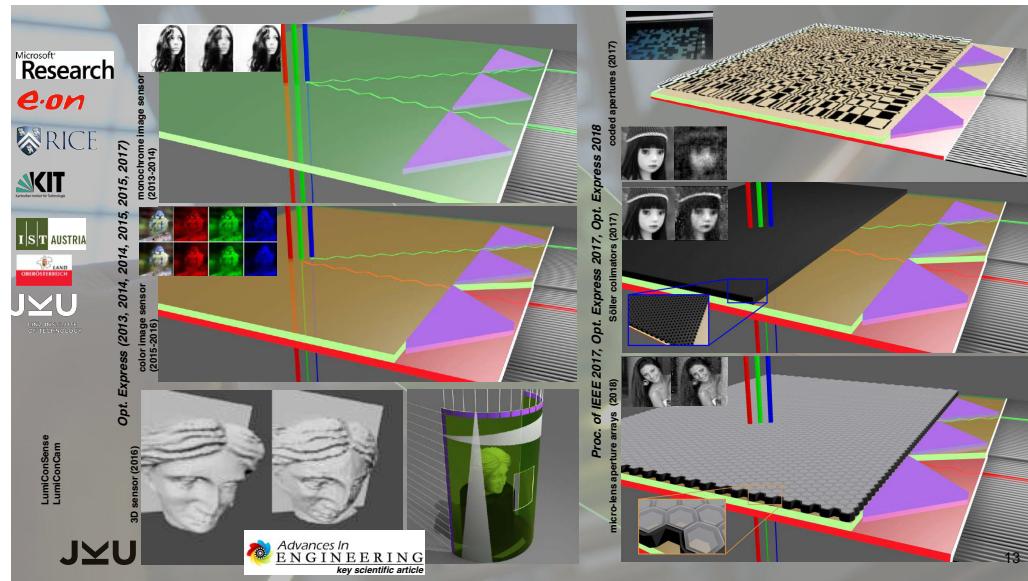
Light field: angular resolution outside,
spatial resolution insight.



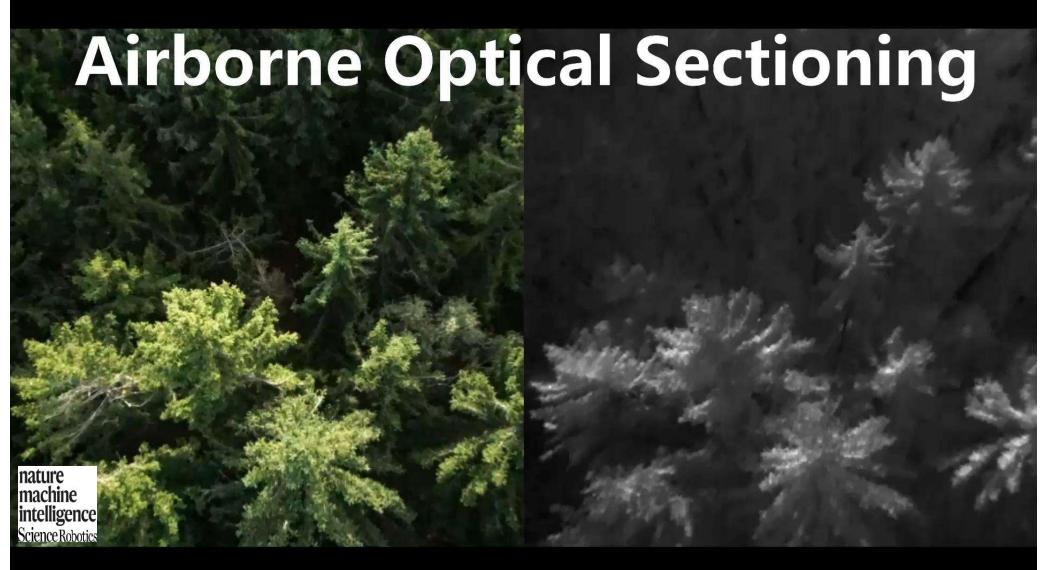
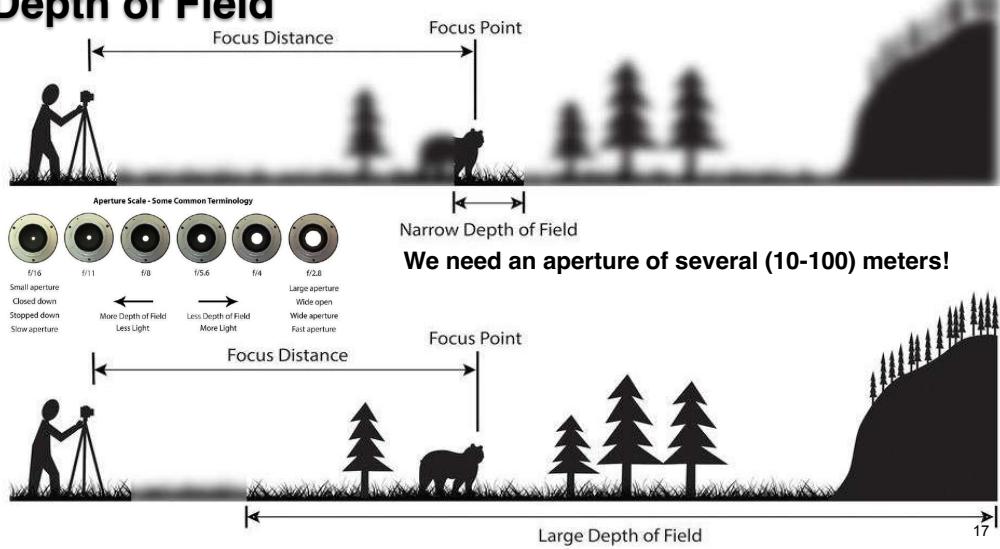
10

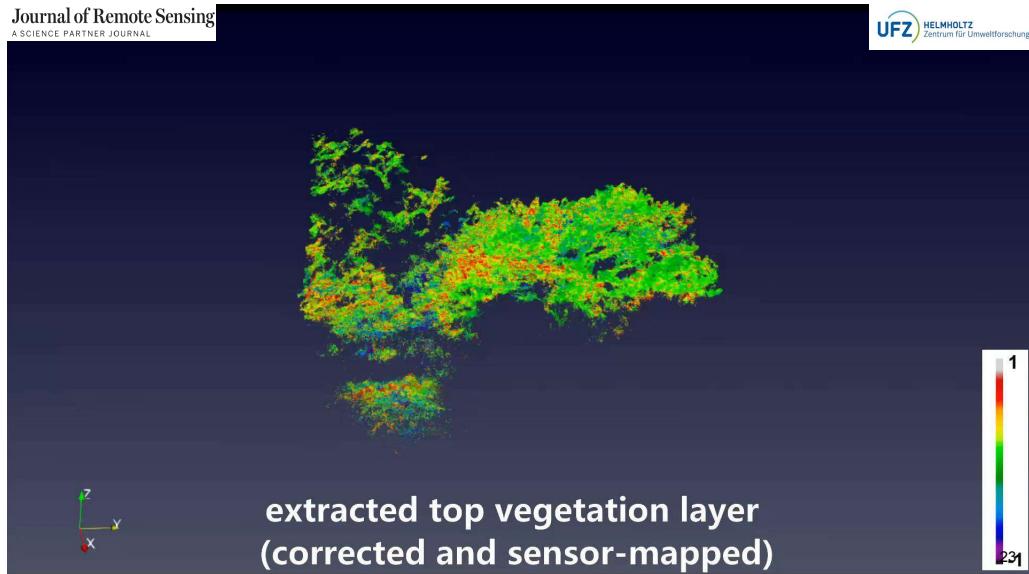
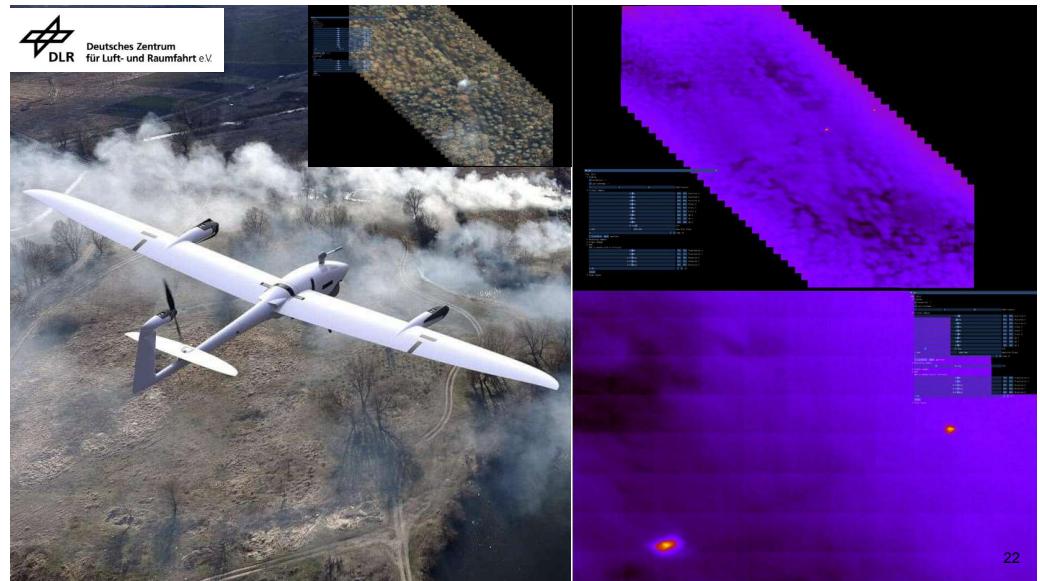
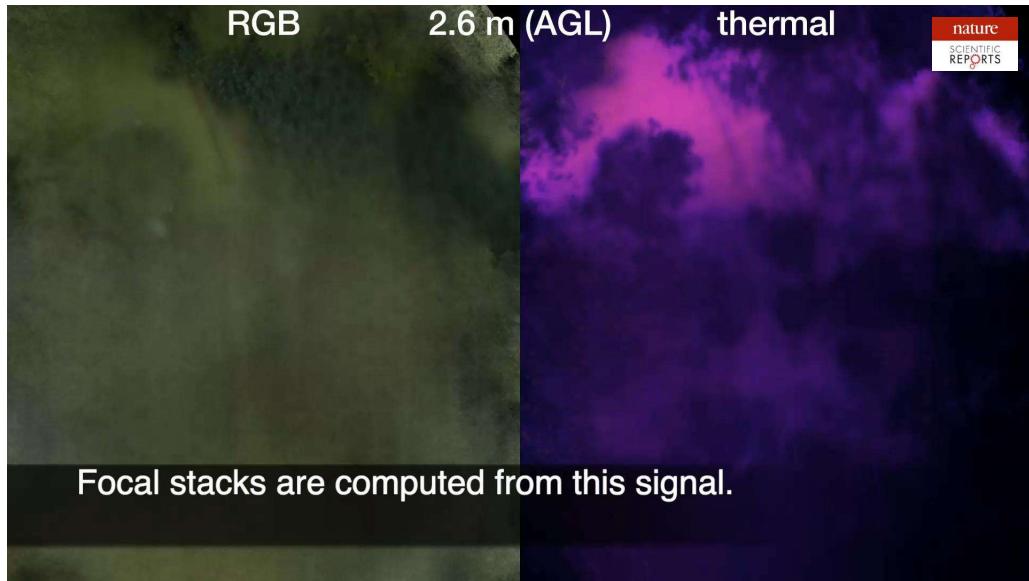


12



Depth of Field

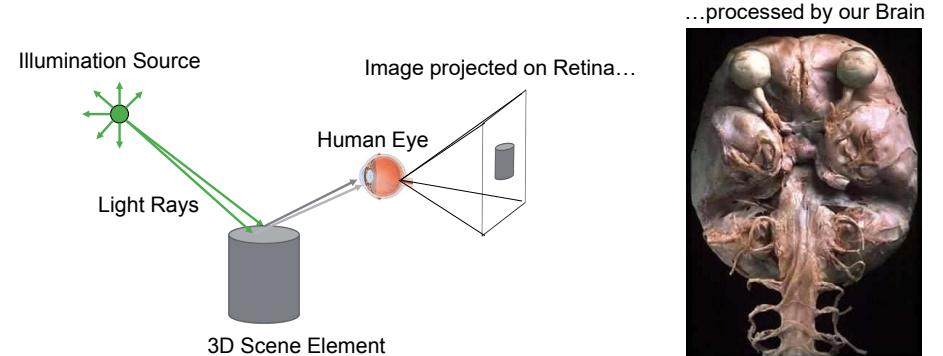






Computer Vision – where do we start?

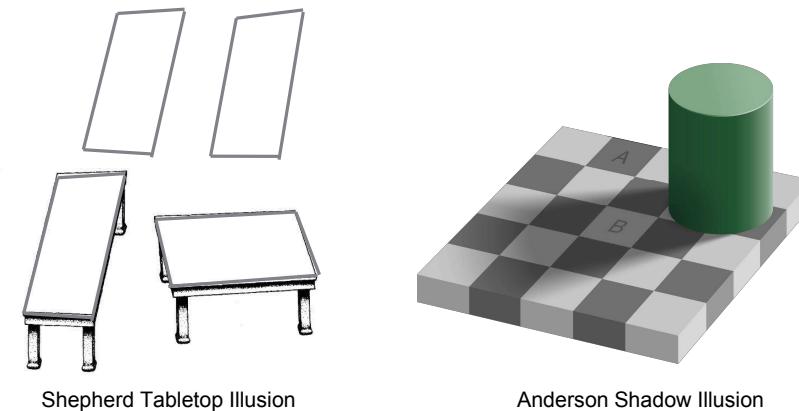
How Humans see the World (in Principle)



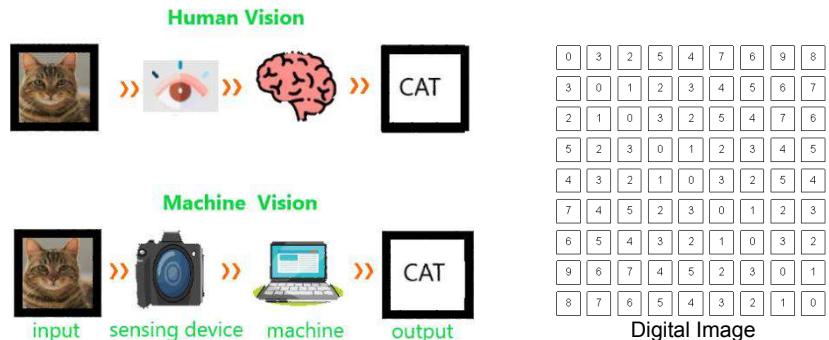
Our Brain is really good at using learned Context



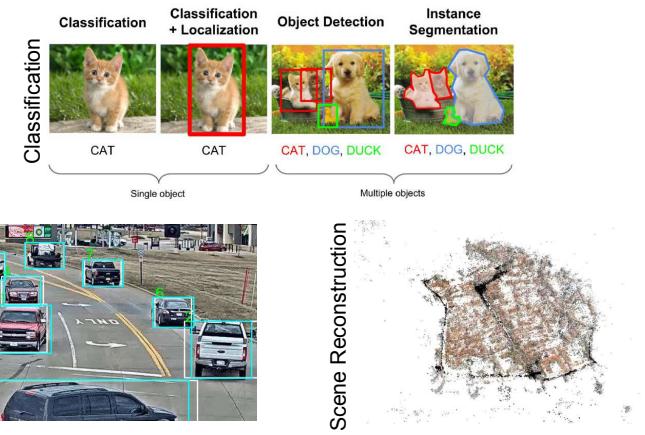
Our Brain is really good at using learned Context



How Computers see the World (in Principle)



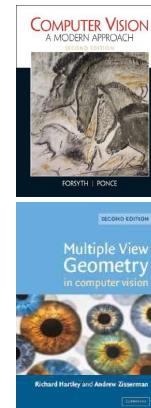
Classical Computer Vision Tasks



For Computers, Vision is not that easily (yet)



First Decades of Computer Vision

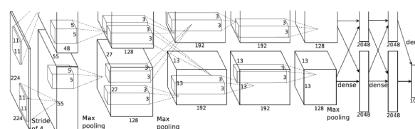


- **1960s and 70s: The “early Years”**
 - Community gets over its Blocks World Phase
 - Canonical recovery Problems are defined and initial Approaches are proposed
 - Ambitious Scene understanding Approaches flower briefly and prematurely
 - Marr’s Book sums up Progress to Date
- **1980s and 90s: The “middle ages”**
 - Progress on many Fronts
 - The Field goes through its Geometric Recognition Phase
 - Multi-view Geometry matures and becomes useful, as summarized in the Hartley & Zisserman book
 - The Field stops being afraid of pixels and discovers Data and Classifiers
- **2000s and 2010s: The “early modern era”?**
 - Local Features “solve” Structure from Motion and Instance Recognition
 - Generic Category Recognition and Detection become central Problems
 - The Field becomes driven by Datasets and Benchmarks

Where are we now?

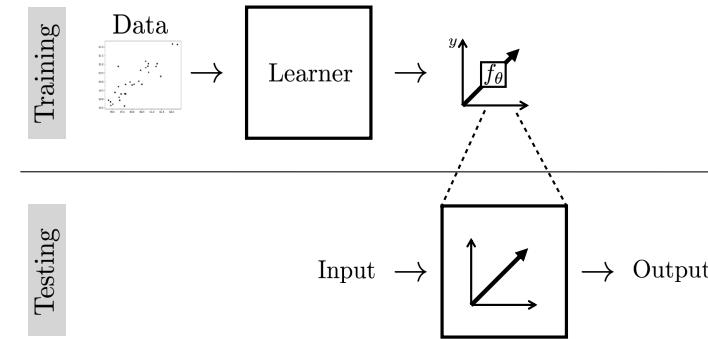


Machine Learning= Data-driven Intelligence

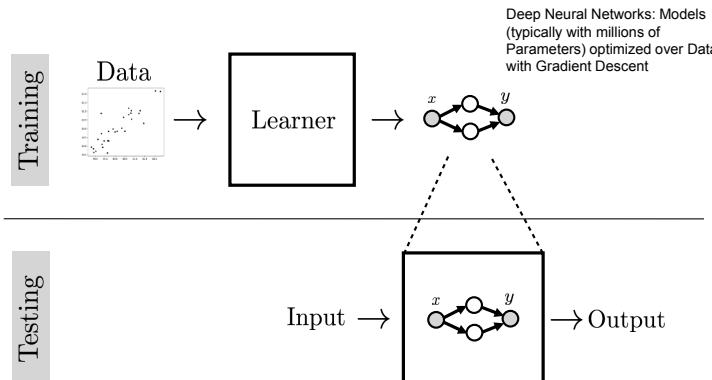


2012: AlexNet

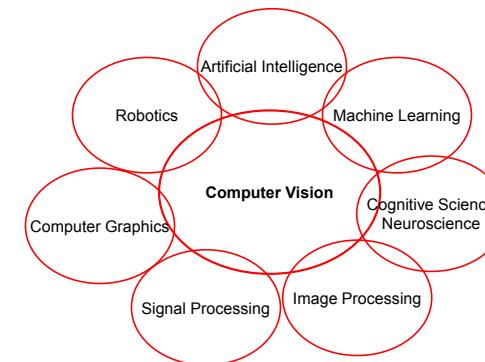
Learning from Data



Learning from Data



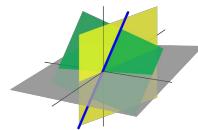
Related Fields



Required Basic Knowledge



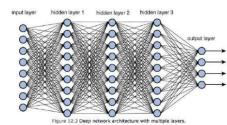
Python



Linear Algebra

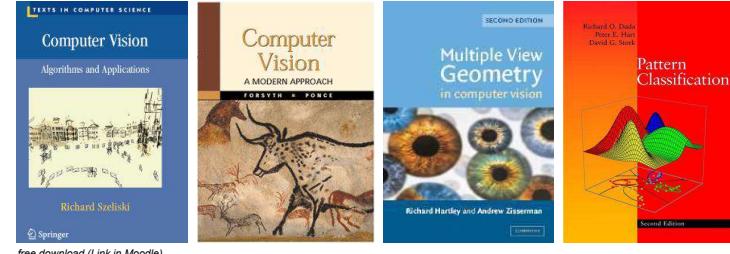


Machine Learning



Neural Networks

Recommended Text Books



Material in Slides is partially compiled from multiple Sources: Lecture Material of other Universities (TU Munich, RICE University, MIT, Stanford University); Supplementary Material of Text Books (Computer Vision: Algorithms and Applications, Computer Vision: A Modern Approach); various Internet Sources.

Course Organization

Lectures (3 ETCS)

- 12 Lecture Appointments
- Zoom only
- Live: Tuesdays, 12:00-13:30
- Recordings and Slides available (Moodle)

Labs and Assignments (1.5 ETCS)

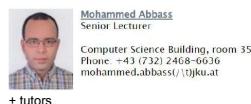
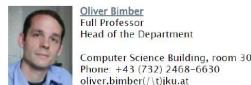
- Basic Python and Machine Learning Skills required
- 2 Onboarding Labs (Zoom only)
- 4 Assignments (related to Lecture Topics)
- 4 Open Labs (Zoom only / Hybrid)
- All Labs and Open Labs Repeated at various Days / Times per Week
- Recordings, Code, Model-Solutions, Slides available (Moodle)

Grading

- Lecture: Moodle Exam / Retry-Exam (100%)
- Lab: Each Assignment 25% (4x = 100%)

Attendance

- No Attendance required in Lectures (but recommended, questions will be addressed in Lectures)
- Physical Presence required for Exams
- Onboarding Labs recommended to refresh required Programming Skills
- Open Labs recommended for addressing Problems and Questions



+ tutors

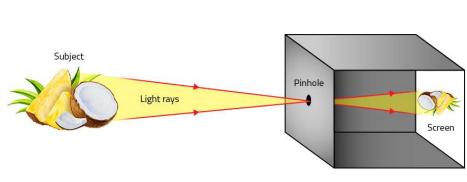
Computer Vision



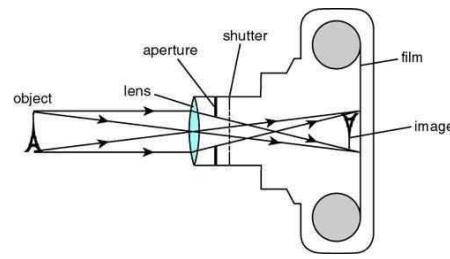
Lecture 2: Capturing Digital Images

Oliver Bimber

2D Imaging

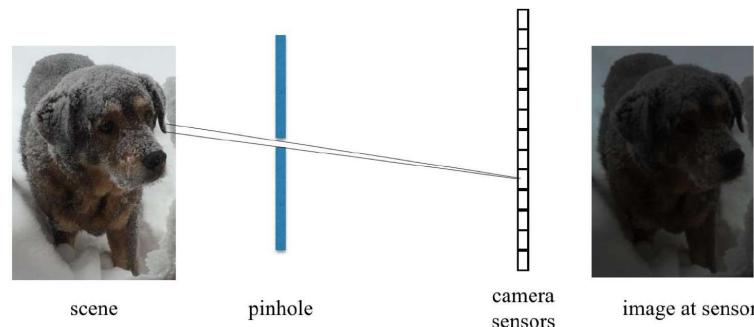


Pinhole Camera (Camera Obscura)

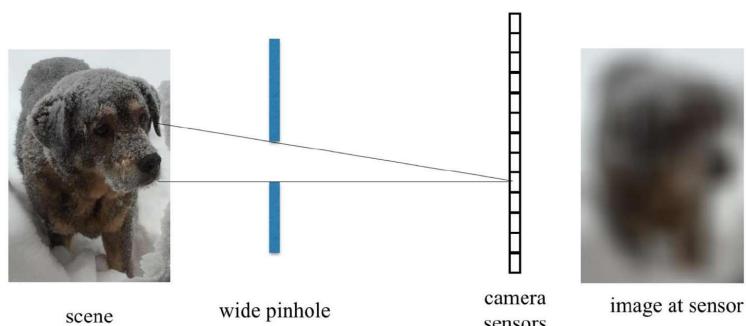


Analog Camera

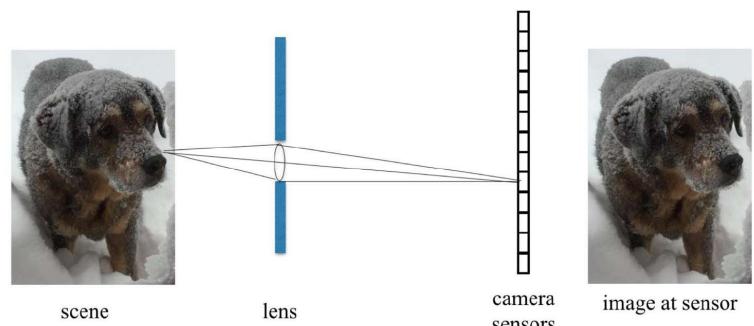
Drawbacks of Pinhole Cameras



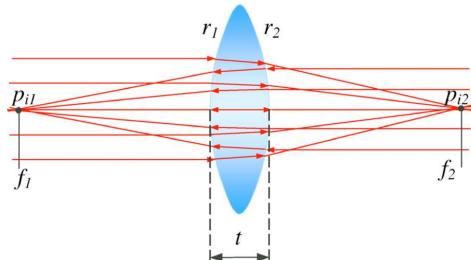
Drawbacks of Pinhole Cameras



Using Lenses to collect and focus Light



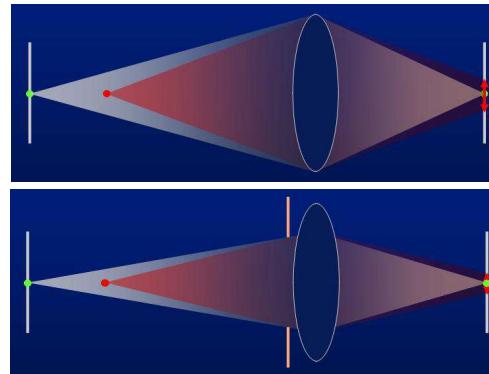
Lenses



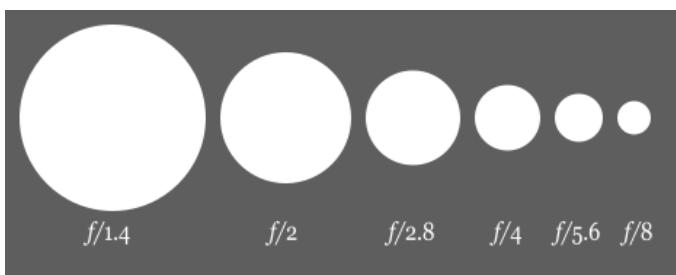
$$\frac{1}{f} = (\eta_2 - 1) \left(\frac{1}{r_1} - \frac{1}{r_2} \right) + \frac{(\eta_2 - 1)^2}{\eta_2} \frac{t}{r_1 r_2}$$

r_1 and r_2 are the two Surface Radii, and t the central Thickness

Depth of Field



f-number

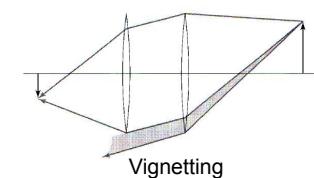
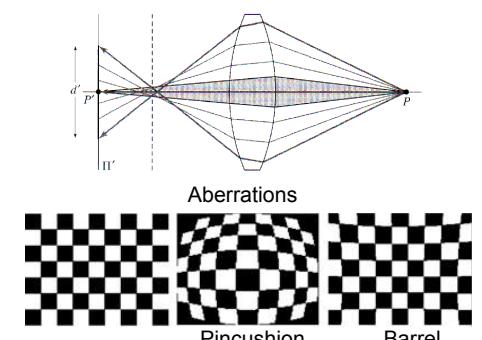
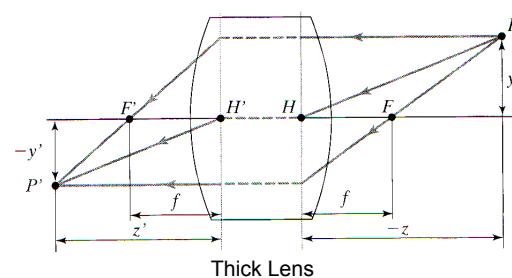


Focal Length /
Lens (Aperture) Diameter

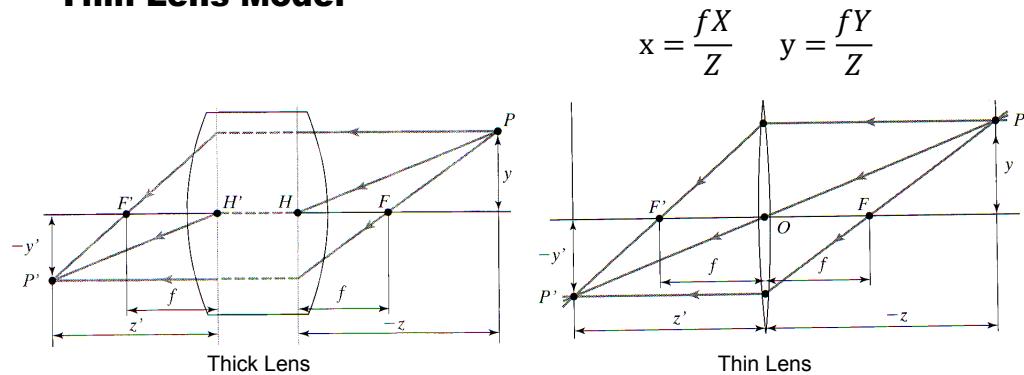
$$f / \# = N = \frac{f}{d}$$

For instance, f/16 means that the Focal Length is 16 times the Aperture Diameter (N=16).

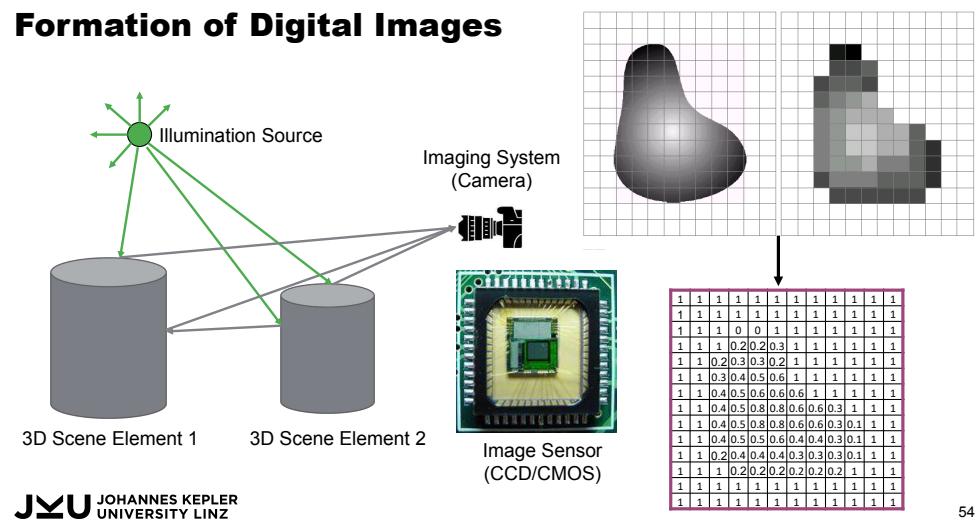
Thick Lenses



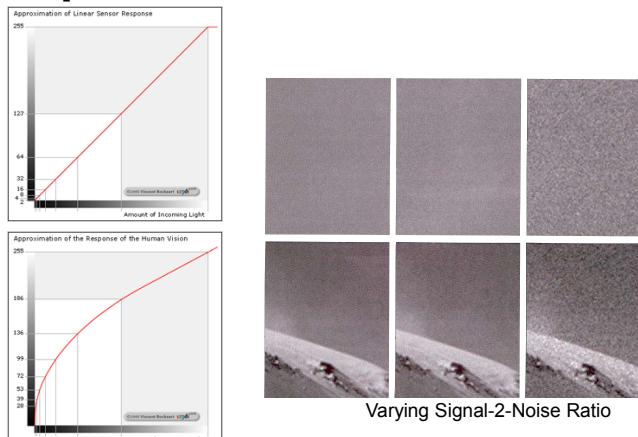
Thin-Lens Model



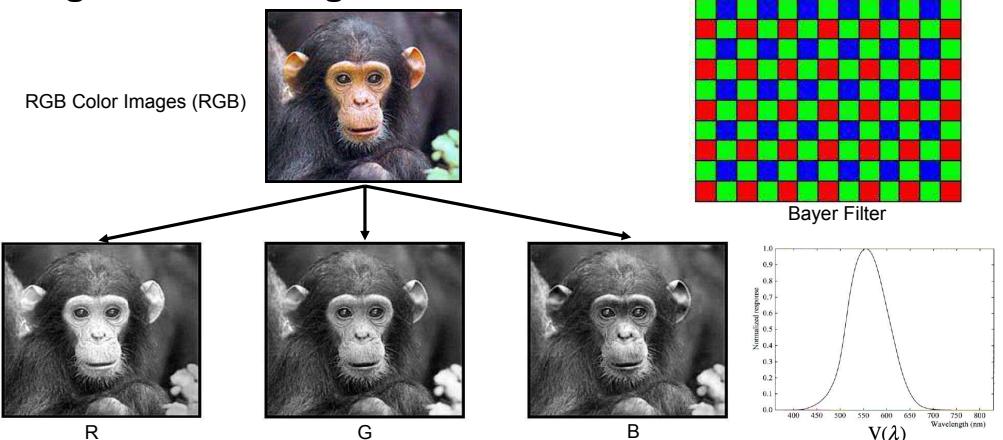
Formation of Digital Images



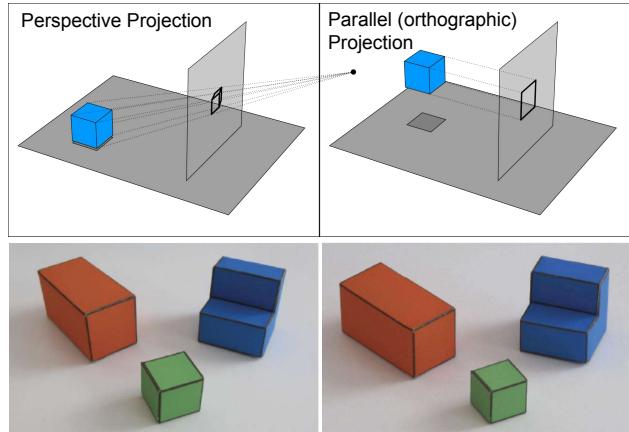
Sensor Response and Noise



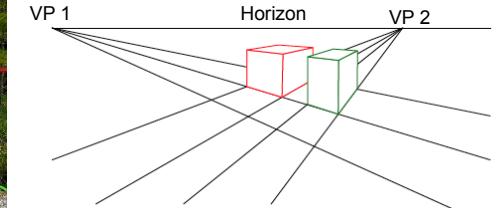
Digital Color Images



Projection

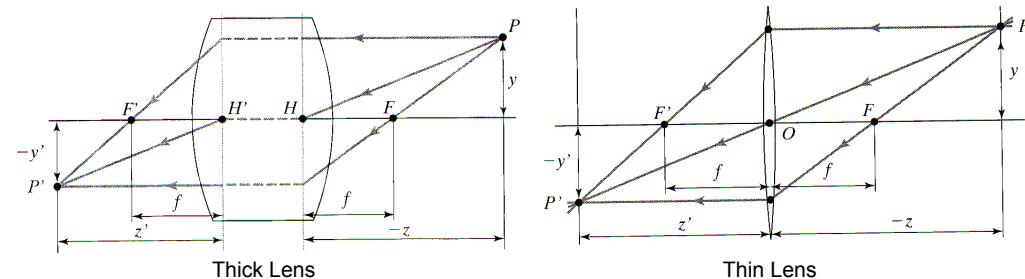


Vanishing Points



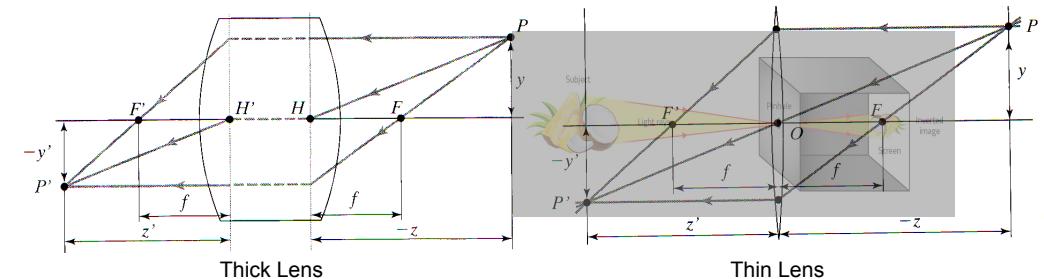
Recap: Thin-Lens Model

$$x = \frac{fX}{Z} \quad y = \frac{fY}{Z}$$

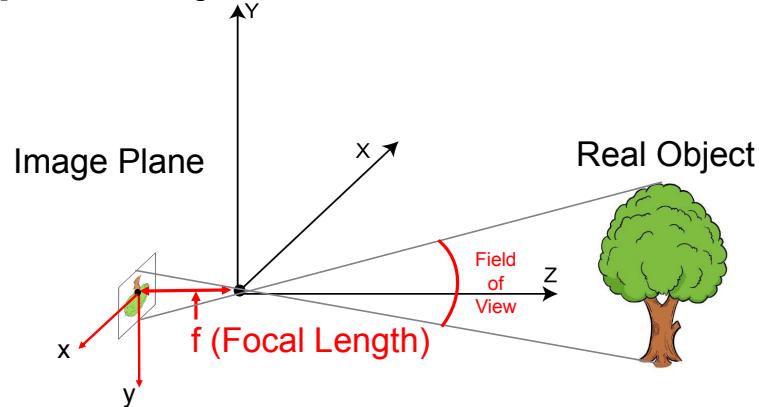


Recap: Thin-Lens Model

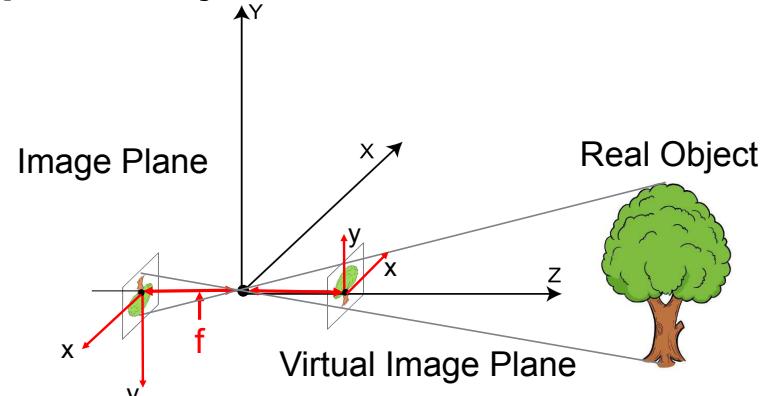
$$x = \frac{fX}{Z} \quad y = \frac{fY}{Z}$$



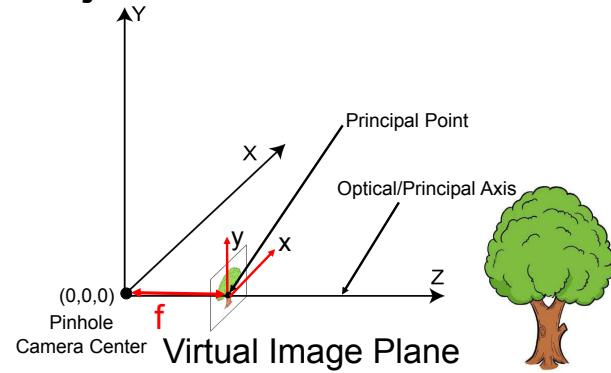
Perspective Projection



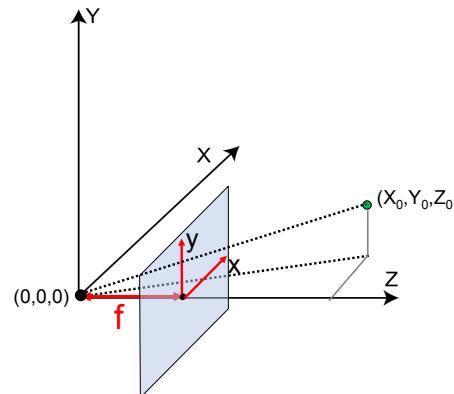
Perspective Projection



Perspective Projection



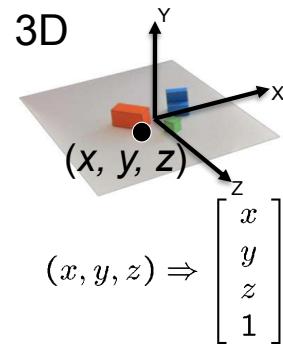
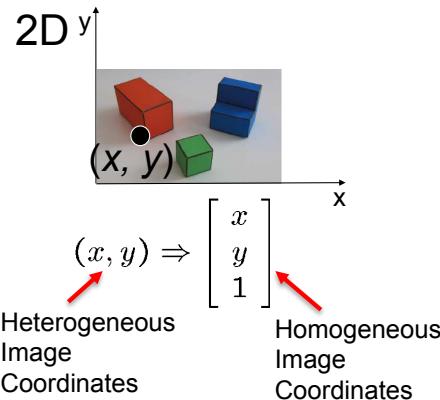
Perspective Projection



$$\begin{array}{l} \rightarrow x_0 = f X_0 / Z_0 \\ \rightarrow y_0 = f Y_0 / Z_0 \end{array}$$

The figure consists of two small diagrams. The top diagram shows a triangle with vertices at $(0,0,0)$, $(x_0, 0, Z_0)$, and $(0, y_0, Z_0)$. The horizontal distance between the first two vertices is labeled f . The bottom diagram shows a triangle with vertices at $(0,0,0)$, $(x_0, 0, Z_0)$, and $(0, y_0, Z_0)$. The vertical distance between the first two vertices is labeled f .

Homogeneous Coordinates



Homogeneous Coordinates

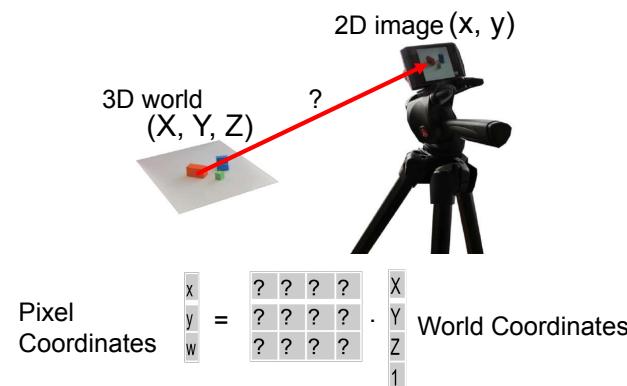
From Heterogeneous to Homogeneous:

$$(x, y) \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

From Homogeneous to Heterogeneous:

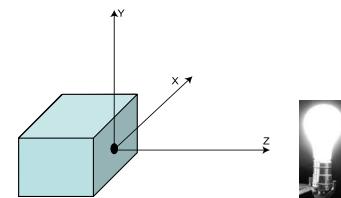
$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \rightarrow \left(\frac{x}{w}, \frac{y}{w} \right)$$

Mathematical Camera Model



Perspective Projection

$$x = \frac{fX}{Z} \quad y = \frac{fY}{Z}$$



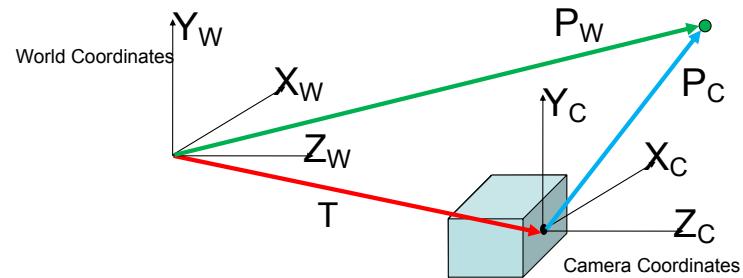
Let's assume the Origin of the World Coordinate System at the Aperture of the Pinhole.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \\ 1 \end{bmatrix} \longrightarrow (fx/Z, fy/Z)$$

When changing to Pixels, there will be an arbitrary Scaling in horizontal and vertical Directions.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} aX \\ bY \\ z \\ 1 \end{bmatrix} \longrightarrow (aX/Z, bY/Z)$$

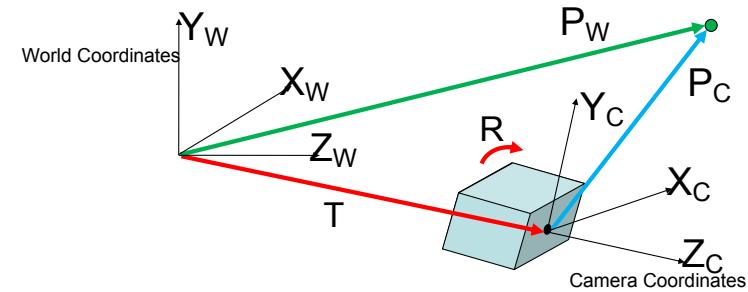
Camera Translation



In Heterogeneous Coordinates:

$$P_C = P_W - T$$

Camera Rotation



In Heterogeneous Coordinates:

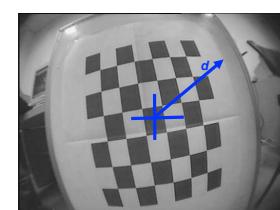
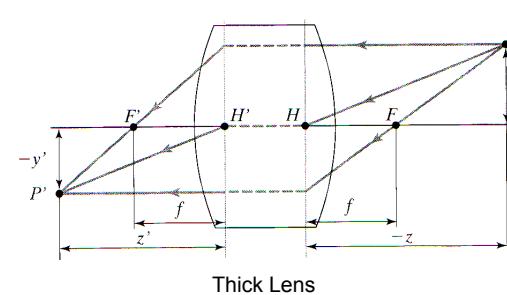
$$P_C = R(P_W - T)$$

Simplified Mathematical Camera Model

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} [3 \times 3] \\ a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Parameters (K)}} \cdot \underbrace{\begin{bmatrix} [3 \times 3] \\ R \end{bmatrix}}_{\text{Extrinsic Parameters}} \cdot \begin{bmatrix} [3 \times 4] \\ I \\ -T \end{bmatrix} \cdot \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

Note, that in Practice there are more intrinsic Parameters: e.g. Principle Point and Skew Angle describing Misalignment of Sensor on Optical Axis.

Radial Distortion



Radial Distortion

If Lens Distortion is considered, our Mathematical Model becomes non-linear (usually a Polynomial Function)!

Camera Calibration



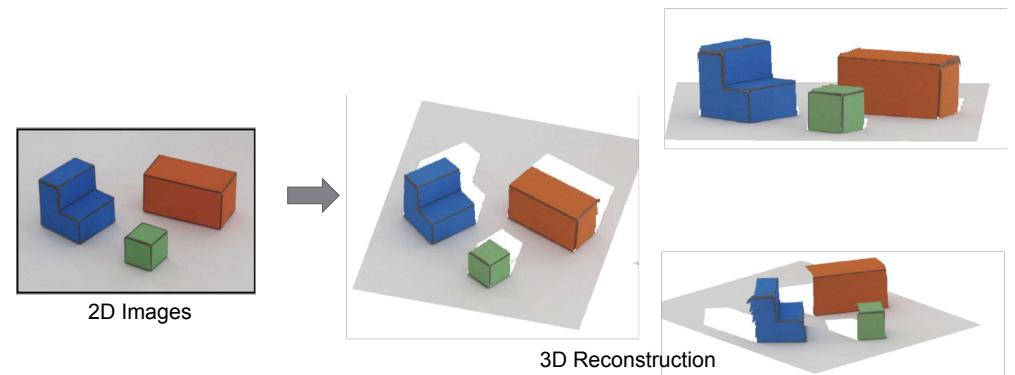
Images of known Calibration Pattern

`cv.calibrateCamera()` returns the Camera Matrix, distortion Coefficients, Rotation and Translation Vectors etc.

Optimization minimizes Re-Projection Error of Checker Corners

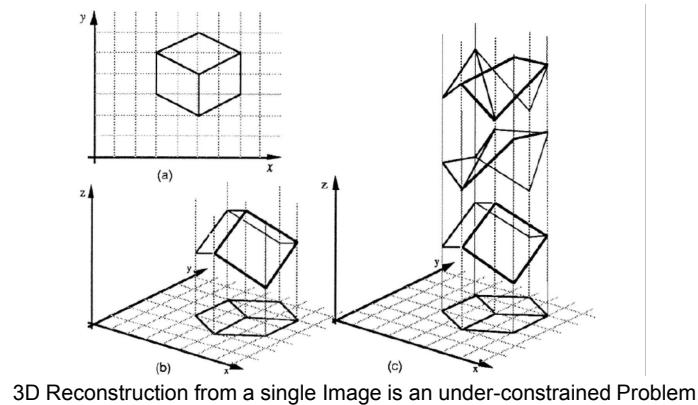
In OpenCV: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

Why do we need all of this?

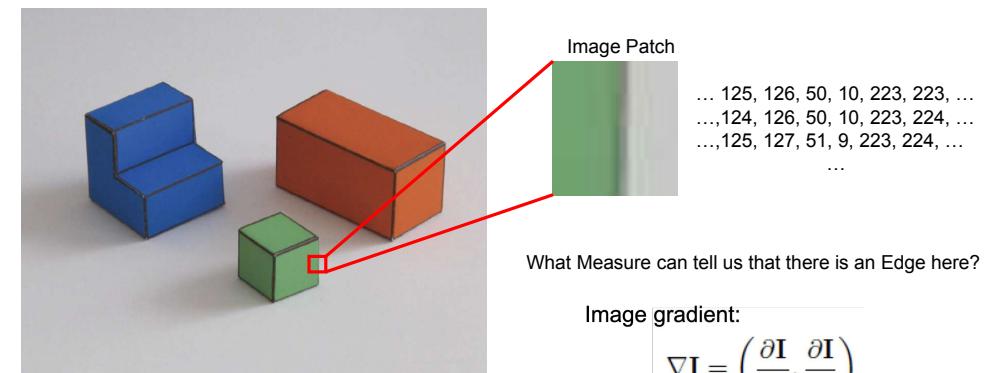


3D Reconstruction

3D Reconstruction is not simple!



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



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

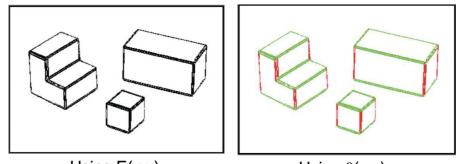
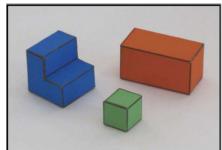


Image gradient:

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

Approximation image derivative:

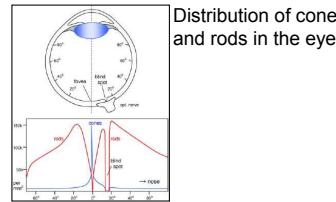
$$\frac{\partial I}{\partial x} \simeq I(x, y) - I(x - 1, y)$$

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

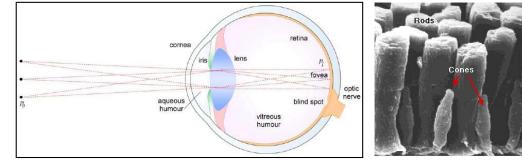
$$\text{Edge orientation: } \theta(x, y) = \angle \nabla I = \arctan \frac{\partial I / \partial y}{\partial I / \partial x}$$

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

How Humans do it?



Distribution of cones and rods in the eye

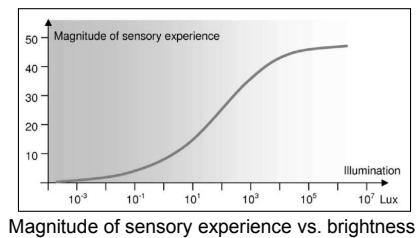


The Human Eye is a biological Camera (Lens, Aperture=Iris, Sensor=Retina) and is used for Measurement=Sensing.

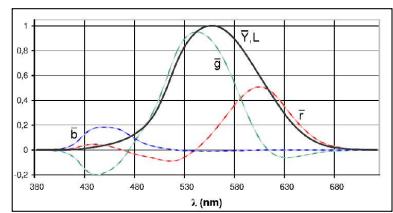


The Perception of the Visual Cortex is really what we want to mimic with Computer Vision!

Human Visual Perception (Sensory Basics)



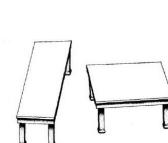
Magnitude of sensory experience vs. brightness



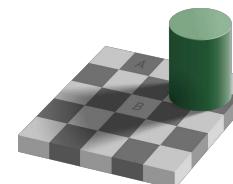
CIE 1931 RGB color matching functions in real ratio (dashed), and resulting luminance (Y) curve (black)

Why is our Brain so efficient?

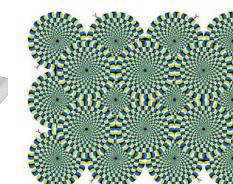
Likely because we are making **prior** assumptions about the world regarding depth, color, shadows, motion, etc.



Shepherd Tabletop
Illusion



Anderson Shadow
Illusion



Spinning Wheels
Illusion



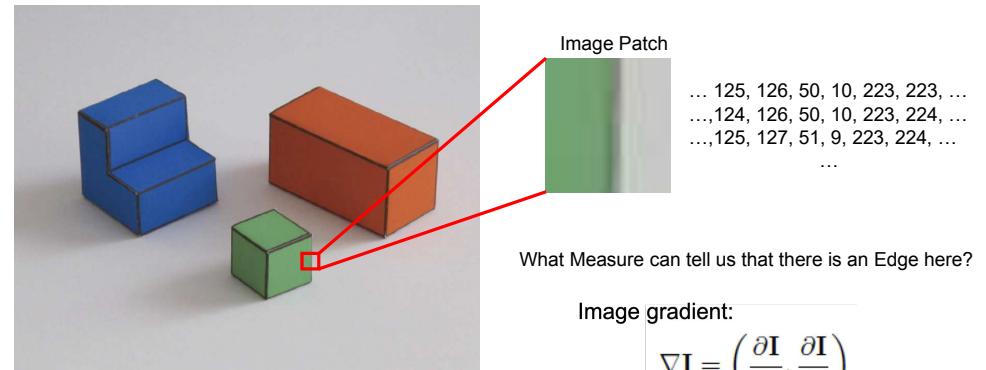
Crooked Lines
Illusion

Computer Vision



Lecture 3: Digital Image Processing
Oliver Bimber

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



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

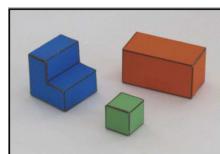


Image gradient:

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

Approximation image derivative:

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

Edge strength

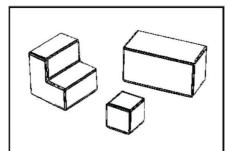
$$E(x, y) = |\nabla I(x, y)|$$

Edge orientation:

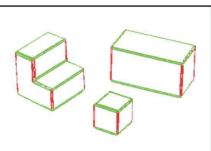
$$\theta(x, y) = \angle \nabla I = \arctan \frac{\partial I / \partial y}{\partial I / \partial x}$$

Edge normal:

$$n = \frac{\nabla I}{|\nabla I|}$$



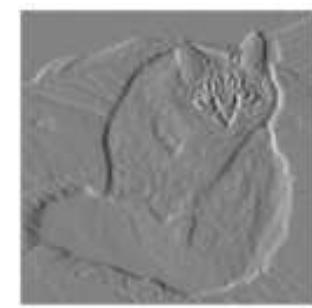
Using $E(x, y)$



Using $\theta(x, y)$

Image Gradients

What happens if we apply this to every Pixel?



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

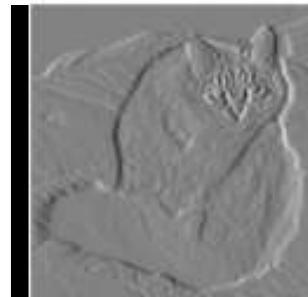
Image Gradients

And what happens if we shift the Image?



$I(x, y)$

This is called Shift-Invariance.



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

Convolution

x

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

Kernel

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

=

y

x

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

Kernel

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

=

y

Convolution

Convolution

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

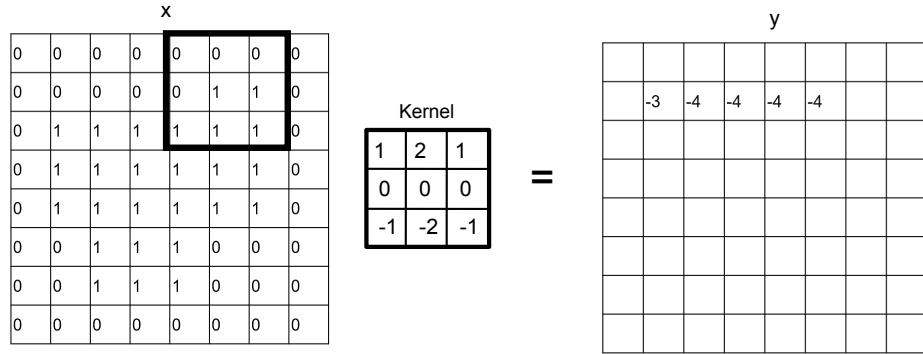
1	2	1
0	0	0
-1	-2	-1

=

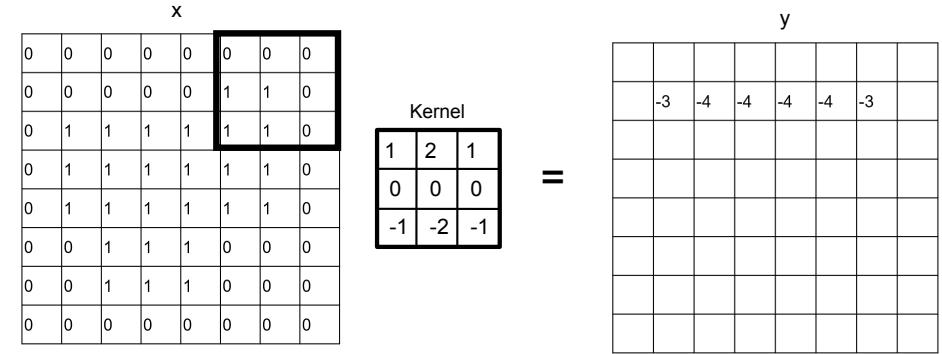
y

Convolution

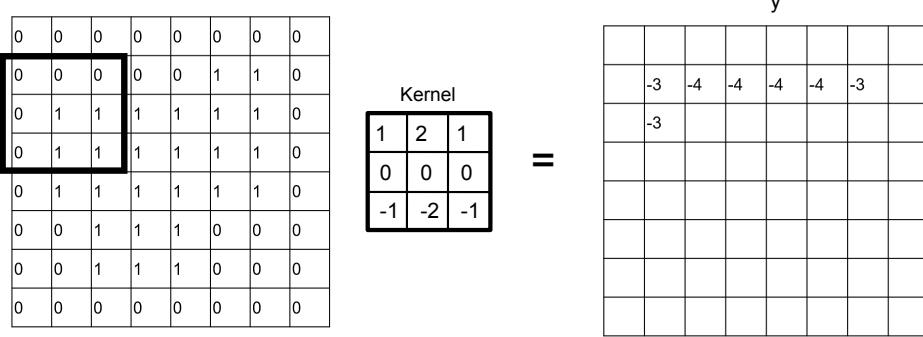
Convolution



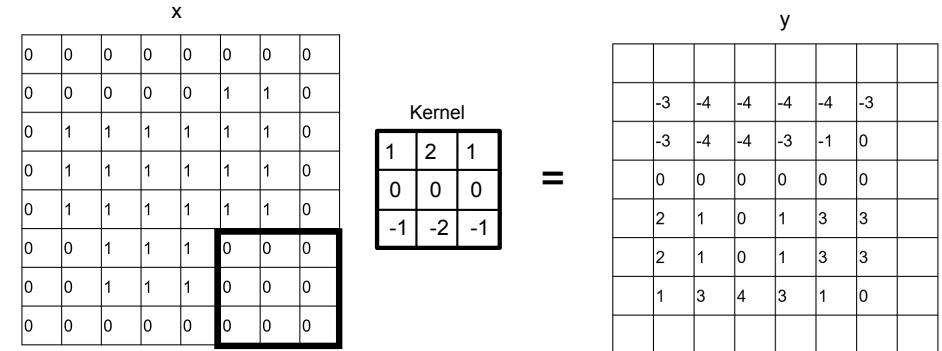
Convolution



Convolution



Convolution



Convolution

	x	
0 0 0 0 0 0 0 0 0		
0 0 0 0 0 0 1 1 0		
0 1 1 1 1 1 1 1 0		
0 1 1 1 1 1 1 1 0		
0 1 1 1 1 1 1 1 0		
0 0 1 1 1 0 0 0 0		
0 0 1 1 1 0 0 0 0		
0 0 0 0 0 0 0 0 0		

Kernel

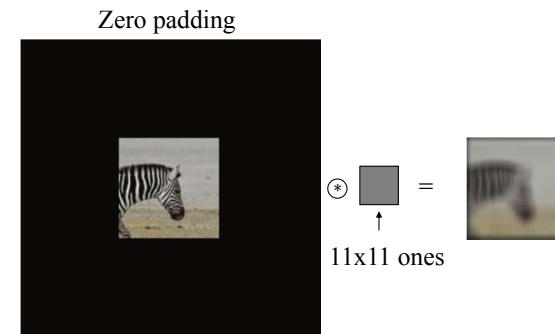
1 2 1	
0 0 0	
-1 -2 -1	

=

?								
-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 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 0	

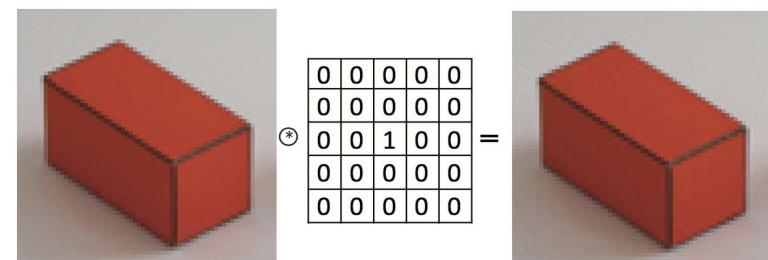
In this case, $N = 1$.

\dagger

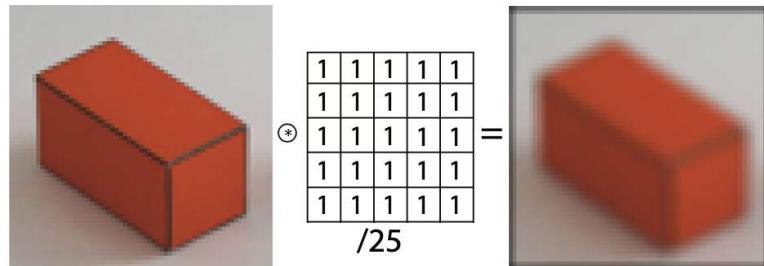
1 -1 -2 -1	
0 0 0 0	
-1 1 2 1	

Convolution is a shift-invariant linear Operation.

Example: Impulse



Example: Blurring (Box Filter)



Recap: Gradients

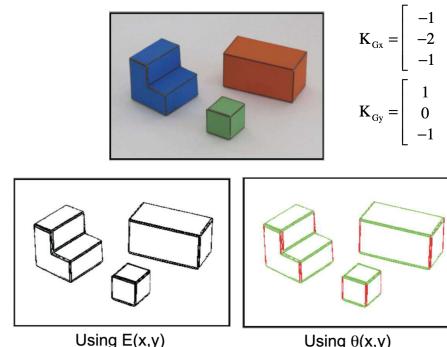


Image gradient:

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

Approximation image derivative:

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

Edge strength

$$E(x, y) = |\nabla I(x, y)|$$

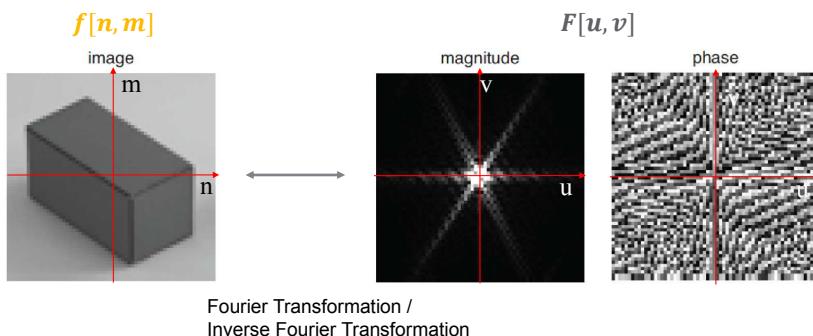
Edge orientation:

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

Edge normal:

$$n = \frac{\nabla I}{|\nabla I|}$$

Spatial vs. Frequency Domain



Spectra of Natural Images



Convolution Theorem

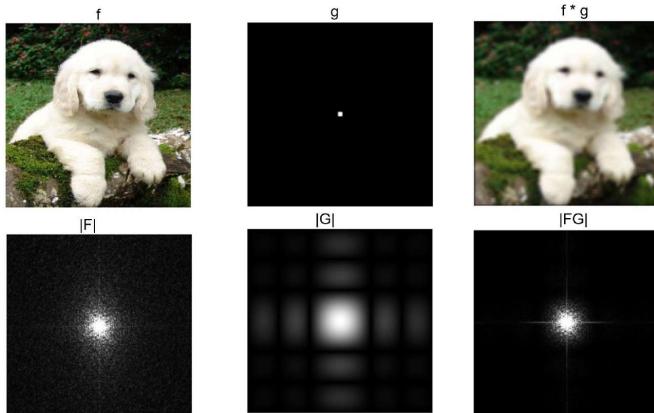
The Fourier transform of the convolution is the product of Fourier transforms

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

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

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

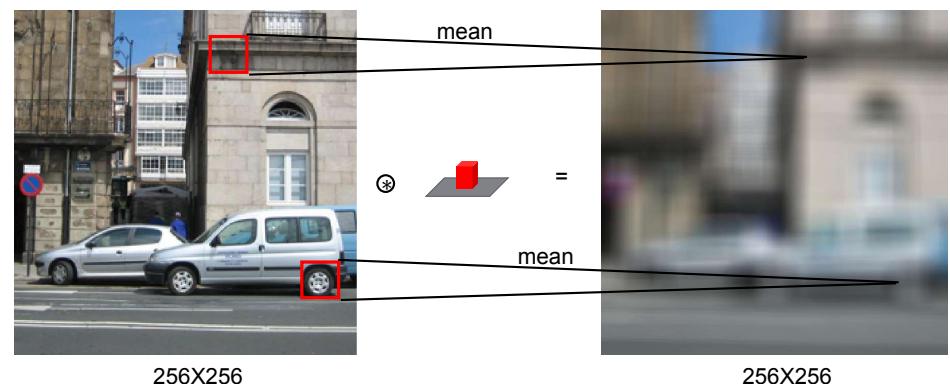
Example: Box Filter



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	$f(x + t) * h = (f * h)(x + t)$	$(e^{i\omega t} F)H = e^{i\omega t}(FH)$

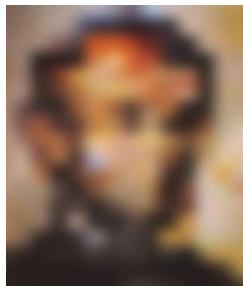
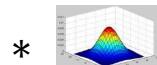
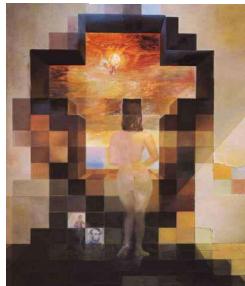
Low-Pass Filter (Box Filter)



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

Low-Pass Filter (Gaussian Filter)

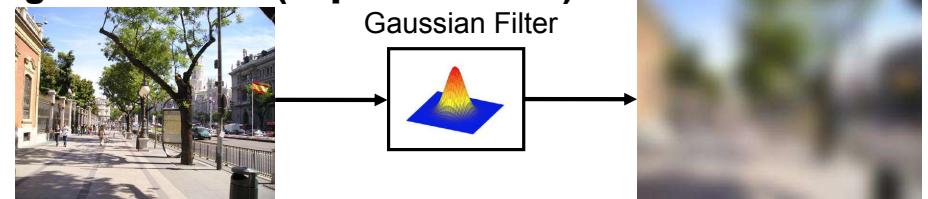
$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$



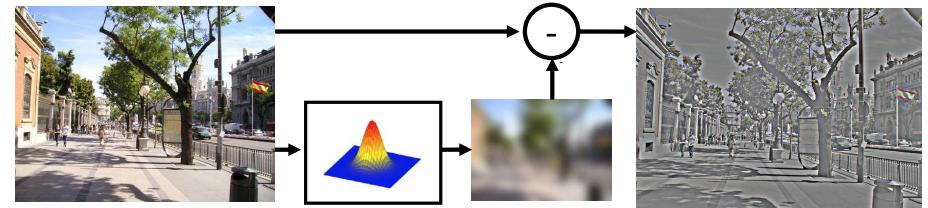
- Convolution of two Gaussians is a Gaussian.
- The (continuous) Fourier Transform of a Gaussian is another Gaussian.
- Gaussians are separable (2x1D Convolution instead of 1x2D).

109

High-Pass Filter (Laplacian Filter)



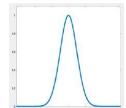
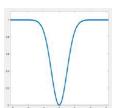
Gaussian Filter



Laplacian Filter

110

High-Pass Filter (Laplacian Filter)

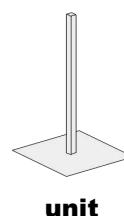


111

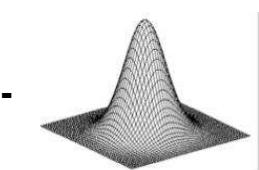
High-Pass Filter (Laplacian Filter)

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

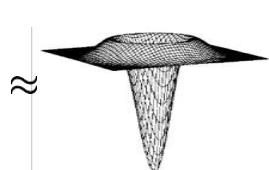
$$K_{\nabla^2} = -1 \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



unit



Gaussian

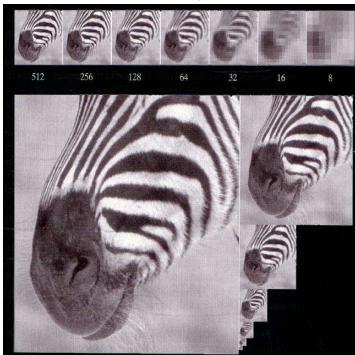


Laplacian

112

Image Pyramids (Scale Invariance)

Gaussian Pyramid

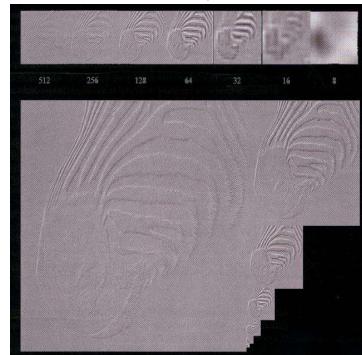


$$P_G^{n+1}(I) = \downarrow (G_s \otimes P_G^n(I))$$

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

JKU JOHANNES KEPLER
UNIVERSITY LINZ

Laplacian Pyramid



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

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

113

Deconvolution / Inverse Filtering

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

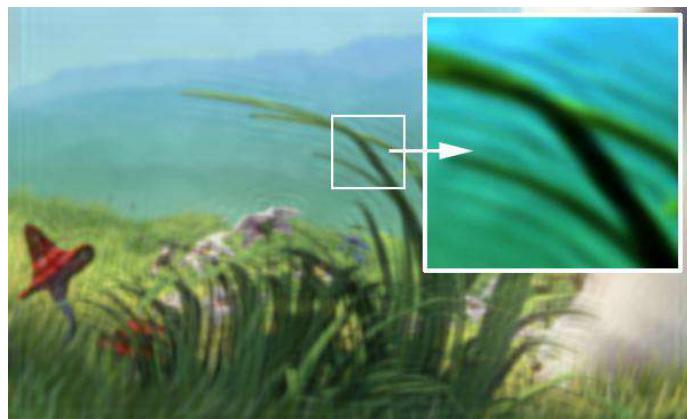
Convolution in Spatial Domain.....is.....Multiplication in Frequency Domain

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

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

114

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

115

Inverse Filtering



Original Image



After Convolution
(Motion Blur)



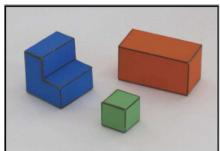
After Inverse Filtering

JKU JOHANNES KEPLER
UNIVERSITY LINZ

JKU JOHANNES KEPLER
UNIVERSITY LINZ

116

Recap: Gradients



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

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

Image gradient:

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

Approximation image derivative:

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

Edge strength

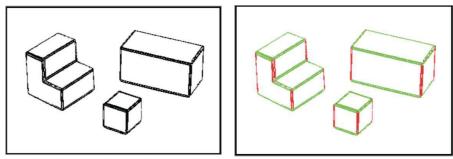
$$E(x, y) = |\nabla I(x, y)|$$

Edge orientation:

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

Edge normal:

$$n = \frac{\nabla I}{|\nabla I|}$$



Spatial vs. Gradient Domain

Gradient Domain

Image gradient:

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

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

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

We can consider this as Coefficients of a Vectorfield.

Derivative / Integral

$$I = \int \nabla I$$

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

Spatial vs. Gradient Domain

Gradient Domain

Image gradient:

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

I_x

I_y

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

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

Laplacian!

We can consider this as Coefficients of a Vectorfield.

Spatial vs. Gradient Domain

Gradient Domain

Image gradient:

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

I_x

I_y

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

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

We can consider this as Coefficients of a Vectorfield.

Spatial Domain

$$I = \int \nabla I$$

Processing the Vectorfield is called
Gradient Domain Processing.

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

Image Domain
 $\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$
 Derivative / Integral
 $I = \int \nabla I$

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

Laplacian!

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

I_x

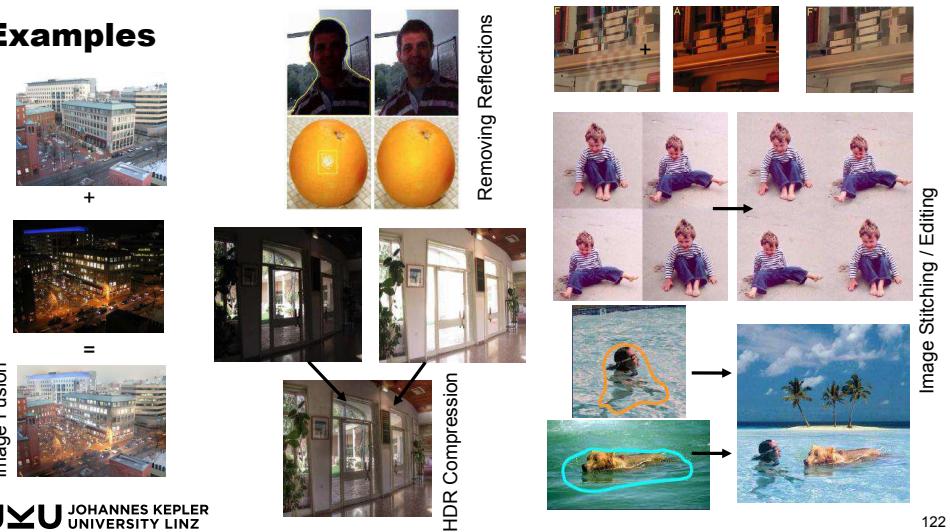
I_y

We can consider this as Coefficients of a Vectorfield.
Processing the Vectorfield is called Gradient Domain Processing.

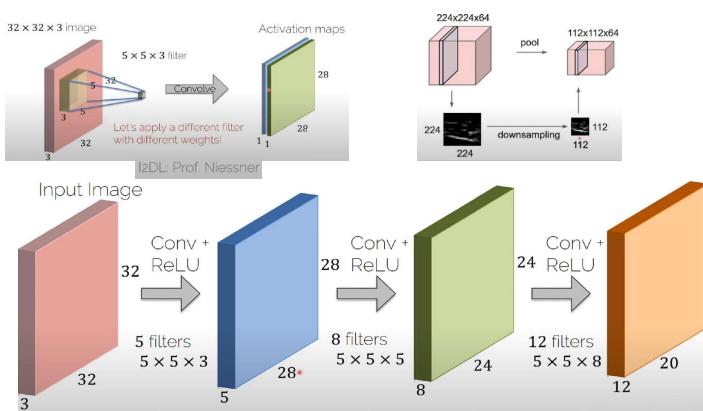
Integrating this Vectorfield results in the original Image as long as the Vectorfield has zero Curl. In Practice: that does not work as the Vectorfields is no longer conservative anymore after processing it (it has non-zero Curl!).

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

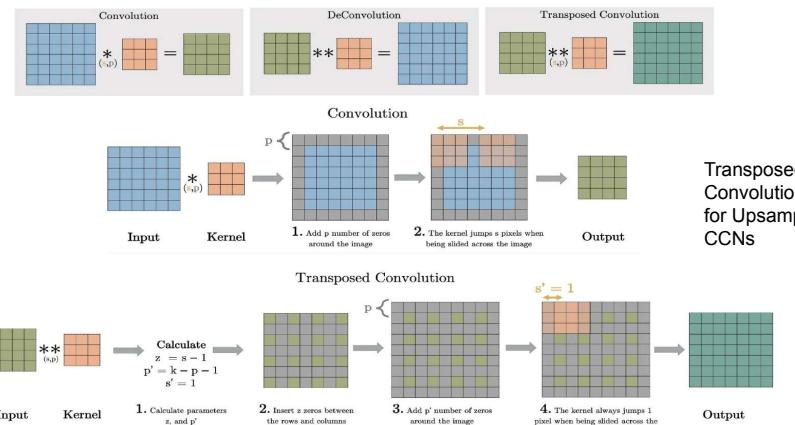
Examples



Convolutional Neural Networks (CNNs)



Transposed Convolution



Computer Vision



Lecture 4: Machine Learning
Oliver Bimber

Research Example: How Robot Dogs See the Unseeable



What does \star do?

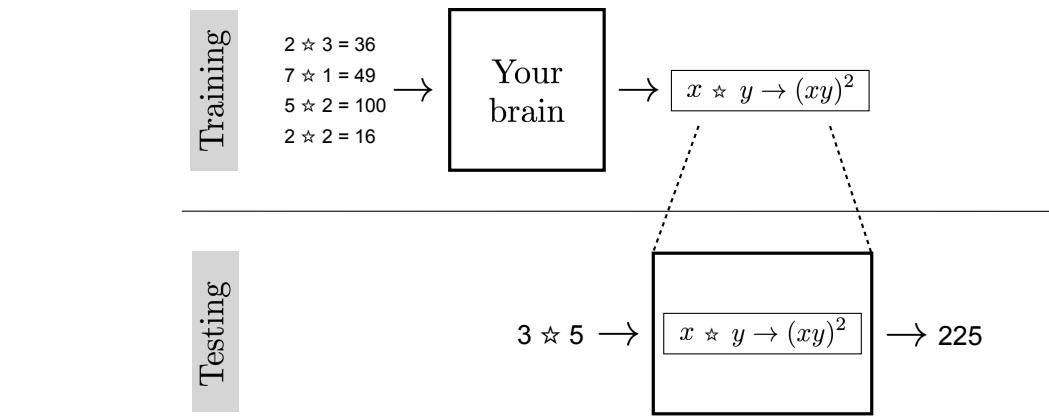
$$2 \star 3 = 36$$

$$7 \star 1 = 49$$

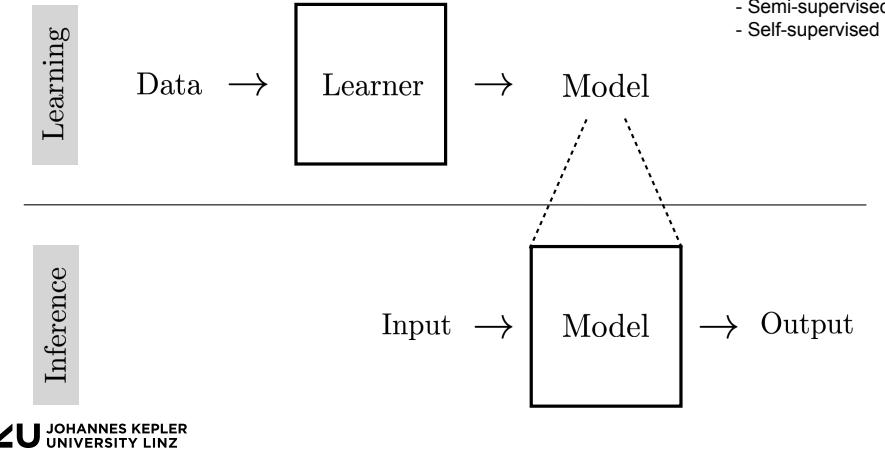
$$5 \star 2 = 100$$

$$2 \star 2 = 16$$

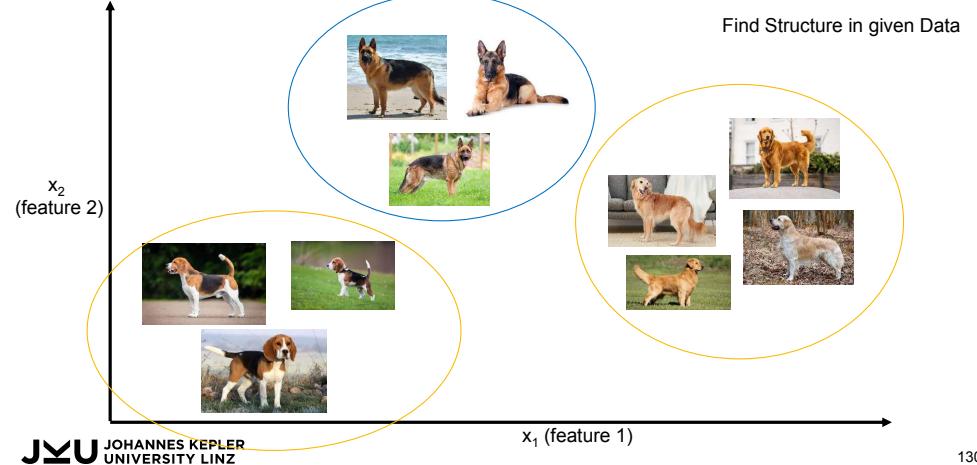
What does \star do?



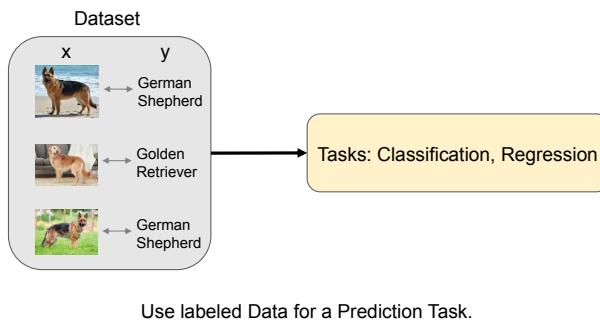
Machine Learning



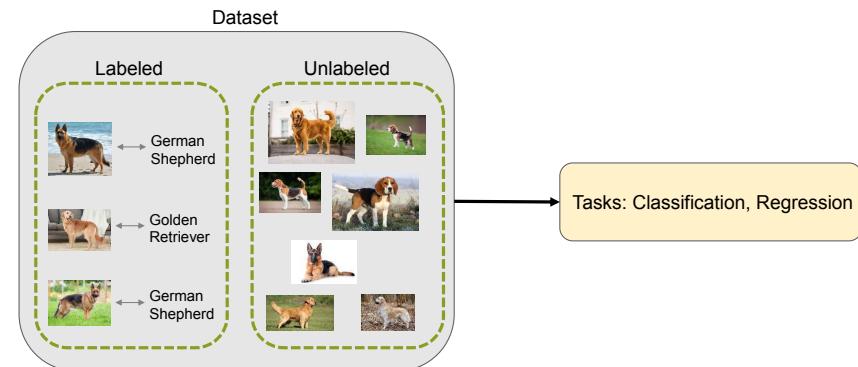
Unsupervised Learning



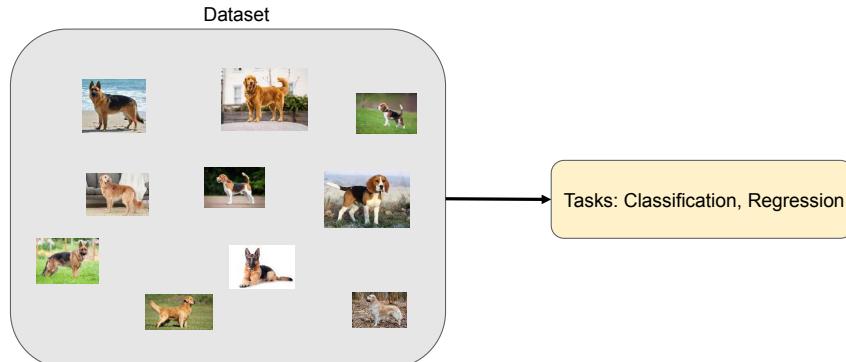
Supervised Learning



Semi-Supervised Learning



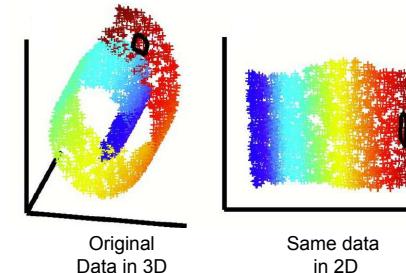
Self-Supervised Learning



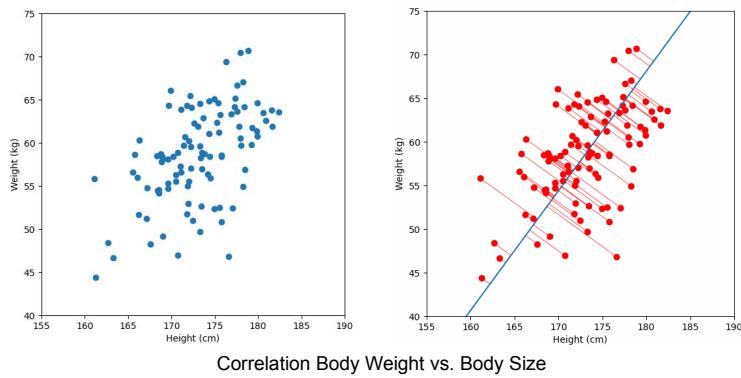
No external Labels. Instead, uses Properties of the Data itself as supervisory Signals. Example (learn how to solve a puzzle): take unlabeled images, rotate them, and train a model to guess the rotation. It needs to understand how the objects look upright → helps to learn how to classify.

Dimensionality Reduction

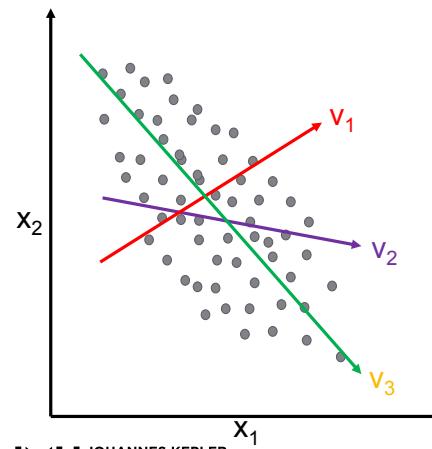
Given Data Points in d Dimensions, convert them to Data Points in $k < d$ Dimensions with minimal loss of Information.



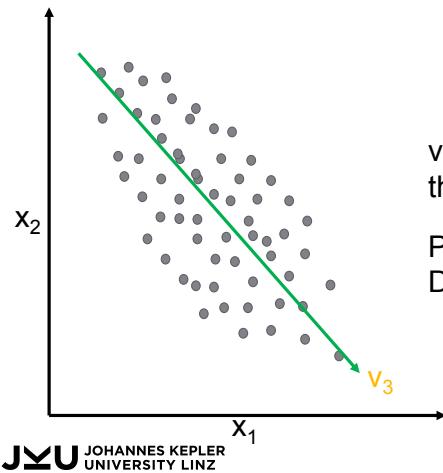
Dimensionality Reduction: Example



What is the best Direction?



What is the best Direction?



The Variance along v of all the Point Projections is:

$$\begin{aligned} \text{var}(v) &= \sum_x \| (x - \bar{x})^T \cdot v \|^2 \\ &= \sum_x v^T (x - \bar{x})(x - \bar{x})^T v \\ &= v^T \left[\sum_x (x - \bar{x})(x - \bar{x})^T \right] v \\ &= v^T C v, \text{ where } C = X^T X \text{ is the covariance matrix of } X \end{aligned}$$

v = Eigenvectors of C
 $\text{var}(v)$ = Eigenvalues of v

PCA is used to compute $v, \text{var}(v)$

We are interested in v with the largest $\text{var}(v)$!

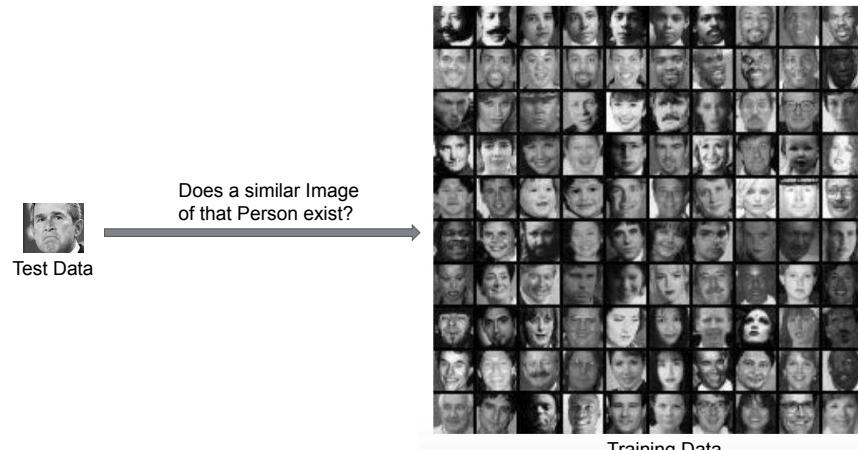
This works also in higher dimensions.

JYU JOHANNES KEPLER UNIVERSITY LINZ

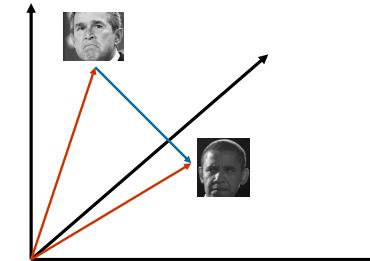
137

138

Example: Eigenfaces



Example: Eigenfaces



Here, an Image represents a Point in a High Dimensional space

- An $N \times M$ (binary) Image is a Point in R^{NM}
- We can define Vectors in this Space

JYU JOHANNES KEPLER UNIVERSITY LINZ

139

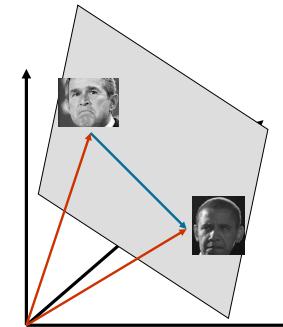
140

Face Space

- When viewed as Vectors of Pixel Values, Face Images are extremely high-dimensional
 - 100×100 Image = **10,000** Dimensional
- But very few **10,000**-dimensional Vectors are valid Faces
- Let's find a lower-dimensional Representation using PCA!



Dimensionality Reduction



The set of Faces is a “Subspace” of the Set of Images

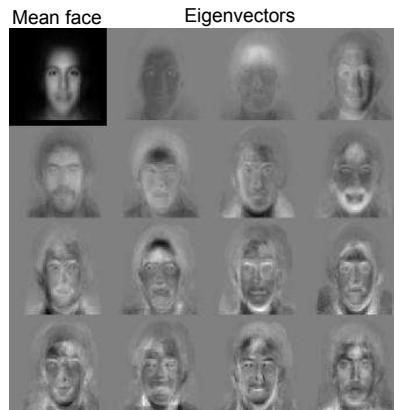
- Suppose it is K -dimensional
- We can find the best Subspace using PCA
- This is like fitting a “Hyperplane” to the Set of Faces spanned by vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K$

$$\text{Any Face: } \mathbf{x} \approx \bar{\mathbf{x}} + a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + \dots + a_K \mathbf{v}_K$$

Eigenfaces

PCA extracts a set of vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots$

Each one of these Vectors is a Direction in Face space. What do these look like?



Projecting onto Eigenfaces

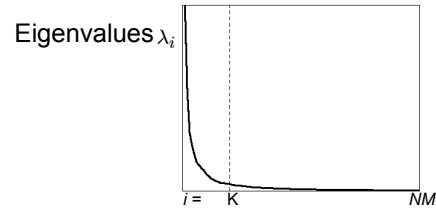
The Eigenfaces $\mathbf{v}_1, \dots, \mathbf{v}_K$ span the Space of Faces. A Face is converted to Eigenface coordinates by

$$\mathbf{x} \rightarrow ((\underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_1}_{a_1}, \underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_2}_{a_2}, \dots, \underbrace{(\mathbf{x} - \bar{\mathbf{x}}) \cdot \mathbf{v}_K}_{a_K})$$

$$\mathbf{x} \approx \bar{\mathbf{x}} + a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + \dots + a_K \mathbf{v}_K$$



Choosing the Dimension K



How many Eigenfaces to use? Look at the Decay of the Eigenvalues

- The Eigenvalue tells you the amount of Variance “in the Direction” of that Eigenface
- Ignore Eigenfaces with low Variance

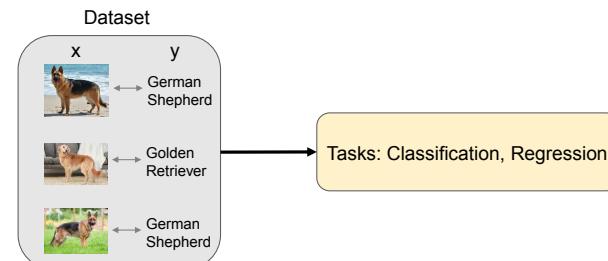
Face Recognition with Eigenfaces

- Project Input Face to Eigenfaces (i.e., represent it by a K-dimensional vector: $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_K$)
- Find the Nearest-Neighbor in Training Image Set (also projected to Eigenfaces)
- Pro: Speed, Size, Fairly Invariant to Resolution
- Cons: Fails quickly if Variation between Training Images and Test Image becomes too large

Dimensionality Reduction Methods

- PCA (Principal Component Analysis):
 - Find Projection that maximize the Variance
- ICA (Independent Component Analysis):
 - Very similar to PCA except that it assumes non-Gaussian features
- Multidimensional Scaling:
 - Find Projection that best preserves Inter-Point Distances
- LDA(Linear Discriminant Analysis):
 - Maximizing the Component Axes for Class-Separation
- ...
- ...

Supervised Learning – Example: Image Classification

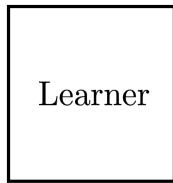


Use labeled Data for a Prediction Task.

Learning from Labeled Training Data

Training data

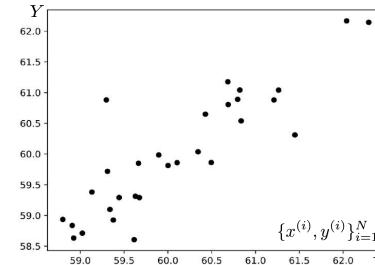
$$\begin{aligned} & \{x^{(1)}, y^{(1)}\} \\ & \{x^{(2)}, y^{(2)}\} \\ & \{x^{(3)}, y^{(3)}\} \\ & \dots \end{aligned}$$



$$f : X \rightarrow Y$$

Linear Regression

Training data



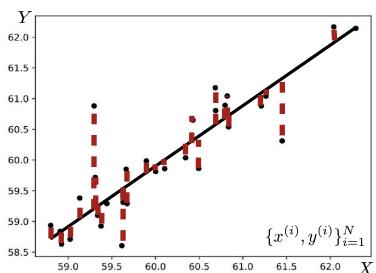
$$f_\theta(x) = \theta_1 x + \theta_0$$

Hypothesis space

The relationship between X and Y is roughly linear: $y \approx \theta_1 x + \theta_0$

Linear Regression

Training data



Search for the Parameters, $\theta = \{\theta_0, \theta_1\}$, that best fit the Data.

$$f_\theta(x) = \theta_1 x + \theta_0$$

Best Fit in what sense?

The Least-Squares Objective (aka Loss) says the best Fit is the Function that minimizes the Squared Error between Predictions and Target Values:

$$\mathcal{L}(\hat{y}, y) = (\hat{y} - y)^2 \quad \hat{y} \equiv f_\theta(x)$$

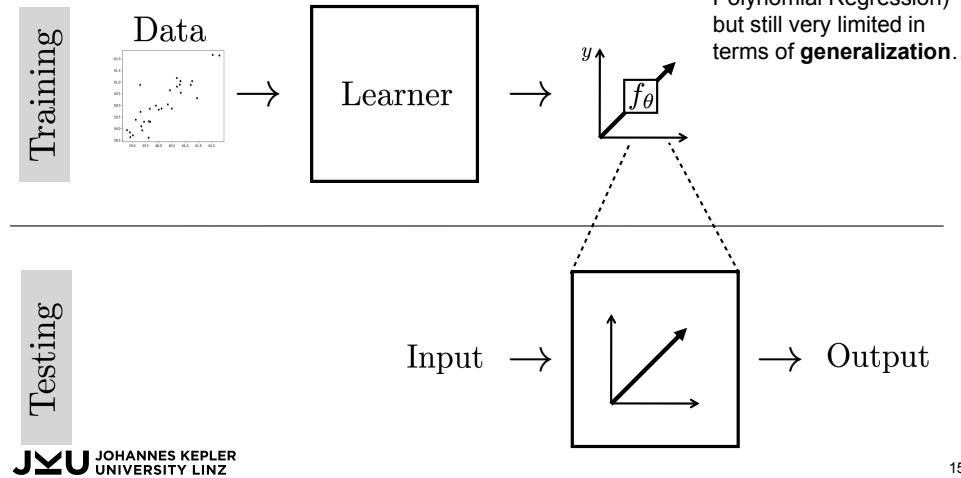
Linear Regression

Learner
Data
 $\{x^{(i)}, y^{(i)}\}_{i=1}^N \rightarrow$

Objective
 $\mathcal{L}(f_\theta(x), y) = (f_\theta(x) - y)^2$
Hypothesis space
 $f_\theta(x) = \theta_1 x + \theta_0$
Optimizer
 $\theta^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$

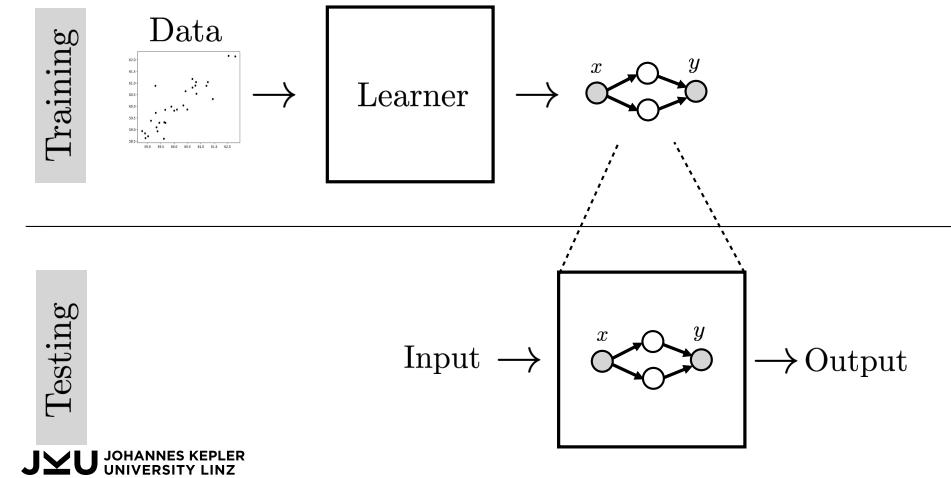
$$\rightarrow f$$

Linear Regression

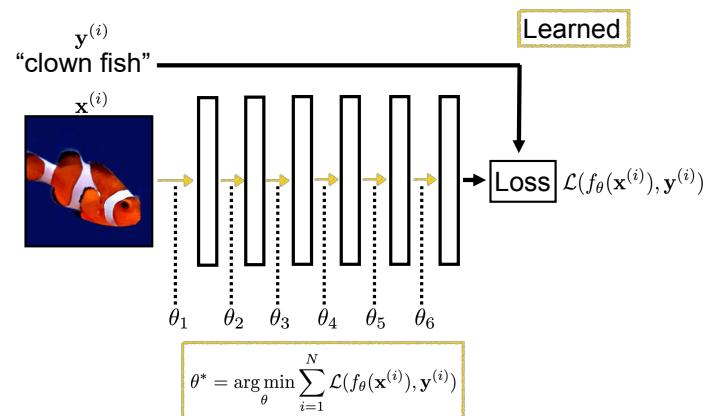


We could assume more complex Functions (like Polynomial Regression) – but still very limited in terms of **generalization**...

Deep Learning



Deep Learning



Deep Learning



Training: Used to train Model Parameters

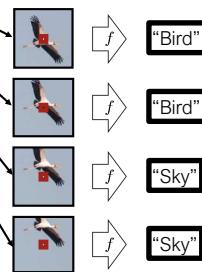
Validation: Used to tune Hyperparameters (e.g., λ in regularized regression)

Testing: Used to report final Accuracy (shouldn't be touched before!)

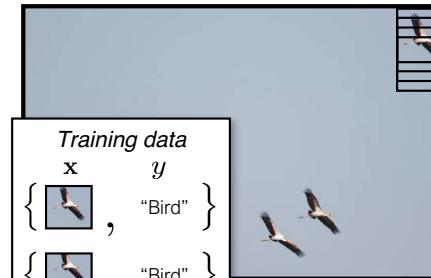
Classification with Convolution



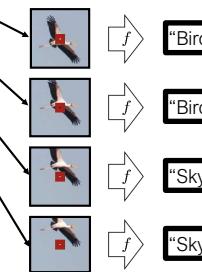
What's the Object Class of the Center Pixel?



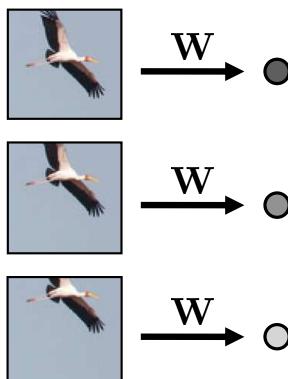
Classification with Convolution



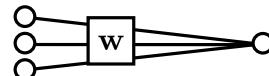
What's the Object Class of the Center Pixel?



Classification with Convolution



W computes a weighted Sum of all Pixels in the Patch

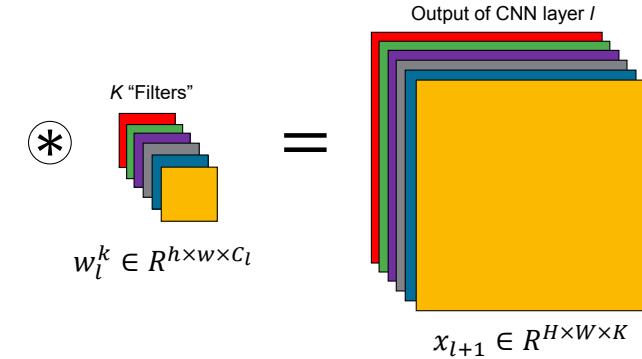
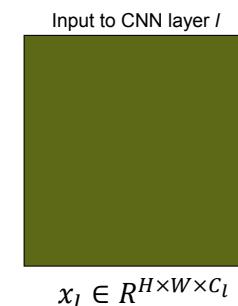


W is a Convolutional Kernel applied to the Full Image!

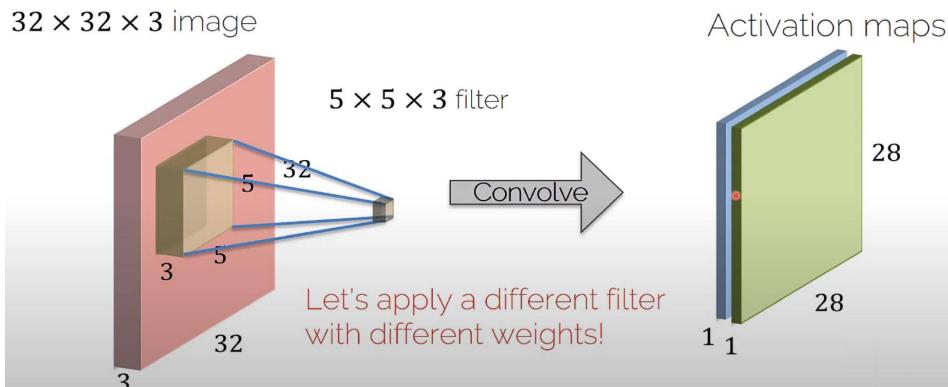
It gives highest Response if the Kernel matches the Image Content

It is Shift Invariant. But how about Scale, Rotation, other Poses, etc.?

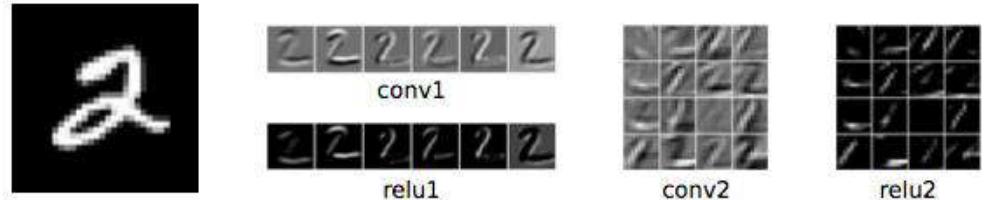
Convolutional Neural Networks (CNNs)



Convolutional Layer

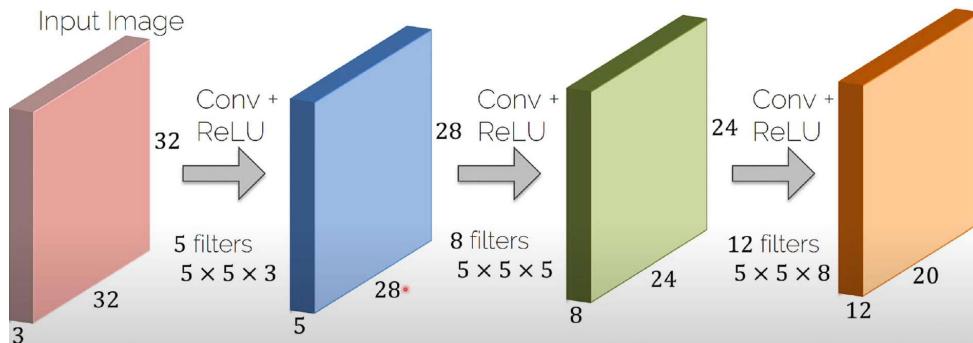


Activation (Feature) Maps



- Each Layer can be thought of as a set of C **Feature Maps** aka **Channels**
- Each Feature Map is an NxM Image
- ReLU (Rectified Linear Unit, $f(x)=\max(0,x)$) Activation Function is used to introduce Nonlinearity in a Neural Network, helping mitigate the Vanishing Gradient Problem

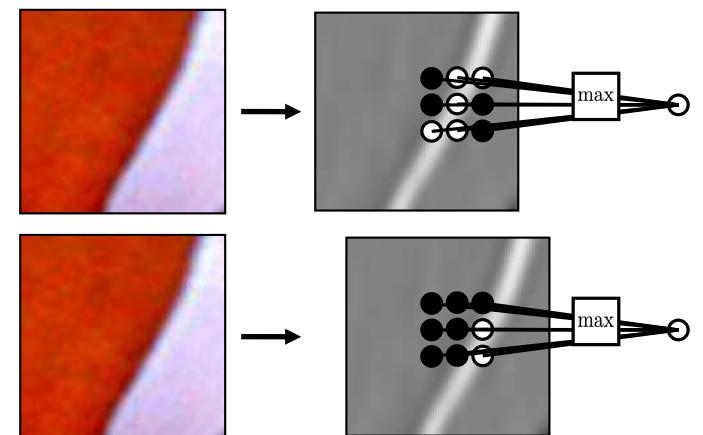
Multiple Convolutional Layers



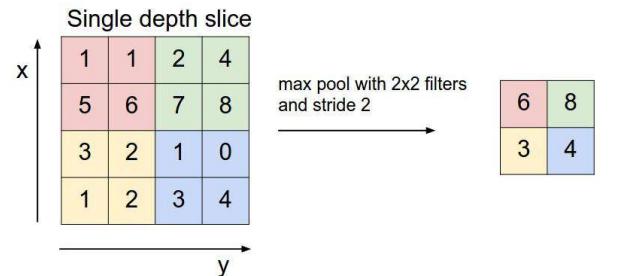
Pooling

Max pooling
(usually 2x2 or 3x3)
 $y_j = \max_{j \in \mathcal{N}(j)} h_j$

provides large Response
regardless of exact Position of Edge

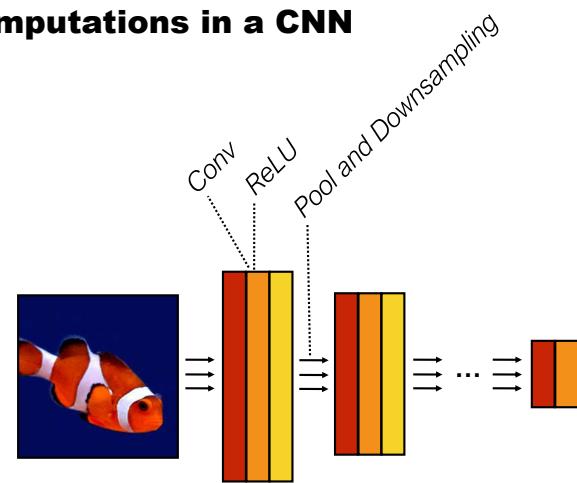


Pooling and Down-Sampling

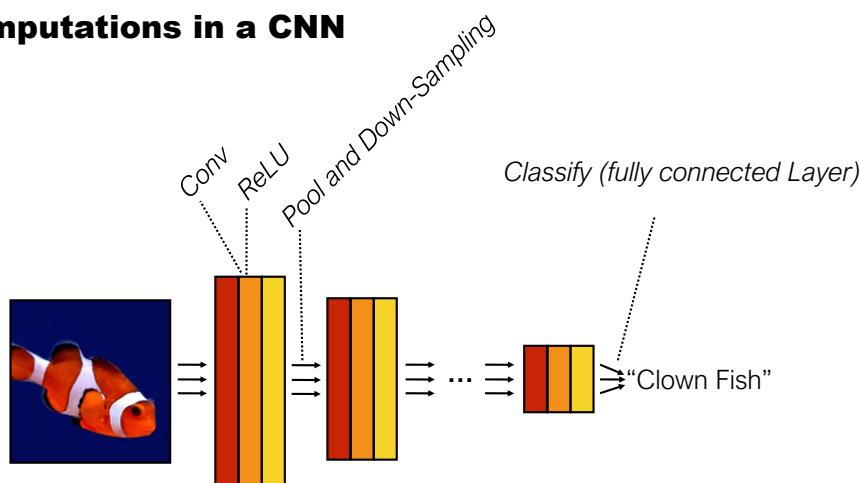


Stride: shift of kernel (for normal Convolution it is 1)

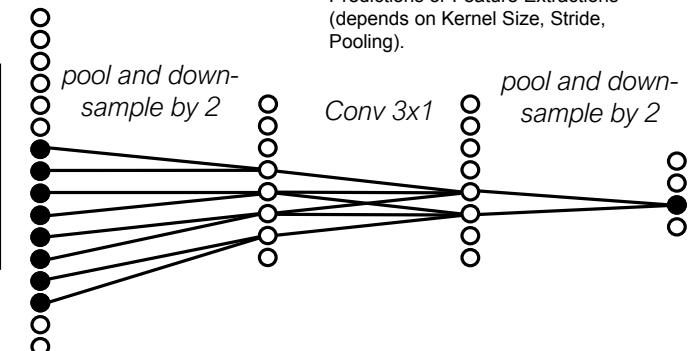
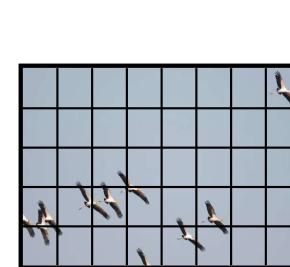
Computations in a CNN



Computations in a CNN



Receptive Field



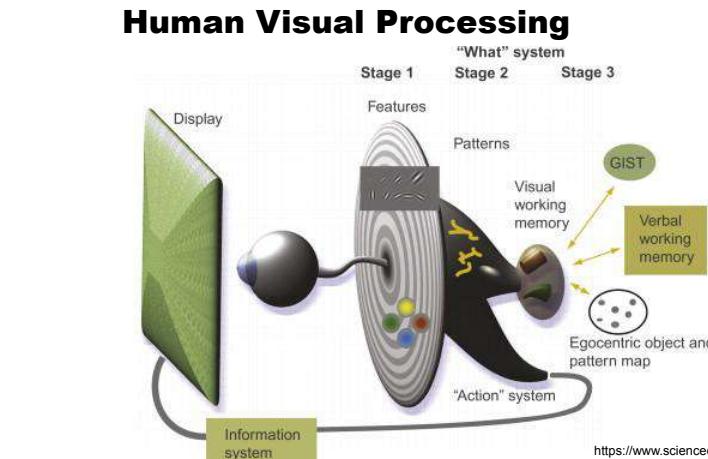
Receptive Field is the Region of the Image that a particular Neuron in a Convolutional Layer is “looking at” or taking into account when making its Predictions or Feature Extractions (depends on Kernel Size, Stride, Pooling).

Computer Vision

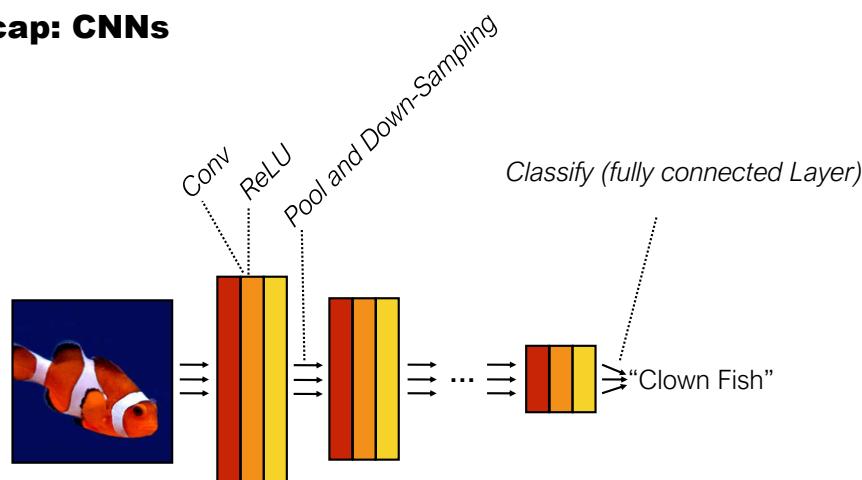


Lecture 5: Feature Extraction

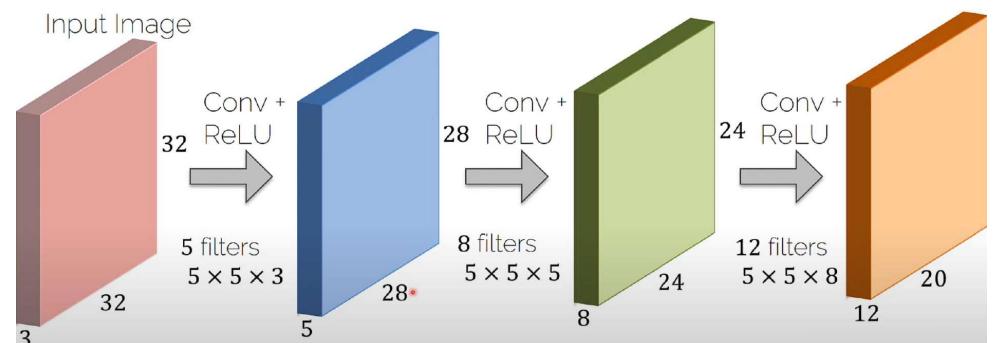
Oliver Bimber



Recap: CNNs



Recap: Convolutional Layers



Early Visual Processing

The visual image is broken down into different kinds of features, particularly color differences, local edge and texture information, and local motion information.

Pattern Perception

Patterns are formed based on low-level features and on the task demands of visual thinking. Patterns consist of entities such as continuous contours, areas of a common texture, color, or motion.

Working Memory

Only a few simple patterns can be held in working memory at any given instant. A visual query pattern can be held in working memory, forming the basis for active visual search through the direction of attention. Attention controls what visual information is held and stored. The semantic meaning or gist of an object or scene can be activated. For items to be processed into long-term memory, deeper semantic coding is needed (and sleep).

Recap: Activation (Feature) Maps



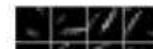
conv1



relu1



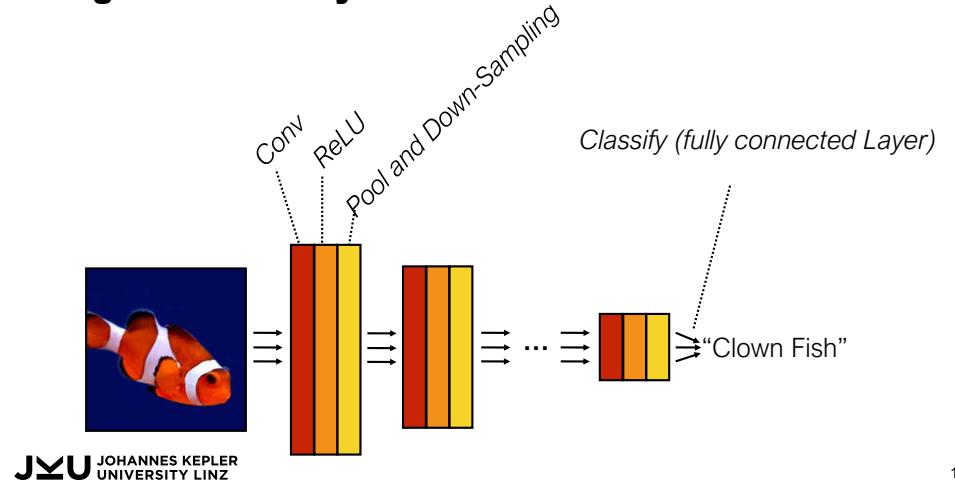
conv2



relu2

- Each Layer can be thought of as a set of C **Feature Maps** aka **Channels**
- Each Feature Map is an NxM Image
- ReLU (Rectified Linear Unit, $f(x)=\max(0,x)$) Activation Function is used to introduce Nonlinearity in a Neural Network, helping mitigate the Vanishing Gradient Problem

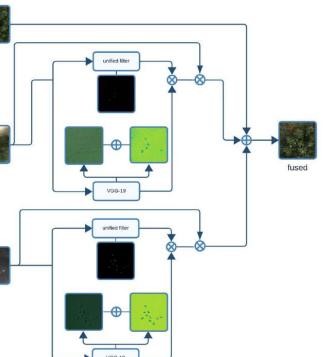
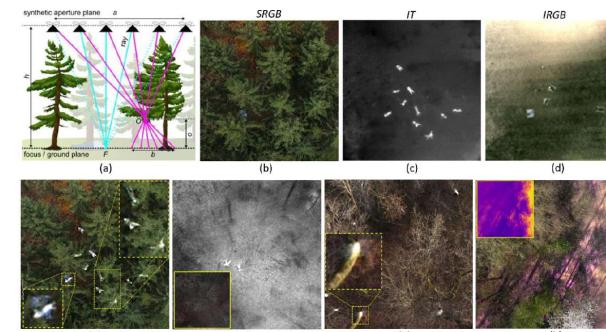
Using Features Beyond Classification



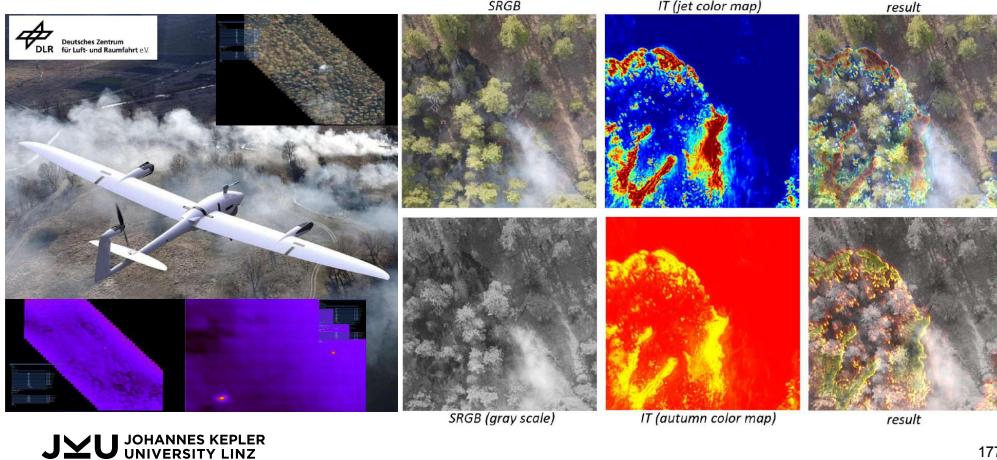
Airborne Optical Sectioning



Research Example: Image Fusion with VGG-Net Features



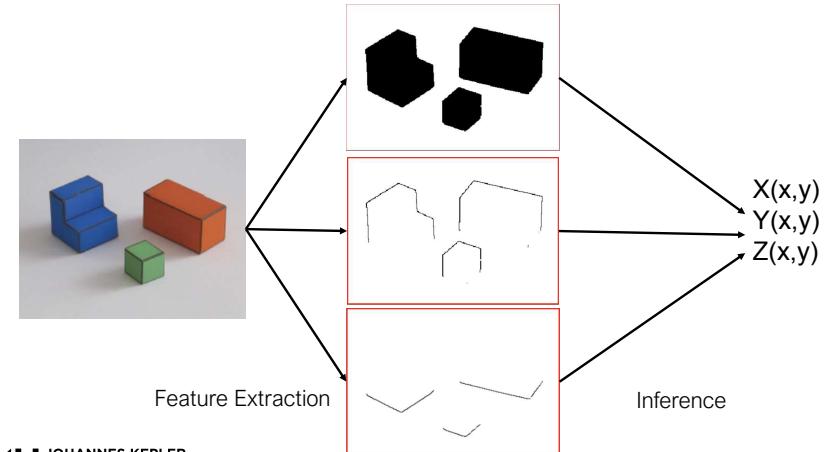
Research Example: Image Fusion with VGG-Net Features



JKU JOHANNES KEPLER
UNIVERSITY LINZ

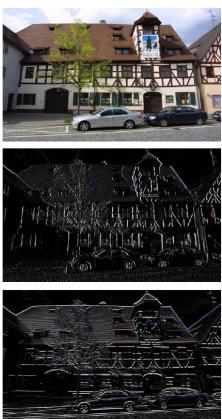
177

Role of Image Features in Computer Vision



178

What are Image Features



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

$$H = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

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

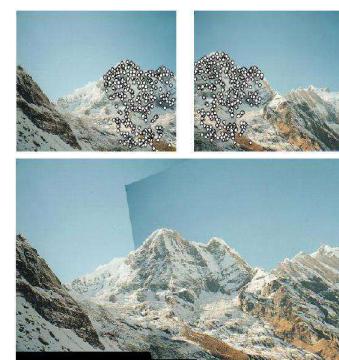
$$H = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad H = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Example: Partial Derivation

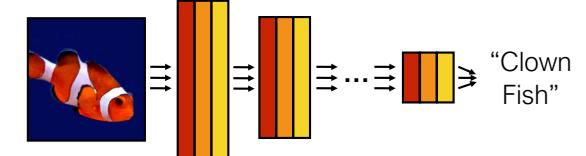
- Not every Pixel in an Image is equally important to understand its Content
- Image Features mark those Pixels that are important
- Convolution Kernels can be used to identify them
- But what are good (i.e., invariant) Features and what are the Kernels to get them?

179

What do we need Image Features for?



Example: Image Registration



Example: Classification

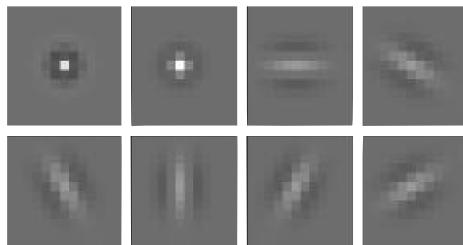
JKU JOHANNES KEPLER
UNIVERSITY LINZ

180

Model-Based vs. Learning-Based Feature Extraction

- **Model-Based:**

- manually designed, general Filter Kernels (e.g., adapted to the Human Visual System -- spots and Bars at different Orientations)

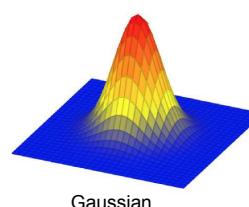


Filter Bank of two Spot and six Bar Filters

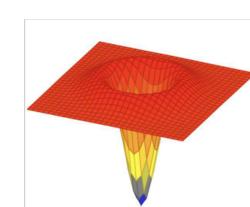
- **Learning-Based:**

- Application specific Filter Kernels are automatically learned from Training Data (e.g., CNN)

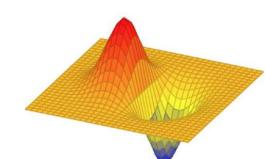
Shift-Invariance (Lots of useful Kernels)



Gaussian



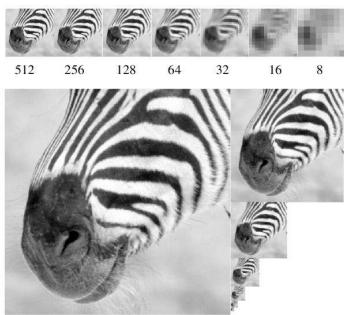
Laplacian



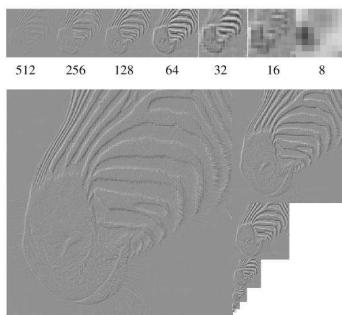
Gaussian Derivative

...and many more...

Scale-Invariance (Lots of Image Pyramids)



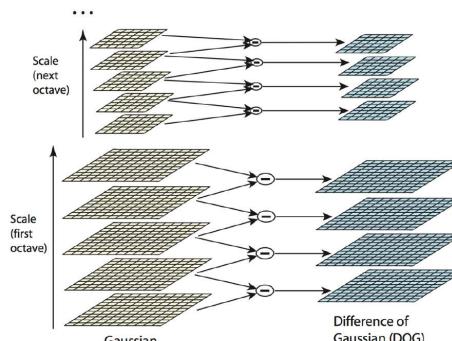
Gaussian Pyr



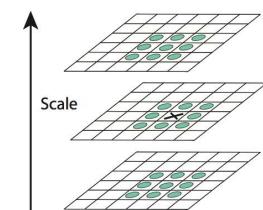
Laplacian Pyr

...and many more...

Example: Scale-Invariant Feature Transform (SIFT)



(1) Compute Scale Space (Difference of Gaussian, DoG = Laplacian Image Pyramid)

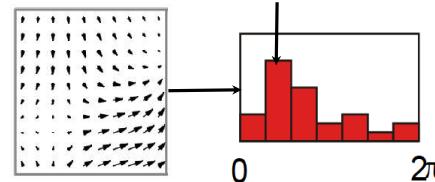


(2) Detect local Extrema in DoG Pyramid Layers (i.e., high Gradients wrt. Neighborhood)

Example: Scale-Invariant Feature Transform (SIFT)

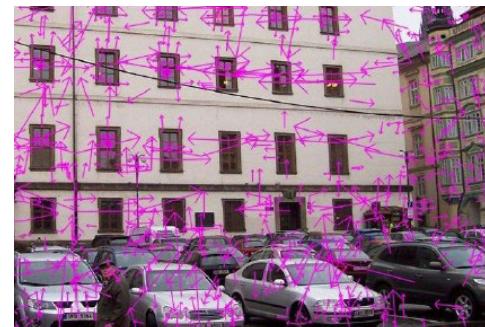


(3) Filter out Features in low-contrast Regions (Noise)



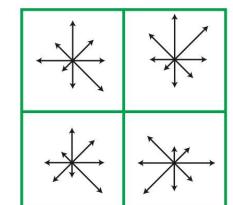
(4) Determine Feature (i.e., Gradient) Orientations and sort them into Histogram (largest Bin = main Orientation)

Example: Scale-Invariant Feature Transform (SIFT)



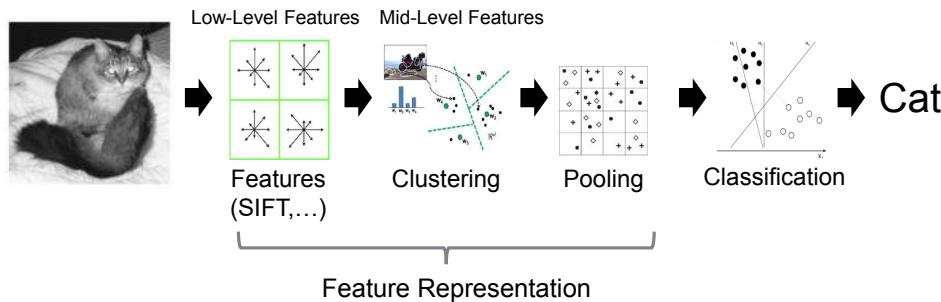
Assignment of Main Orientations to Feature Points

(5) Define Descriptor for each Feature. Example: for 4 neighboring Image Tiles (each 4x4 Pixels) determine Orientation Histogram (e.g., 8 Bins) $\rightarrow 4 \times 8 = 32$ Entries of Feature Descriptor (SIFT used 4x4=16 Tiles instead of 2x2=4 Tiles $\rightarrow 16 \times 8 = 128$ Entries of Feature Descriptor)



Orientation Histograms (8 bins) for each Tile

Model-Based Classification



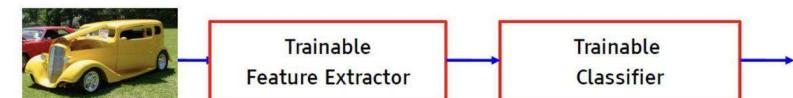
Problem: Manually constructed Features. Can we learn them from Data?

Model-Based vs. Learning-Based Feature Extraction

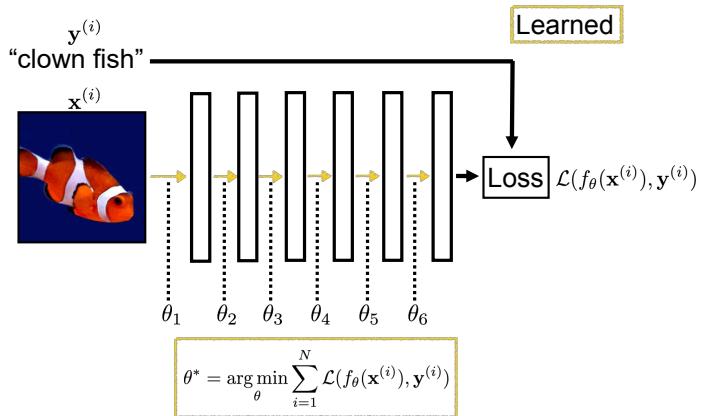
- Fixed engineered features (or kernels) + trainable classifier



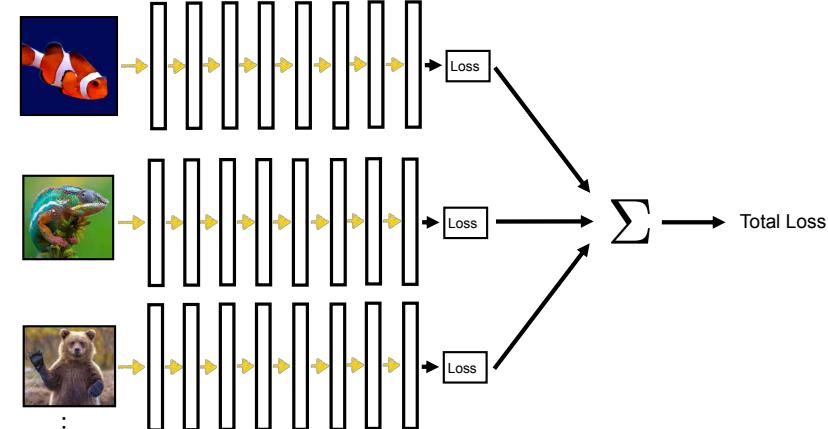
- End-to-end learning / feature learning / deep learning



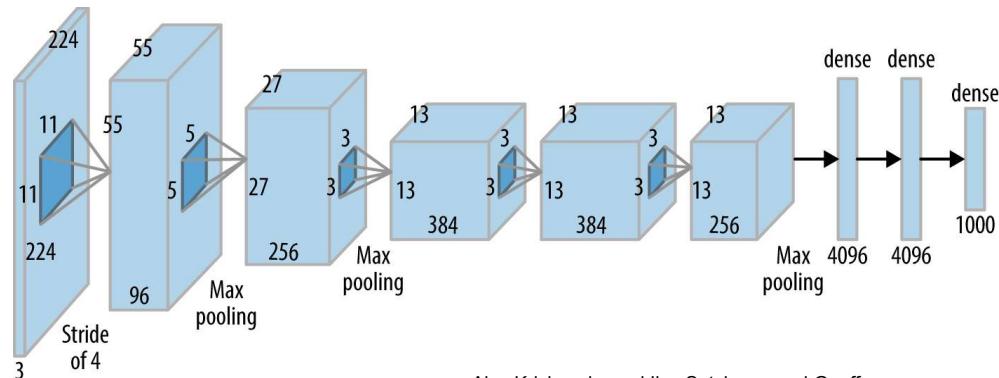
Feature Extraction (for Classification) with Deep NNs



Feature Extraction (for Classification) with Deep NNs

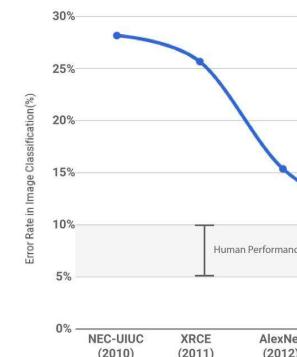


Example: AlexNet (2012)



Alex Krizhevsky and Ilya Sutskever and Geoffrey

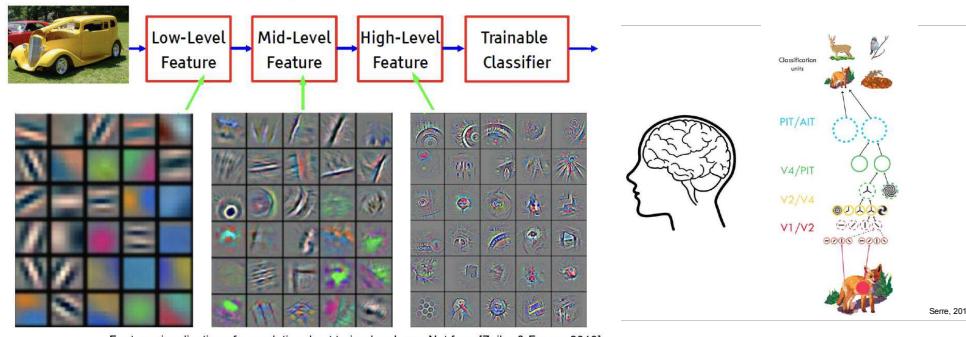
Development of Architectures



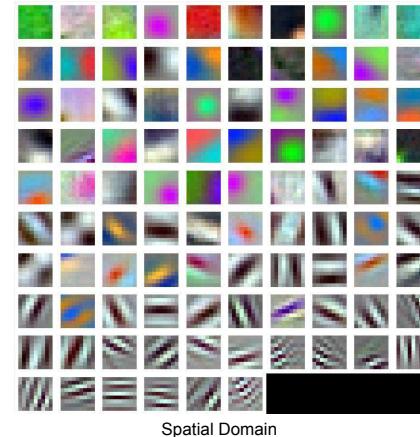
Decline in top 5 Error Rate for Neural Network Architecture for Image Classification

Top 5 Error Rate = correct Label not among top 5 Classifications

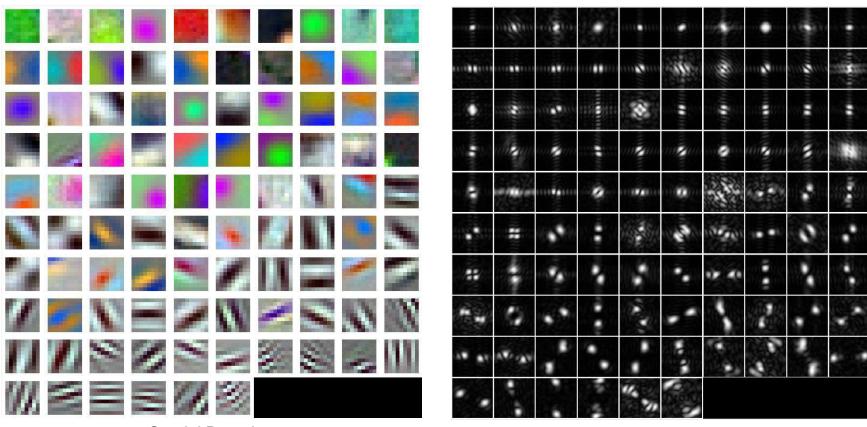
From Low-Level to High-Level Features



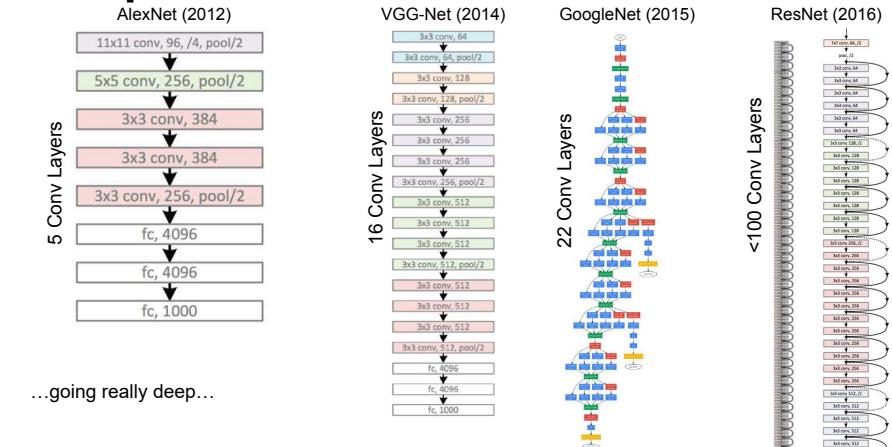
Example: 96 Kernels in Conv1



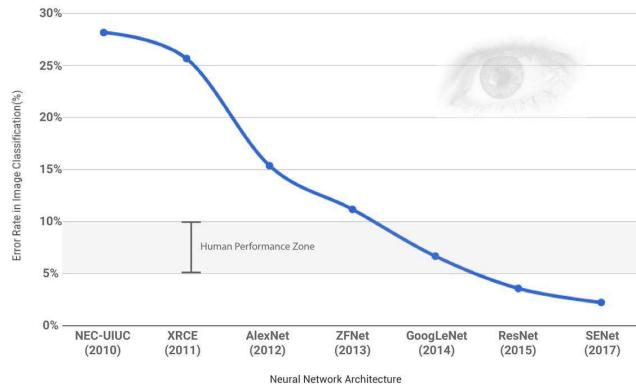
Example: 96 Kernels in Conv1



Development of Architectures



Development of Architectures



Decline in top 5 Error
Rate for Neural Network
Architecture for Image
Classification

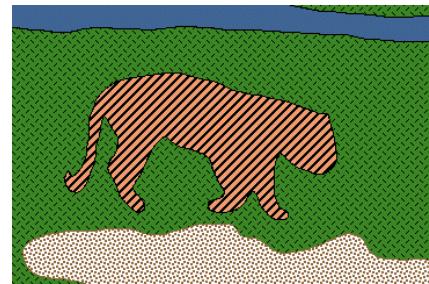
Top 5 Error Rate = correct
Label not among top 5
Classifications

Computer Vision



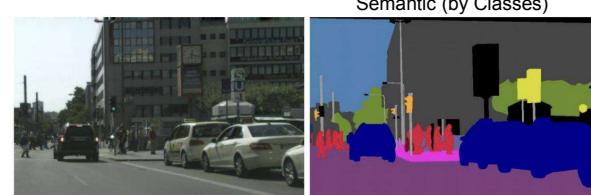
Lecture 6: Segmentation
Oliver Bimber

What is Segmentation?



Identify Groups of Pixels that go together

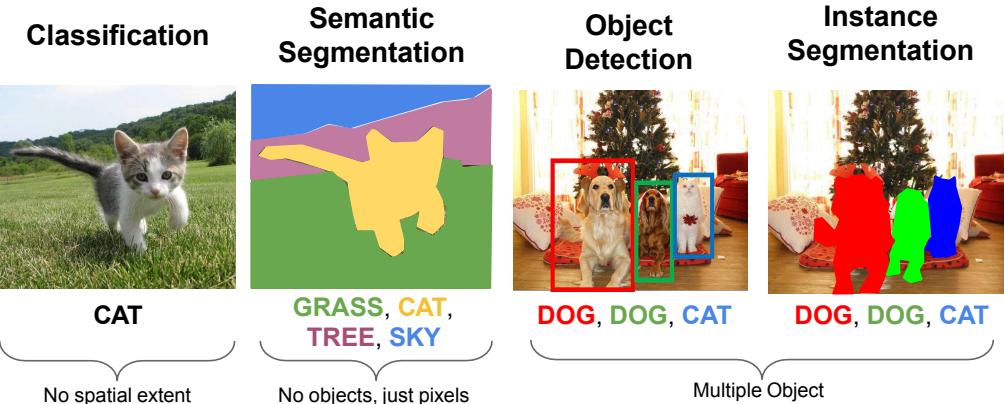
Types of Segmentation



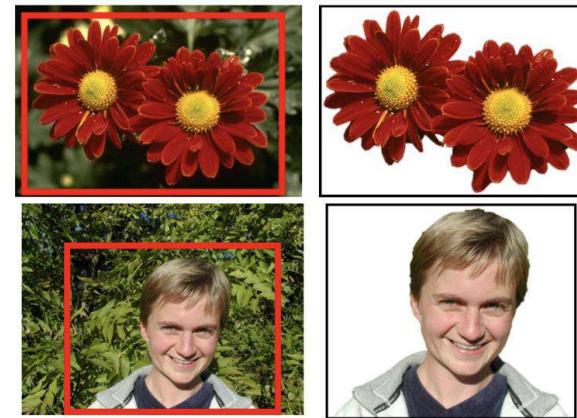
Semantic (by Classes)



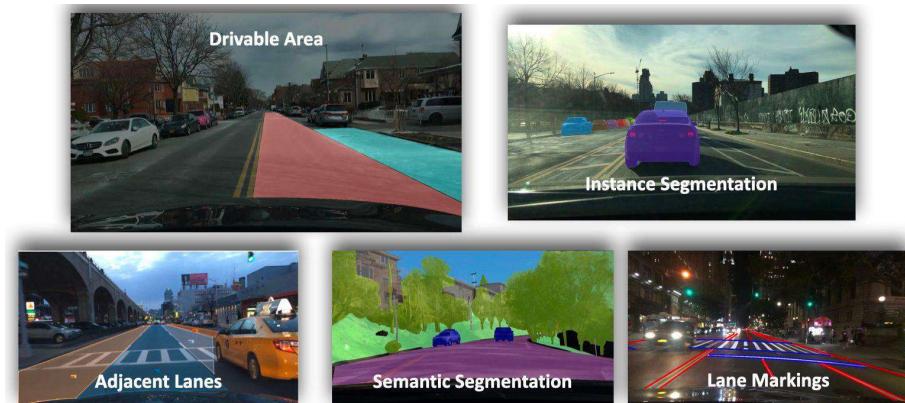
Classification vs. Segmentation



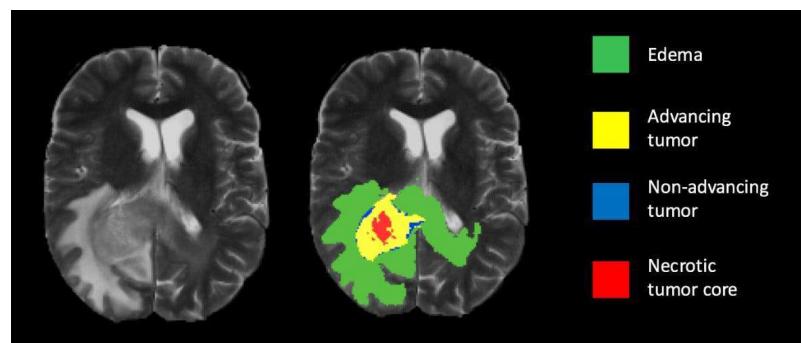
Example: Image Editing



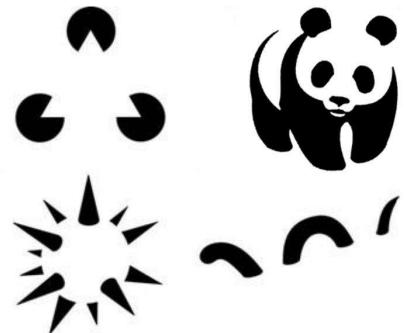
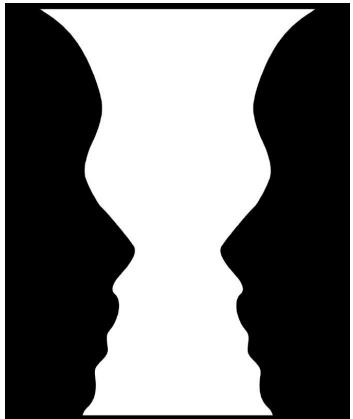
Example: Autonomous Driving



Example: Medical Imaging

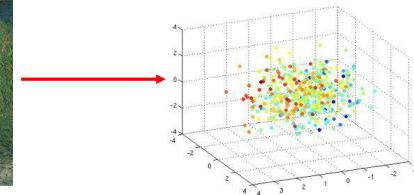


Segmentation done by Humans



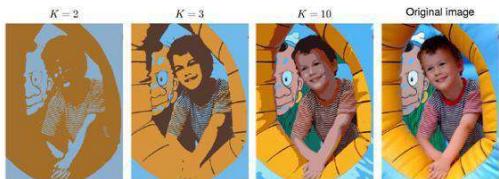
Illusory or subjective contours are perceived

Segmentation by Clustering



- Pixels are Points in a high-dimensional Space, eg.:
 - Color: 3d
 - Color + Location: 5d
- Cluster Pixels into Segments, eg.:
 - K-Means Clustering

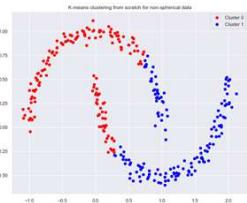
Example: K-Means Clustering



$$\Phi_{(\text{cluster}, \text{data})} = \sum_{i \in \text{cluster}} \left\{ \sum_{j \in \text{cluster}(i)} (x_j - c_i)^T (x_j - c_i) \right\}$$

K-means:

choose k points by random to act as cluster centers
until cluster centers are unchanged
 allocate each point to nearest cluster
 (ensure that each cluster has at least one point)
 replace cluster centers with mean of new cluster points
end



Example: K-Means Clustering

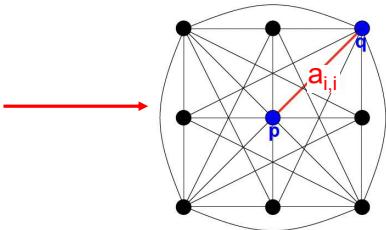
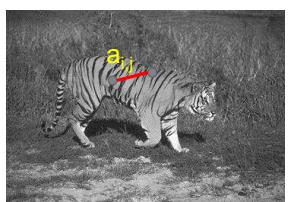


$$\Phi_{(\text{cluster}, \text{data})} = \sum_{i \in \text{cluster}} \left\{ \sum_{j \in \text{cluster}(i)} (x_j - c_i)^T (x_j - c_i) \right\}$$

K-means:

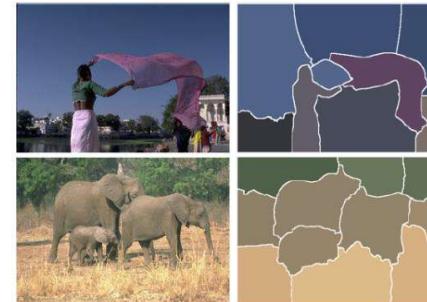
choose k points by random to act as cluster centers
until cluster centers are unchanged
 allocate each point to nearest cluster
 (ensure that each cluster has at least one point)
 replace cluster centers with mean of new cluster points
end

Segmentation with Graphs



- Pixels are Nodes in a Graph
 - Edge between Pairs of Pixels (i,j)
 - Affinity Weight $a_{i,j}$ for each Edge measures "Similarity"
- Cluster Pixels into Segments, eg.:
 - Clustering by Graph Eigenvectors, Graph Cut, Grab Cut

Example: Clustering by Graph Eigenvectors



Let's assume your Graph is represented with an Affinity Matrix \mathbf{A} (size: $N \times N$ for N Elements)

A good Cluster is one where Elements that are strongly connected to the Cluster also have large Values connecting one another in the Affinity Matrix

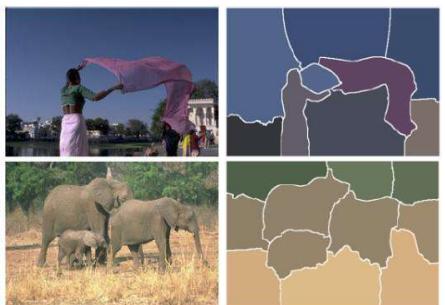
Let's assume you make an Assignment for Cluster n using a Weight Vector \mathbf{w}_n (with N elements, high values indicate strong connectivity to the Cluster)

When is use Assignment a good one?

Our objective from above rephrased mathematically:

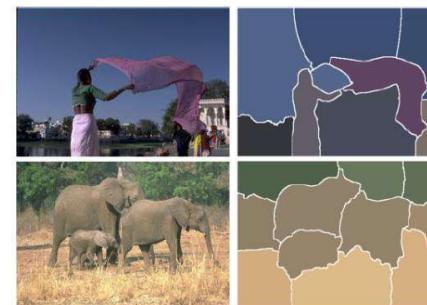
$$\text{Max}(\mathbf{w}_n^T \mathbf{A} \mathbf{w}_n = \lambda)$$

Example: Clustering by Graph Eigenvectors



$$\begin{array}{c}
 \text{affinity between element } i \text{ and } j \\
 X \\
 \text{association of elements } j \text{ with cluster } n \\
 \uparrow \\
 \boxed{\mathbf{w}_n^T \mathbf{A} \mathbf{w}_n = \lambda} \quad \mathbf{A} \mathbf{w}_n = \lambda \mathbf{w}_n \\
 \downarrow \\
 \text{association of elements } j \text{ with cluster } n \quad X \\
 \left[\begin{array}{c} \mathbf{w}_{n,0} \dots \mathbf{w}_{n,j} \end{array} \right] \left[\begin{array}{ccc} a_{0,0} & \dots & a_{i,0} \\ \dots & \dots & \dots \\ a_{0,j} & \dots & a_{i,j} \end{array} \right] \left[\begin{array}{c} \mathbf{w}_{n,0} \\ \dots \\ \mathbf{w}_{n,j} \end{array} \right]
 \end{array}$$

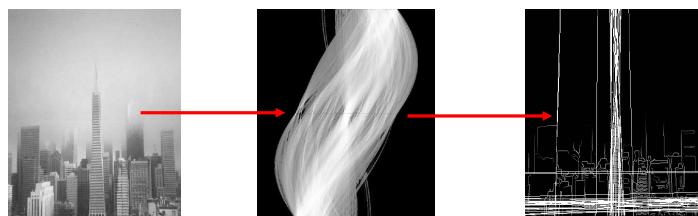
Example: Clustering by Graph Eigenvectors



Clustering by Graph Eigenvectors:
 construct affinity matrix \mathbf{A}
 compute eigenvalues and eigenvectors of \mathbf{A}
 until there are sufficient clusters
 take the eigenvector corresponding to
 next largest eigenvalue
 assign elements to cluster (multiply
 eigenvector with \mathbf{A} and threshold)
 zero out all clustered elements in \mathbf{A}

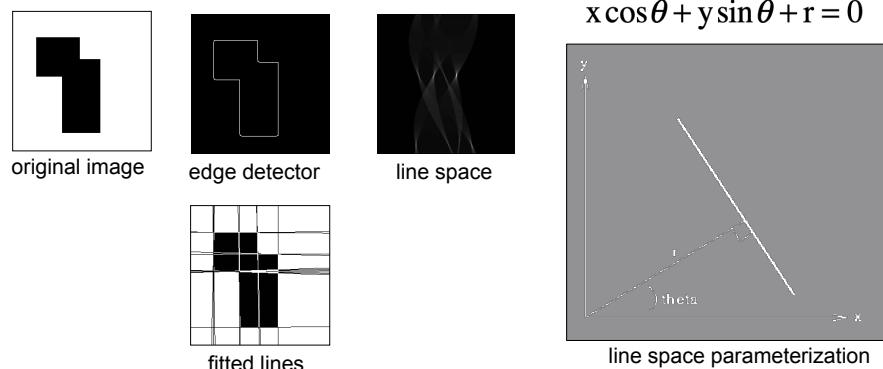
end

Segmentation by Fitting



- Knowing the Shapes of the Segments
 - Segments can be parameterized
 - Transform Pixels into Parameter Space, eg.:
 - Hough Transform

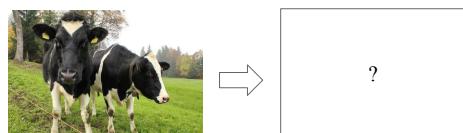
Example: Hough Transform



Segmentation by Learning



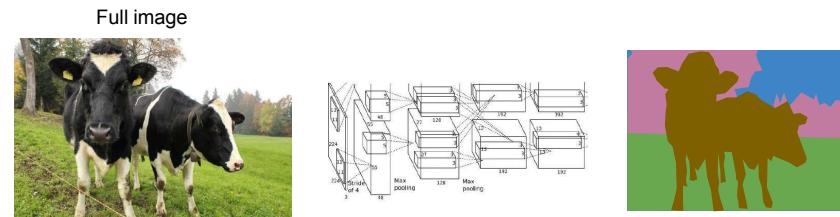
GRASS, CAT, TREE, SKY, ...



At Test Time, Classify each Pixel of a new Image

Paired Training Data: for each Training Image, each Pixel is labeled with a Semantic Category

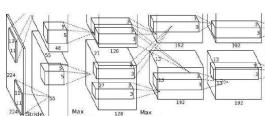
Segmentation using CNNs



An intuitive Idea: encode the entire Image with a CNN, and do Semantic Segmentation on top

Segmentation using CNNs

Full image

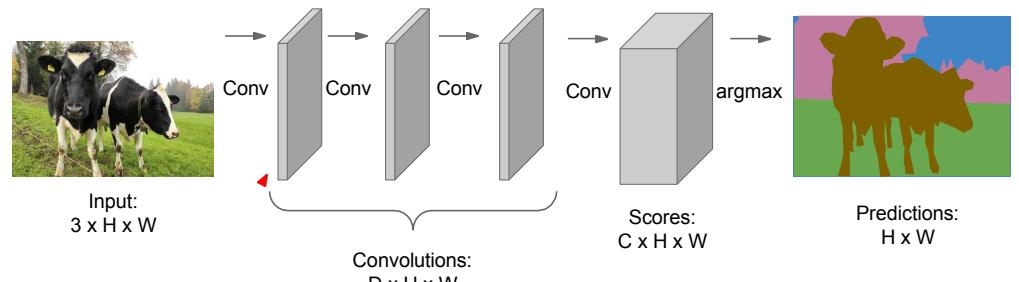


An intuitive Idea: encode the entire Image with a CNN, and do Semantic Segmentation on top

Problem: Classification Architectures often reduce Feature spatial Sizes to go deeper, but Semantic Segmentation requires the Output Size to be the same as Input Size

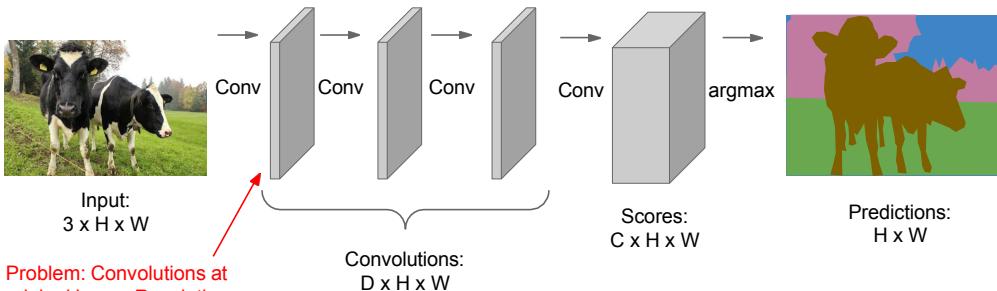
Segmentation using CNNs

Design a Network with only Convolutional Layers without Downsampling Operators to make Predictions for Pixels all at once!



Segmentation using CNNs

Design a Network with only Convolutional Layers without Downsampling Operators to make Predictions for Pixels all at once!



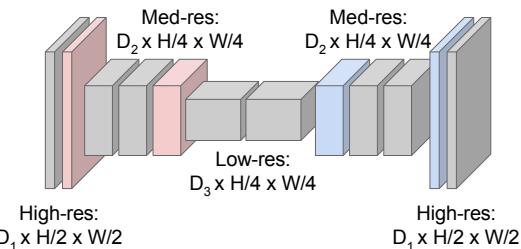
Segmentation using CNNs

Downsampling:
Pooling, Strided
Convolution



Input:
 $3 \times H \times W$

Design Network as a Bunch of Convolutional Layers, with **Downsampling** and **Upsampling** inside the Network!

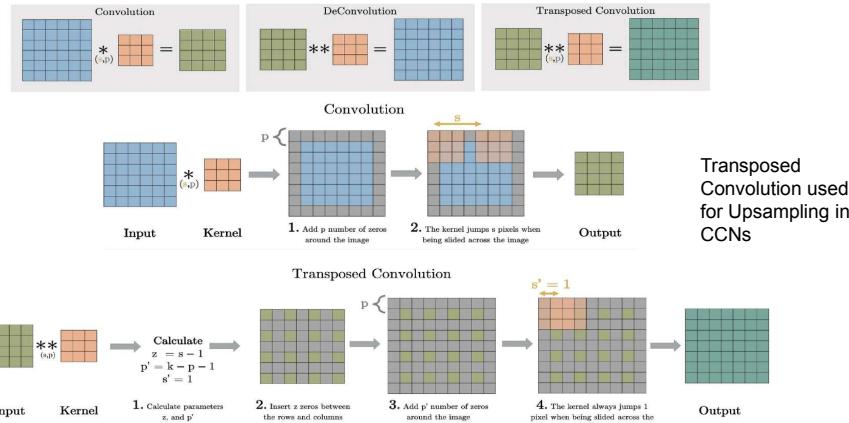


Upsampling:
Strided Transposed
Convolution (or Unpooling)

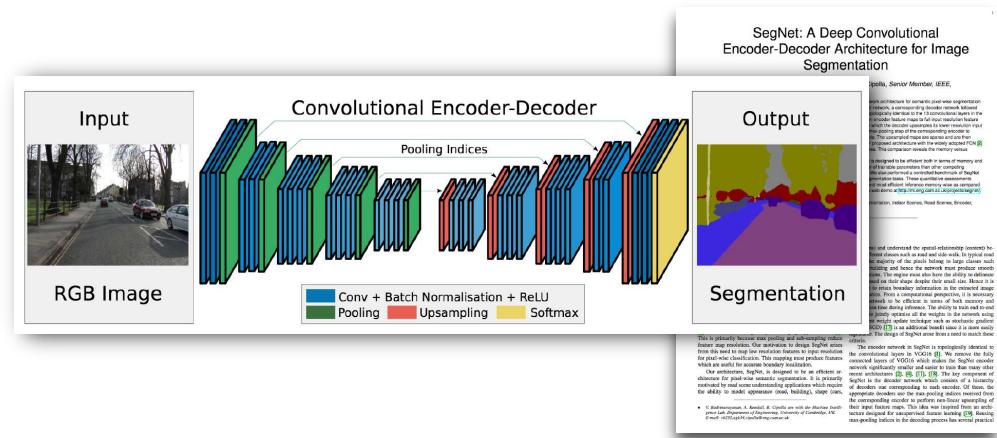


Predictions:
 $H \times W$

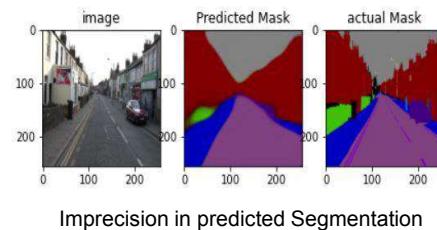
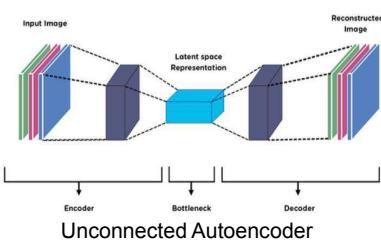
Recap: Transposed Convolution



Autoencoder: Encoder-Decoder Architectures

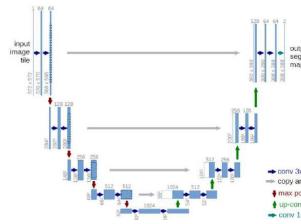


Unconnected Autoencoders

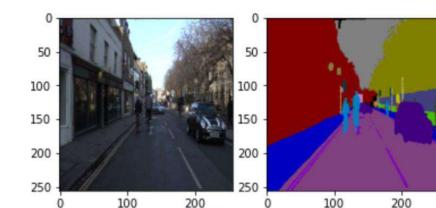


However, convolution and pooling during encoding and transposed convolution during decoding leads to a imprecise feature prediction

Connected Autoencoders (U-Nets)



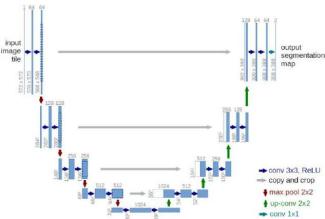
Connected Autoencoder (U-Net)



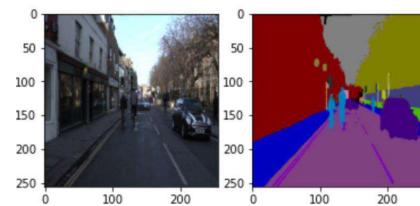
Predicted Segmentation

- U-Nets overcome this problem by connecting corresponding encoder-decoder layers with skip connections:
 - the output of an encoder level is skip-connected (concatenation) with the input of the corresponding decoder level

Connected Autoencoders (U-Nets)



Connected Autoencoder (U-Net)



Predicted Segmentation

Computer Vision



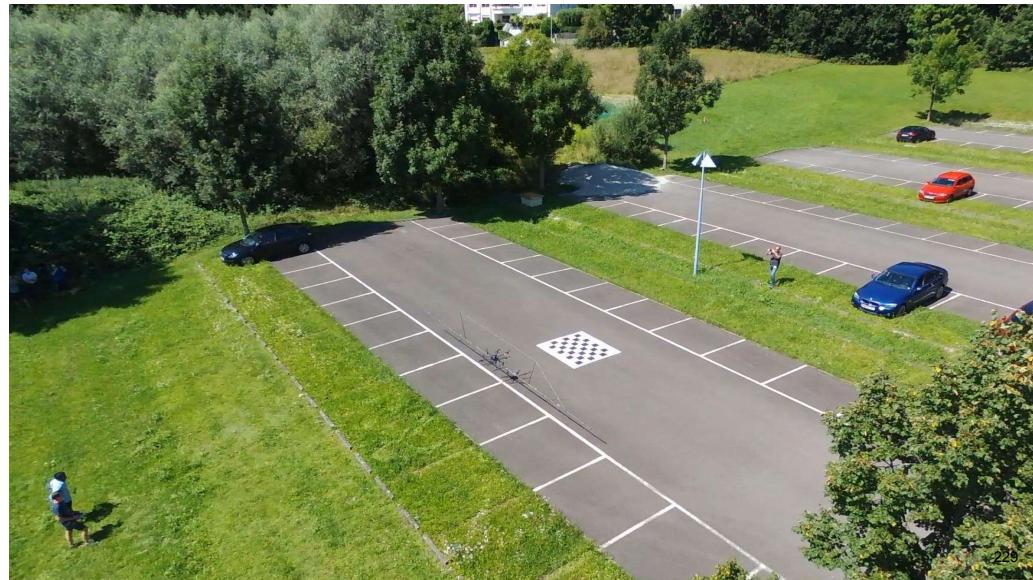
Lecture 7: Optical Flow
Oliver Bimber



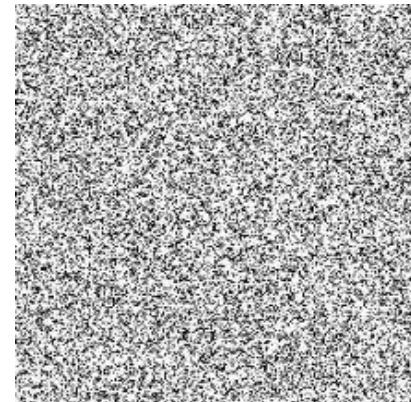
Research Example: Through-Foliage
Detection and Tracking



Research Example: Through-Foliage
Detection and Tracking



Perception of Motion

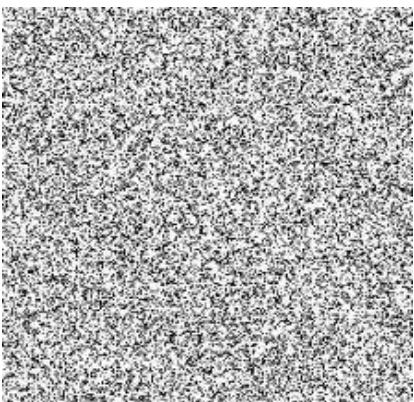


Can you see a Patter?

JKU JOHANNES KEPLER
UNIVERSITY LINZ

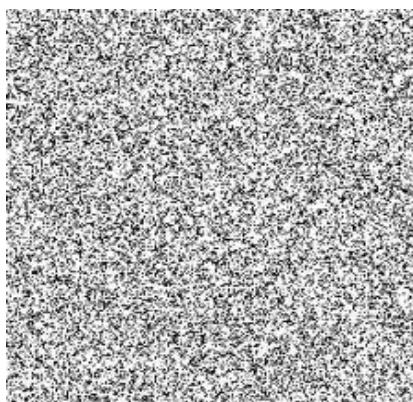
230

Perception of Motion



Can you see a Patter?

JKU JOHANNES KEPLER
UNIVERSITY LINZ

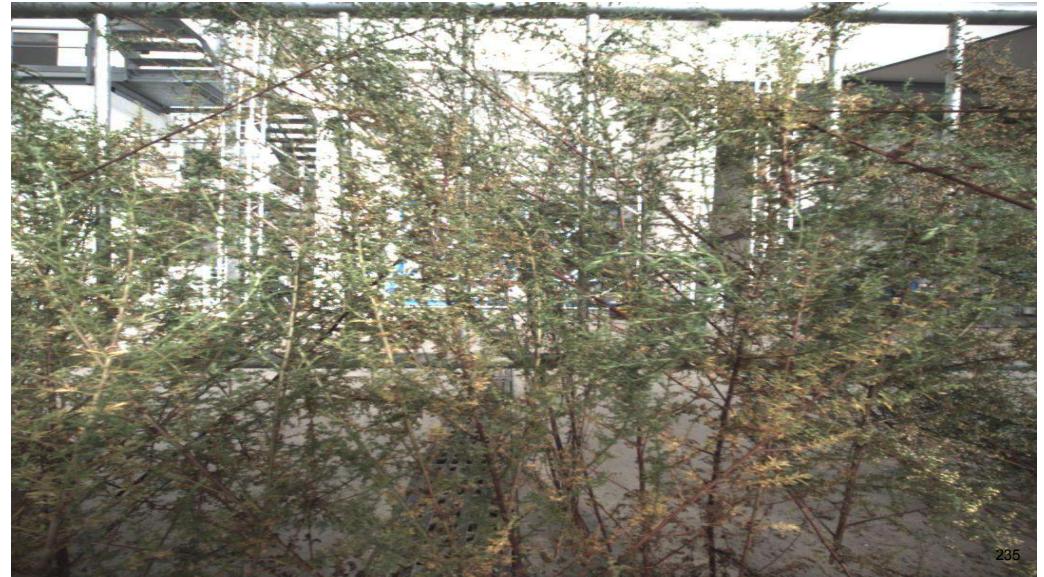
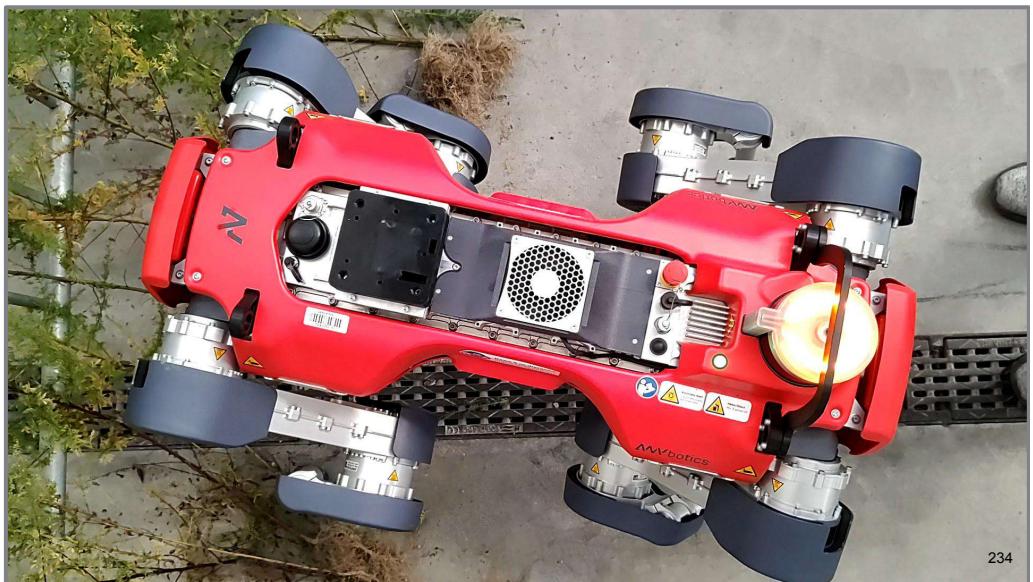


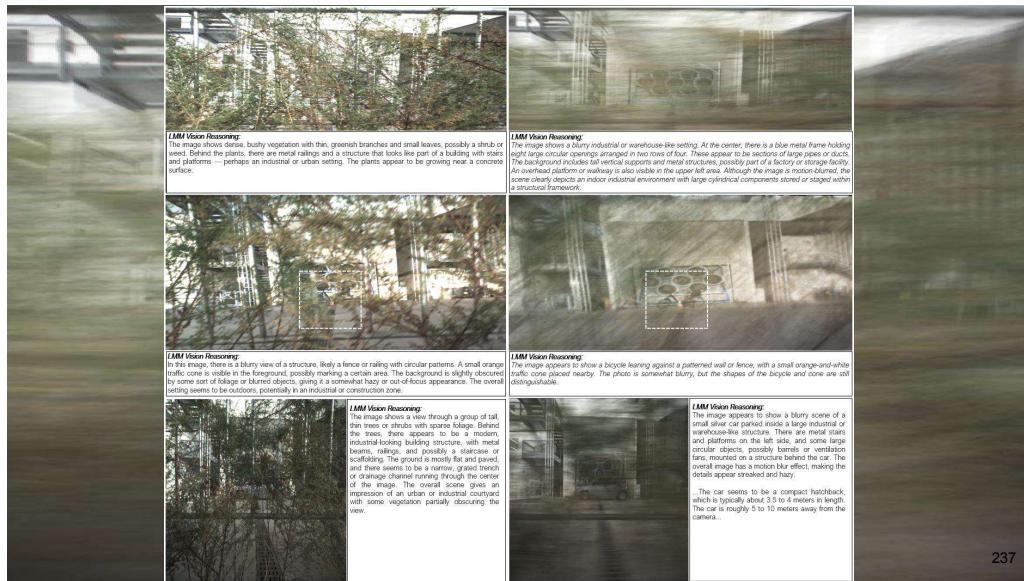
Sometimes Motion is the only Cue

231



232





Perception of Motion



Can you see a Patter?

JYU JOHANNES KEPLER
UNIVERSITY LINZ

238

Perception of Motion



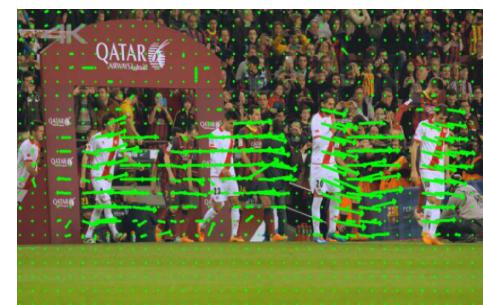
Can you see a Patter?

Sometimes Motion is the only Cue
(even for sparse Data)

JYU JOHANNES KEPLER
UNIVERSITY LINZ

239

What is Optical Flow?



Sparse Optical Flow (Flow Vector per Region)



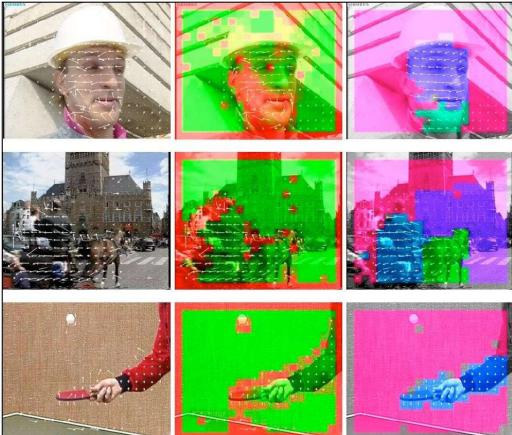
Dense Optical Flow (Flow Vector per Pixel)

Motion Vector of Pixel in Time Series (two consecutive Video Frames at Times t and t+1)

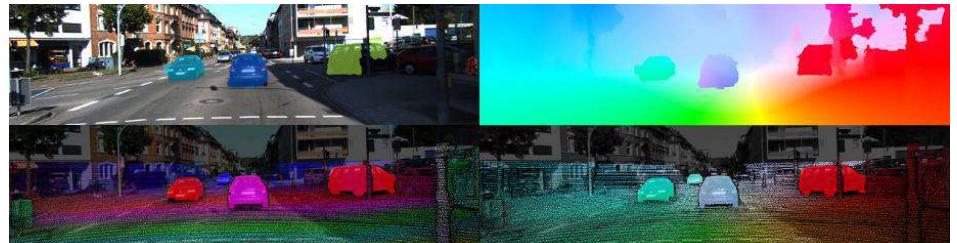
JYU JOHANNES KEPLER
UNIVERSITY LINZ

240

Example: MPEG Compression



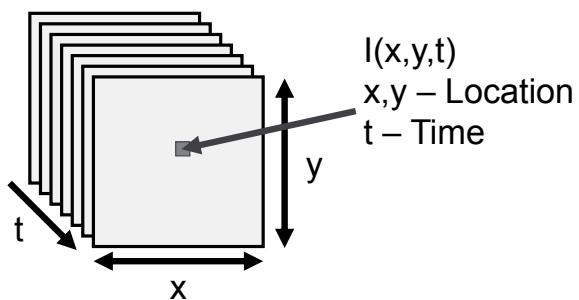
Example: Autonomous Driving



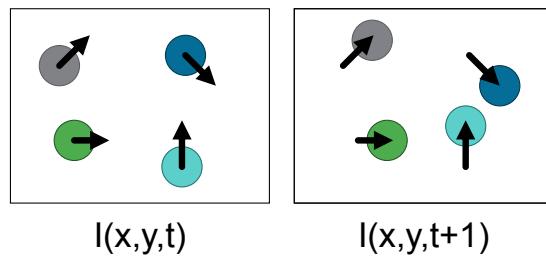
Example: Traffic Monitoring



Video: A Sequence of Frames over Time

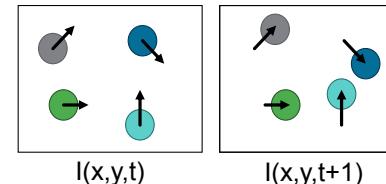


Optical Flow



Want to estimate Pixel Motion from Image $I(x,y,t)$ to Image $I(x,y,t+1)$

Assumption



Solve correspondence Problem: given Pixel at Time t , find **nearby** Pixels of the **same Color** at Time $t+1$

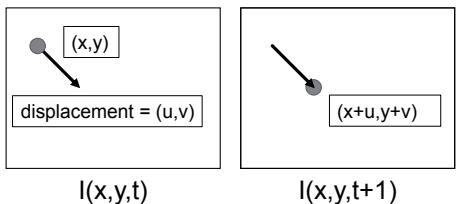
Key assumptions:

- **Color/Brightness Constancy**: Point at Time t looks same at Time $t+1$
- **Small Motion**: Points do not move very far

The Optical Flow Equation

Brightness Constancy:

$$I(x + u, y + v, t + 1) = I(x, y, t)$$

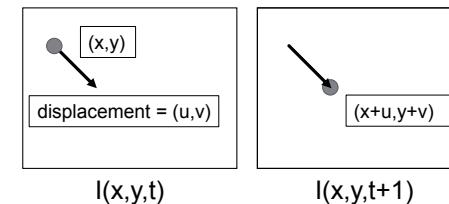


The Optical Flow Equation

Brightness Constancy:

$$I(x + u, y + v, t + 1) = I(x, y, t)$$

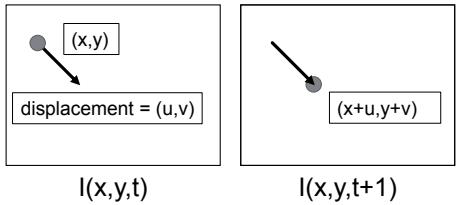
$$0 \approx I(x + u, y + v, t + 1) - I(x, y, t)$$



Taylor Expansion:

$$\begin{aligned} &\approx I(x, y, t + 1) + I_x u + I_y v - I(x, y, t) \\ &= I(x, y, t + 1) - I(x, y, t) + I_x u + I_y v \end{aligned}$$

The Optical Flow Equation



Brightness Constancy:

$$I(x + u, y + v, t + 1) = I(x, y, t)$$

$$0 \approx I(x + u, y + v, t + 1) - I(x, y, t)$$

Taylor Expansion:

$$\approx I(x, y, t + 1) + I_x u + I_y v - I(x, y, t)$$

$$= I(x, y, t + 1) - I(x, y, t) + I_x u + I_y v$$

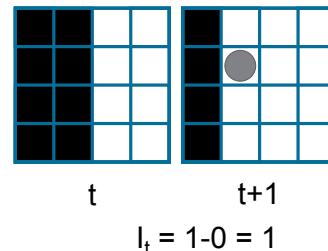
Optical Flow Equation:

$$0 = I_t + I_x u + I_y v$$

$$= I_t + \nabla I \cdot [u, v]$$

Example

$$I_x u + I_y v + I_t = 0$$



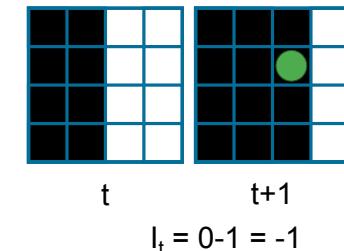
@ ●

$$I_t = 1-0 = 1$$

$$I_y = 0$$

$$I_x = 1-0 = 1$$

What's u?



@ ●

$$I_t = 0-1 = -1$$

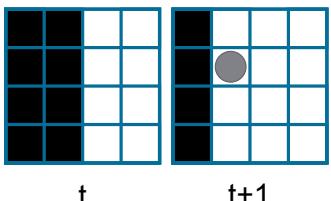
$$I_y = 0$$

$$I_x = 1-0 = 1$$

What's u?

Example

$$I_x u + I_y v + I_t = 0$$



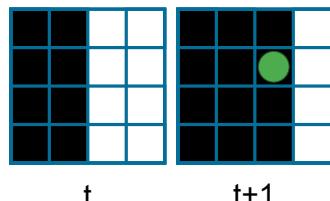
@ ●

$$I_t = 1-0 = 1$$

$$I_y = 0$$

$$I_x = 1-0 = 1$$

What's u?



@ ●

$$I_t = 0-1 = -1$$

$$I_y = 0$$

$$I_x = 1-0 = 1$$

What's u?

How to overcome this Problem?

Lucas-Kanade 1981

$$I_t + I_x u + I_y v = 0$$

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

- 2 Unknowns $[u, v]$, 1 Equation per Pixel
- How do we get more Equations?
- Assume **Spatial Coherence**: Pixel's Neighbors have same $[u, v]$ (same Optical Flow on local Region)
- Example: 5x5 Window gives 25 Equations

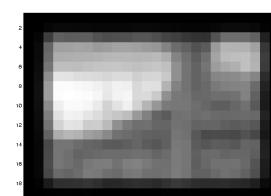
What if Motion is larger than one Pixel?



JKU JOHANNES KEPLER
UNIVERSITY LINZ

253

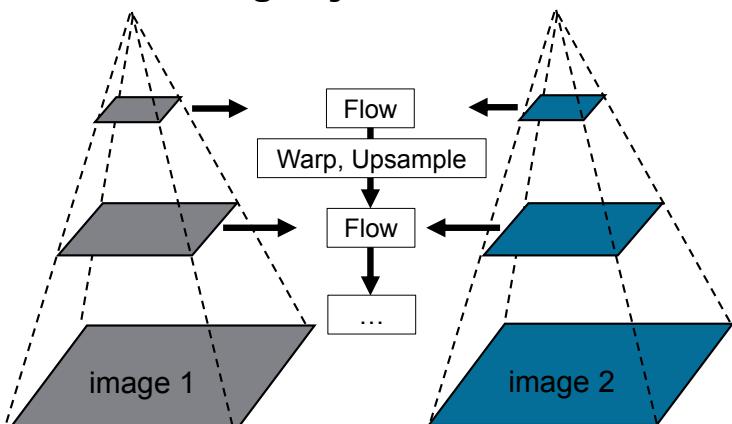
Reduce Resolution!



JKU JOHANNES KEPLER
UNIVERSITY LINZ

254

Using Gaussian Image Pyramids



JKU JOHANNES KEPLER
UNIVERSITY LINZ

255

Lucas-Kanade Optical Flow



JKU JOHANNES KEPLER
UNIVERSITY LINZ

256

Dense Optical Flow



Key Assumption:

- Most Objects in the World are rigid or deform elastically and move together coherently
- We expect the flow fields to be **SMOOTH**

Basic Idea:

- Enforce Brightness Constancy for every Pixel
- Enforce Smoothness Constraint for Flow Vectors

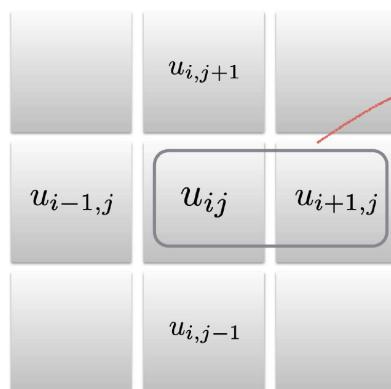
Enforce Brightness Constancy

$$I_x u + I_y v + I_t = 0$$

For every Pixel (i,j):

$$\min_{u,v} [I_x(i,j)u_{ij} + I_y(i,j)v_{ij} + I_t(i,j)]^2$$

Enforce Smoothness Constraint



$$\min_u (u_{i,j} - u_{i+1,j})^2$$

Same for v

Objective Function

Horn-Schunck 1981

$$\min_{u,v} \sum_{i,j} \{ E_d(i,j) + \lambda E_s(i,j) \}$$

λ = regularization constant (larger \rightarrow more smooth optical flow)

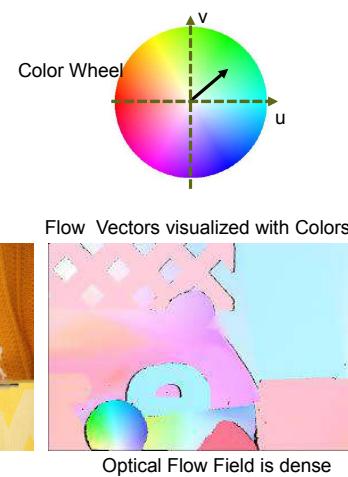
Brightness Constraint

$$E_d(i,j) = [I_x(i,j)u_{ij} + I_y(i,j)v_{ij} + I_t(i,j)]^2$$

Smoothness Constraint

$$E_s(i,j) = \frac{1}{4} [(u_{ij} - u_{i+1,j})^2 + (u_{ij} - u_{i,j+1})^2 + (v_{ij} - v_{i+1,j})^2 + (v_{ij} - v_{i,j+1})^2]$$

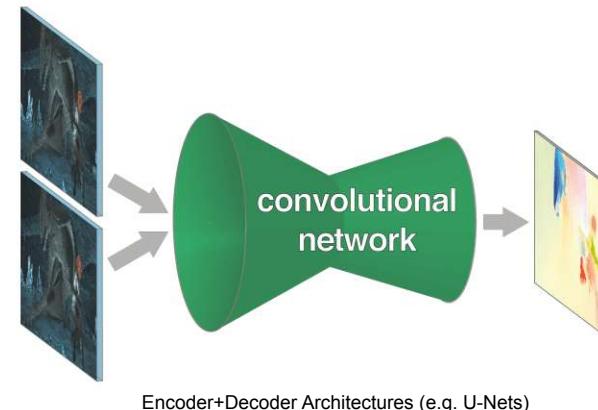
Horn-Schunck Optical Flow



JKU JOHANNES KEPLER
UNIVERSITY LINZ

261

Optical Flow and Machine Learning

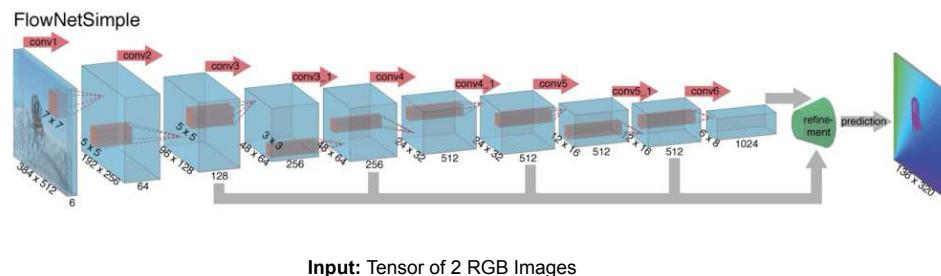


JKU JOHANNES KEPLER
UNIVERSITY LINZ

262

Example: FlowNetS (Simple)

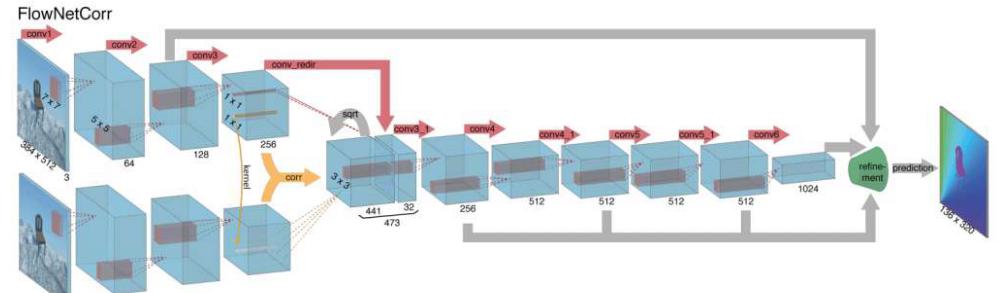
<https://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15/flownet.pdf>



263

Example: FlowNetCorr (Correlation)

<https://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15/flownet.pdf>



JKU JOHANNES KEPLER
UNIVERSITY LINZ

264

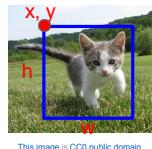
Computer Vision



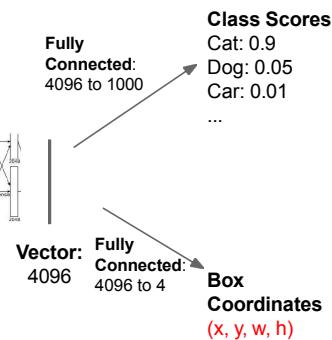
Lecture 8: Object Detection
Oliver Bimber

JKU JOHANNES KEPLER
UNIVERSITY LINZ

Object Detection: Single Object

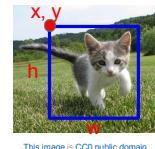


This image is CC0 public domain

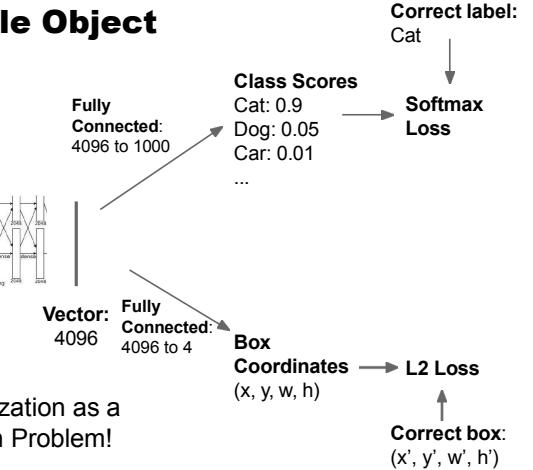


267

Object Detection: Single Object



This image is CC0 public domain

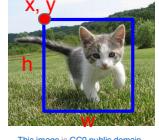


268

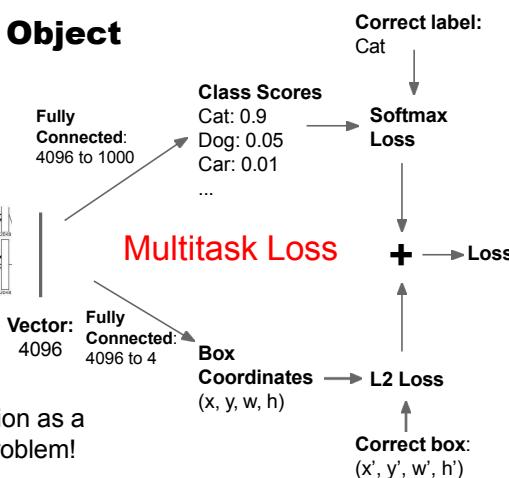
JKU JOHANNES KEPLER
UNIVERSITY LINZ

JKU JOHANNES KEPLER
UNIVERSITY LINZ

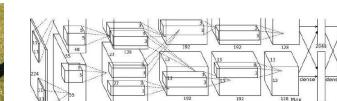
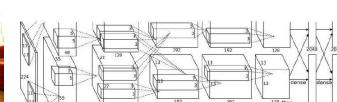
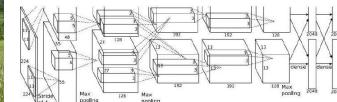
Object Detection: Single Object



Treat localization as a Regression Problem!

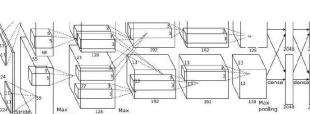
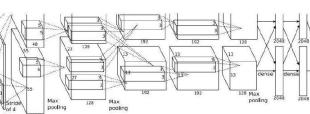


Object Detection: Multiple Objects



Object Detection: Multiple Objects

Each Image needs a different Number of Outputs!

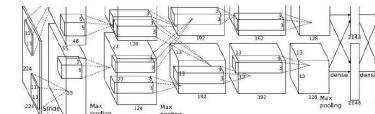
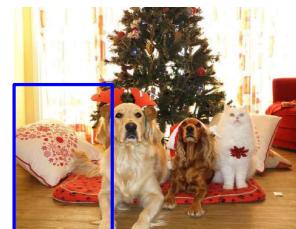


DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

Many numbers!

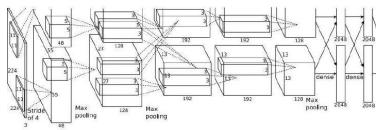
Object Detection: Multiple Objects

Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background



Object Detection: Multiple Objects

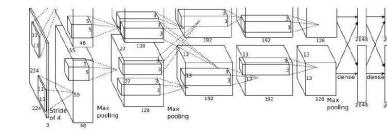
Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

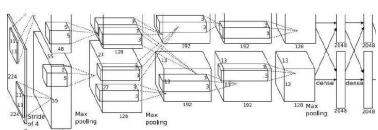
Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background

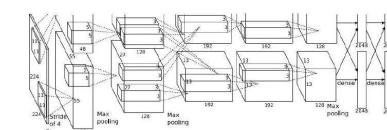
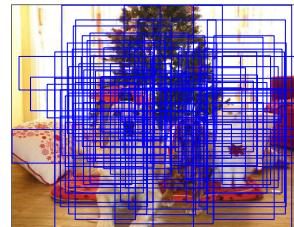


Dog? NO
Cat? YES
Background? NO

Q: What's the Problem with this Approach?

Object Detection: Multiple Objects

Apply a CNN to many different Crops of the Image, CNN classifies each Crop as Object or Background

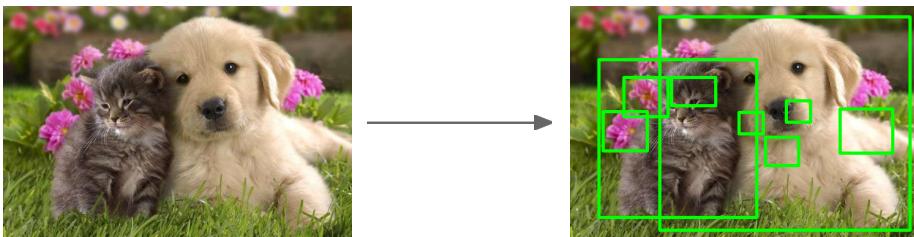


Dog? NO
Cat? YES
Background? NO

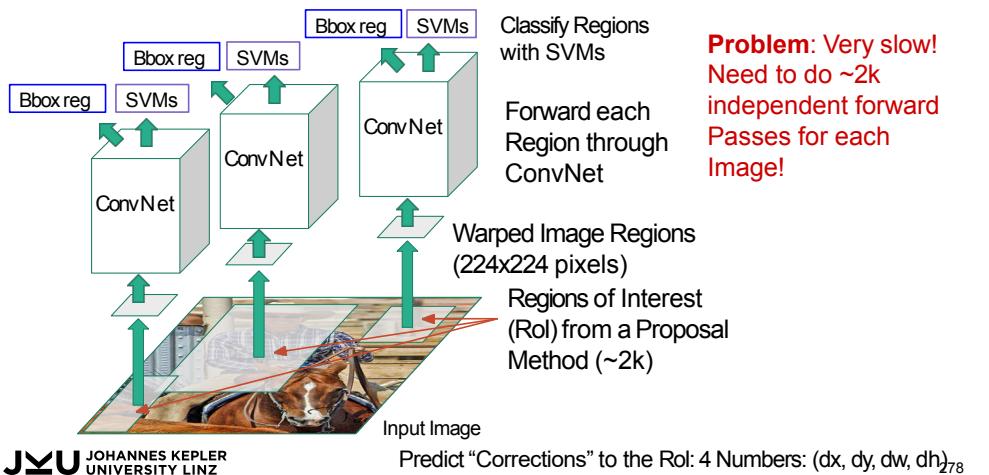
Problem: Need to apply CNN to huge Number of Locations, Scales, and Aspect Ratios, very computationally expensive!

Region Proposal: Selective Search

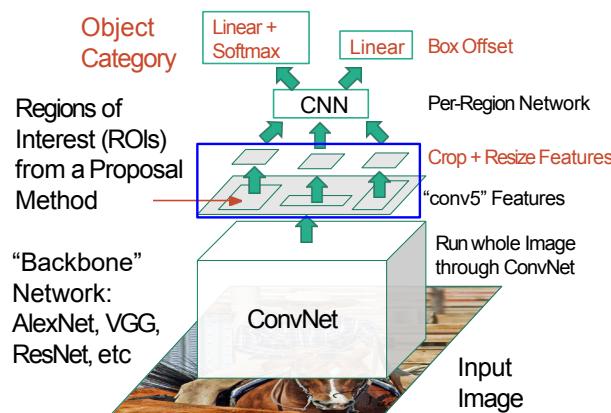
- Find “blobby” Image Regions that are likely to contain Objects
- Relatively fast to run; e.g. Selective Search gives 2000 Region Proposals in a few Seconds on CPU



Regional-Based (R)-CNN



Fast R-CNN

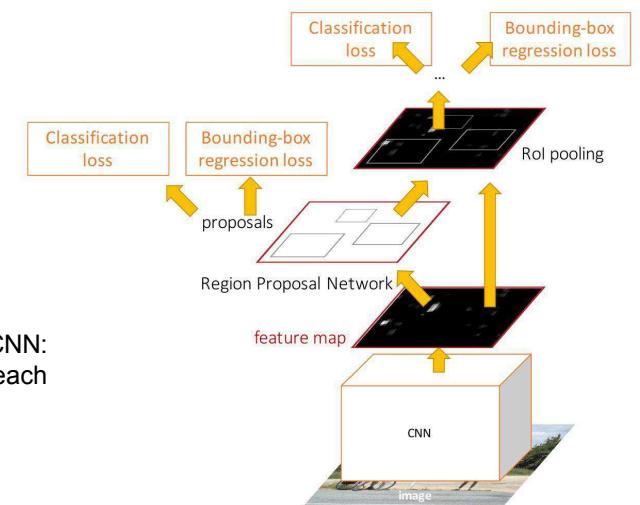


Faster R-CNN

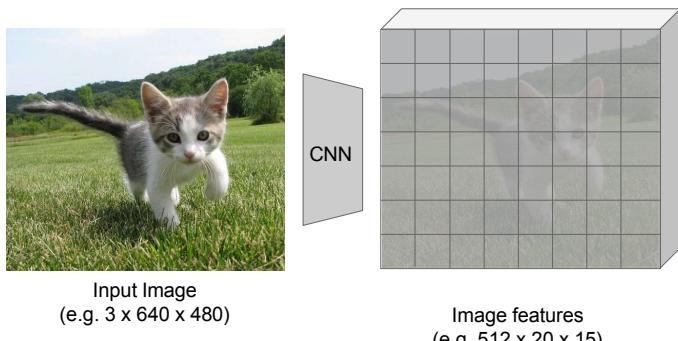
Make CNN do Proposals!

Insert Region Proposal Network (RPN) to predict Proposals from Features

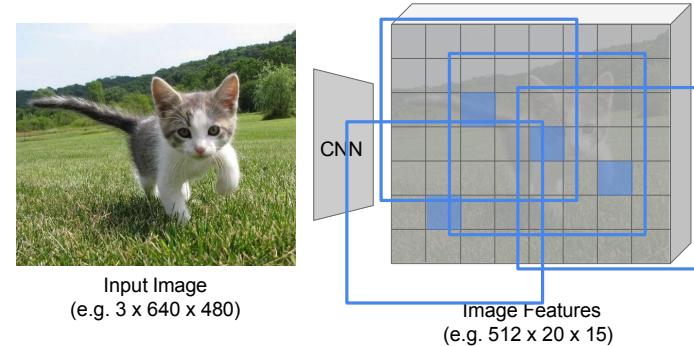
Otherwise same as Fast R-CNN:
Crop Features for each Proposal, Classify each one



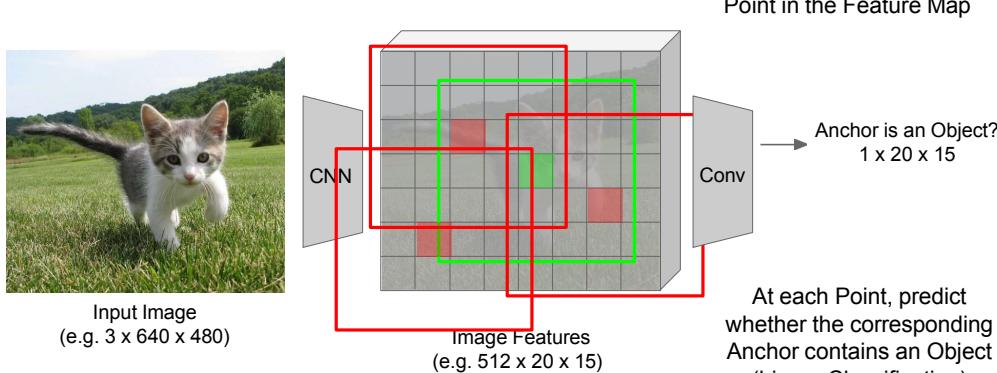
Region Proposal Network



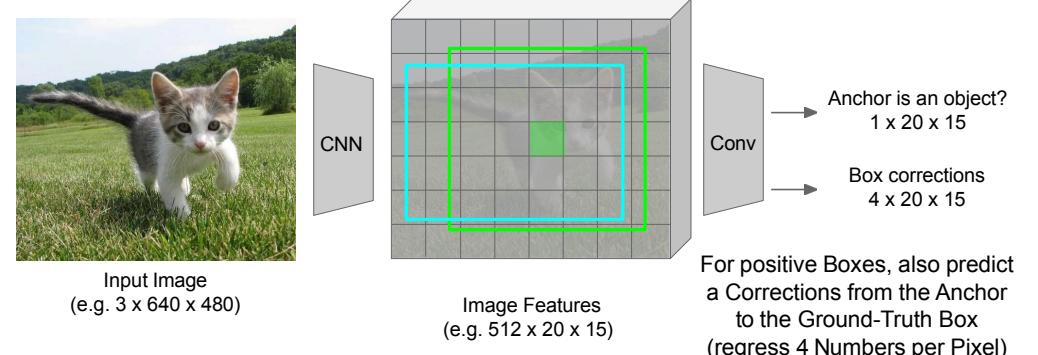
Region Proposal Network



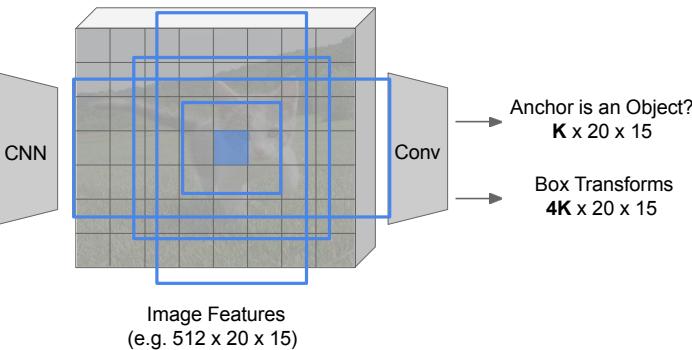
Region Proposal Network



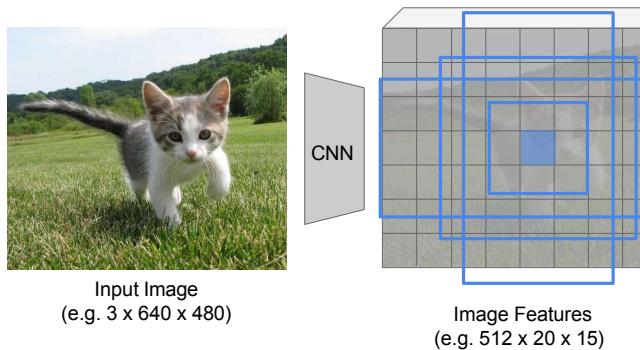
Region Proposal Network



Region Proposal Network



Region Proposal Network



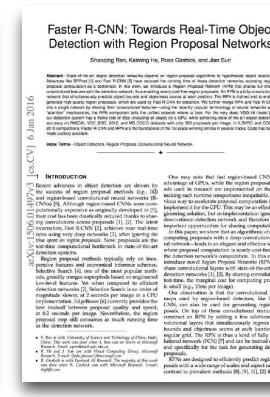
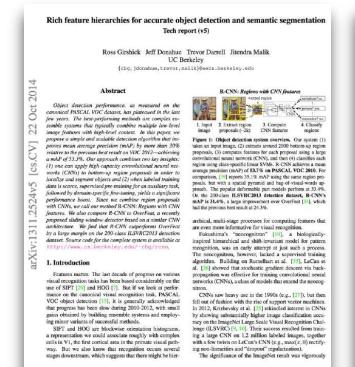
In practice use K different Anchor Boxes of different Size / Scale at each Point

Anchor is an Object?
 $K \times 20 \times 15$

Box Transforms
 $4K \times 20 \times 15$

Sort the $K \times 20 \times 15$ Boxes by their "Objectness" Score,
take top ~ 300 as our
Proposals

R-CNN, Fast R-CNN, Faster R-CNN



https://arxiv.org/pdf/1311.2524.pdf

https://arxiv.org/pdf/1504.08083.pdf

https://arxiv.org/pdf/1506.01497.pdf

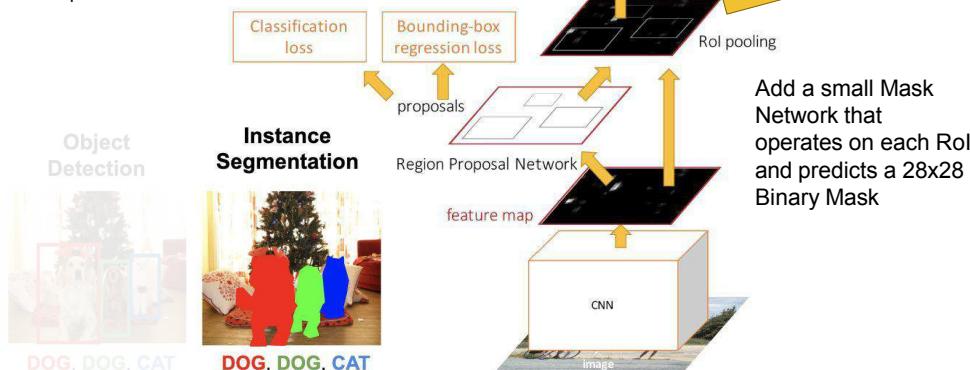
Object Detection

Example: Faster R-CNN



Instance Segmentation

Example: Masked R-CNN

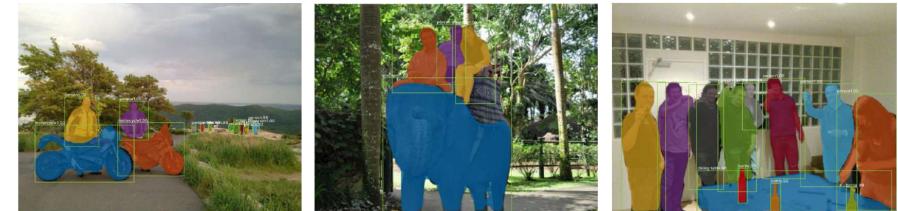


He et al., "Mask R-CNN", ICCV 2017

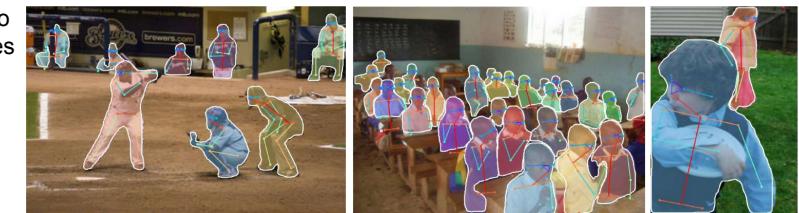
JKU JOHANNES KEPLER
UNIVERSITY LINZ

289

Instance Segmentation



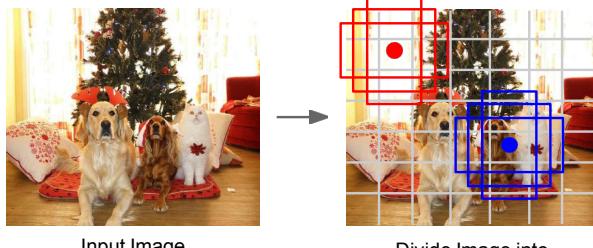
Can also do Poses



JKU JOHANNES KEPLER
UNIVERSITY LINZ

290

Single-Shot Object Detectors: YOLO/SSD/RetinaNet



Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al., "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al., "Focal Loss for Dense Object Detection", ICCV 2017

JKU JOHANNES KEPLER
UNIVERSITY LINZ

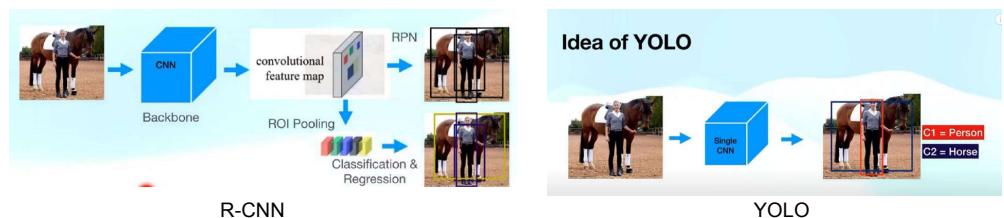
- Within each Grid Cell:
- Regress from each of the B Base Boxes to a final Box with 5 Numbers: (dx, dy, dh, dw, confidence)
- Predict Scores for each of C Classes (including Background as a Class)
- Looks a lot like RPN, but category-specific!
- Only a single Stage needed (Classification and prediction of BB in the same step)

Output:
 $7 \times 7 \times (5 * B + C)$

291

YOLO: You Only Look Once

One CNN to predict Bounding Boxes and Classes

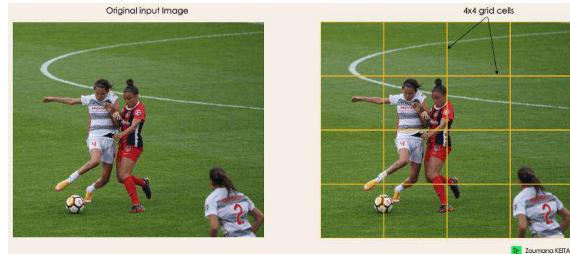


JKU JOHANNES KEPLER
UNIVERSITY LINZ

292

YOLO: You Only Look Once

Step 1 (Residual Blocks):
Split image into Grid of SxS
(in original paper: 7x7)
equal Cells that go through
the CNN



The Key Question is, which Cell is responsible for predicting an Object? → The Cell where the center of the object falls

YOLO: You Only Look Once

Step 2 (Bounding Box Regression):

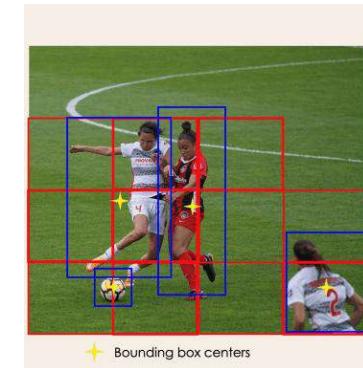
For each Cell predict (regression) Bounding Box Parameters:

$$Y = [pc, bx, by, bh, bw, c1, c2, \dots]$$

pc: Probability of containing an Object
(Confidence Score)

bx,by,bh,bw: bounding box coordinates wrt.
cell center

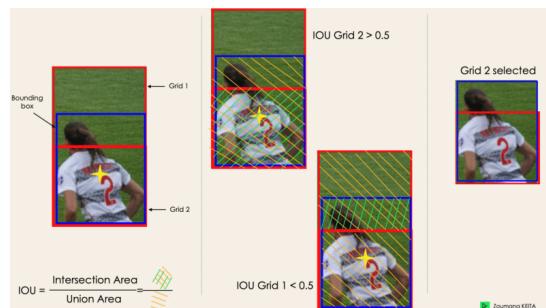
c1,c2,...: contained classes



YOLO: You Only Look Once

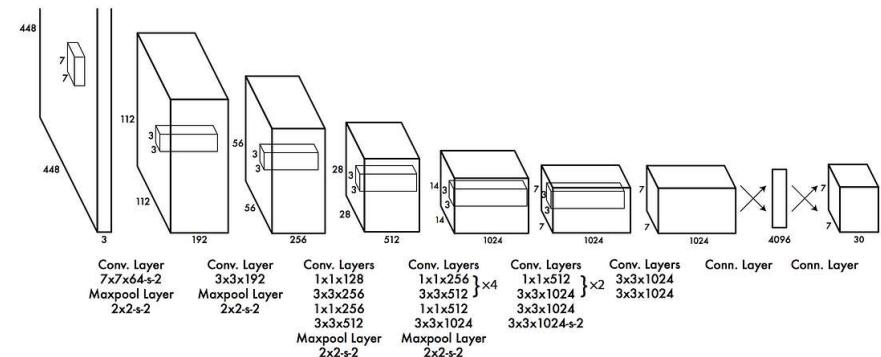
Step 3 (Intersection over Union):
discard cells with too small IoU (ignore
predictions)

Step 4: (Non-Maximum Suppression):
reduce too many noisy BBs per Object
- Select BB with highest Confidence
Score and suppress all other BBs with
high IoU wrt. Selection
- Repeat until no BB can be selected
anymore

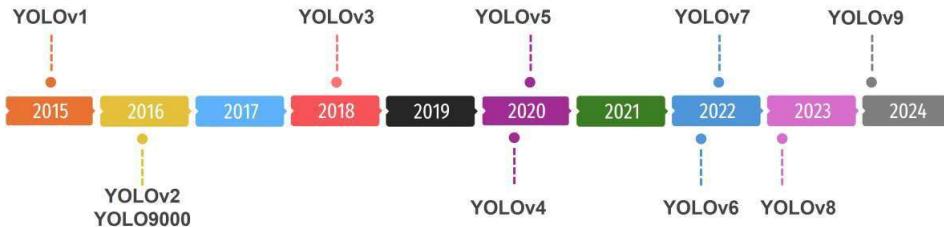


YOLO: You Only Look Once

YOLOv1: Architecture



YOLO: You Only Look Once



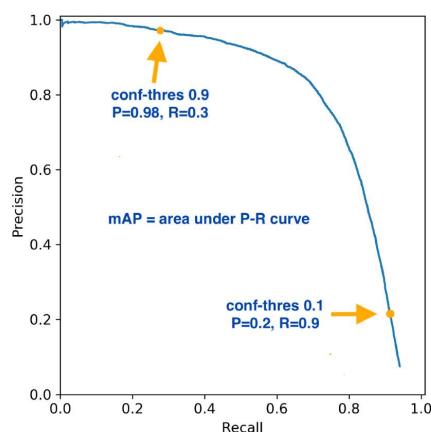
Precision over Recall (P-R) Curve

Let's now consider many Confidence Score Thresholds, each leading to individual Precision and Recall Values

A well trained Classifier has a large Area under the P-R curve (for all Classes)!

That area is called **Average Precision**, **AP** (0..1)

AP can be determined for each Class, and the mean over them is called **Mean Average Precision**, **mAP** (0..1)



Evaluating Classification Performance

We decide (e.g., based on a confidence score threshold) whether a classification is considered correct or not, and this leads to the following cases:

	Predicted Positive	Predicted Negative
Actual Positive	True Positives (TP)	False Negative (FN)
Actual Negative	False Positives (FP)	True Negatives (TN)

These cases can now be formalized into different metrics:

Accuracy = $(TP+TN)/(TP+TN+FP+FN)$: the proportion of all classifications that were correct, whether positive or negative

Precision = $TP/(TP+FP)$: proportion of all the model's positive classifications that are actually positive

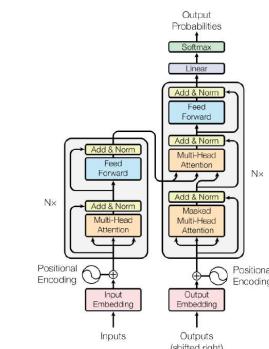
Recall = $TP/(TP+FN)$: the proportion of all actual positives that were classified correctly as positives

False Positive Rate = $FP/(FP+TN)$: the proportion of all actual negatives that were classified incorrectly as positives

A Note on (Vision-) Transformers

- Transformers used for sequences (e.g., in NLP)

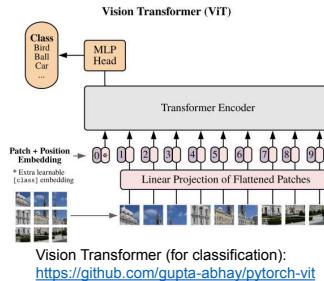
- in contrast to deep CNNs or RNNs can better handle long-distance dependencies (vanishing / exploding gradient problem)
- relies on attention (e.g., determining context = semantic relations for words at particular positions in a sentence)



Transformer:
<https://arxiv.org/abs/1706.03762>

A Note on (Vision-) Transformers

- Vision Transformers (ViTs) used for computer vision (e.g. classification, semantic labeling, etc.)
 - sequence of words = sequence of image patches
 - only the encoder part is used
 - flattened patches = reshape to 1D vector
 - linear projection = flattened patches * weight vector + bias (weight vector + bias has lower dimension and is learned)
 - position encoded added to ensure transformer knows about order of patches (= tokens)
 - for classification, add class token
 - attention determined/learned between all tokens in transformer encoder network (result are feature vectors)
 - example classification: output of transformer encoder (e.g., only class feature vector) = input of MLP / softmax for classification
 - **key idea:** not only features are relevant (like in CNNs) but their relationship (context) to each other



Vision Transformer (for classification):
<https://github.com/gupta-abhay/pytorch-vit>

Computer Vision



Lecture 9: Multi-View Geometry

Oliver Bimber

Goal: Depth-Estimation



Depth from Stereo(-scopic) Images) = two Views

Two Problems:

- 1.) Finding matching Features
- 2.) Computing Depth of corresponding Scenepoint (relative to Cameras)

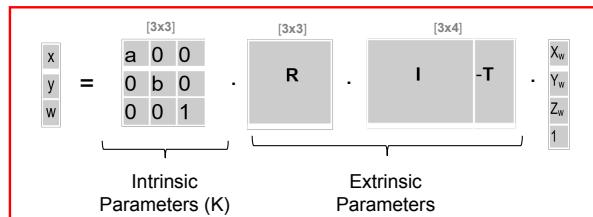
Recap: Simplified Mathematical Camera Model

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} [3x3] & \\ a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Parameters (K)}} \cdot \underbrace{\begin{bmatrix} [3x3] & \\ R & \end{bmatrix} \cdot \begin{bmatrix} [3x4] & \\ I & -T \end{bmatrix}}_{\text{Extrinsic Parameters}} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Note, that in Practice there are more intrinsic Parameters: e.g. Principle Point and Skew Angle describing Misalignment of Sensor on Optical Axis.

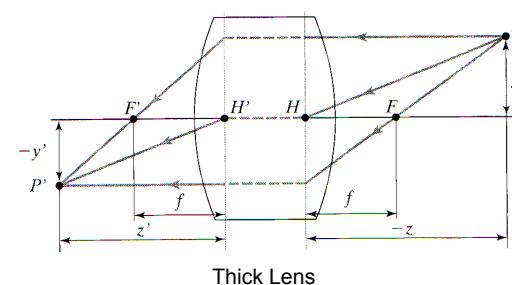
Recap: Simplified Mathematical Camera Model

$$p = MP \longrightarrow p = \frac{1}{z} MP \quad M = K(R \quad t)$$

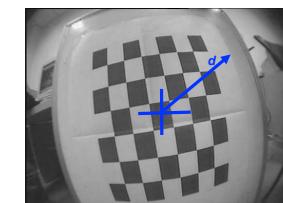
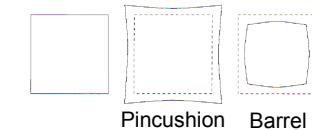


Note, that in Practice there are more intrinsic Parameters: e.g. Principle Point and Skew Angle describing Misalignment of Sensor on Optical Axis.

Recap: Radial Distortion



If Lens Distortion is considered, our Mathematical Model becomes non-linear (usually a Polynomial Function)!



Radial Distortion

Recap: Camera Calibration

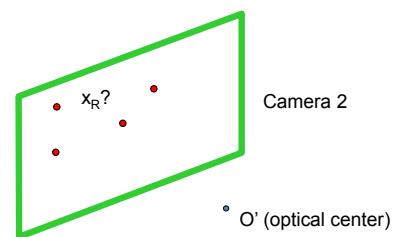
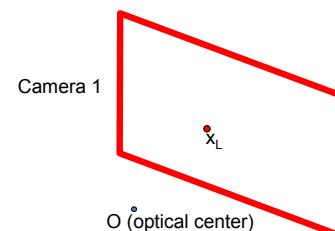


Images of known Calibration Pattern

[`cv.calibrateCamera\(\)`](#) returns the camera matrix, distortion coefficients, rotation and translation vectors etc.

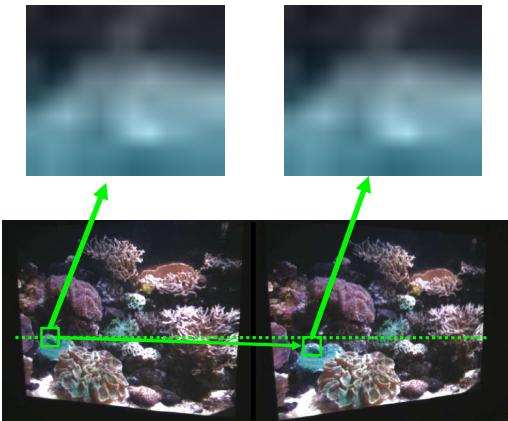
In OpenCV: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

Two Calibrated Cameras



If we see a Point in Cam 1, are there any Constraints on where we will find it in Cam 2?

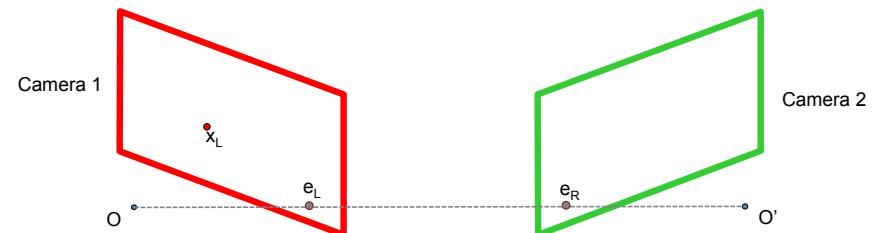
Feature Matching Problem



JKU JOHANNES KEPLER
UNIVERSITY LINZ

Is it really a 2D Search Problem? 309

Epipolar Constraints

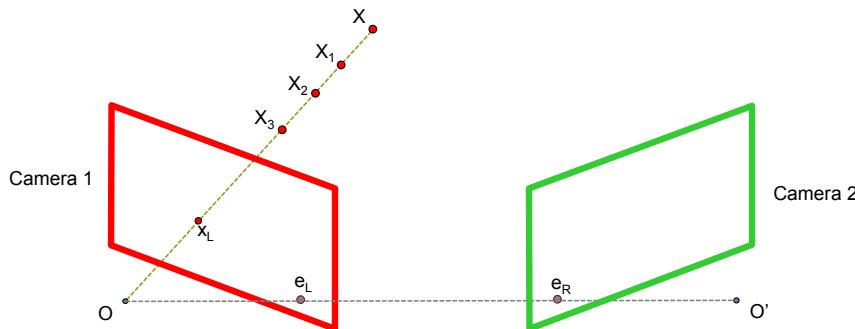


JKU JOHANNES KEPLER
UNIVERSITY LINZ

Epipoles: project each Optical Center onto other Image Plane
(Distance between Optical Centers is called Baseline)

310

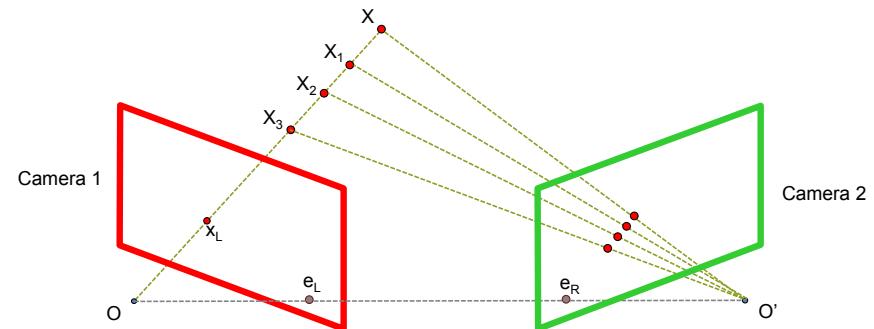
Epipolar Constraints



JKU JOHANNES KEPLER
UNIVERSITY LINZ

311

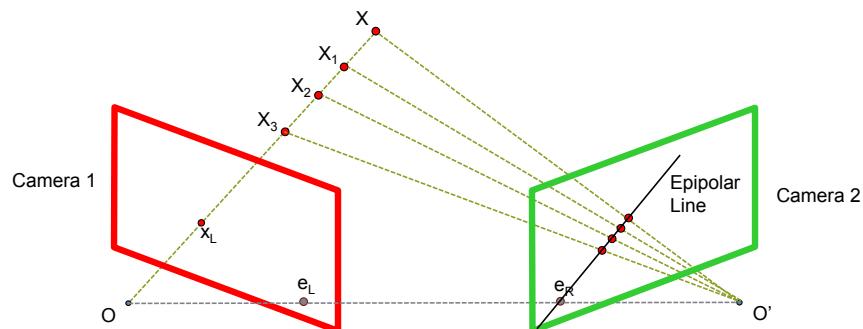
Epipolar Constraints



JKU JOHANNES KEPLER
UNIVERSITY LINZ

312

Epipolar Constraints



It's a 1D Search Problem!
But how do we get the Epipolar Line for
a given Point?

313

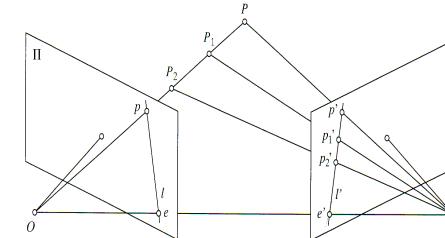
Epipolar Constraints

Calibrated Case
 p, p' on normalized
Image Plane,
Intrinsic K known:

$$p^T \epsilon p' = 0$$

$$l = \epsilon p', l' = \epsilon^T p$$

Line Parameters of
Epipolar Line on
normalized Image
Plane, $\epsilon \rightarrow 3 \times 3$
Essential Matrix



Uncalibrated Case
 p, p' on physical Image
Plane, Intrinsic K
unknown:

$$p^T F p' = 0$$

$$l = F p', l' = F^T p$$

Line Parameters of
Epipolar Line on
physical Image
Plane, $F \rightarrow 3 \times 3$
Fundamental Matrix

314

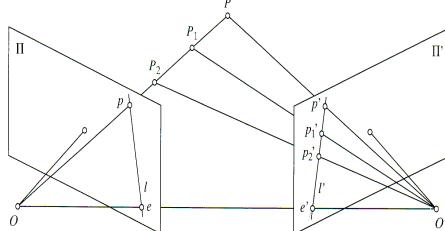
Epipolar Constraints

Calibrated Case
 p, p' on normalized
Image Plane,
Intrinsic K known:

$$p^T \epsilon p' = 0$$

$$l = \epsilon p', l' = \epsilon^T p$$

Line Parameters of
Epipolar Line on
normalized Image
Plane, $\epsilon \rightarrow 3 \times 3$
Essential Matrix



How do we get the Matrices?

Example: The uncalibrated Case

$$p^T F p' = 0$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}^T \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0$$

$$E = \sum_{i=1}^n (p_i^T F p'_i)^2$$

To get the Matrix-Coefficients, you
need to have sufficient amount of
known corresponding point-pairs (>8)

315

Image Rectification

If cameras are only displaced horizontally, Epipolar Lines
are horizontal!

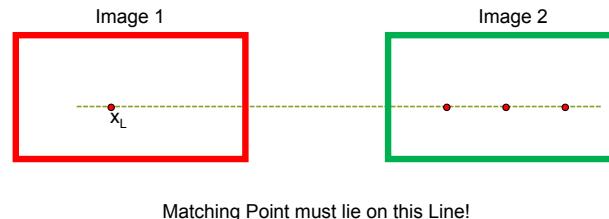
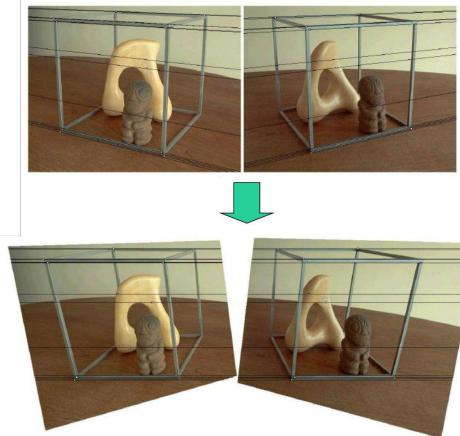


Image Rectification

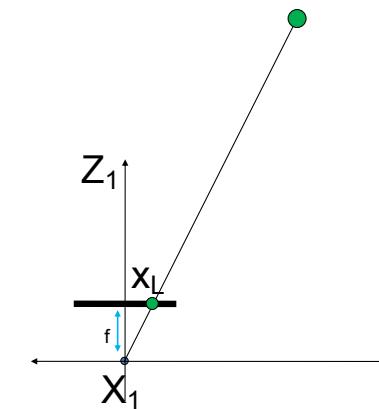


Correspondence Search

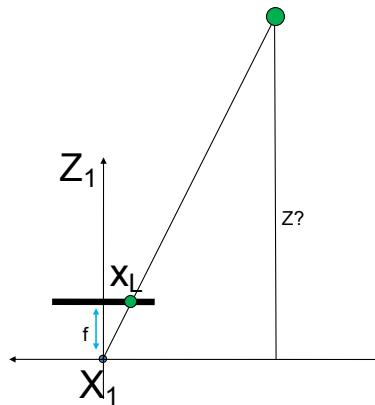


Assuming Images are rectified, we only need to search for Matches along horizontal Scanlines.

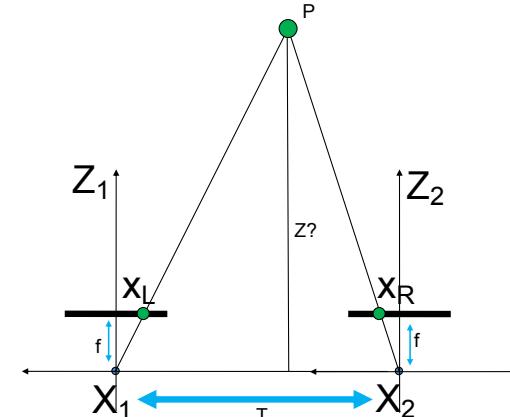
Stereoscopic Depth-Reconstruction



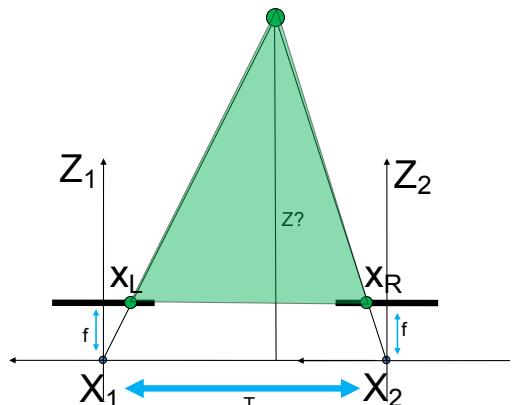
Stereoscopic Depth-Reconstruction



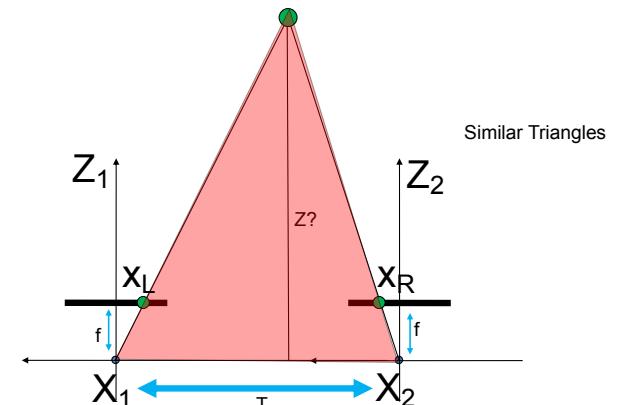
Stereoscopic Depth-Reconstruction



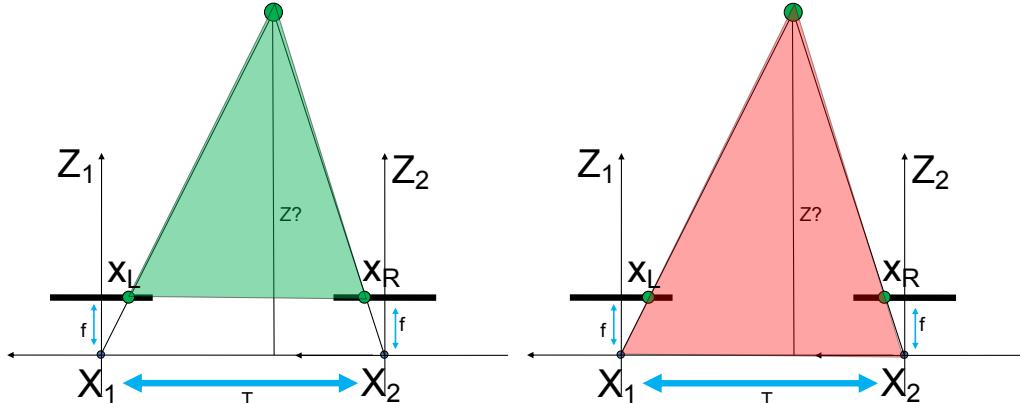
Stereoscopic Depth-Reconstruction



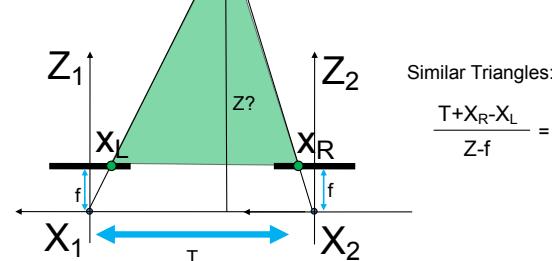
Stereoscopic Depth-Reconstruction



Stereoscopic Depth-Reconstruction



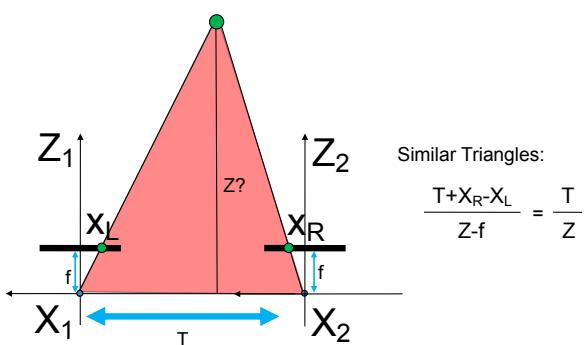
Stereoscopic Depth-Reconstruction



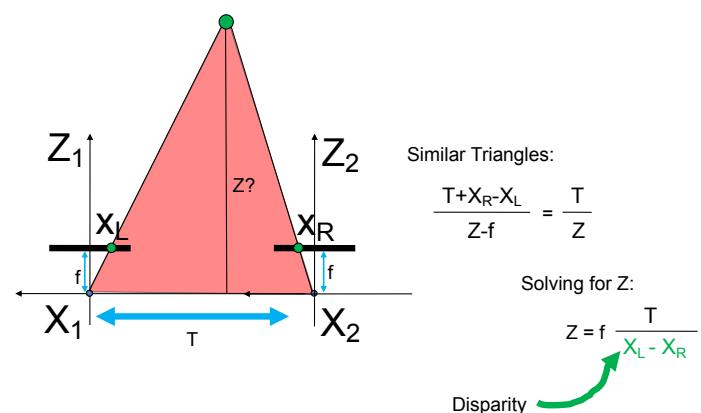
Similar Triangles:

$$\frac{T + X_R - X_L}{Z - f} =$$

Stereoscopic Depth-Reconstruction



Stereoscopic Depth-Reconstruction



So Z (depth) is proportional to Disparity.

Measuring Disparity



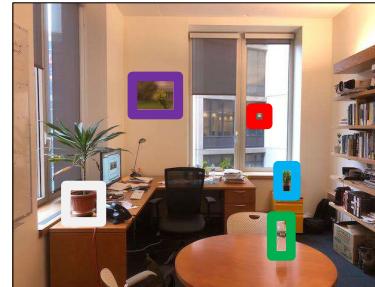
Left Image



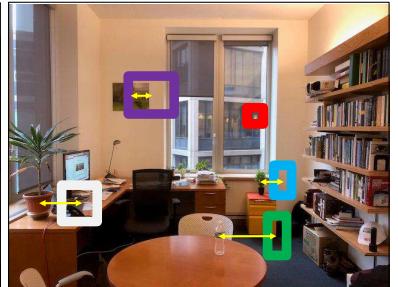
Right Image

(Taken by moving ~1m horizontally to the right)

Measuring Disparity

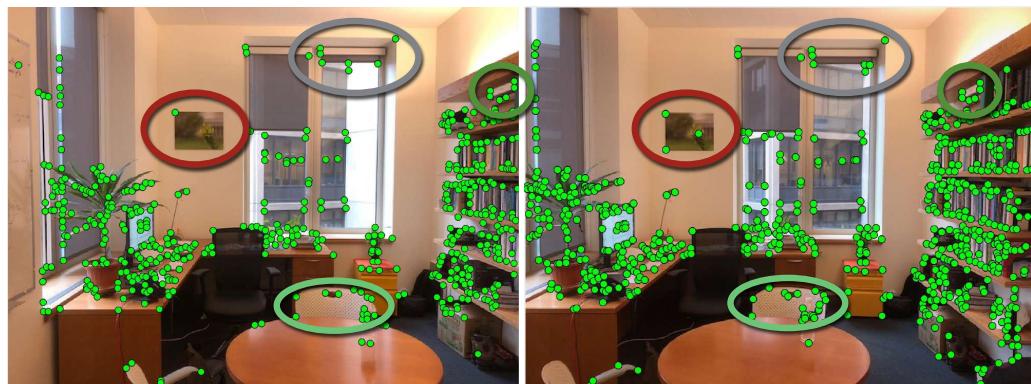


Left Image

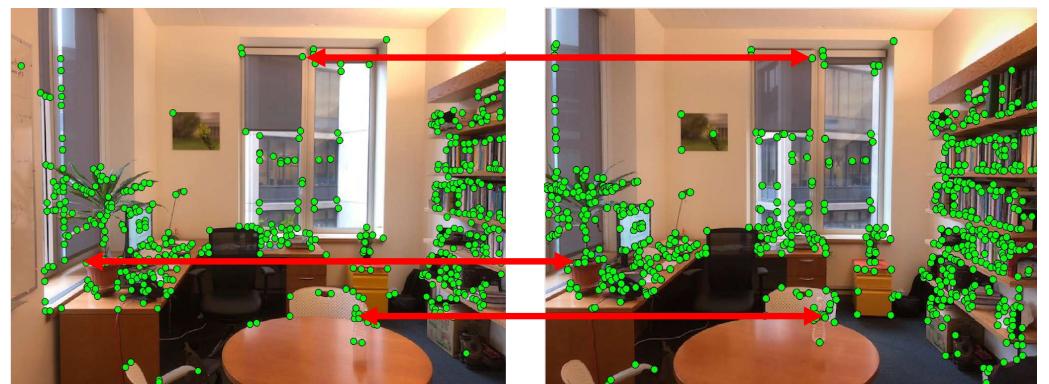


Right Image

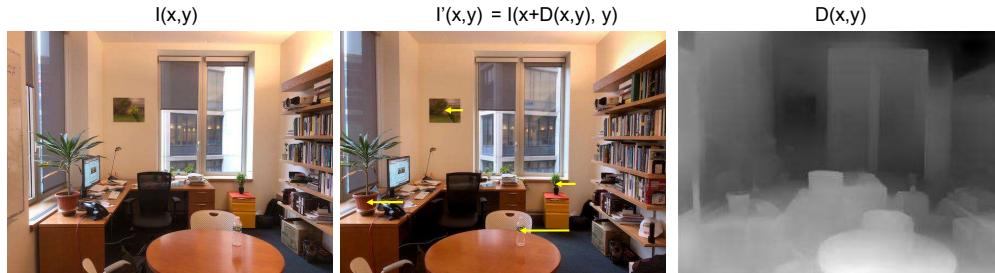
Example: SIFT Features



Example: SIFT Features



Disparity Maps



$$Z(x,y) \propto \frac{1}{D(x,y)}$$

Structure-from-Motion



Can we estimate Depth (= Structure) and Camera Pose (= Motion) simultaneously from multiple Images?

Structure-from-Motion in a Nutshell (and Simplified)

$P_j (j=1, \dots, n)$ 3D Points
 $i = 1, \dots, m$ Images (Camera View)
 $j = 1, \dots, n$ Points

$$p_{ij} = M_i \begin{pmatrix} P_j \\ 1 \end{pmatrix}$$

This is what is known

This is what is unknown

Structure-from-Motion in a Nutshell (and Simplified)

$P_j (j=1, \dots, n)$ 3D Points
 $i = 1, \dots, m$ Images (Camera View)
 $j = 1, \dots, n$ Points

$$p_{ij} = M_i \begin{pmatrix} P_j \\ 1 \end{pmatrix}$$

This is what is known

This is what is unknown

This is an Equation-System: How many Unknowns vs. how many Equations, and when can it be solved?

Structure-from-Motion in a Nutshell (and Simplified)

$P_j (j = 1, \dots, n)$ 3D Points
 $i = 1, \dots, m$ Images (Camera View)
 $j = 1, \dots, n$ Points

$$p_{ij} = M_i \begin{pmatrix} P_j \\ 1 \end{pmatrix}$$

This is what is known ↑ ↓ This is what is unknown

This is an Equation-System: How many Unknowns vs. how many Equations, and when can it be solved?

Unknowns: 12m Coefficients in all m M + 3n Coefficients in all n P , **Equations:** 2 Equations for all nm p

Structure-from-Motion in a Nutshell (and Simplified)

$P_j (j = 1, \dots, n)$ 3D Points
 $i = 1, \dots, m$ Images (Camera View)
 $j = 1, \dots, n$ Points

$$p_{ij} = M_i \begin{pmatrix} P_j \\ 1 \end{pmatrix}$$

This is what is known ↑ ↓ This is what is unknown

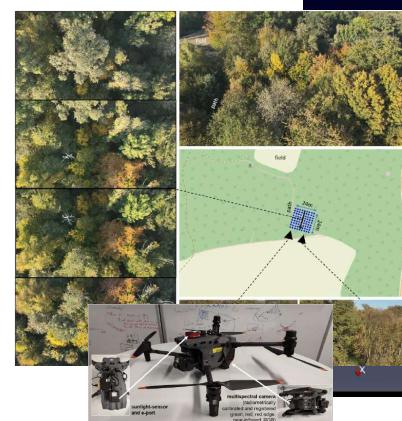
This is an Equation-System: How many Unknowns vs. how many Equations, and when can it be solved?

Unknowns: 12m Coefficients in all m M + 3n Coefficients in all n P , **Equations:** 2 Equations for all nm p
Solvable (simplified): $2nm > 12m+3n$ (e.g., $m=2$ and $n=25$)

Structure-from-Motion



Research Example: Reconstructing 3D Forest Health



extracted top vegetation layer
(corrected and sensor-mapped)



Computer Vision



Lecture 10: 3D Vision
Oliver Bimber

JKU JOHANNES KEPLER
UNIVERSITY LINZ

We have seen how to predict 2D Shapes of Objects

Classification



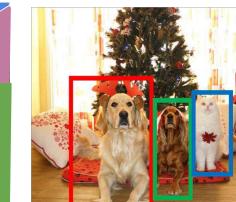
CAT

Semantic Segmentation



GRASS, CAT, TREE,
SKY

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

341

JKU JOHANNES KEPLER
UNIVERSITY LINZ

342

But we interpret the World in 3D



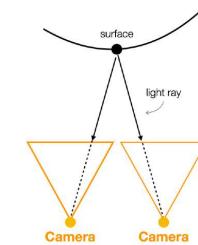
JKU JOHANNES KEPLER
UNIVERSITY LINZ

343

Recap: Depth from Stereo (and multiple Views)



JKU JOHANNES KEPLER
UNIVERSITY LINZ



Key Idea: use difference in Perspective to estimate Depth

Solution: develop Models for Feature Extraction, Matching, Depth Computation, Camera Calibration, Pose estimation, etc.

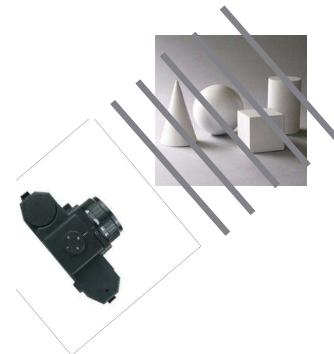
344

Depth from X



X = Shading

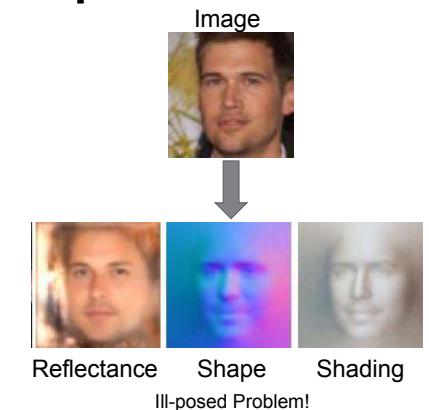
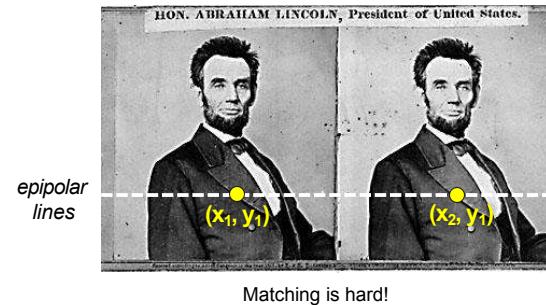
Key Idea: use difference in Shading to estimate Depth
Solution: develop Models for Inverse Shading



X = Defocus

Key Idea: use difference in Focus to estimate Depth
Solution: develop Models for Focus Estimation

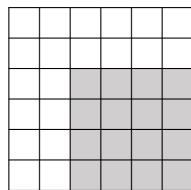
Why Learning how to estimate Depth?



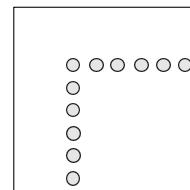
How to represent Depth

∞
 ∞
2
2
2
2
2

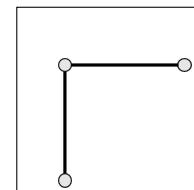
Depth Map



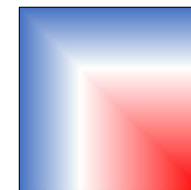
Voxel Grid



Point Cloud



Mesh

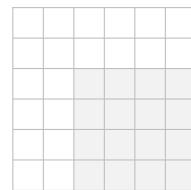


Implicit Surface

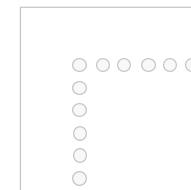
Depth Maps

∞
 ∞
2
2
2
2
2

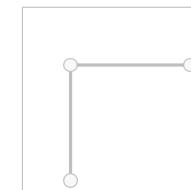
Depth Map



Voxel Grid



Point Cloud



Mesh



Implicit Surface

How to measure Depth Maps

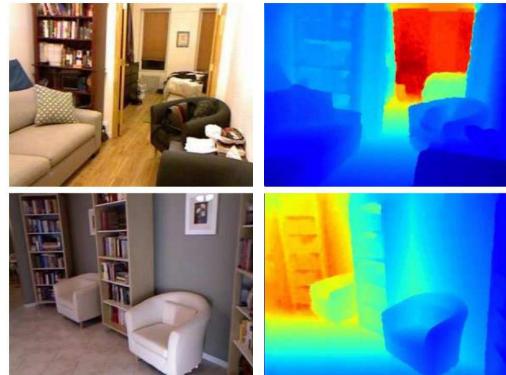
A Depth Map gives the Distance from the Camera to the Object in the World at each Pixel

RGB Image + Depth Map
= RGB-D Image (2.5D)

This Type of Data can be recorded directly for some Types of 3D Sensors (e.g. Microsoft Kinect)



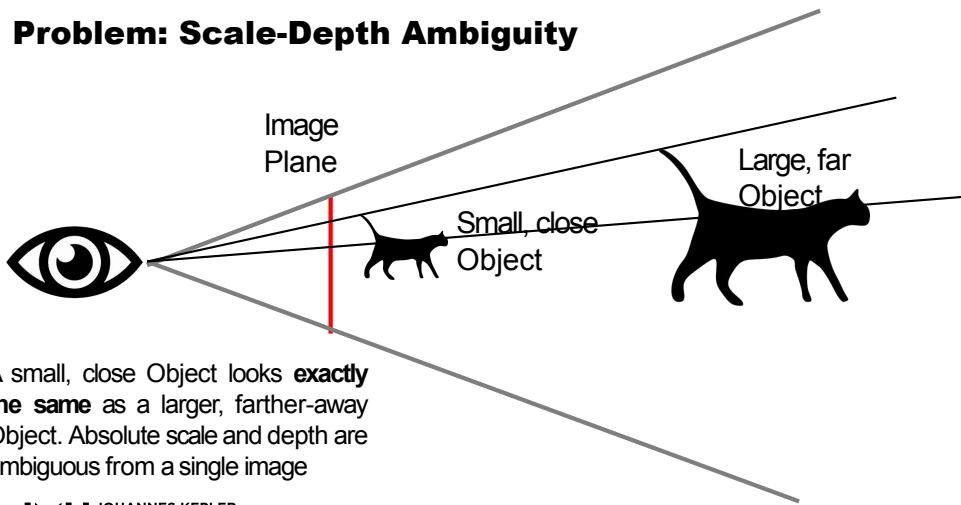
JKU JOHANNES KEPLER
UNIVERSITY LINZ



RGB Image: $3 \times H \times W$ Depth Map: $H \times W$

349

Problem: Scale-Depth Ambiguity



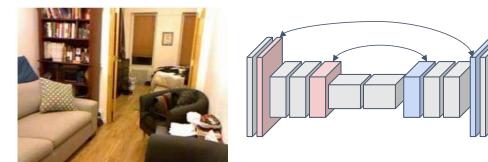
JKU JOHANNES KEPLER
UNIVERSITY LINZ

351

Predicting Depth Maps

Estimate log Depth instead of Depth. Defining y_i as the Ground Truth Depth on Pixel i, and y_i^* its estimated Depth:

$$D_{L2}(y, y^*) = \frac{1}{n} \sum_{i=1}^n (\log y_i - \log y_i^*)^2$$

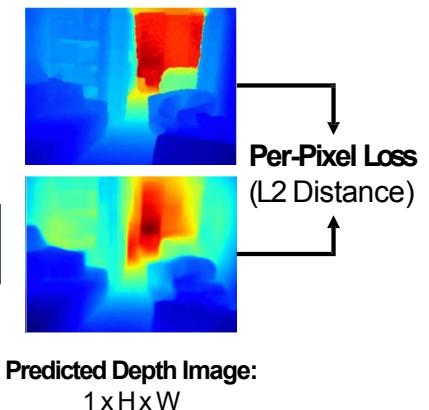


RGB Input Image:
 $3 \times H \times W$

JKU JOHANNES KEPLER
UNIVERSITY LINZ

CNN

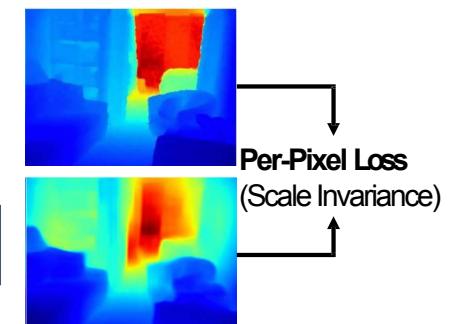
Measured Depth Image:
 $1 \times H \times W$



Predicted Depth Image:
 $1 \times H \times W$

350

Measured Depth Image:
 $1 \times H \times W$

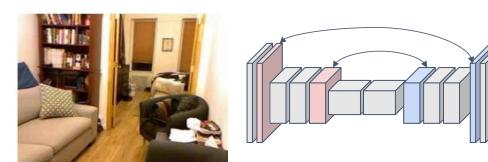


Scale Invariant Error

The global scale of a scene is a fundamental Ambiguity in Depth Prediction! So it is considered in the Loss Function.

$$D_{SI}(y, y^*) = \frac{1}{n} \sum_{i=1}^n (\log y_i - \log y_i^* + \alpha(y, y^*))^2$$

$$\alpha(y, y^*) = \frac{1}{n} \sum_{j=1}^n (\log y_j - \log y_j^*)$$



RGB Input Image:
 $3 \times H \times W$

JKU JOHANNES KEPLER
UNIVERSITY LINZ

CNN

Predicted Depth Image:
 $1 \times H \times W$

<http://arxiv.org/pdf/1406.2283>

352

Surface Normals

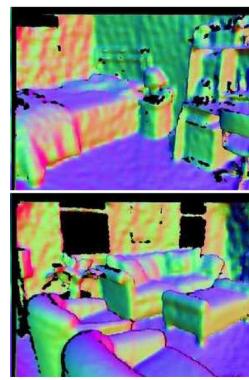
For each Pixel, its Surface Normal is the Normal Vector (i.e., Unit Vectors perpendicular to the tangential Surface) of the Pixel

We can compute Surface Normals from given Depth (Gradients), and we can compute Depth from given Surface Normals (Integration)

$$\frac{\partial f}{\partial x} = \frac{a}{c}$$



RGB Image: $3 \times H \times W$

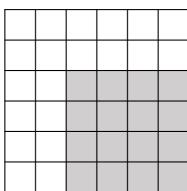


Normals: $3 \times H \times W$

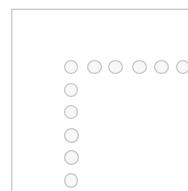
Voxel Grids

∞
 ∞
2
2
2
2

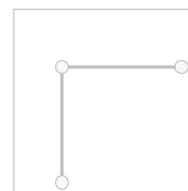
Depth
Map



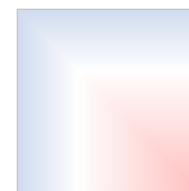
Voxel
Grid



Point
Cloud



Mesh



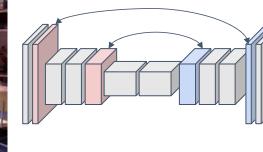
Implicit
Surface

Predicting Surface Normals

Here, x is the estimated and y the ground truth surface normal. The loss function considers the solid angle difference.

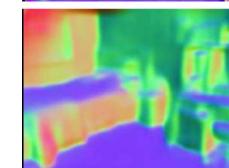
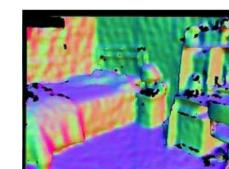


RGB Input Image:
 $3 \times H \times W$



Fully Convolutional
network

Ground-truth Normals:
 $3 \times H \times W$



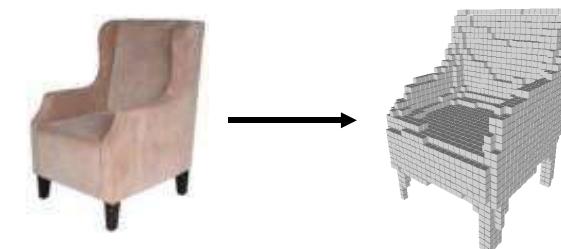
Predicted Normals:
 $3 \times H \times W$

Per-Pixel Loss:
 $(x \cdot y) / (|x||y|)$

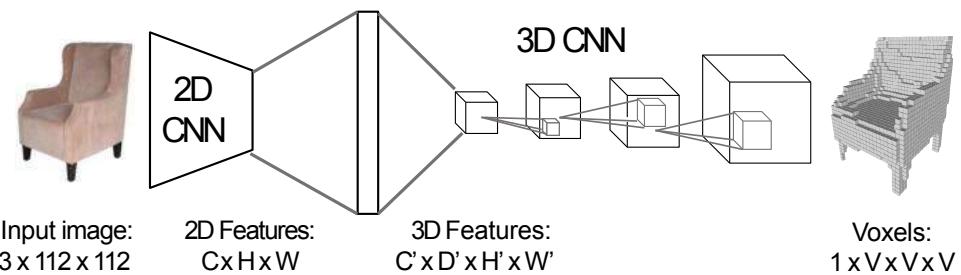
Recall:
 $x \cdot y$
 $= |x| |y| \cos \theta$

Voxel Grids

- Represent a Shape with a $V \times V \times V$ Grid of Occupancies
- Basically just like Segmentation, but in 3D!
- (+) Conceptually simple: just a 3D grid!
- (-) Need high spatial Resolution to capture fine Structures
- (-) Scaling to high Resolutions is nontrivial!

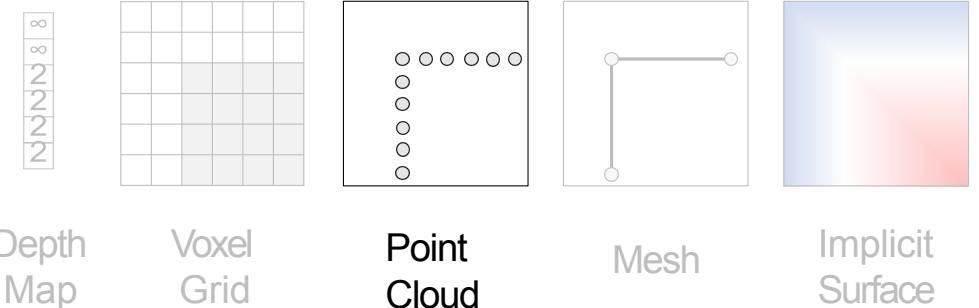


Example for Generating Voxel Shapes



Choy et al., "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction", ECCV/2016

Point Clouds

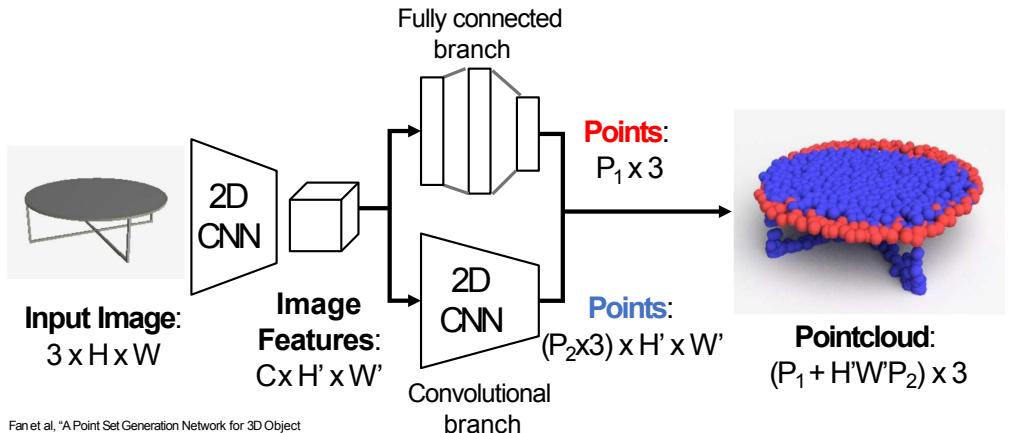


Point Clouds

- Represent Shape as a set of P Points in 3D space
- (+) Can represent fine Structures without huge Numbers of Points
- (-) Requires new Architecture, losses, etc
- (-) Doesn't explicitly represent the Surface of the Shape: extracting a Mesh for rendering or other Applications requires Post-Processing

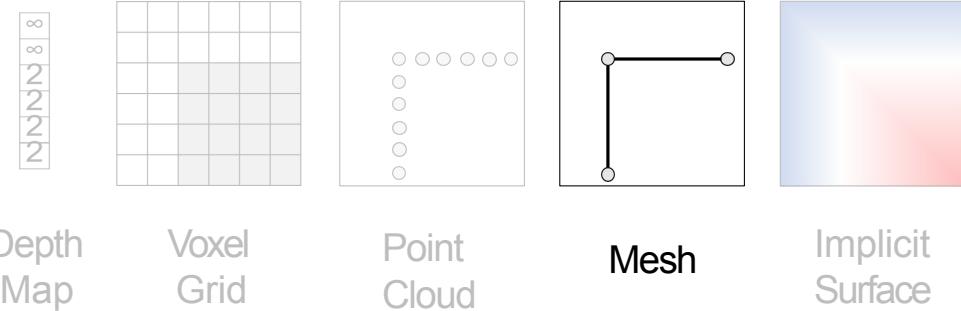


Example for Generating Point Clouds



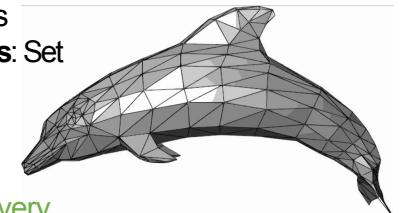
Fan et al., "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR2017

Meshes

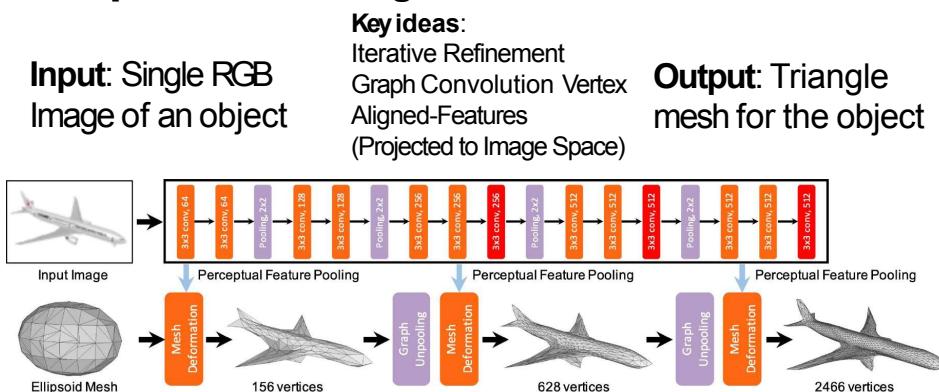


Meshes

- Represent a 3D Shape as a set of Triangles
- **Vertices:** Set of V points in 3D space **Faces:** Set of Triangles over the Vertices
- (+) Standard Representation for Graphics
- (+) Explicitly represents 3D Shapes
- (+) Adaptive: Can represent flat Surfaces very efficiently, can allocate more Faces to Areas with fine Detail
- (+) Can attach Data on Verts and interpolate over the whole Surface: RGBColors, Texture Coordinates, Normal Vectors, etc.

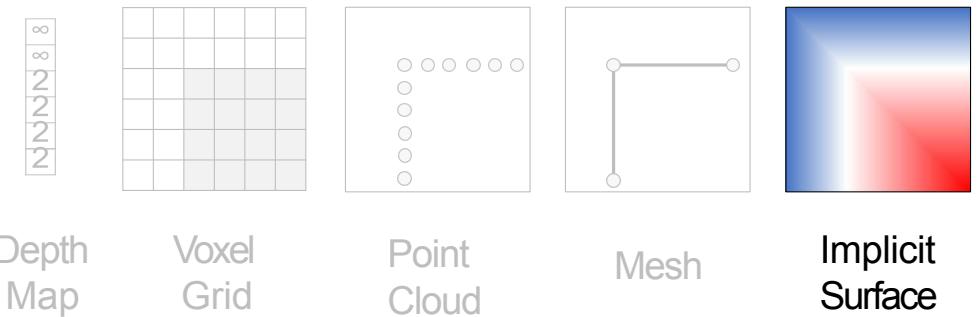


Example for Predicting Meshes



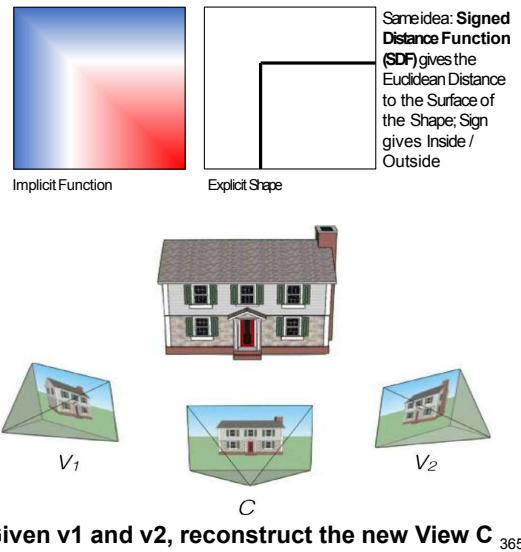
Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Implicit Surfaces

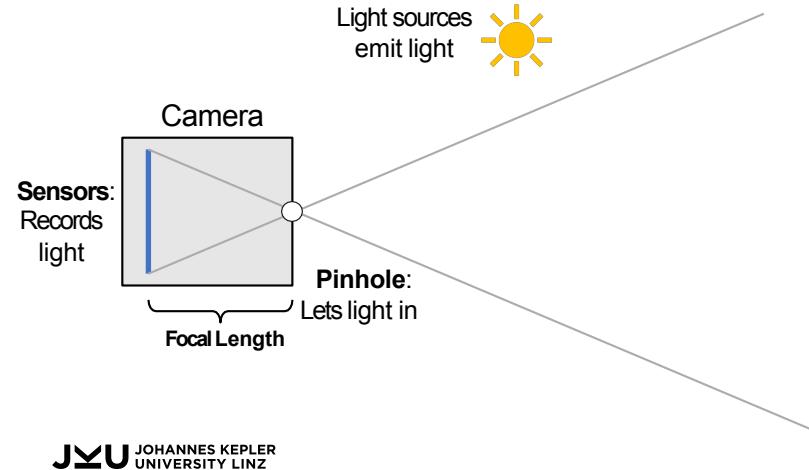


Implicit Surfaces

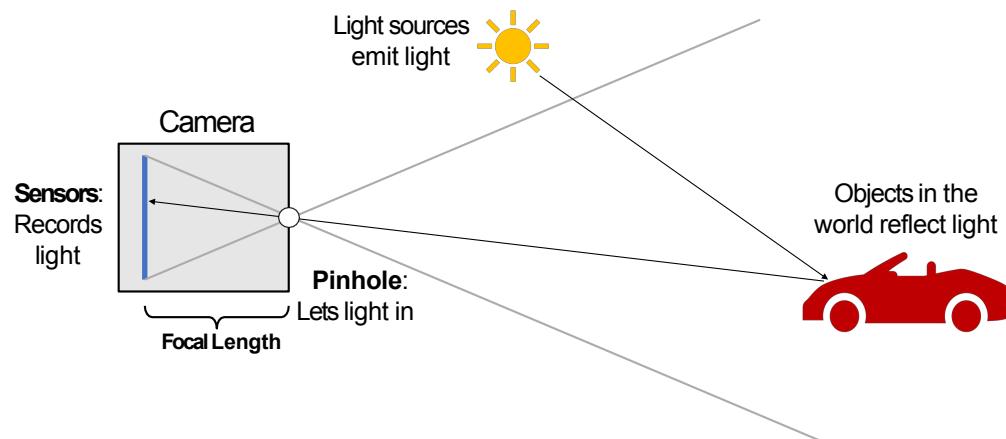
- Learn a Function to classify arbitrary 3D Points as inside / outside the shape
- Application: Novel View Synthesis



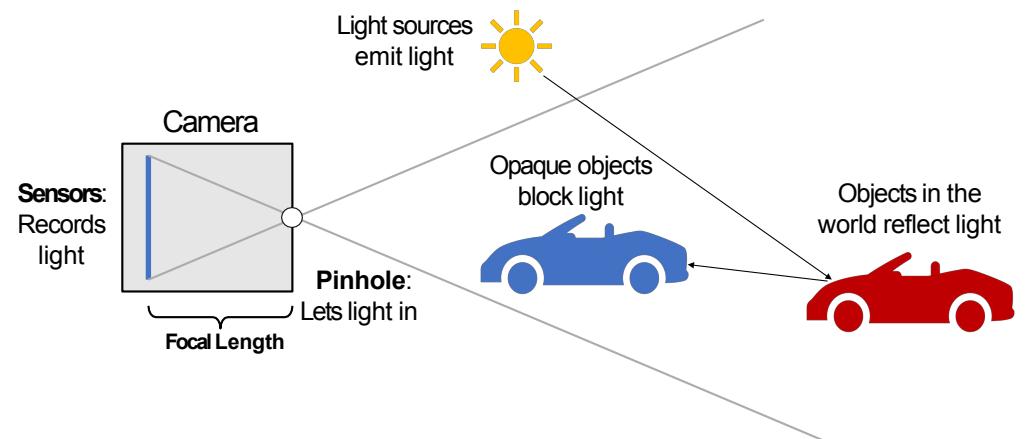
Recap: Pinhole Cameras



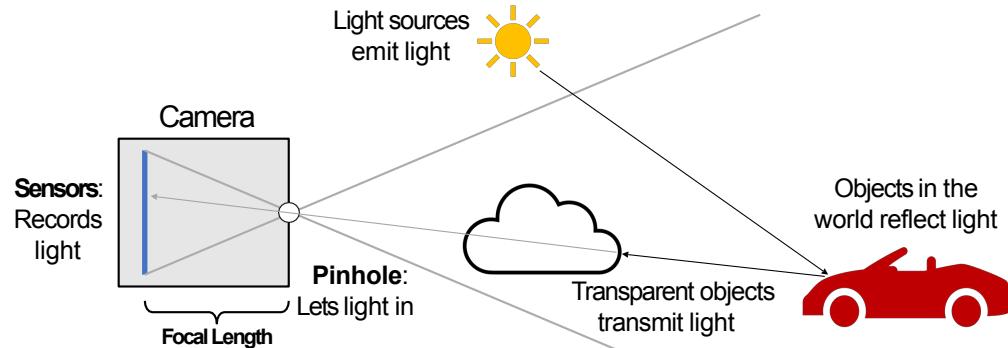
Recap: Pinhole Cameras



Recap: Pinhole Cameras

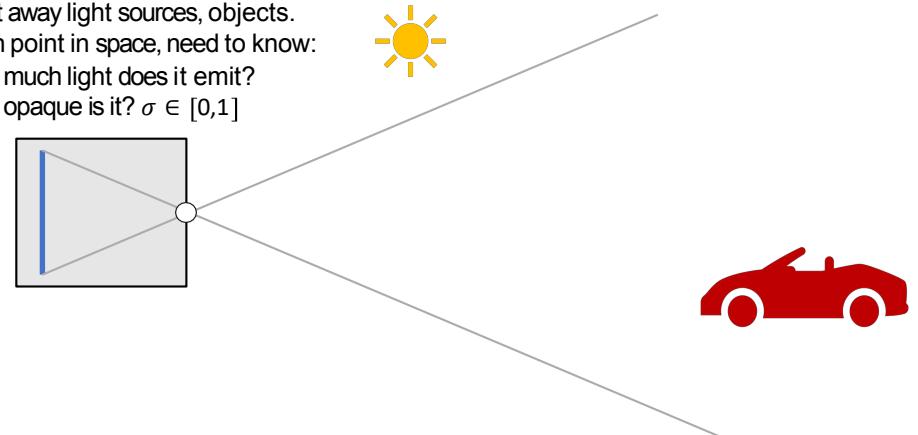


Recap: Pinhole Cameras



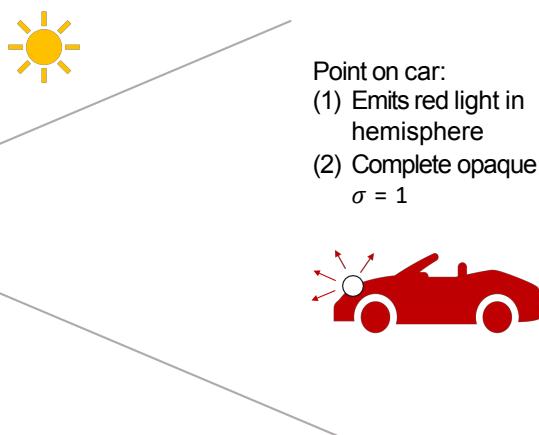
Volume Rendering

Abstract away light sources, objects.
For each point in space, need to know:
(1) How much light does it emit?
(2) How opaque is it? $\sigma \in [0,1]$



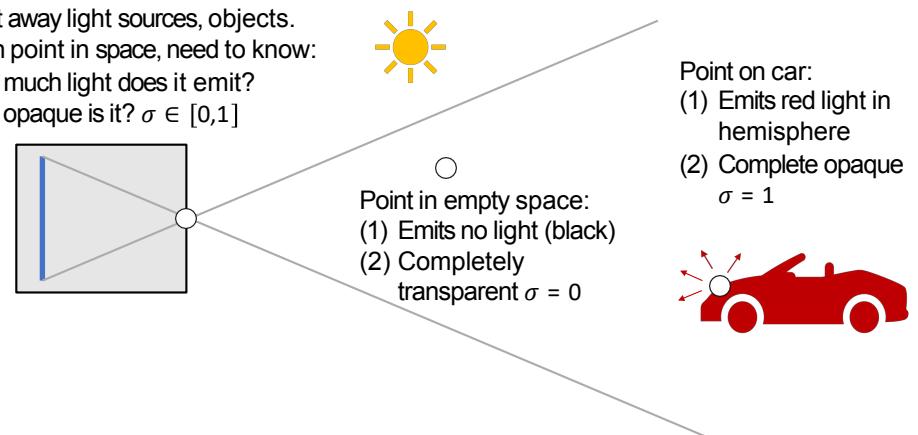
Volume Rendering

Abstract away light sources, objects.
For each point in space, need to know:
(1) How much light does it emit?
(2) How opaque is it? $\sigma \in [0,1]$

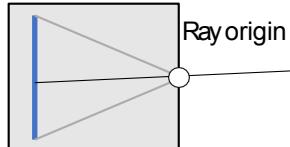


Volume Rendering

Abstract away light sources, objects.
For each point in space, need to know:
(1) How much light does it emit?
(2) How opaque is it? $\sigma \in [0,1]$



Volume Rendering

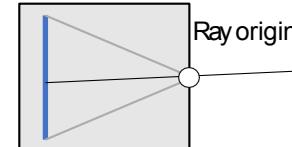


Parameterize each ray as origin plus direction: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

Color that a point \mathbf{p} emits in direction \mathbf{d} is $c(\mathbf{p}, \mathbf{d}) \in [0,1]^3$

Volume Rendering

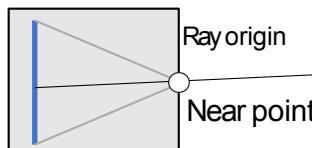


Parameterize each ray as origin plus direction: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

Color that a point \mathbf{p} emits in direction \mathbf{d} is $c(\mathbf{p}, \mathbf{d}) \in [0,1]^3$

Volume Rendering



Parameterize each ray as origin plus direction: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

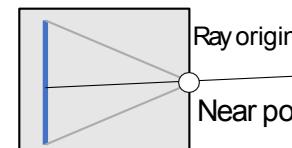
Color that a point \mathbf{p} emits in direction \mathbf{d} is $c(\mathbf{p}, \mathbf{d}) \in [0,1]^3$

Color observed by the camera given by volume rendering equation:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(\mathbf{r}(t))\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \mathbf{d})dt$$

Near point: t_n Current point: t Far point: t_f

Volume Rendering



Parameterize each ray as origin plus direction: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

Color that a point \mathbf{p} emits in direction \mathbf{d} is $c(\mathbf{p}, \mathbf{d}) \in [0,1]^3$

Near point: t_n Current point: t Far point: t_f

Color observed by the camera given by volume rendering equation:

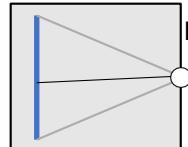
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(\mathbf{r}(t))\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \mathbf{d})dt$$

Transmittance: How much light from the current point will reach the camera?

Volume Rendering

Color observed by the camera given by volume rendering equation:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(\mathbf{r}(t)) \sigma(\mathbf{r}(t)) c(\mathbf{r}(t), \mathbf{d}) dt$$



Ray origin

Near point: t_n

Current point: t

Far point: t_f

Parameterize each ray as origin plus direction: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

Color that a point \mathbf{p} emits in direction \mathbf{d} is $c(\mathbf{p}, \mathbf{d}) \in [0,1]^3$



377

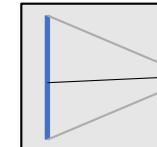
Transmittance: How much light from the current point will reach the camera?

Opacity: How opaque is the current point?

Volume Rendering

Color observed by the camera given by volume rendering equation:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(\mathbf{r}(t)) \sigma(\mathbf{r}(t)) c(\mathbf{r}(t), \mathbf{d}) dt$$



Ray origin

Near point: t_n

Current point: t

Far point: t_f

Parameterize each ray as origin plus direction: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

Color that a point \mathbf{p} emits in direction \mathbf{d} is $c(\mathbf{p}, \mathbf{d}) \in [0,1]^3$



378

Transmittance: How much light from the current point will reach the camera?

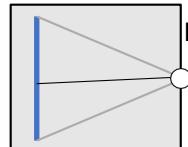
Opacity: How opaque is the current point?

Color: What color does the current point emit along the direction toward the camera?

Volume Rendering

Color observed by the camera given by volume rendering equation:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(\mathbf{r}(t)) \sigma(\mathbf{r}(t)) c(\mathbf{r}(t), \mathbf{d}) dt$$



Ray origin

Near point: t_n

Current point: t

Far point: t_f

Parameterize each ray as origin plus direction: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

Color that a point \mathbf{p} emits in direction \mathbf{d} is $c(\mathbf{p}, \mathbf{d}) \in [0,1]^3$



379

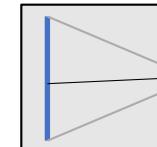
Transmittance: How much light from the current point will reach the camera?

Compute transmittance by accumulating volume density up to current point

Neural Radiance Fields (NeRFs)

Color observed by the camera given by volume rendering equation:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(\mathbf{r}(t)) \sigma(\mathbf{r}(t)) c(\mathbf{r}(t), \mathbf{d}) dt$$



Ray origin

Near point: t_n

Current point: t

Far point: t_f

Parameterize each ray as origin plus direction: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

Color that a point \mathbf{p} emits in direction \mathbf{d} is $c(\mathbf{p}, \mathbf{d}) \in [0,1]^3$



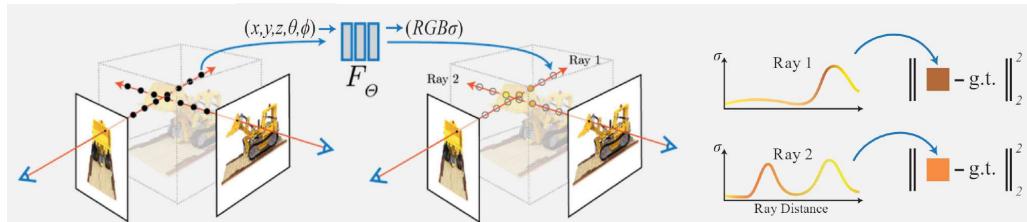
Train a neural network to input position \mathbf{p} and direction \mathbf{d} , output $\sigma(\mathbf{p})$ and $c(\mathbf{p}, \mathbf{d})$

380

$$T(\mathbf{r}(t)) = \exp \left(- \int_{t_n}^t \sigma(\mathbf{r}(s)) ds \right)$$

Neural Radiance Fields (NeRFs)

Fully-connected Network: Input Position $p=x,y,z$ and Direction $d=\theta,\Phi$, and output Volume Density (σ) and RGB color



Mildenhall et al, "Representing Scenes as Neural Radiance Fields for View Synthesis", ECCV 2020

Neural Radiance Fields (NeRFs)



Neural Radiance Fields (NeRFs)



Some Pre-Trained Models (try out)

<https://github.com/ByteDance-Seed/Depth-Anything-3>



<https://github.com/gangweix/pixel-perfect-depth>

<https://github.com/DepthAnything/Video-Depth-Anything>

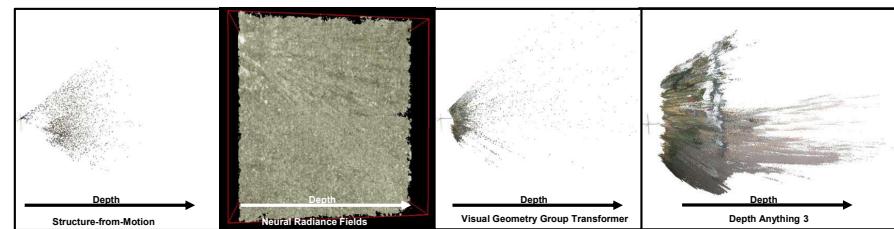
<https://github.com/DepthAnything/Depth-Anything-V2>

<https://github.com/facebookresearch/vggt>

<https://github.com/NVlabs/instant-ngp>



Research Example: Occlusion, however...



JKU JOHANNES KEPLER
UNIVERSITY LINZ

385

Thank You



JKU JOHANNES KEPLER
UNIVERSITY LINZ



Happy Holidays

386

Computer Vision



Lecture 11: Trends in Computer Vision
Oliver Bimber

JKU JOHANNES KEPLER
UNIVERSITY LINZ

387

Not covered: Generative Models for Image Synthesis

- Variational Autoencoders (VAEs)
 - Generative Adversarial Networks (GANs)
 - Diffusion Models
 - etc.
-
- What is this good for in CV?
 - Training Data Augmentation
 - Solving Inverse Problems
 - etc.



a shiba inu wearing a beret and black turtleneck



a close up of a hand palm with leaves growing from it

388

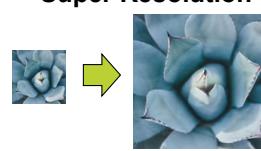
JKU JOHANNES KEPLER
UNIVERSITY LINZ

Example: Inverse Problems

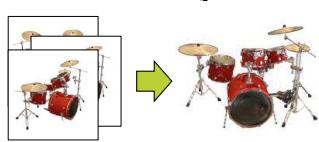
Inverse Tomography



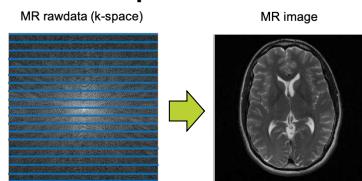
Super Resolution



Novel View Synthesis



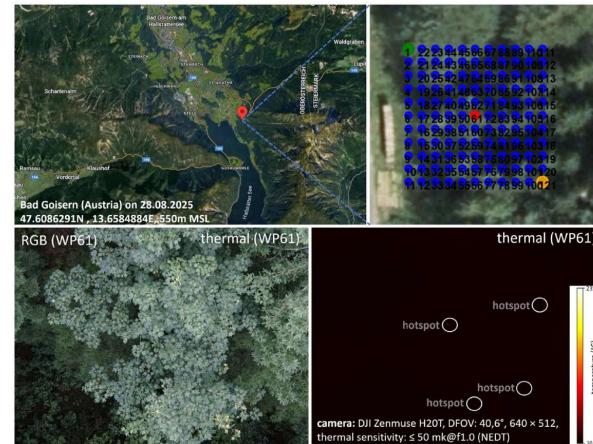
Sparse MRI



JKU JOHANNES KEPLER
UNIVERSITY LINZ

389

Example from our Research



JKU JOHANNES KEPLER
UNIVERSITY LINZ

<https://arxiv.org/pdf/2511.12572>

390

Example from our Research

Wildfire at Grafenbergalm (Austria) on 17.10.2023
47.46754N , 13.75525E, 1780m MSL

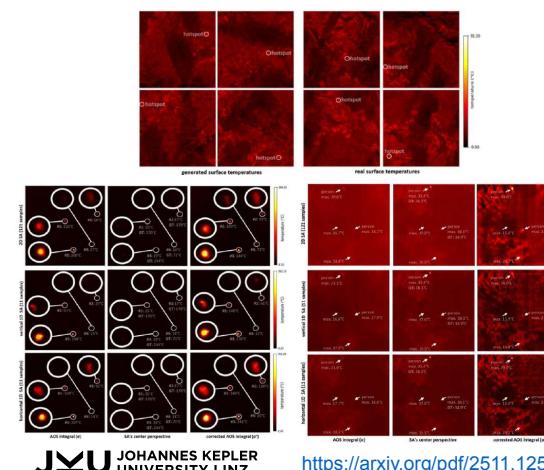


JKU JOHANNES KEPLER
UNIVERSITY LINZ

<https://arxiv.org/pdf/2511.12572>

391

Example from our Research



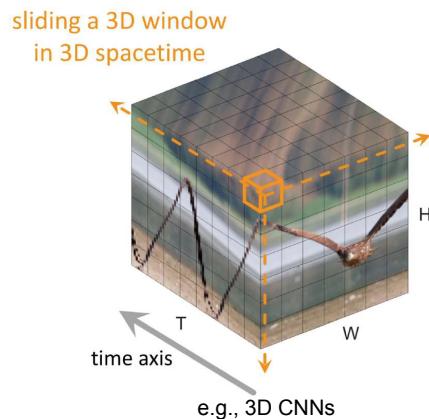
JKU JOHANNES KEPLER
UNIVERSITY LINZ

<https://arxiv.org/pdf/2511.12572>

392

Not covered: Sequence Problems

- Sequences in Computer Vision
= Videos
- ConvNets for Sequence Problems
- Recurrent Neural Networks
- Transformers
- etc.



Not covered: Self-Supervised Vision

Common Trick:

- Convert “unsupervised” Problem into “supervised” Empirical Risk Minimization
- Do so by cooking up “Labels” (prediction Targets) from the raw Data itself

Supervised vision

Hand-curated training data
+ Informative
- Expensive
- Limited to teacher's knowledge



Vision in nature

Raw unlabeled training data
+ Cheap
- Noisy
- Harder to interpret



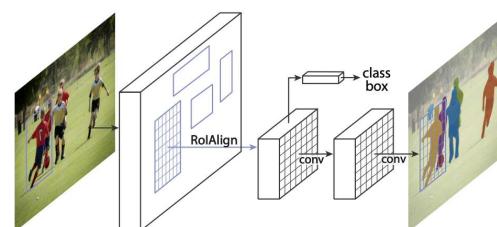
Limits of Machine Learning

“The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs—not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called **generalization**.

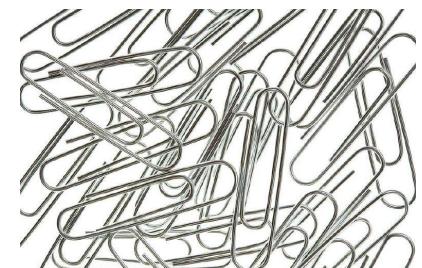
... [this is what] separates machine learning from optimization.”

— Deep Learning Textbook (Goodfellow et al.)

Example: Object Detection

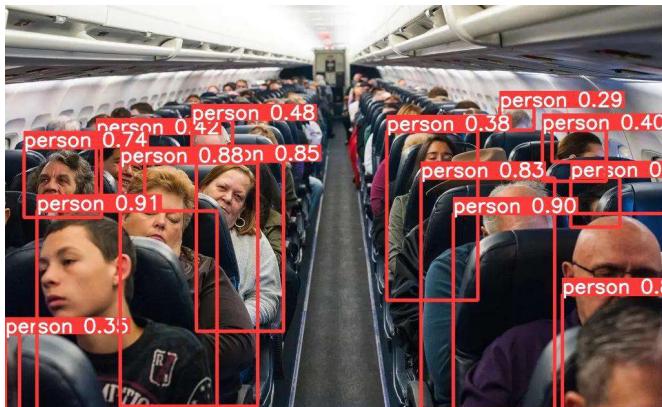


CNN can detect Humans, Cars, etc.



But it can't count these Binder Clips because it's never seen them before! It has not learned what an ‘object’ is in general.

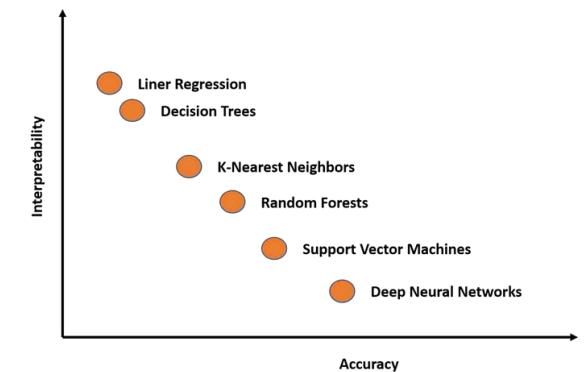
Example: Person Detection



Yolo-v8 can't detect the people in the back.
But we can!

Problem: Interpretability vs. Accuracy

- What Features of an Image is the Model using?
- How does the Model make its Decisions?
- Accuracy-based Training (how about a Bias towards a Majority in the Training Data)?



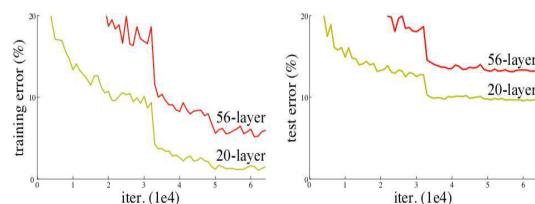
Lessons Learned

Data is not the only Answer

"If you torture the Data long enough,
it will confess to anything"

— *How to Lie With Statistics*
by Darrell Huff

More Parameters != better



Human Knowledge can help (e.g. as done with Regularization in Loss-Functions)

Large Multi-Model Models (LMMs)

- A large multi-modal model **handles multiple data types** (text, images, audio, video) in one system.
- It **learns connections across modalities**, not each one in isolation.
- Examples:
 - GPT (OpenAI): text, image, audio, and video understanding and generation
 - Gemini (Google DeepMind): strong multimodal reasoning with very long context
 - Claude (Anthropic): advanced text + image understanding
 - etc.



Example: stereopair of a dog.

Object Detection with LMMs



what is in the image?

This image features a very cute, small dog, likely a puppy or a toy breed, standing on a stone path.

The Dog

- **Breed Appearance:** It looks like a small terrier or poodle mix (often called a "designer breed" like a Morkie, Maltipoo, or Shichon).
- **Coat:** The fur is white with light tan or cream patches, particularly around the ears and eyes. The texture appears soft and slightly shaggy.
- **Features:** It has dark, expressive eyes, a black nose, and a small tail poking up in the back.

The Setting

- **Background:** The dog is standing in front of a **rustic stone retaining wall** and a set of **stone steps** to the left.
- **Ground:** It is standing on a light-colored concrete or stone walkway with some fallen leaves scattered nearby, suggesting an outdoor garden or patio setting.

Would you like me to help you identify the specific breed or suggest some names for this little pup?

Segmentation with LMMs



generate an image with the dog segmented



More difficult Example with LMMs



Count how many binder clips are in the image.

There are **zero** binder clips in this image.

The items shown here are actually **paperclips** (specifically the "Gem" style). While they both hold paper together, binder clips are the ones with the black triangular bodies and silver folding arms.

In this pile, there are approximately **30** paperclips.

Would you like me to help you identify the difference between other types of office supplies, or perhaps help you find where to buy actual binder clips?

Depth from Single Images with LMMs



Here is an image. Generate an image that visualizes reconstructed depth.



Depth from Stereo with LMMs



Here is a stereopair. Generate an image that visualizes reconstructed depth.

Novel View Generation with LMMs



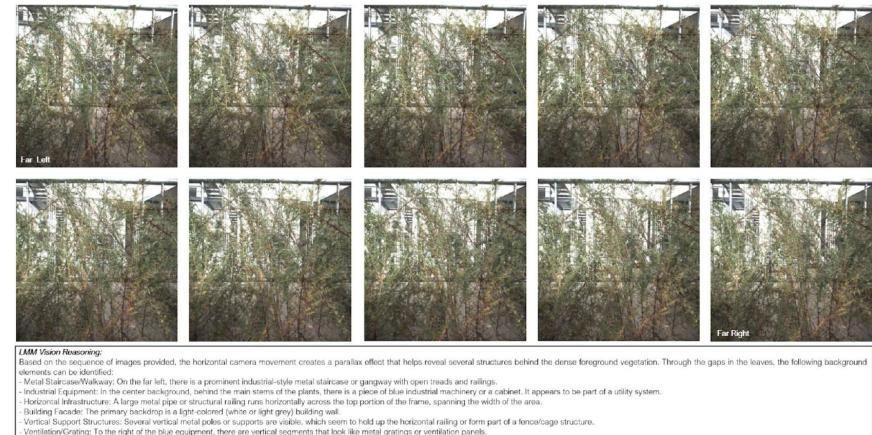
generate an image that shows the dog from the side perspective.



Example from our Research



Example from our Research



Example from our Research



JKU JOHANNES KEPLER
UNIVERSITY LINZ

<http://arxiv.org/abs/2511.16262>

409

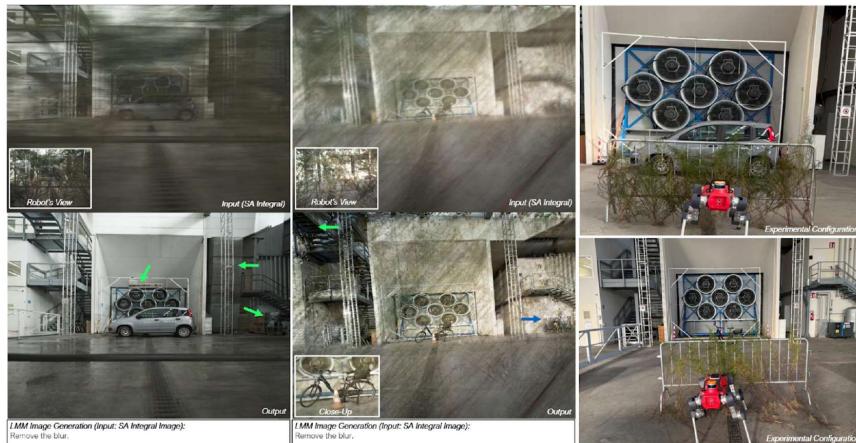
Example from our Research



<http://arxiv.org/abs/2511.16262>

410

Example from our Research



JKU JOHANNES KEPLER
UNIVERSITY LINZ

<http://arxiv.org/abs/2511.16262>

411

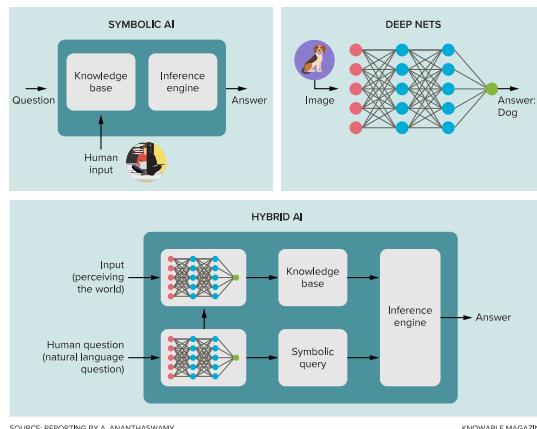
LLM Limitations

- Todays LMMs are **huge (e.g., transformer) models**, trained on a **huge amount of (general) data**.
- They can be tweaked with **in-context learning** (adding background information to huge context windows).
- They represent a **general solution to computer vision** (ignoring performance, cost, energy consumption, etc. for now).
- Nevertheless, they are not always right! **They hallucinate** if they can't generalize.
- So, do they **really reason**? Probably not! They only compile and apply the statistical essence of data – but they don't understand.
- And they are still **mainly based on natural language**.
- They don't know much of the **rules** on how our world works.

JKU JOHANNES KEPLER
UNIVERSITY LINZ

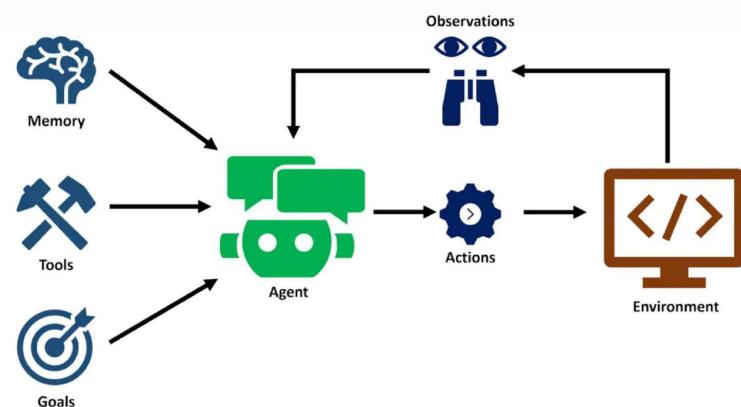
412

Neurosymbolic AI



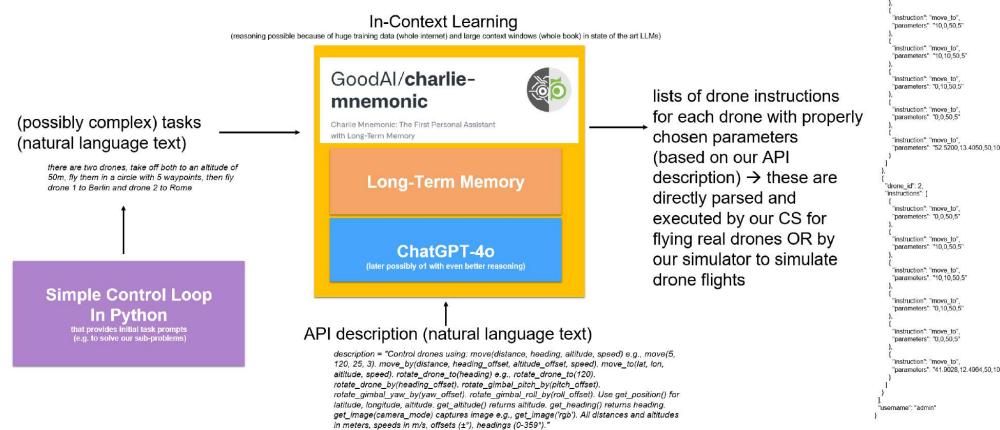
JKU JOHANNES KEPLER
UNIVERSITY LINZ

Agentic AI

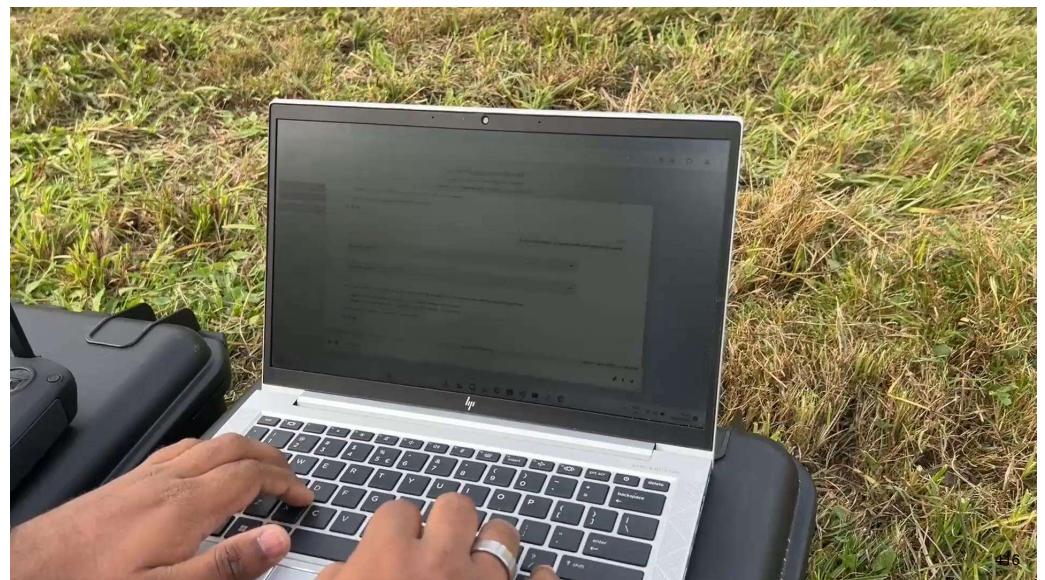


413

Example from our Research



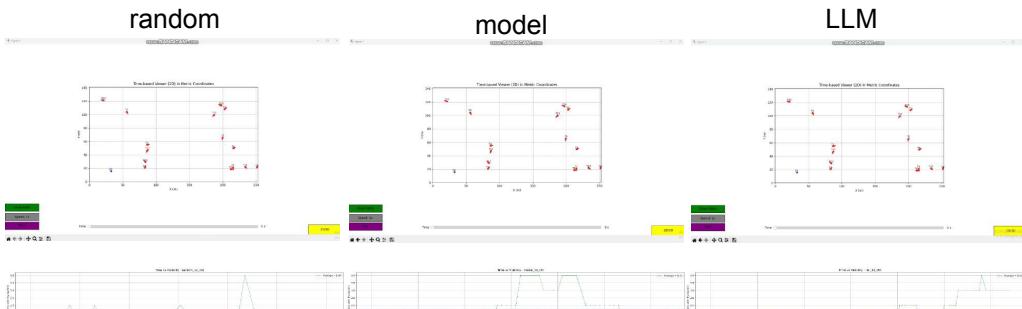
415



414

Example from our Research

1 drone, 15 targets

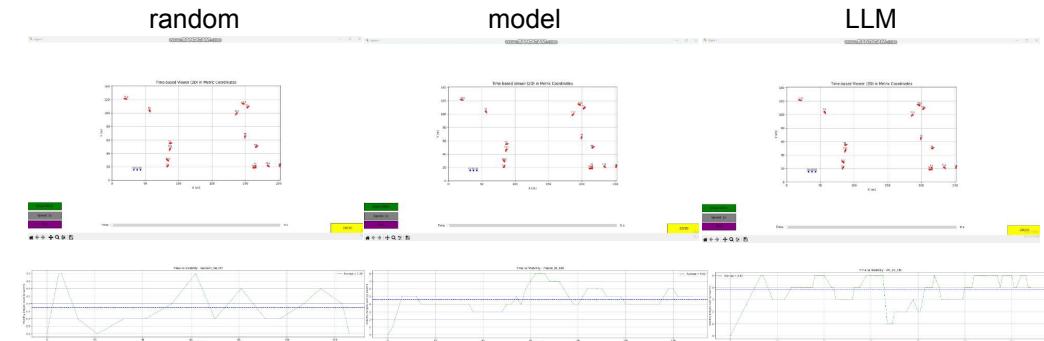


JKU JOHANNES KEPLER
UNIVERSITY LINZ

417

Example from our Research

3 drones, 15 targets



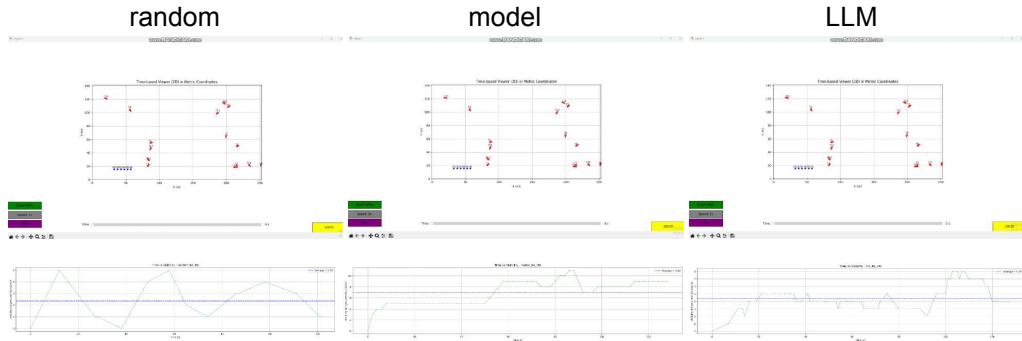
JKU JOHANNES KEPLER
UNIVERSITY LINZ

JOHANNES KEPLER
UNIVERSITY LINZ

418

Example from our Research

6 drones, 15 targets

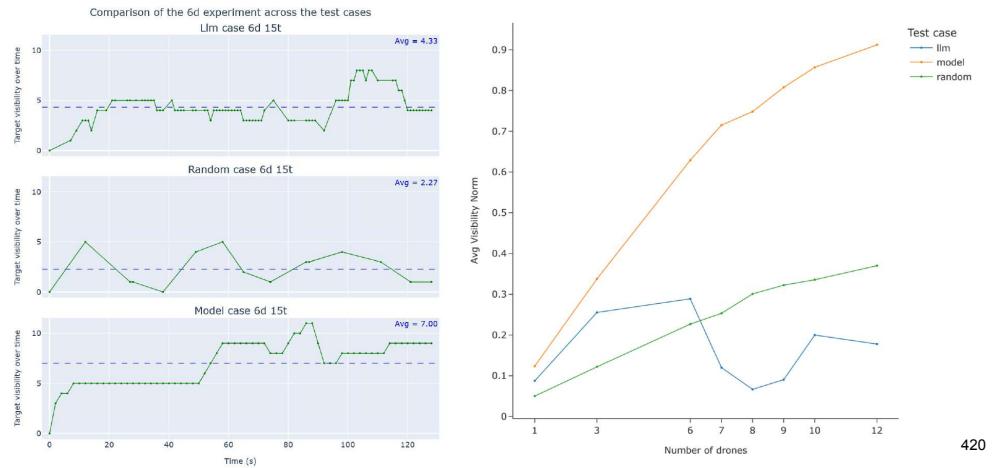


JOHANNES KEPLER
UNIVERSITY LINZ

419

Example from our Research

Normalized average visibility across all experiments



420

Conventional Imaging

Has not changed since 350BC (first mentioning of Camera Obscura = Pinhole Camera, by Aristotle)

Became analog in 1826 (first permanent Photograph taken in France)

Became digital in 1975 (Kodak invented the first digital Camera Sensor)

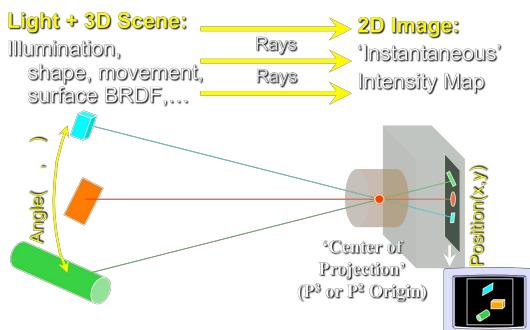
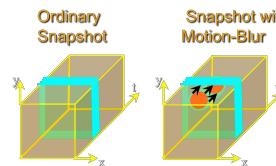


Image Aggregation

Ordinary Snapshot

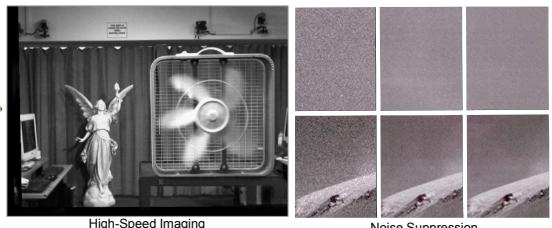


Snapshot with Motion-Blur



'Motion Picture'

(missing time!)



High-Speed Imaging

Noise Suppression

Depth from Defocus

Combine multiple conventional Images computationally

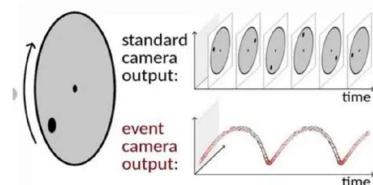


Super-Resolution



Panorama Imaging

Event Cameras



2026 Trends in Computer Vision

Gemini

Top Computer Vision Trends in 2026

- **Edge-Native Inference**: massive shift from cloud-based processing to local Vision NPUs (Neural Processing Units). Devices now perform complex real-time analysis on-device to ensure data privacy and near-zero latency.
- **Multimodal World Model**: Computer vision is no longer isolated. Models now integrate visual data with audio and text simultaneously, allowing AI to "reason" about a scene (e.g., a security camera explaining why a situation looks suspicious in plain English).
- **4D Imaging & Spatial Intelligence**: The integration of 4D imaging radar and LiDAR has become standard in autonomous systems. This allows for precise depth and velocity sensing in "all-weather" conditions where traditional RGB cameras fail.
- **Event-Based Neuromorphic Vision**: Instead of traditional frame-by-frame recording, sensors now only capture pixel changes. This "event-based" approach allows for ultra-high-speed motion tracking with a fraction of the power consumption.
- **Synthetic Data as the Gold Standard**: To solve the "data gap" for rare edge cases (like perfectly labeled synthetic training environments), AI researchers are turning to generative AI to create photorealistic, perfectly labeled synthetic training environments.
- **Foundation Models for Video (SAM 2.0 Beyond)**: The "Segment Anything" trend has evolved. Systems can now track and segment any object across video frames with perfect consistency, revolutionizing automated video editing and medical surgery assistance.
- **Human-Centric Behavior Analytics**: Beyond facial recognition, 2026 focuses on intent detection. Computer vision in retail and automotive now analyzes micro-gestures and eye dynamics to predict a person's next move or state of impairment.

how

where

what

Chat-GPT

Core Technology & Models

- **Fusion & multimodal vision models** — large vision models that combine image, text, and language understanding for richer interpretation and reasoning. [\[more\]](#)
- **Vision-language action systems** powering embedded AI & robotics (integrating vision with motion control). [\[more\]](#)
- **Vision Transformers & advanced architectures** replacing older CNNs for better performance. [\[more\]](#)
- **Deployment & Architecture**

 - **Real-time edge vision** — device inference for low latency and better privacy. [\[more\]](#)
 - **Autonomous, digital twin validation tools** — virtual sensor/test environments for safer training/testing across autonomous systems. [\[more\]](#)

- **Advanced Sensing & Data**

 - **3D vision & LiDAR integration** — richer spatial understanding for AIV, robotics, AIVR, [\[more\]](#)
 - **Event-based and neuromorphic sensors** — highly efficient, low-power vision for dynamic scenes. [\[more\]](#)

- **Ethics & Privacy**

 - **Privacy-first vision systems** (anonymization, regulatory compliance) as visual AI spreads into society. [\[more\]](#)

- **Applications & Impact**

 - **Autonomous mobility & robotics** — vision as a core enabler of safe autonomy. [\[more\]](#)
 - **Healthcare imaging & diagnostics** — improved clinical insights from visual analysis. [\[more\]](#)
 - **Smart infrastructure & surveillance** — real-time monitoring for cities and industry. [\[more\]](#)

Claude

Based on the latest information, here are the key computer vision trends for 2026:

- **Physical AI and edge deployment**: Real-time inference running on local AI chips with many award-winning products at CES 2026. [\[Basic\]](#)
- **Edge AI for real-time processing**: Processing data locally on devices for autonomous driving, surveillance, and industrial automation, reducing cloud dependency. [\[Software Solutions\]](#)
- **Vision Perception (ViT)**: Neural networks using self-attention mechanisms outperforming CNNs in image recognition and object detection. [\[Visual\]](#)
- **Generative AI Integration**: Using GANs and diffusion models to create synthetic data and expand training datasets. [\[Visual\]](#)
- **3D computer vision**: Depth sensing using structured light, time-of-flight sensors, and stereo vision for VR, AR, and robotics. [\[Visual\]](#)
- **4D imaging/radar fusion**: Combining multiple radar sensors with deep learning for cost-effective autonomous vehicle perception. [\[Basic\]](#)
- **Ethical AI and bias mitigation**: Increasing focus on addressing biases in image recognition and analysis systems. [\[more\]](#)
- **Multimodal AI**: Combining computer vision with natural language processing for richer understanding.

Interested in Research Projects?

ICG

Basic Research: Light Fields
Displays
Computational Imaging & Optics
Computer Vision / Image Processing

Areas:
Aerial Imaging, Microscopy, Thin-Film Sensing, 3D Displays, Autonomous Drone and Swarms

Applications:
Search and Rescue, Wildlife Observation, Wildfire Detection, Forest Ecology, Optogenetics, User Interfaces

Funding: FWF, DFG, Upper Austria, LIT, EU, Microsoft Research, FFG, Helmholtz Association, Nokia, e.on

Current Collaborators:
Cambridge U, EPFL, DLR, BOKU, OÖELFV, U. Magdeburg, Helmholtz UFZ

oliver.bimber@jku.at

INSTITUTE OF COMPUTER GRAPHICS

Exam

Places: HS1 (Linz), S1+S3 (Vienna), R4/R5 (Bregenz).

Process: We will check your student IDs at 12:00-12:20. The exam will be open from 12:20-13:20. Results are available after 14:00. On average you need 3 minutes per question (60 minutes in total for 20 questions). However, some answers need more time, while others need less. Please read the questions and answer options very carefully! If you are asked for numbers, please type them in without units (only numbers).

Regulations: This is a closed-book exam. You are allowed to use your laptop (only the test page open - nothing else), clean paper, and pencil only (no additional mobiles, no headsets). Moodle logfiles will be checked for consistent user-IDs and IP-addresses. Any suspected chicanery will lead to a failed attempt of the entire course (without option to attend the retry exam in 2026). All students are responsible to ensure a laptop, browser, internet connection, and access to the Moodle exam from the lecture hall/seminar rooms. Every student who is not present for checking the student ID will not be able to participate. On your desktop only the Moodle exam is allowed to be open.

Rules: Questions and answer options might contain typos. If you have difficulties in understanding (or are unsure about) a question or an answer option, inform the supervisor immediately during the exam. We will neither discard questions with typos from the grading later on, nor accept complaints that questions could not be answered correctly due to misunderstandings. Any results that you will see in Moodle are preliminary results until you receive your final grade in KUSSS. Preliminary results might be subject to changes (e.g., when questions have to be removed after the exam) that might have a (positive or negative) influence to your final grade. You have no right to insist on preliminary grading results. Your grades in KUSSS are final.