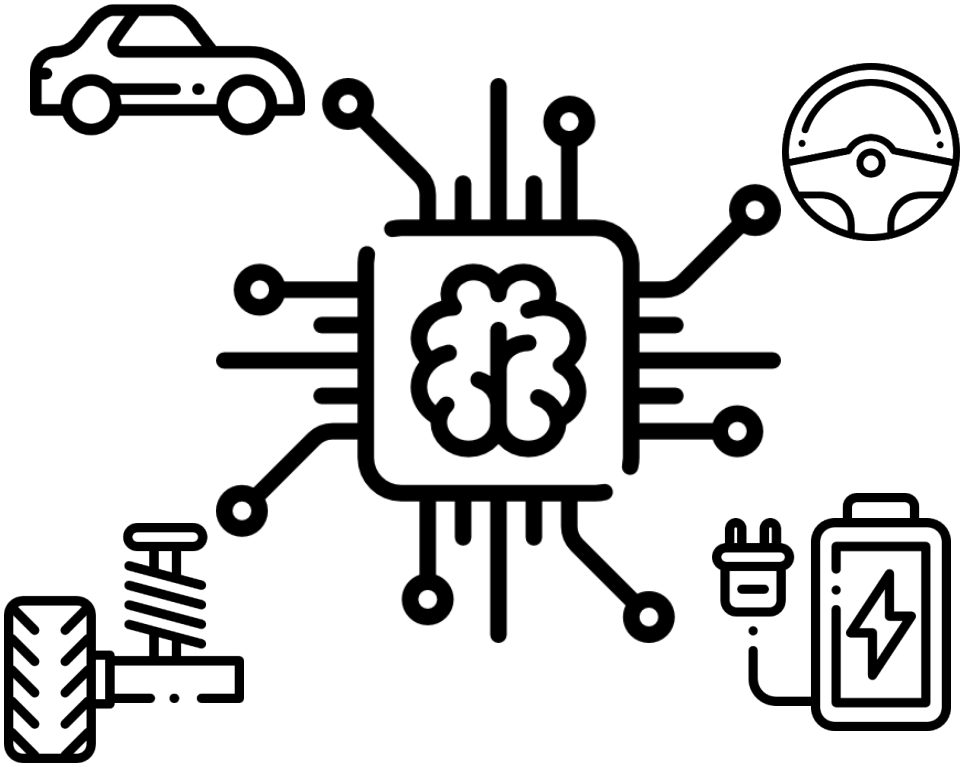
Institute of Automotive Technology
TUM School of Engineering and Design
Technical University of Munich

Artificial Intelligence in Automotive Technology

Maximilian Geißlinger / Fabian Netzler

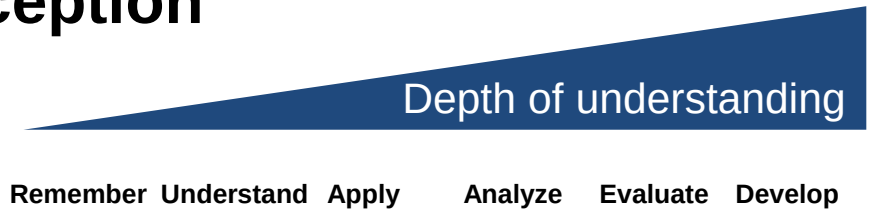
Prof. Dr.-Ing. Markus Lienkamp



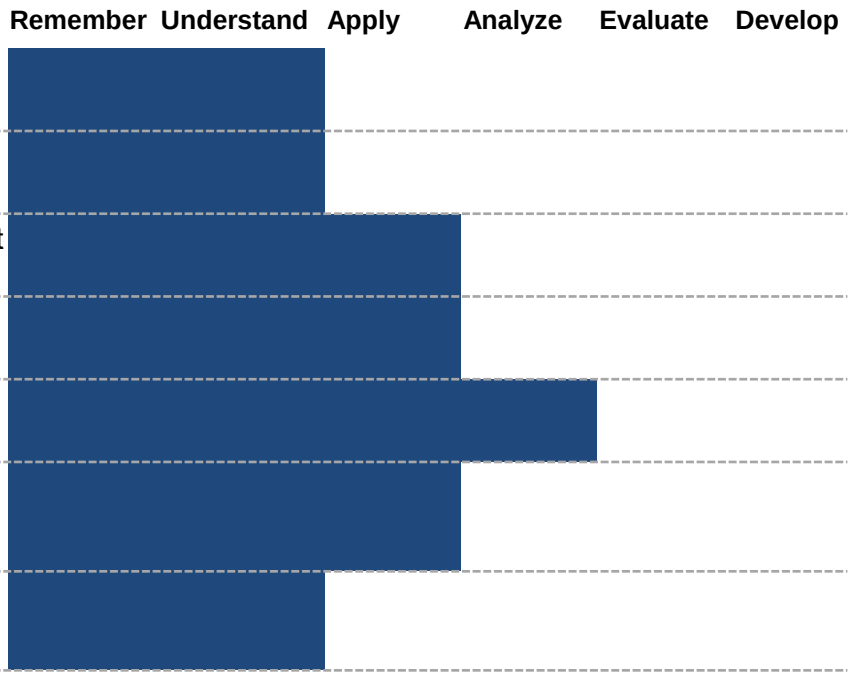
Lecture Overview

Lecture 16:15-17:45 | Practice 17:45-18:30	
1 Introduction: Artificial Intelligence	20.10.2022 – Maximilian Geißlinger
2 Perception	27.10.2022 – Sebastian Huber
3 Supervised Learning: Regression	03.11.2022 – Fabian Netzler
4 Supervised Learning: Classification	10.11.2022 – Andreas Schimpe
5 Unsupervised Learning: Clustering	17.11.2022 – Andreas Schimpe
6 Introduction: Artificial Neural Networks	24.11.2022 – Lennart Adenaw
7 Deep Neural Networks	08.12.2022 – Domagoj Majstorovic
8 Convolutional Neural Networks	15.12.2022 – Domagoj Majstorovic
9 Knowledge Graphs	12.01.2023 – Fabian Netzler
10 Recurrent Neural Networks	19.01.2023 – Matthias Rowold
11 Reinforcement Learning	26.01.2023 – Levent Ögretmen
12 AI-Development	02.02.2023 – Maximilian Geißlinger
13 Guest Lecture	09.02.2023 – to be announced

Objectives for Lecture 2: Perception



After the lecture you are able to…



… understand the problems behind computer vision and apply them to autonomous driving.

…. understand the individual steps of the computer vision pipeline.

… connect any camera to the computer and extract an image from that camera.

… remember and apply all variants of image processing.

…. analyze different images regarding their features.

…. remember variants of the feature extraction and apply them.

…. know and understand variants of feature analysis

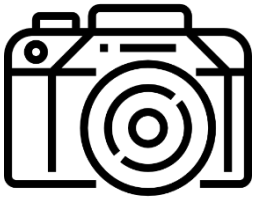
Perception

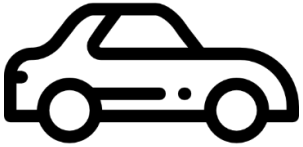
Prof. Dr.-Ing. Markus Lienkamp

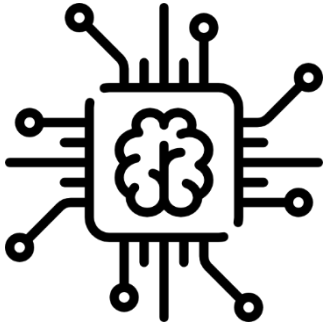
Sebastian Huber

Agenda

1. Introduction
2. Computer Vision
3. Image Processing
4. Feature Extraction
5. Image Classification
6. Video Analysis
7. Summary



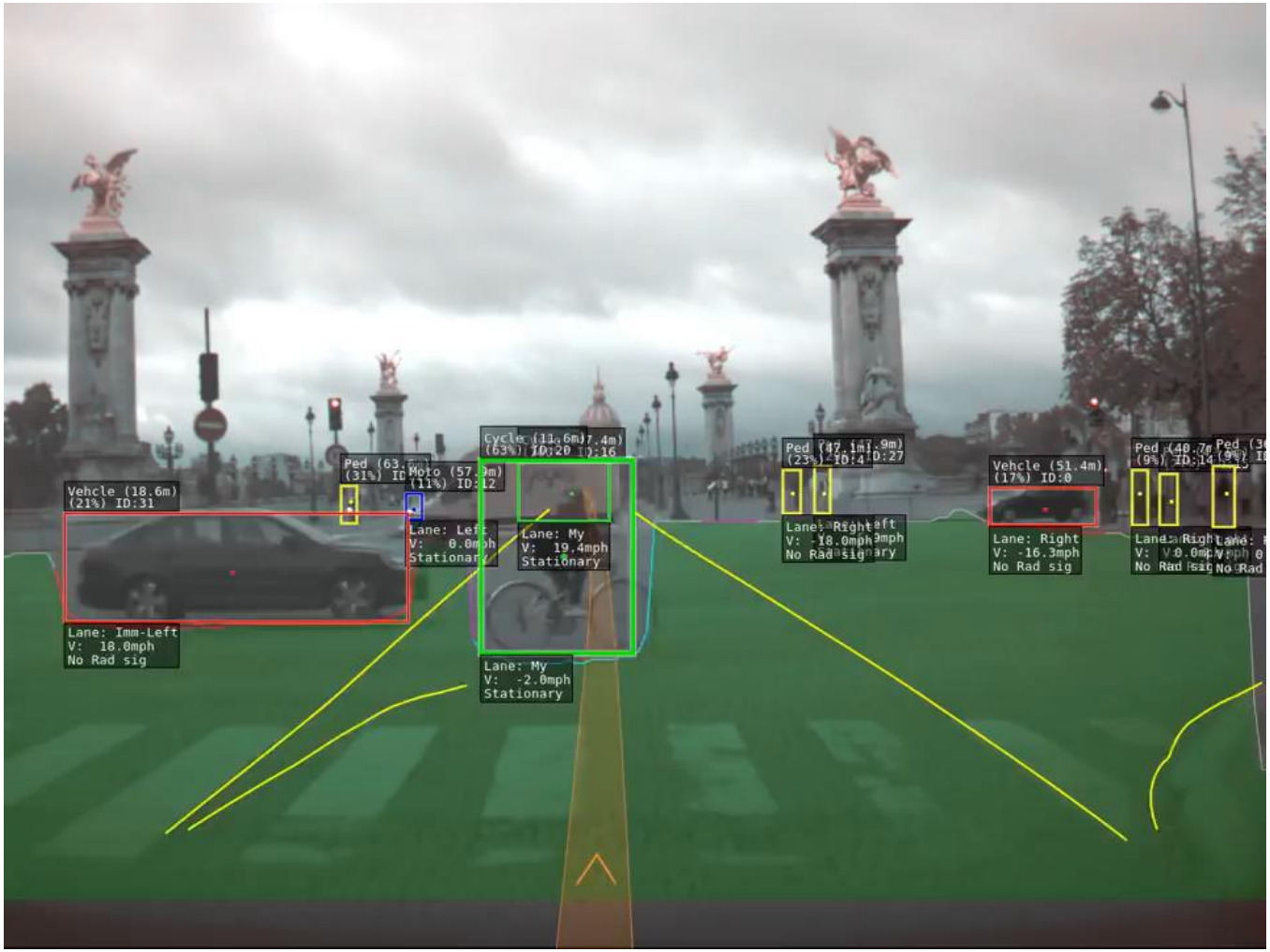




The world is a complex and dynamic place

https://www.youtube.com/watch?v=FXfGZF2-sUU

The world is a complex and dynamic place: Perception



„Paris Streets in the eyes of Tesla Autopilot“ https://www.youtube.com/watch?v=_1MHGUC_BzQ

The world is a complex and dynamic place: Edge Cases

https://www.youtube.com/watch?v=7UF-S2czdCk
https://www.reddit.com/r/teslamotors/comments/nrs8kf/you_think_ice_cream_truck_stop_signs_are_a_problem/

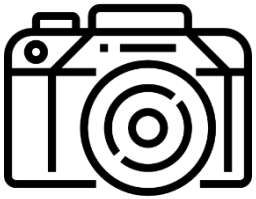
Perception

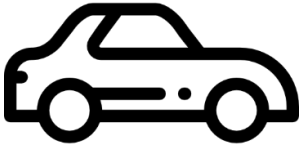
Prof. Dr.-Ing. Markus Lienkamp

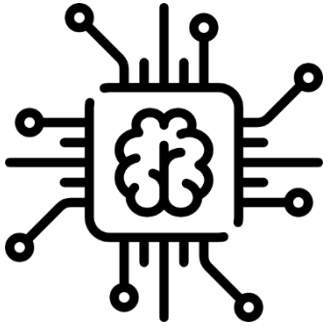
Sebastian Huber

Agenda

1. Introduction
2. **Computer Vision**
3. Image Processing
4. Feature Extraction
5. Image Classification
6. Video Analysis
7. Summary







Computer Vision: A Subfield of Machine Perception

Machine Perception:

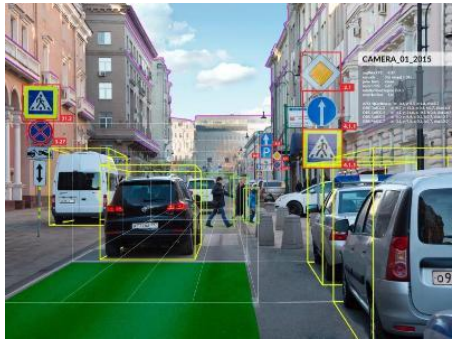
- A computer is acquiring the ability to interpret data related to the environment
- Sensors are used as input: Camera, Lidar, Radar, Ultrasonic, Microphones, …
- **Computer Vision (CV):** The camera input (images/videos) is analyzed and information is extracted

Focus of this lecture: Cameras and Computer Vision

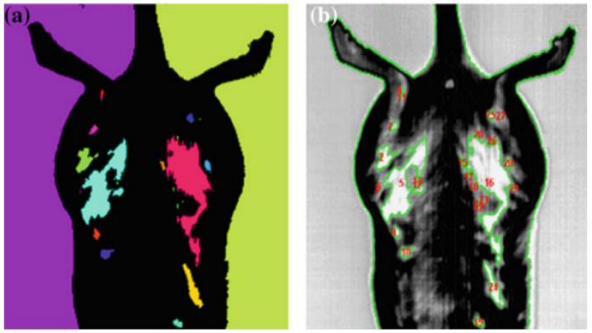
- Cheapest sensor of the car
- Similar to human eyes
- Includes most important information about the environment
- Images can be processed by classical CV-Algorithms and Deep Learning-Algorithms

https://www.bhphotovideo.com/c/product/1308820-REG/nikon_d5600_dslr_camera_with.html

Computer Vision: Why it matters



Autonomous Driving



Health

Fun & Games

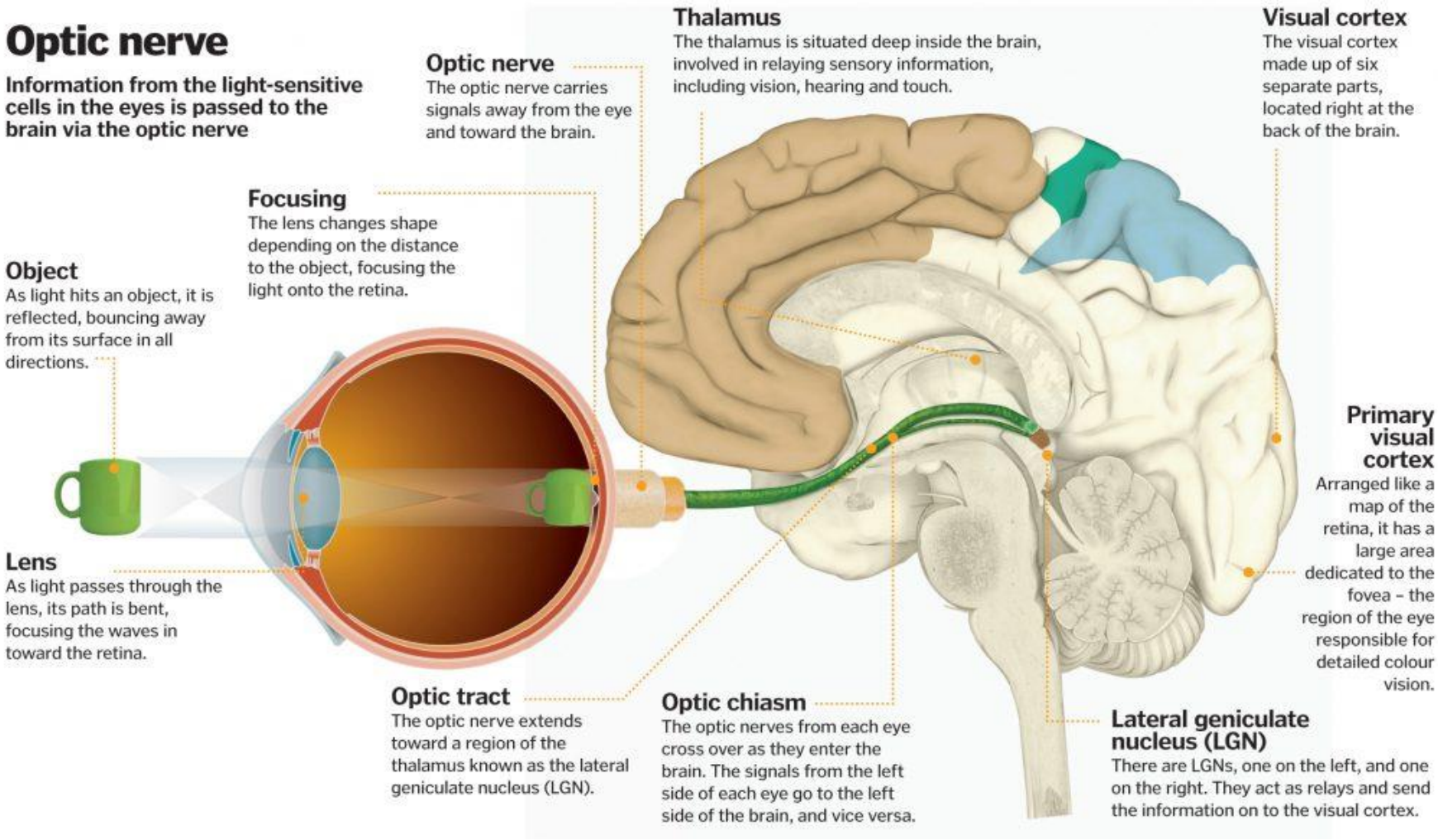
Comfort

Security

Access

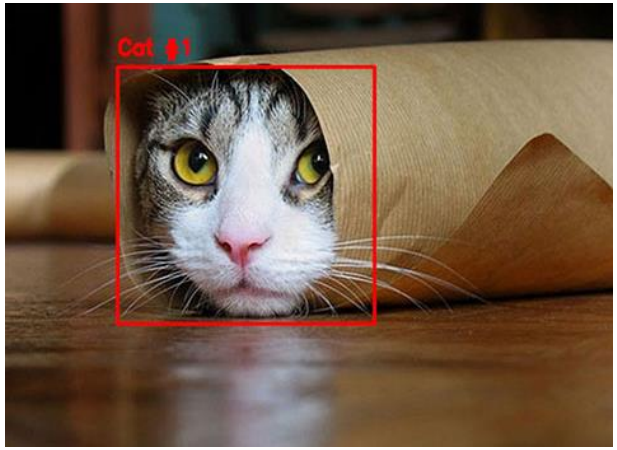
http://www.magazine.unimore.it/site/home/international-news/articolo820043703.html
https://medium.com/health-ai/heat-screening-for-breast-cancer-computer-vision-for-dieting-brain-wave-passwords-c5294b5e73b
https://www.express.de/duesseldorf/duesseldorf-archiv/fahndungsfoto-hier-ueberfallen-zwei-raeuber-eine-tankstelle-in-duesseldorf--24421170
http://gamechanger.co.ke/product/xbox-360-kinect-sensor, http://osxdaily.com/2017/11/10/can-use-iphone-x-without-face-id

Computer Vision vs. Human Vision



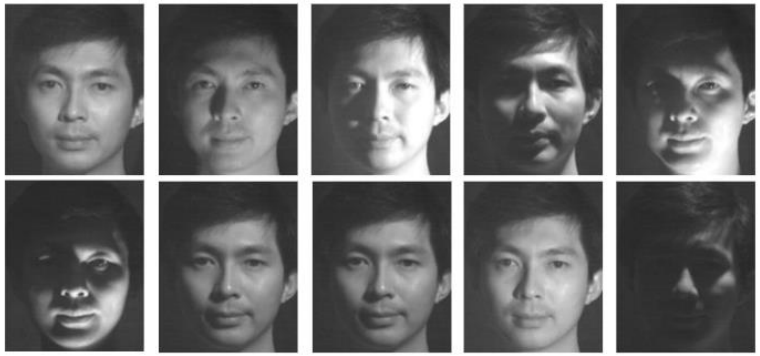
https://www.howitworksdaily.com/science-of-vision-how-do-our-eyes-enable-us-to-see/

Computer Vision: Challenges



Occlusion

Viewpoint variation

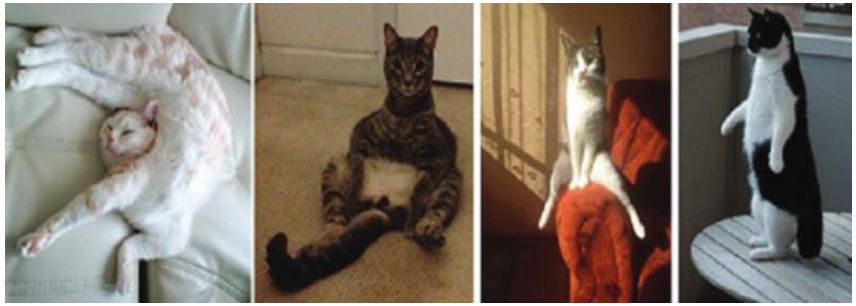


Illumination

Background clutter

https://www.pyimagesearch.com/2016/06/20/detecting-cats-in-images-with-opencv/, https://slideplayer.com/slide/7773132, https://pdfs.semanticscholar.org/af7e/5f06a88fe8d63f22406846022b29dc1fd3f2.pdf

Computer Vision: Challenges



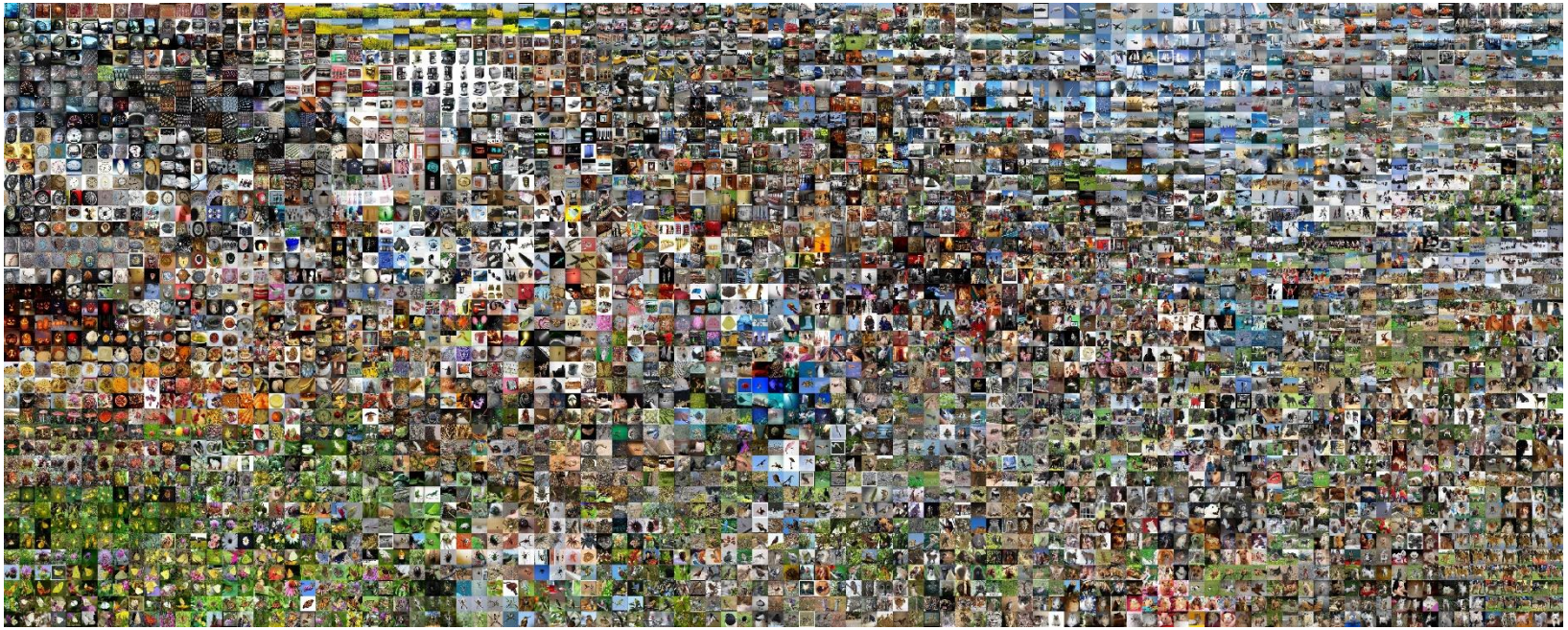
Deformation

Intra-class variation

Inter-class similarities

https://pdfs.semanticscholar.org/af7e/5f06a88fe8d63f22406846022b29dc1fd3f2.pdf, https://barkpost.com/humor/doodle-or-fried-chicken-twitter/
https://www.researchgate.net/publication/325311506_Training_Convolutional_Networks_with_Web_Images

Computer Vision: Challenges



We need a lot of labelled images!

ImageNet:

- >14 million annotated images
- >20,000 categories, e.g. 120 dog breeds
- hierarchical structure

https://awesomeopensource.com/project/matteo-dunnhofer/imagenet

Computer Vision: History

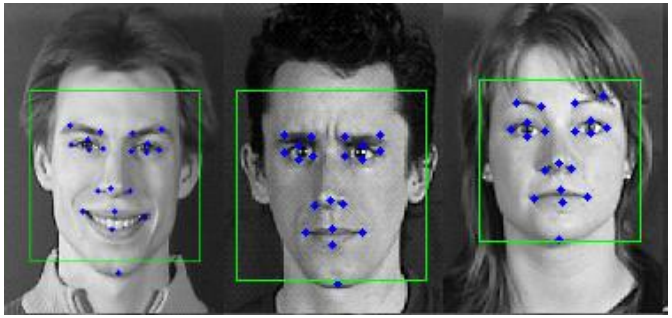
- 1966: Marvin Minsky assigns computer vision as an undergrad summer project
- 1970's: 1st AI winter
- 1980's: **Rule based** expert systems; 2nd AI winter
- 1990's: Face recognition; statistical analysis en vogue
- 2000's: Broader recognition; large annotated datasets available; video processing starts
- 2010's: Deep learning with ConvNets
  → **Data Driven**

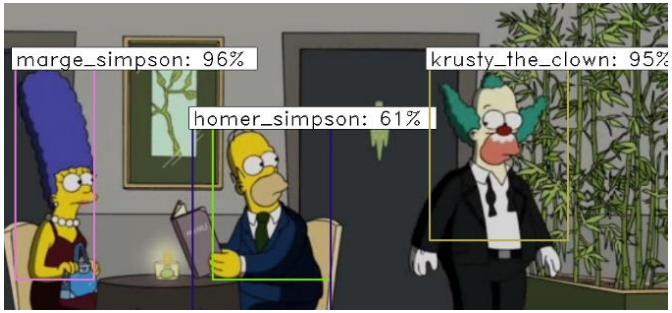
Marvin Minsky, MIT
Turing award, 1969

"In 1966, Minsky hired a first-year undergraduate student and assigned him a problem to solve over the summer:

connect a camera to a computer and get the machine to describe what it sees."

Crevier 1993, pg. 88

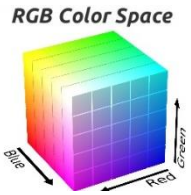


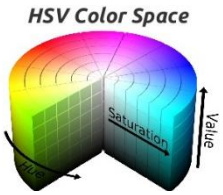


https://www.slideshare.net/tw_dsconf/computer-vision-crash-course-56934508
https://ibug.doc.ic.ac.uk/research/detection-static-geometric-facial-features
https://computervisionblog.wordpress.com/2018/03/25/deep-learning-approaches-for-object-detection

Computer Vision: The CV-Pipeline

OpenCV





1. Choose Hardware: Camera, Lens,…

→

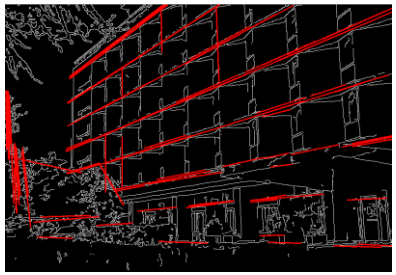
2. Acquire an image/video stream

→

3. Integrate the image in your software pipeline

→

4. Preprocess the image for further evaluation



5. Detect/Extract features in the image

←

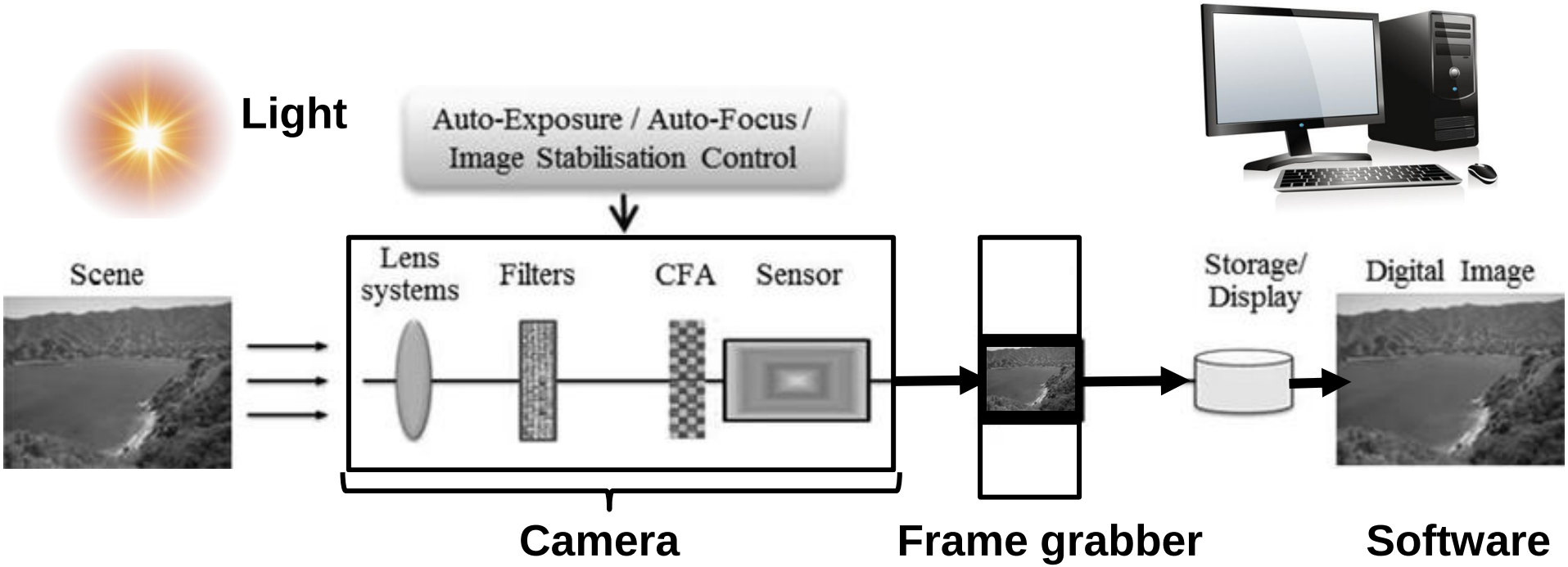
6. Localize object

←

7. Classify object

https://picclick.com/Point-Grey-Flea2-FL2-08S2M-IEEE-1394-Mono-Digital-Camera-252748955187.html, https://www.pexels.com/photo/man-holding-black-silver-bridge-camera-taking-photo-during-daytime-167571, https://de.wikipedia.org/wiki/Datei:OpenCV_Logo_with_text.png, https://www.researchgate.net/publication/323952018_Practices_and_pitfalls_in_inferring_neural_representations/figures, https://stackoverflow.com/questions/35299878/probabilistic-hough-transform-line-segment-connection, https://www.autoscout24.de/auto/audi/audi-r8/

Computer Vision: Image Acquisition



Lens

https://www.researchgate.net/publication/281476188_Smartphone_image_acquisition_forensics_using_sensor_fingerprint/figures?lo=1, https://www.activesilicon.com/de/product-category/frame-grabbers-de
https://picclick.com/Point-Grey-Flea2-FL2-08S2M-IEEE-1394-Mono-Digital-Camera-252748955187.html, https://de.wikipedia.org/wiki/Datei:OpenCV_Logo_with_text.png
http://home.bt.com/tech-gadgets/computing/give-your-pc-a-new-lease-of-life-5-upgrade-options-you-should-be-considering-11363940257353

Computer Vision: Image Acquisition

- **Lights:** The human eye can see well over a wide range of lighting conditions, but a machine vision system is **not as capable.** You must therefore carefully illuminate the scene under observation so that the machine vision system can **"see" it clearly.**
  → **Problem for Autonomous Driving:** Light changes during the day, weather etc.

- **Optics and lenses:** The lens gathers the light reflected (or transmitted) from objects in the camera's field-of-view, and forms an image in the camera sensor. The proper lens allows you to adjust the **field-of-view (FOV)** and to place the camera at a convenient working distance from the scene. To pick the proper lens you will first need to know the **FOV** and the working distance. The **FOV** is the size of the area you want to capture.

https://www.teledynedalsa.com/en/learn/knowledge-center/machine-vision-101-an-introduction/

Computer Vision: Image Acquisition

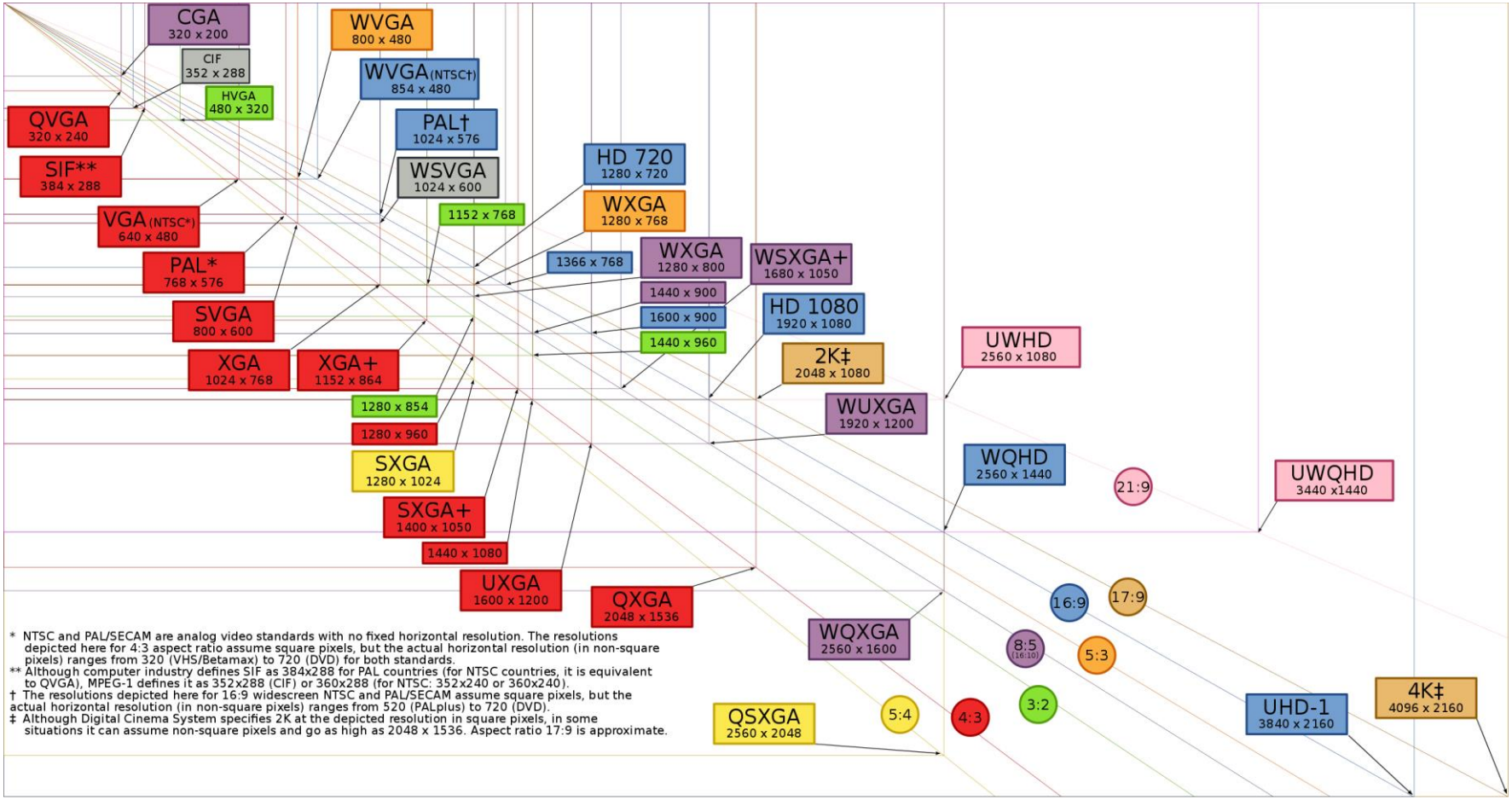
- **Camera:** The camera contains a **sensor** that converts light from the lens into electrical signals. These signals are digitized into an array of values called **pixels.** The resolution (precision) of the inspection depends upon the working distance, the field-of-view (FOV), and the number of physical pixels in the camera's sensor. A standard VGA camera has 640 x 480 physical pixels (= width x height), and each physical pixel is about 7.4 microns square

- **Frame grabber and Software:** Normally, a frame grabber sends the digitized image over a **bus system** (FireWire, USB, Ethernet) to the computer. The grabbing can be done asynchronous or synchronous. After a frame was grabbed, the image is put in the computer memory and can be processed by specific software, e.g. OpenCV

https://www.teledynedalsa.com/en/learn/knowledge-center/machine-vision-101-an-introduction/

Additional Slide

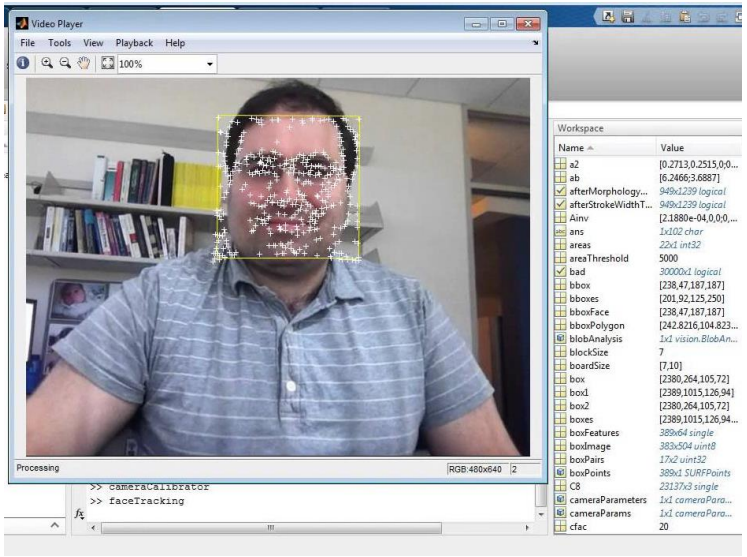
Image Resolution



https://en.wikipedia.org/wiki/Display_resolution#/media/File:Vector_Video_Standards8.svg

Computer Vision Software: MATLAB

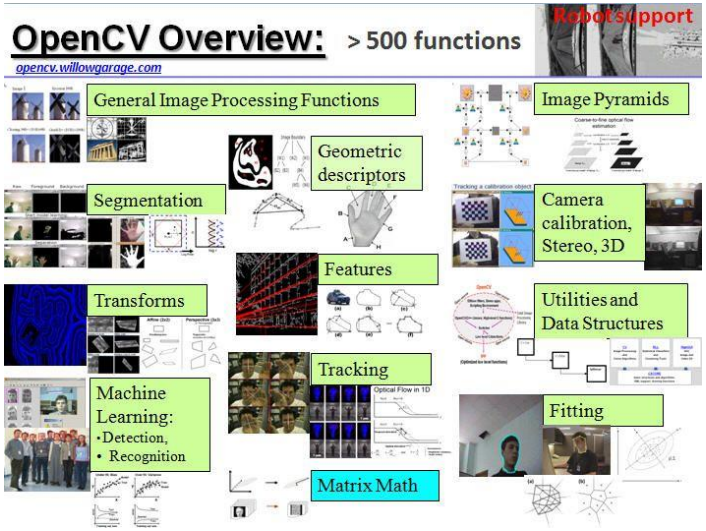
- Commercial Software (Free for **students)**
- Computer Vision Toolbox
- **Operating System:** Linux, macOS, Windows
- **Includes:** Input/Output graphics, CV-basics, Camera Calibration, Feature Detection/Extraction, Deep Learning
- **Pros:** Easy to use, good documentation, GPU boost
- **Cons:** Closed Environment, Performance



https://www.youtube.com/watch?v=yipHjyLUzA4

Computer Vision Software: OpenCV

- Free software for **everyone**
- OpenCV (Open Source Computer Vision)
- **Language:** C++, Python
- **Operating System:** Linux, macOS, Windows
- **Includes:** Input/Output Graphics, CV Basics, Camera Calibration, Feature Detection/Extraction, Deep Learning, Egomotion, Motion Tracking
- **Pros:** Everything you need, Good Documentation, powerful, GPU boost
- **Cons:** Installation (especially GPU)



https://www.pinterest.de/pin/174092341823170778/?lp=true

Demo

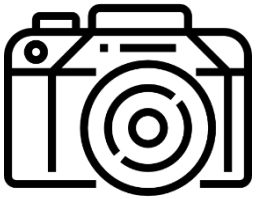
01_camera_input.py

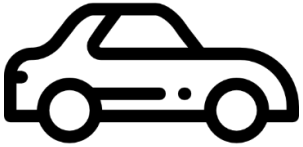
Perception
Prof. Dr.-Ing. Markus Lienkamp

Sebastian Huber

Agenda

1. Introduction
2. Computer Vision
3. **Image Processing**
4. Feature Extraction
5. Image Classification
6. Video Analysis
7. Summary





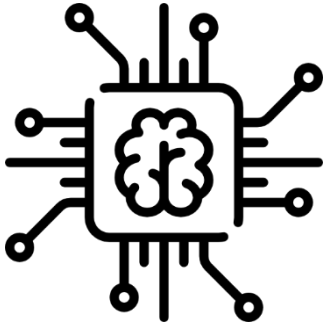


Image Processing

Image processing is a method to perform operations on an image in order to get an **enhanced image** or to enable the **extraction** of useful information from it.

In image processing, different **computer vision algorithms / mathematical operations** are used.

https://shutterstoppers.com/post-processing-macro-photograph-photoshop

Image Processing: The Pipeline

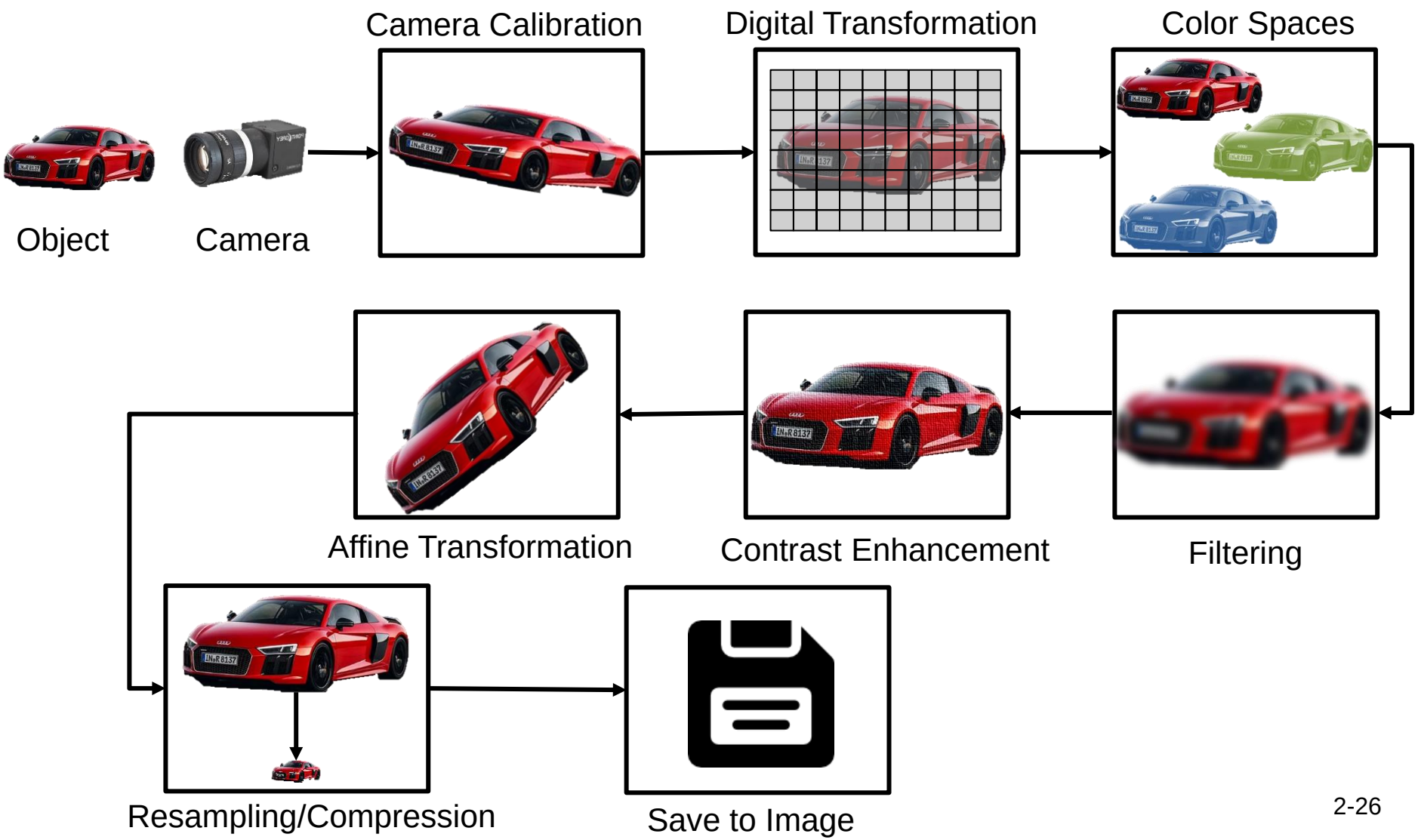


Image Processing: Camera Calibration

- A camera projects 3D world points onto a 2D image plane
- Most cameras produce distortion (radial, tangential)
- **Camera Calibration:** Finding the parameters of the camera and lens that affect the image processing
  - Extrinsic camera parameters: Position of camera center, camera heading,...
  - Intrinsic camera parameters: Focal length, image sensor format,…
  - Lens distortion parameters

https://en.wikipedia.org/wiki/Rectilinear_lens

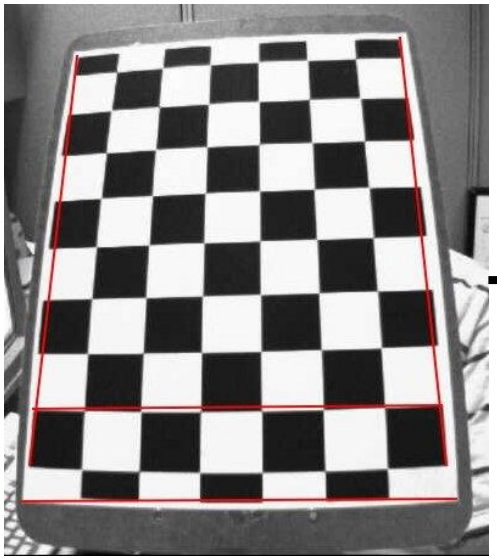
Additional Slide

Types of Distortion

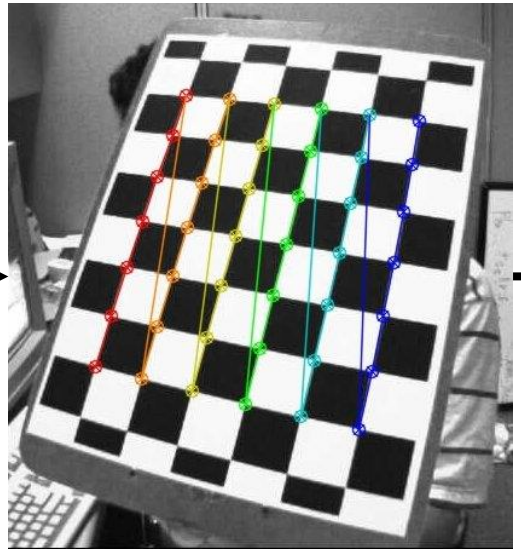
Real cameras use curved lenses to form an image, and light rays often bend a little too much or too little at the edges of these lenses. This creates an effect that distorts the edges of images, so that lines or objects appear more or less curved than they actually are. This is called **radial distortion**, and it's the most common type of distortion.

Another type of distortion, is **tangential distortion**. This occurs when a camera's lens is not aligned perfectly parallel to the imaging plane, where the camera film or sensor is. This makes an image look tilted so that some objects appear farther away or closer than they actually are.

Image Processing: Camera Calibration Process



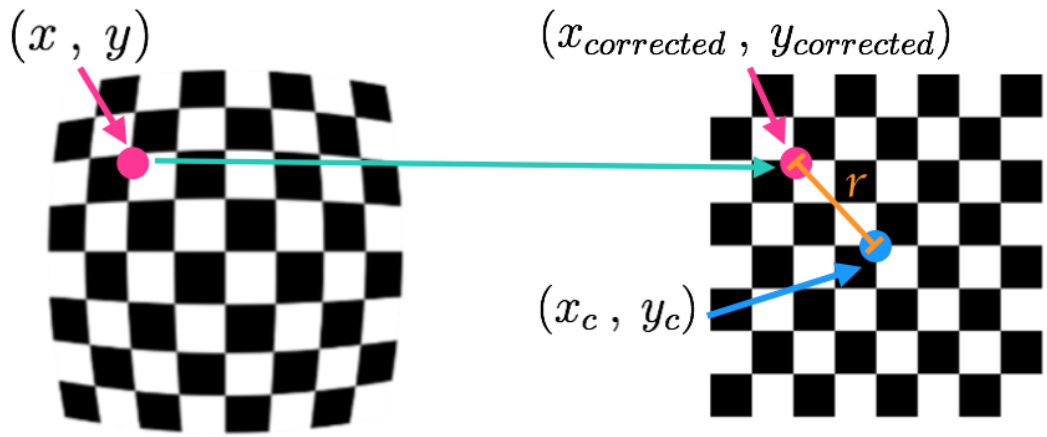
1. Distorted Picture: Lines or objects appear more curved or less curved than they actually are.



2. Find and draw corners in the picture → Find the distortion coefficients and correction

3. Undistortion: Compute the distortion matrix and apply it to the picture

Additional Slide



Distorted **Undistorted**

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

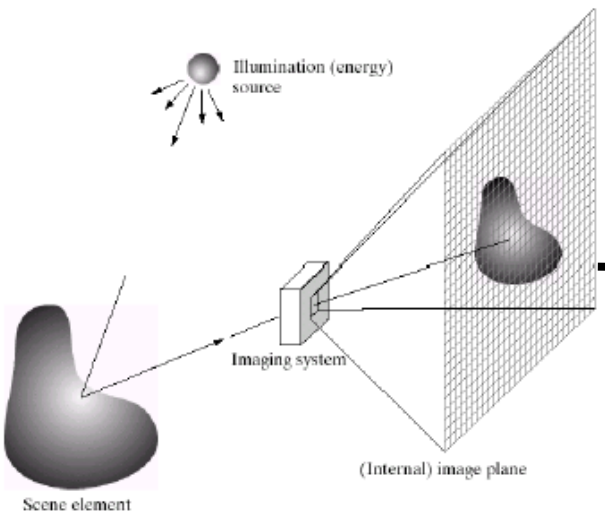
$$x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

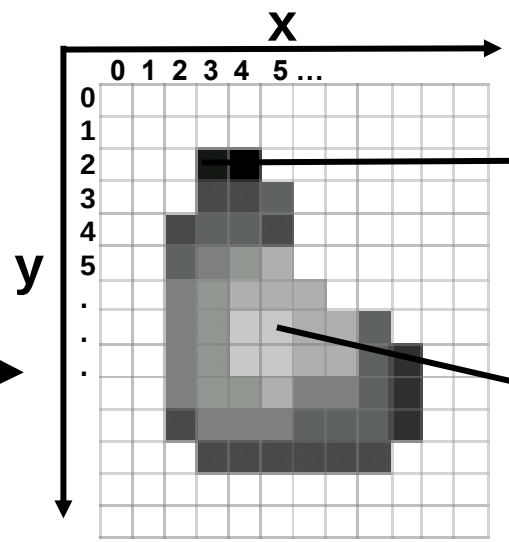
$$Distortion\ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Image Processing: What is an image?

A **digital image** is a numeric representation, normally binary, of a two-dimensional image.
→ A digital image is just a **matrix of values** based on color.



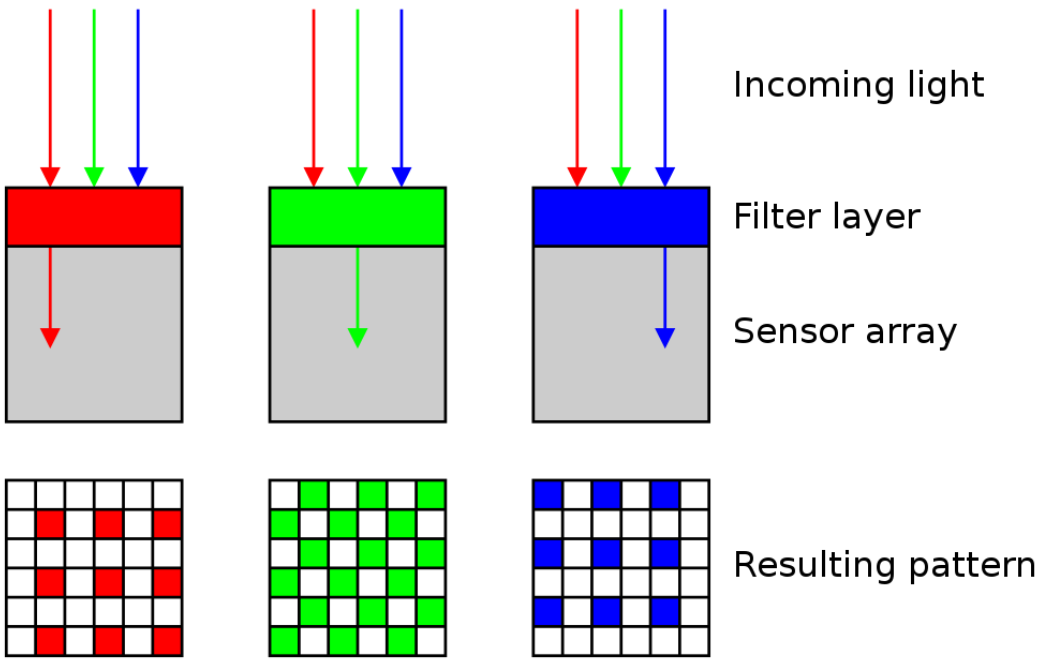


Pixel (3,2):
Red: 20
Green 20
Blue 20

Pixel (5,7):
Red: 150
Green: 150
Blue: 150

Image Processing: What is an image?

- A digital image consists of a rectangular grid of **evenly spaced pixels**.
- Each pixel can be thought of as a measurement or sample of the light from a subject.
- Commonly, the original samples are obtained using a scanner or digital camera by **averaging the amount of red, green and blue light** that falls on the sensitive area of each of its CCD sensing elements.



https://en.wikipedia.org/wiki/Bayer_filter

Image Processing: What is an image?

- Array size is based on the camera resolution
- Suppose, we have a **n×m** RGB image with a variable called “img”
  - img[0, 0, 0] = top-left pixel value in R-channel
  - img[x, y, 1] = x pixels down, y pixels right in G-channel
  - img[m, n, 2] = bottom-right pixel in B-channel

column (n) →

row (m) ↓

Red (R)

6	10	85	45	13	1	1	1	40	34	35
1	40	1	1	1	40	34	1	1	34	34
1	40	1	40	45	1	40	8	13	40	40
45	1	45	1	8	13	39	39	20	34	39
8	13	8	13	13	1	10	10	1	40	34
13	1	13	1	13	13	1	1	13	39	40
13	13	13	13	1	1	8	13	45	10	39
1	1	1	1	45	45	102	3	8	13	10
13	8	13	1	1	8	10	45	1	1	0.99
13	8	13	1	1	1	1	1	1	1	1
13	8	13	8	13	8	13	8	13	8	13

Green (G)

34	35									
34	34									
40	40									
34	39									
40	34									
39	40									
10	39									
13	10									
1	35									
1	1									
13	8	13	1	1	1	1	1	1	1	1
13	8	13	8	13	8	13	8	13	8	13

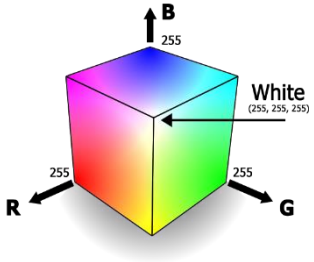
Blue (B)

34	35									
34	34									
40	40									
34	39									
40	34									
39	40									
10	39									
13	10									
1	35									
1	1									
13	8	13	1	1	1	1	1	1	1	1
13	8	13	8	13	8	13	8	13	8	13

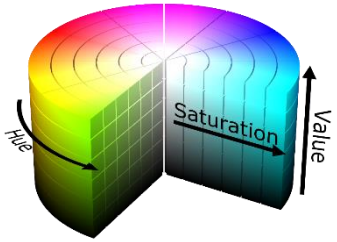
Image Processing: Color Spaces

- Reproducing human color perception/vision
- **Different color spaces for different problems** (monitor, printer, …)
- Different color spaces for more exact color extraction
- Most common color spaces are **RGB** and **HSV**

RGB

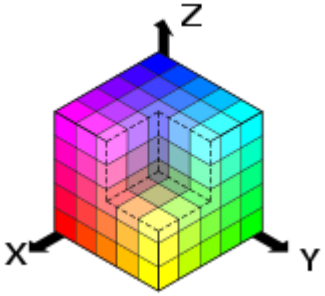


HSV



CMYK

CIE

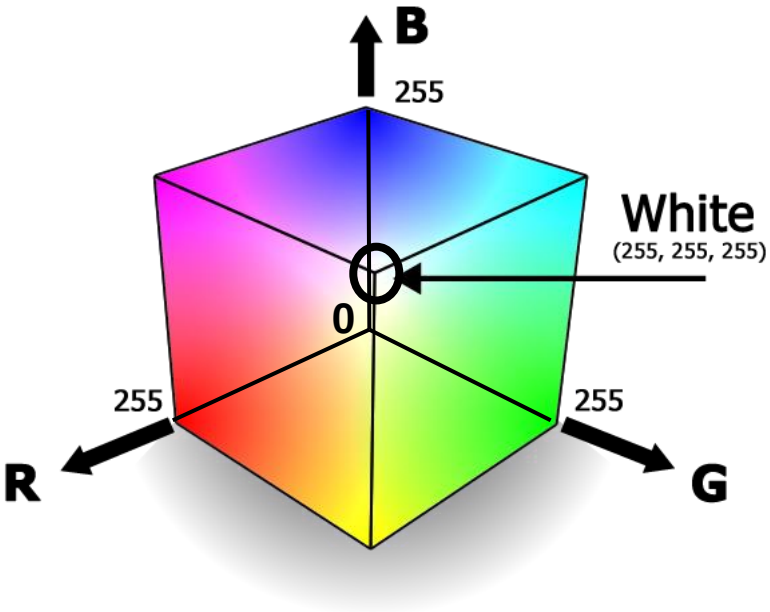


https://old.medialooks.com/mformats/docs/CK%20Advanced.html
https://cs.wikipedia.org/wiki/Soubor:HSV_color_solid_cylinder.png
https://design.tutsplus.com/articles/advanced-color-theory-what-is-color-management--cms-26307
https://www.codeproject.com/articles/243610/the-known-colors-palette-tool-final-revision-hopef

Image Processing: RGB Color Space

RGB – **R**ed **G**reen **B**lue

- Values between 0 – 255 (8bit)
- 0,0,0: Black
- 255,255,255: White



R = 255
(G=0,B=0)

G = 255
(R=0,B=0)

B = 255
(R=0,G=0)

https://old.medialooks.com/mformats/docs/CK%20Advanced.html

Additional Slide

RGB

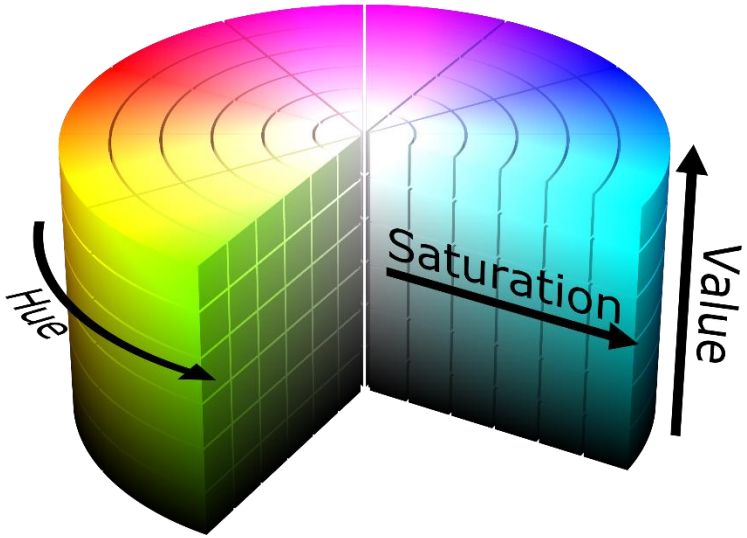
The most common is the RGB color space with 8 bits per channel, corresponding to $(2^8)^3$ = 16,777,216 (approx. 16.8 million) theoretically possible colors.

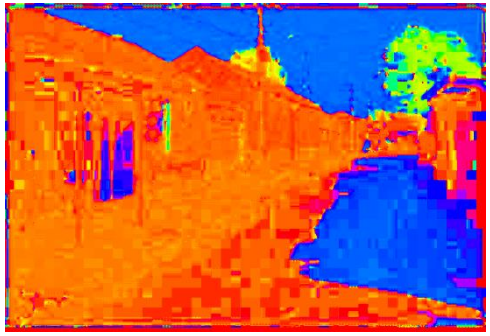
With 16 bits (per channel) this results in 281,474,976,710,656 (281 trillion) color options

Image Processing: HSV Color Space

HSV – **H**ue **S**aturation **V**alue

- Hue (0° Red, 120° Green, 240° Blue)
- Saturation (0% Gray – 100% saturated Color)
- Value (0% dark – 100% light)





H
(S=1,V=1)

S
(H=1,V=1)

V
(H=1,S=0)

https://cs.wikipedia.org/wiki/Soubor:HSV_color_solid_cylinder.png

Demo

02_color_spaces.py

Image Processing: Filtering

- Preprocessing image to **emphasize information** for next steps
- **Convolution** operation with **kernel** (filter/mask/convolution matrix)
- Kernel size and entries depending on purpose for further processing

1	2	3	2	...	...	...
2	1	4	4	...	...	...
1	2	3	4	...	...	...
2	2	2	5	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...

Image

$*$

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

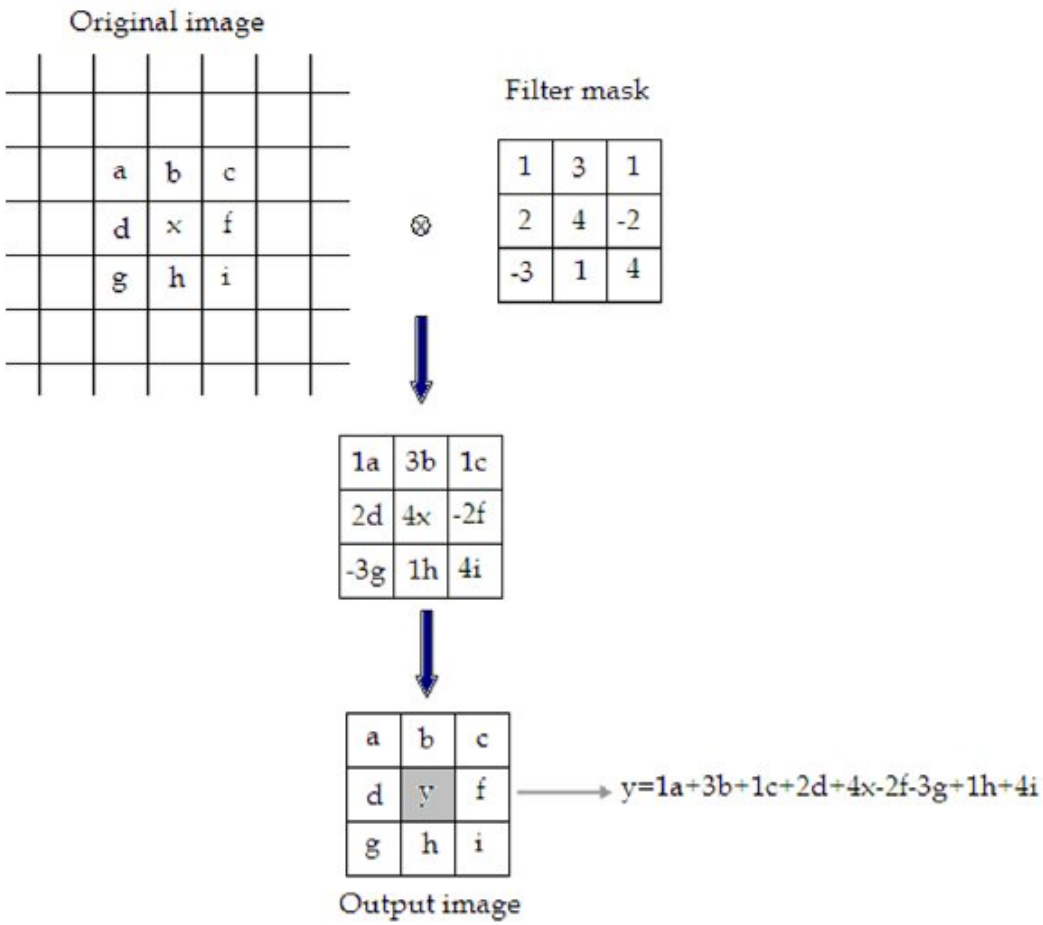
Kernel

$\longrightarrow$

2	3	...	...	...
2	3	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

New Image

Additional Slide



a	b	c
d	e	f
g	h	i

$$\longrightarrow \frac{1}{9}(a+b+c+d+e+f+g+h+i)$$

Image Processing: Filtering Examples

Mean

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian Blur 3x3

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian Blur 5x5

$$\frac{1}{256}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

Sharpen

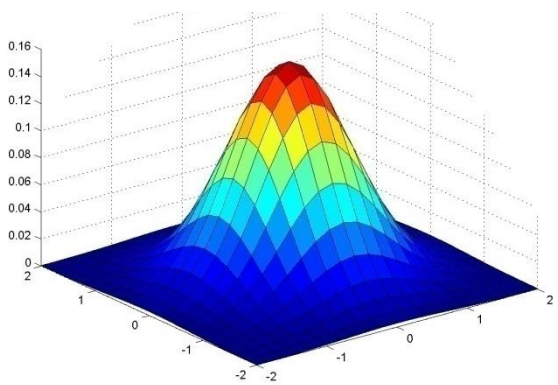
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

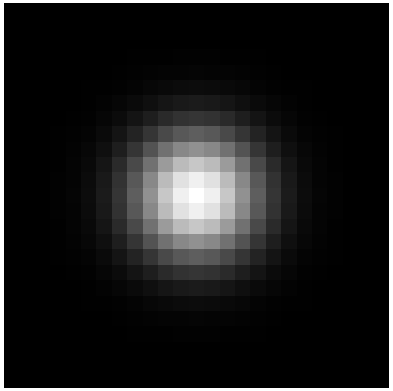
Additional Slide

Gaussian Filter

- Remove “high-frequency” components from the image (low-pass filter)

  Images become more smooth
- Convolution with itself is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

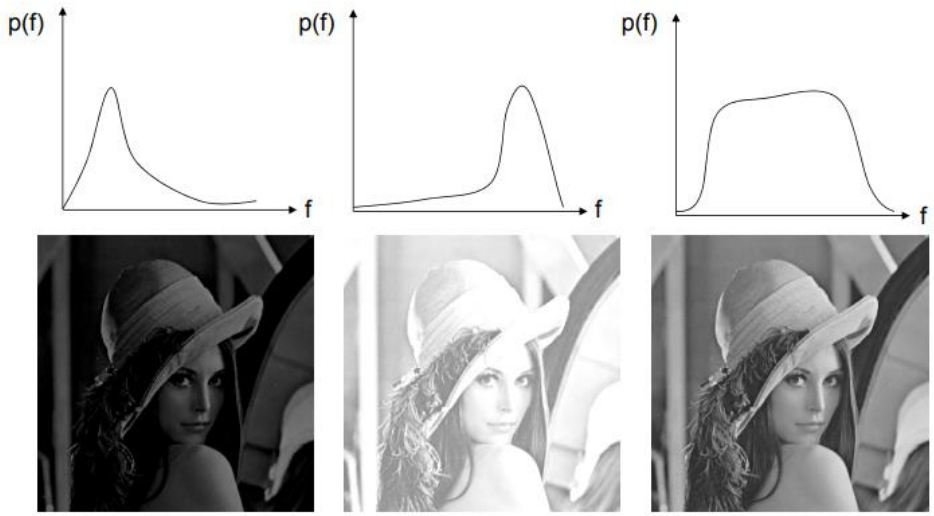
Demo

03_filtering.py

Image Processing: Contrast Enhancement

Contrast: Range of difference in colors and brightness

- **Low contrast**
  → Image values concentrated in a narrow range
- Contrast of an image can be revealed by its histogram
- **Contrast enhancement**
  → Change the image value distribution to cover a wider range.

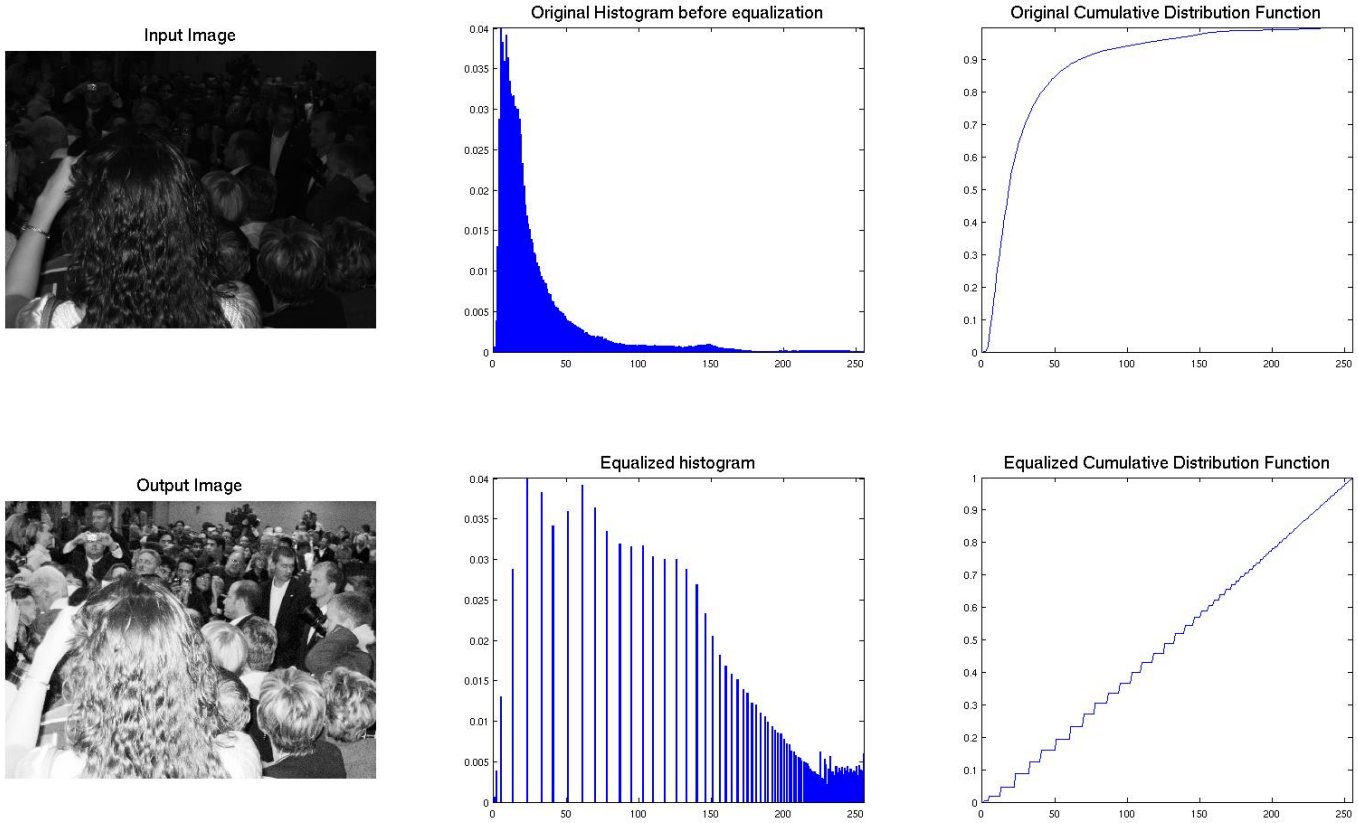


Low contrast | Low contrast | High contrast

http://eeweb.poly.edu/~yao/EL5123/lecture3_contrast_enhancement.pdf

Image Processing: Contrast Enhancement

Histogram Equalization

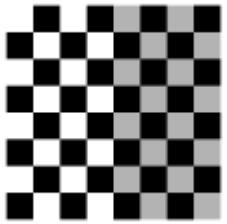


Goal: Linear Cumulative Distribution Function

http://www.sci.utah.edu/~acoste/uou/Image/project1/Arthur_COSTE_Project_1_report.html

Image Processing: Affine Transformations

- Preserves points, straight lines and planes
- Sets of parallel lines remain parallel after an affine transformation
- It does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line.

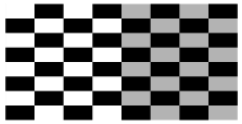


Rotate $\begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Reflect $\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Shear $\begin{pmatrix} 1 & cx=0.5 & 0 \\ cy=0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

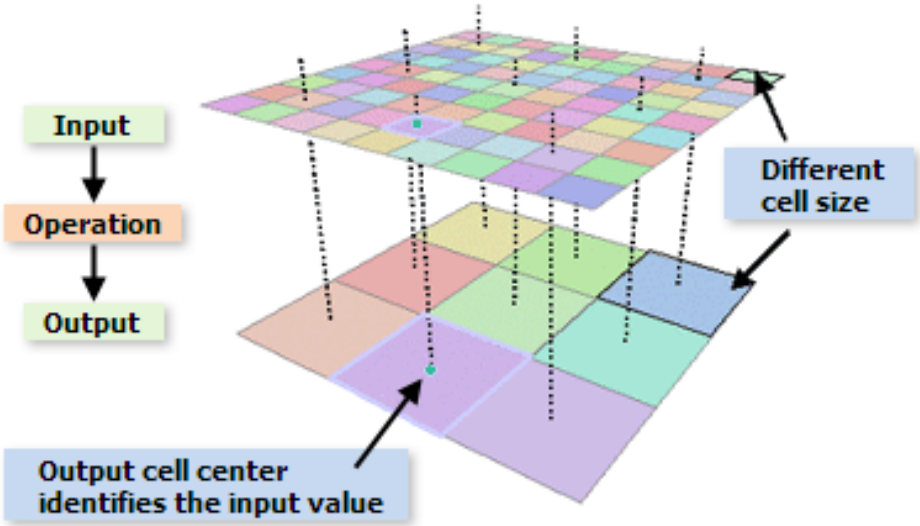
Scale $\begin{pmatrix} cx=2 & 0 & 0 \\ 0 & cy=1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$



https://en.wikipedia.org/wiki/Affine_transformation

Image Processing: Resampling

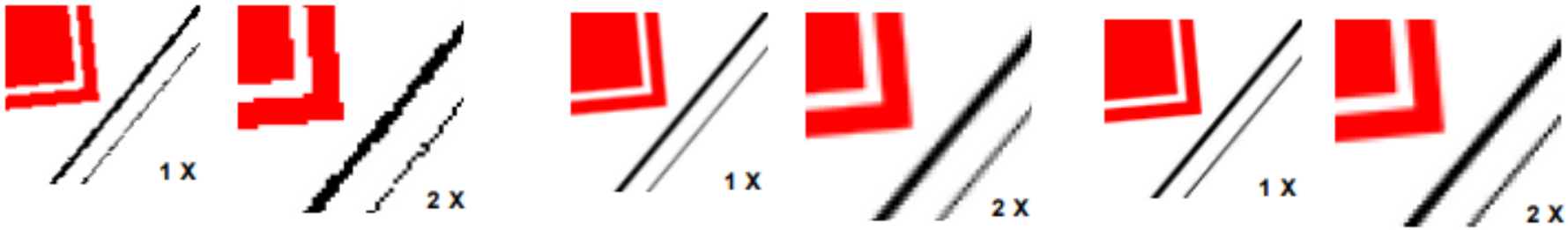
- Change the resolution (number of pixels) of an image
- **Downsampling:** Decreasing the resolution (e.g. mean/max/min filter)
- **Upsampling:** Increasing the resolution (interpolation)
- A value for each cell in the new raster object must be computed by sampling or interpolating over some neighborhood of cells in the corresponding position in the original raster object.

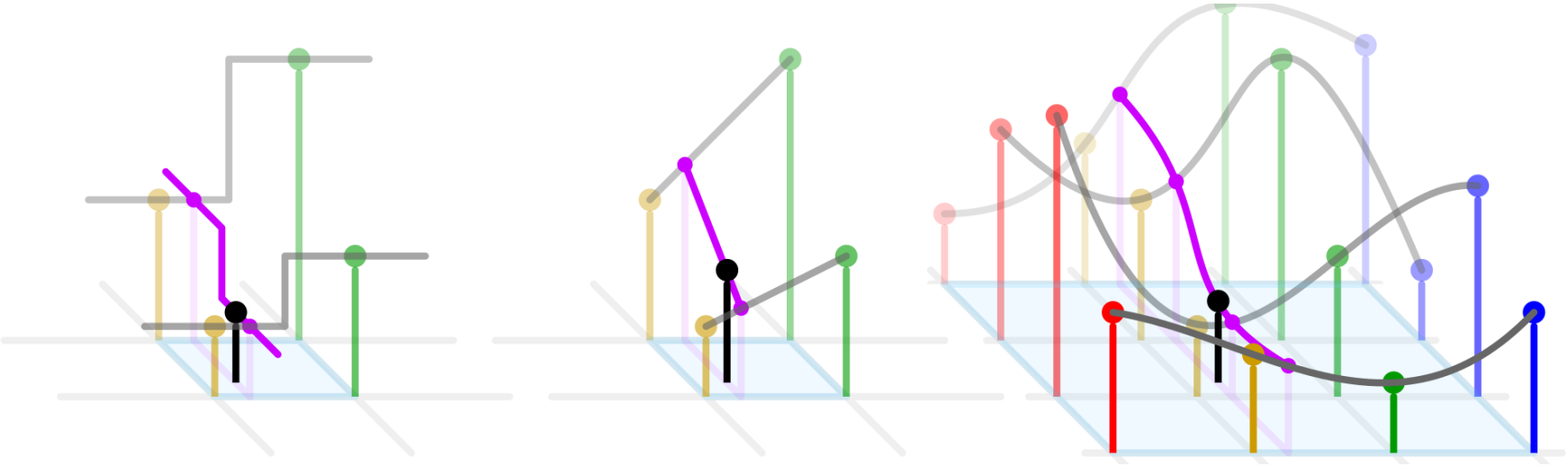


https://desktop.arcgis.com/de/arcmap/latest/extensions/spatial-analyst/performing-analysis/cell-size-and-resampling-in-analysis.htm

Image Processing: Upsampling

Nearest Neighbor **Bilinear** **Bicubic**





http://www.dl-c.com/Temp/downloads/Whitepapers/Resampling.pdf
https://en.wikipedia.org/wiki/Bicubic_interpolation

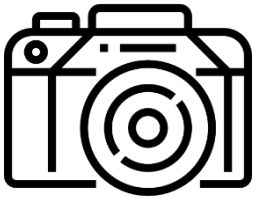
Perception

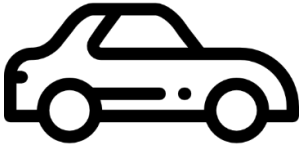
Prof. Dr.-Ing. Markus Lienkamp

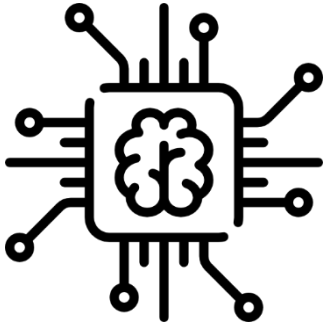
Sebastian Huber

Agenda

1. Introduction
2. Computer Vision
3. Image Processing
4. **Feature Extraction**
5. Image Classification
6. Video Analysis
7. Summary







Feature Extraction

Which features could be useful in the identification of lane lines on the road?

Feature Extraction

- A **Feature** is a piece of information which is relevant for solving the computational task related to a certain application

- **Feature Detection** includes methods for computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point.
  - The resulting features will be subsets of the image domain, often in the form of isolated points, continuous curves or connected regions.
  - Examples: Edges, Corners, Lines, Circles,…

- **Feature Extraction:** Once features have been detected, a local image patch around the feature can be extracted. For Computer Vision, this means the isolation of various desired portions or shapes (features) of a digitized image or video stream.

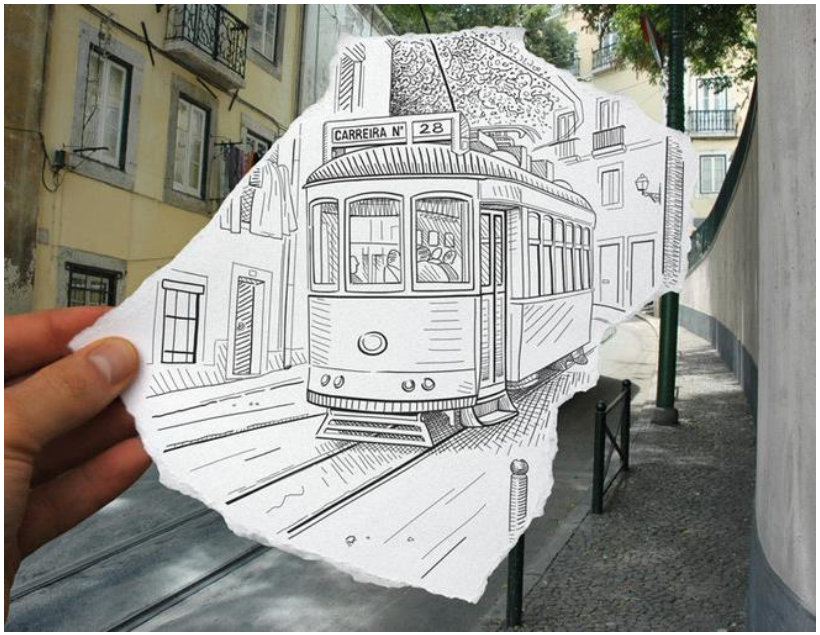
Feature Extraction

Feature points are used for:

- Image alignment
- 3D reconstruction
- Motion tracking
- Robot navigation
- **Object detection**

Feature Extraction: Edge Detection

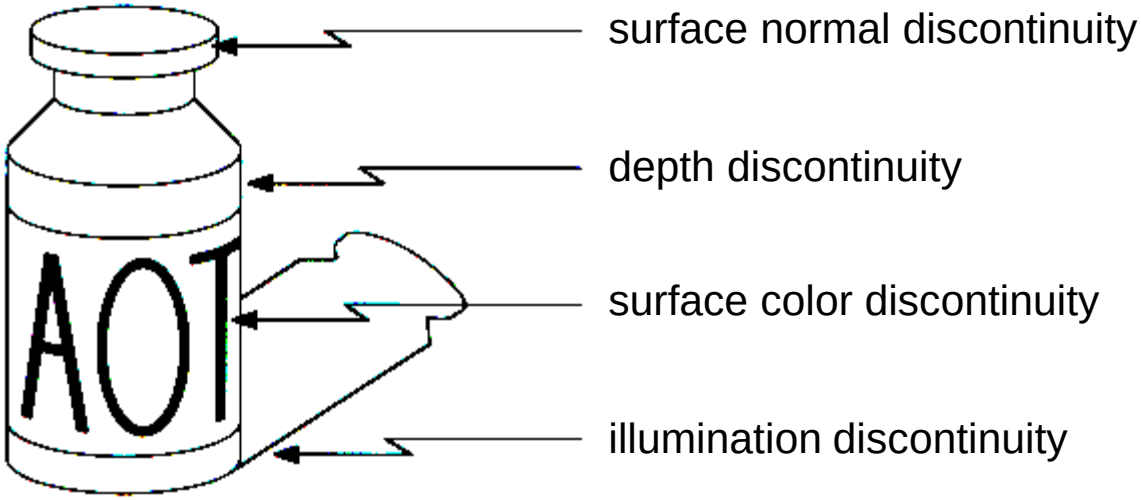
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** Artist's line drawing (but artist is also using object-level knowledge)



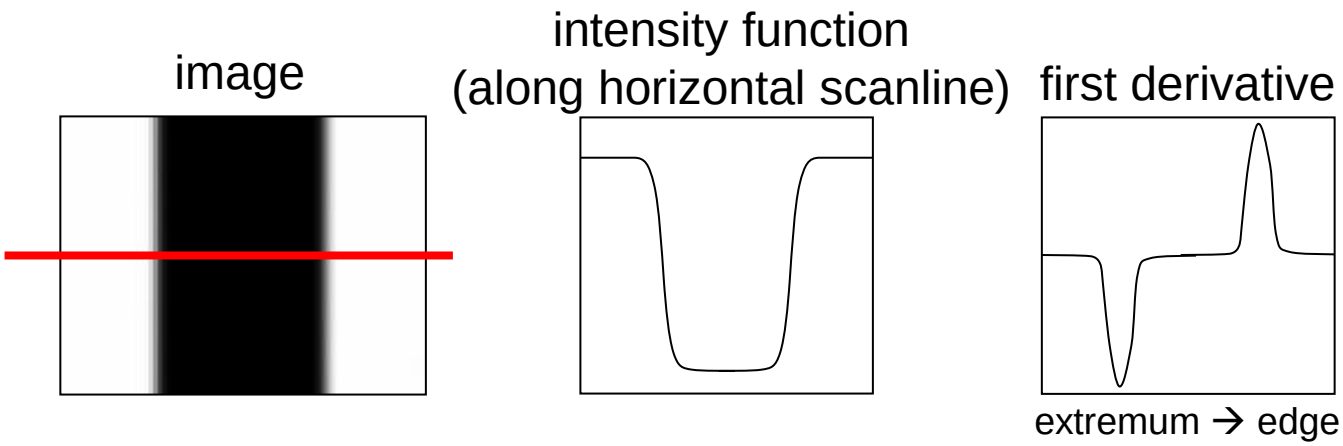
https://curiator.com/art/ben-heine/1

Feature Extraction: Edge Detection

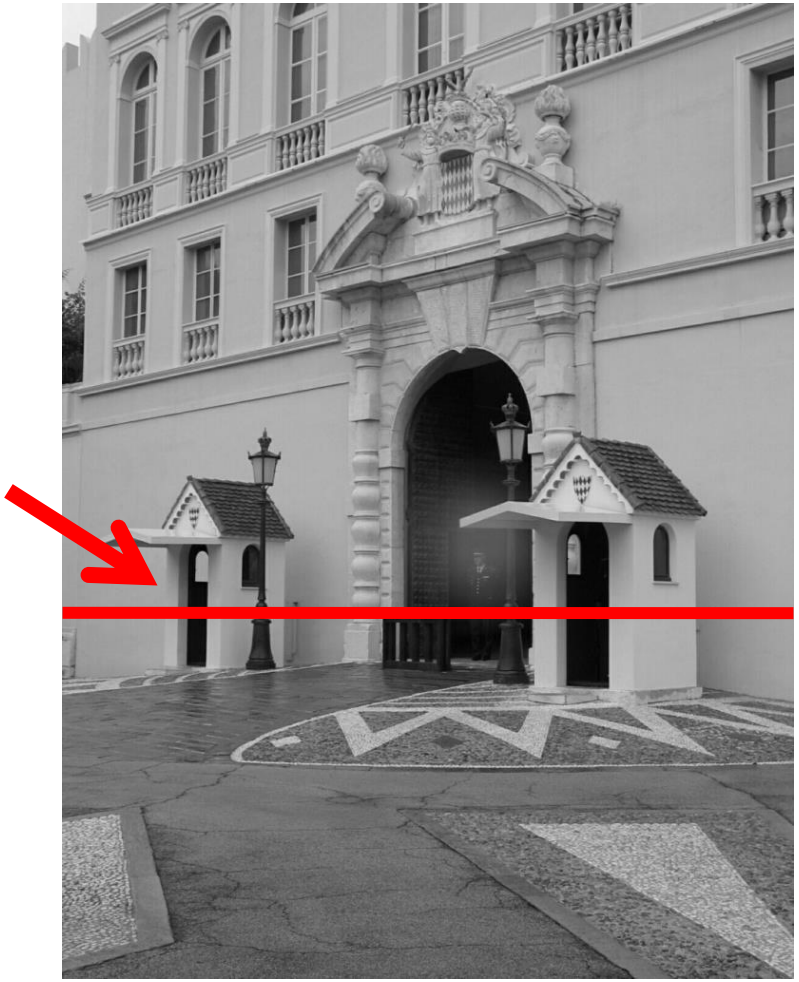
Edges are caused by a variety of factors

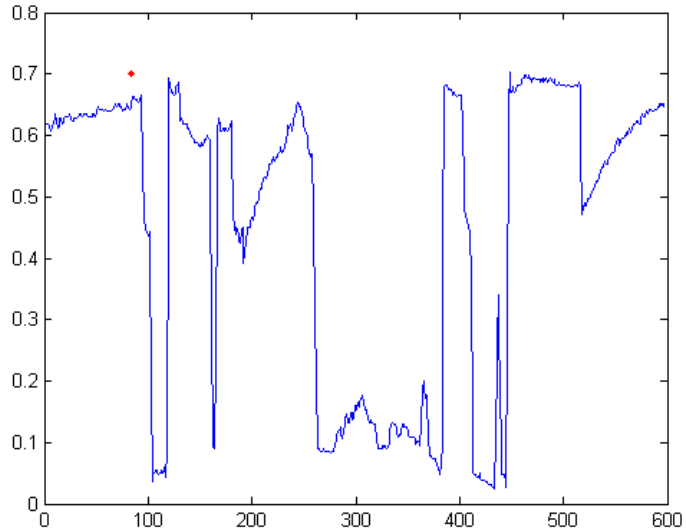


An edge is a place of rapid change in the image **intensity function**

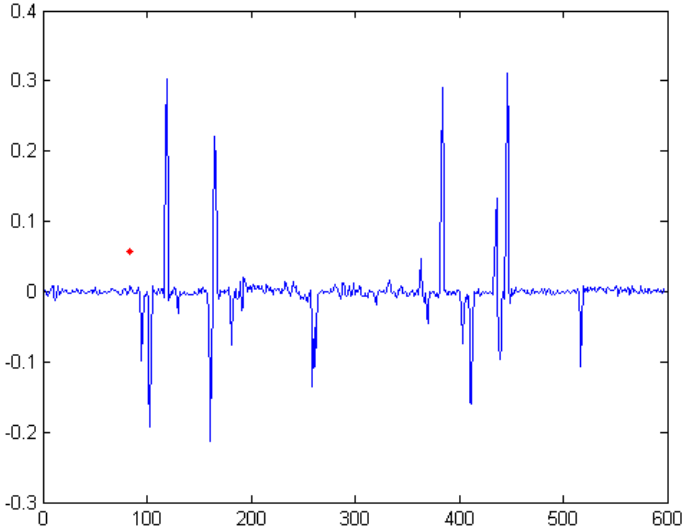


Feature Extraction: Edge Detection





intensity function



first derivative

Feature Extraction: Canny Edge Detection

Probably the most widely used edge detector in Computer Vision

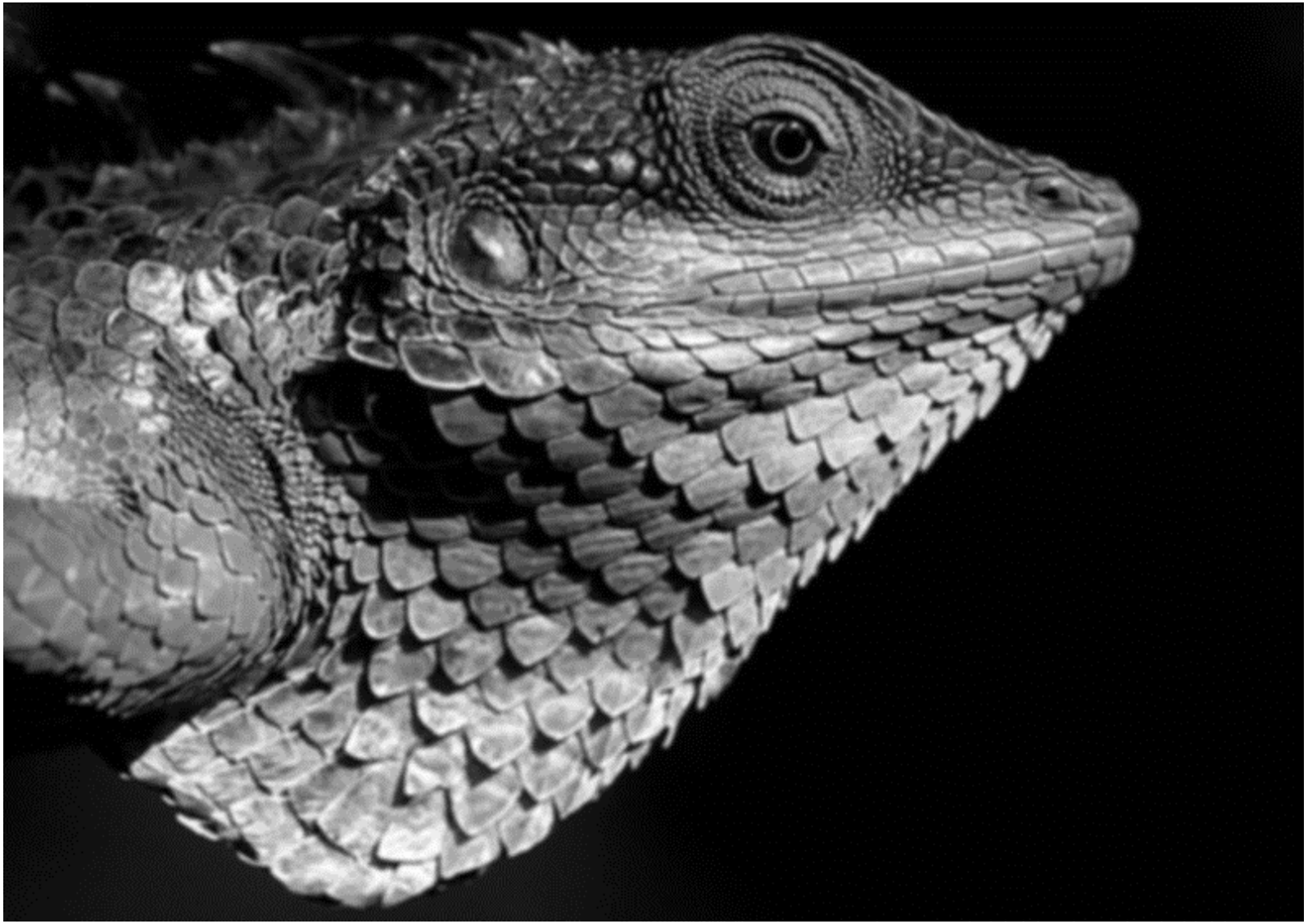
1. Gaussian filter is used to smooth images and remove noise (larger kernel size decreases sensitivity to noise, but increases localization error)
2. Compute intensity gradients for each pixel
3. Non-maximum suppression: Thin multi-pixel wide “ridges” down to single pixel width
4. Apply lower and upper threshold for edges (hysteresis):
   - Intensities above the upper threshold are used to start an edge curve
   - If the neighboring pixels are above the lower threshold, the edge is continued
   - Intensities below the lower thresholds are discarded as noise

Canny Edge Detection Example

Original Image

https://en.wikipedia.org/wiki/Canny_edge_detector

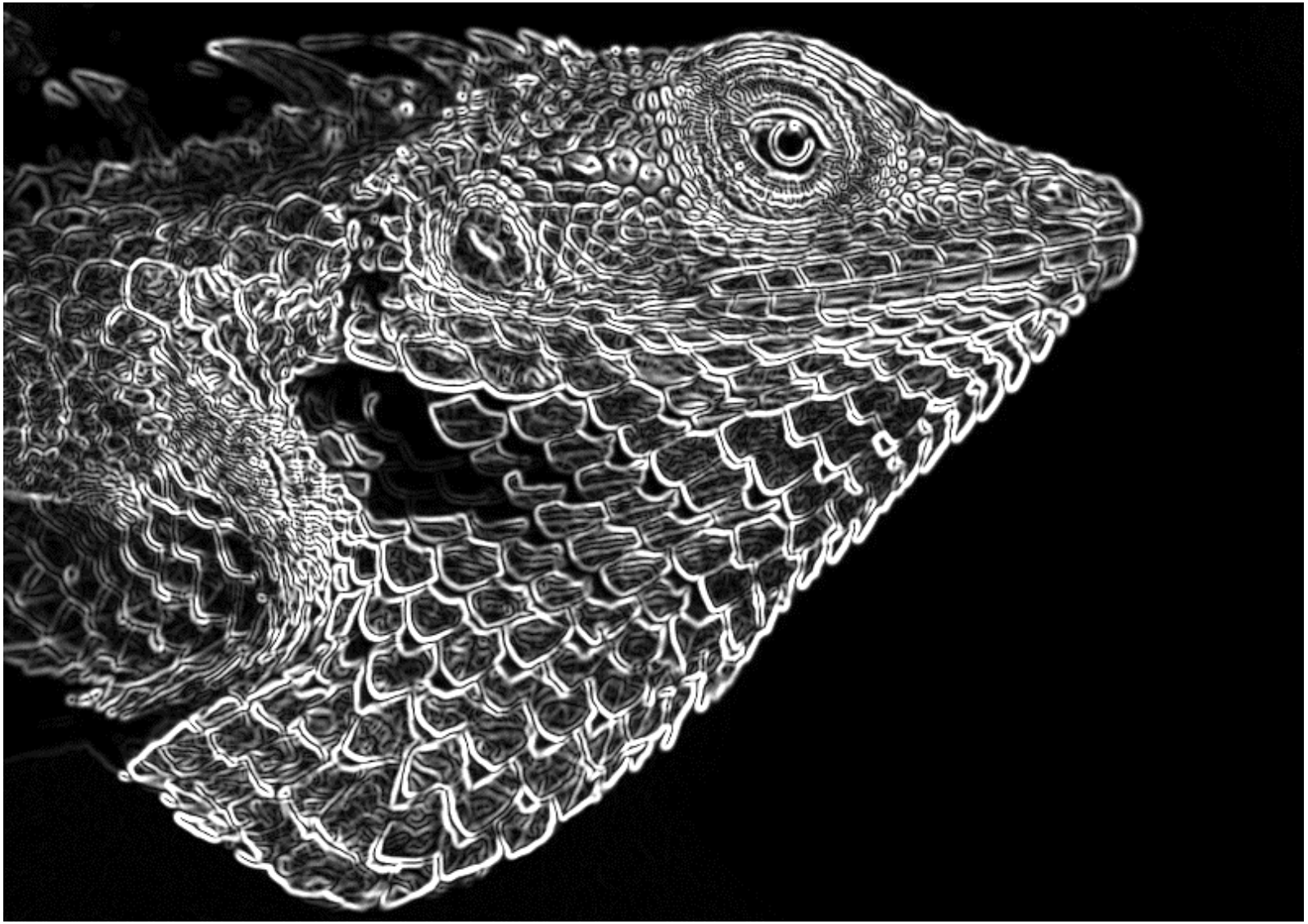
Canny Edge Detection Example



Step 1: Grayscale and 5x5 Gaussian filter

https://en.wikipedia.org/wiki/Canny_edge_detector

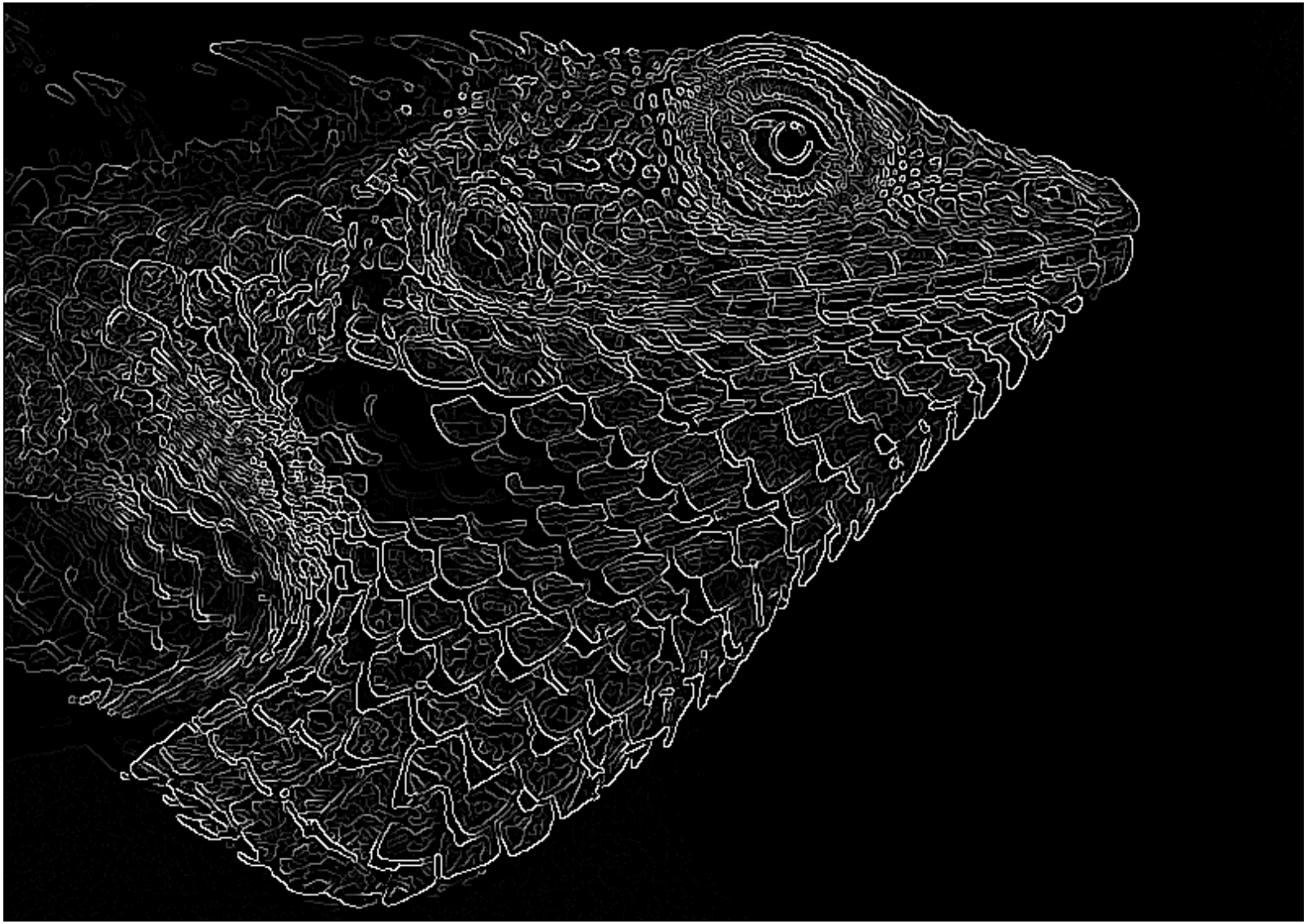
Canny Edge Detection Example



Step 2: Intensity Gradients

https://en.wikipedia.org/wiki/Canny_edge_detector

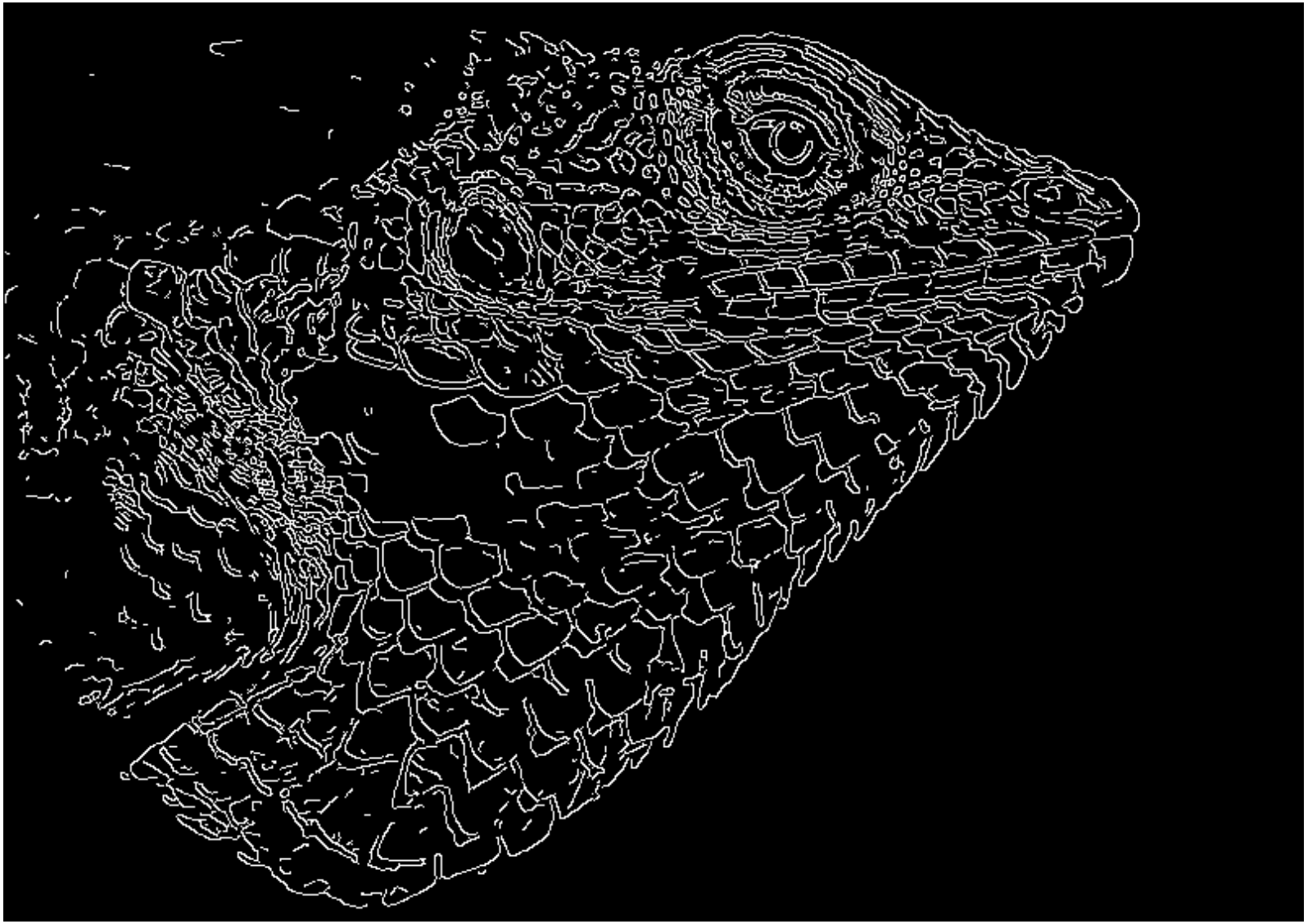
Canny Edge Detection Example



Step 3: Non-maximum suppression

https://en.wikipedia.org/wiki/Canny_edge_detector

Canny Edge Detection Example



Step 4: Thresholds and Hysteresis

https://en.wikipedia.org/wiki/Canny_edge_detector

Demo

04_edge_detection.py

The Problem

- Edge detection makes it possible to considerably reduce the amount of data in an image. However, the output from an edge detector is still an image **described by its pixels.**
- If lines, ellipses and so forth could be defined by their characteristic equations, the amount of data would be reduced even more.
- We can use the **Hough Transform**, which was originally developed to recognize lines and has later been generalized to cover arbitrary shapes, e. g. circles

Feature Extraction: Hough Transform

- Each line in two-dimensional space can be written in the form of $y = ax + b$, where a is the slope and b is the intercept.
- Hough Space is also two-dimensional and features a and b as its axes → A line L in Image Space corresponds to a single point π in Hough Space

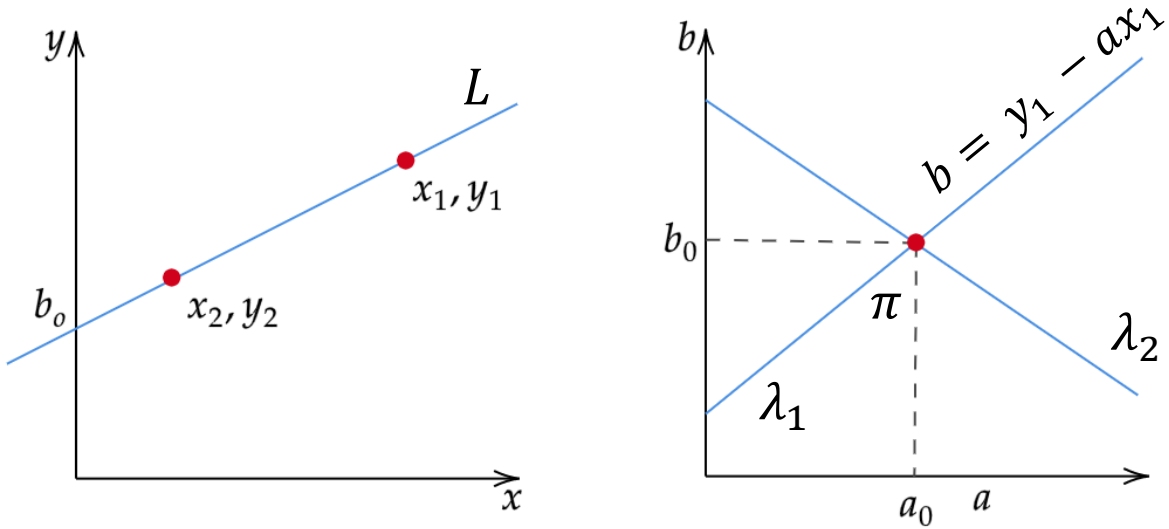


Image Space Hough Space

https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549

Feature Extraction: Hough Transform

- However, a single point $p_i = (x_i, y_i)$ in Image Space can be part of infinite lines and therefore creates infinite points in Hough Space. These points satisfy the condition $b = y_i - ax_i$, i.e. they form a line in Hough Space.
- Consequently, each point p_i of a line L in Image Space creates a separate line λ_i in Hough Space.

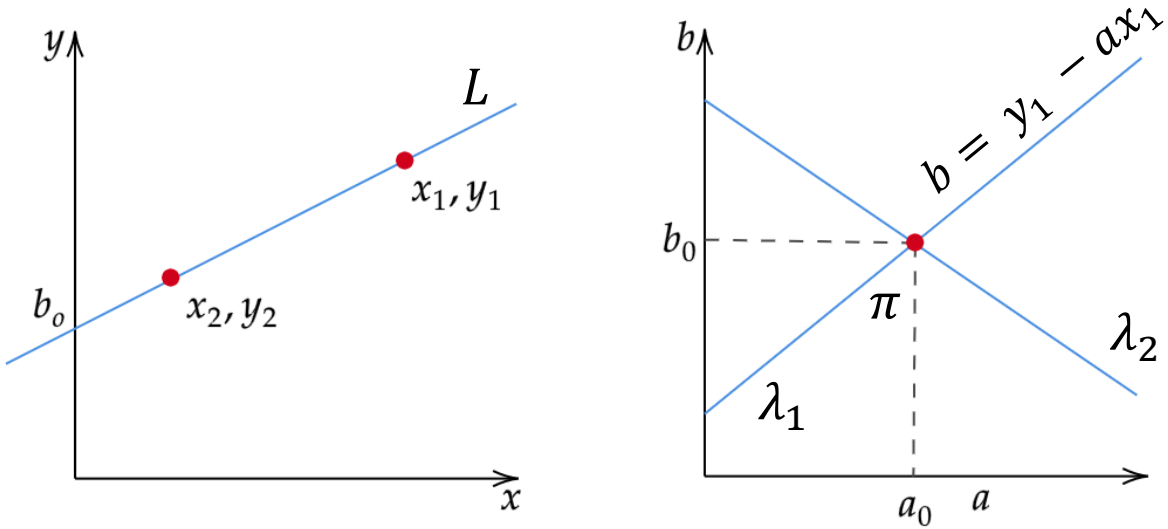


Image Space Hough Space

https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549

Feature Extraction: Hough Transform

- To comply with the principle that L corresponds to π, all λ_i must intersect in π.
- Thus, a point with many intersecting lines in Hough Space corresponds to a single line in Real Space.

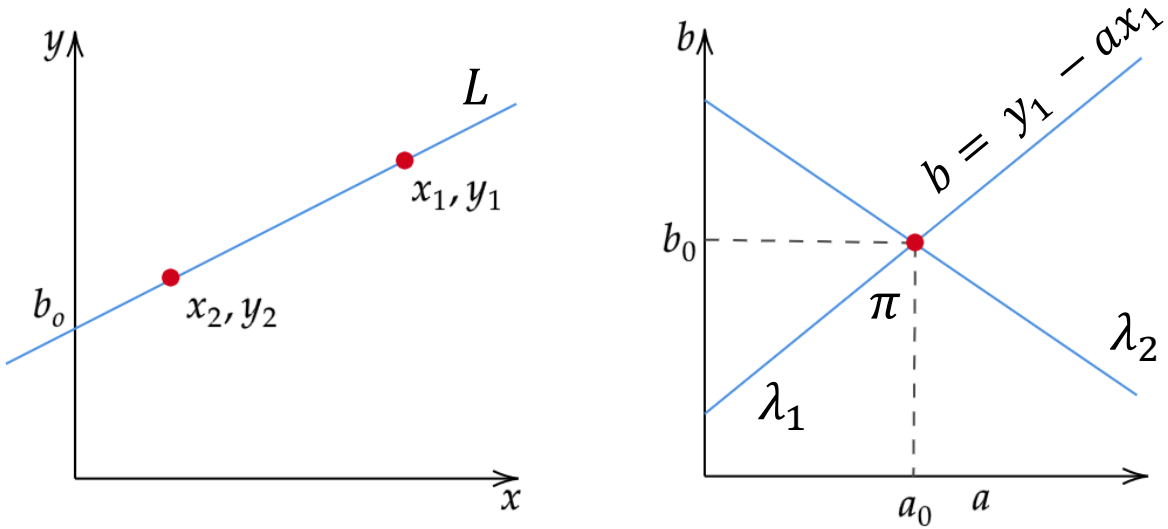


Image Space

Hough Space

https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549

Feature Extraction: Hough Transform

Original

Edge Detection

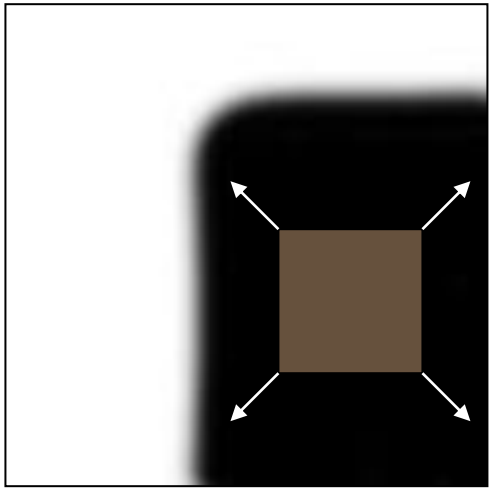
Hough Transform

Demo

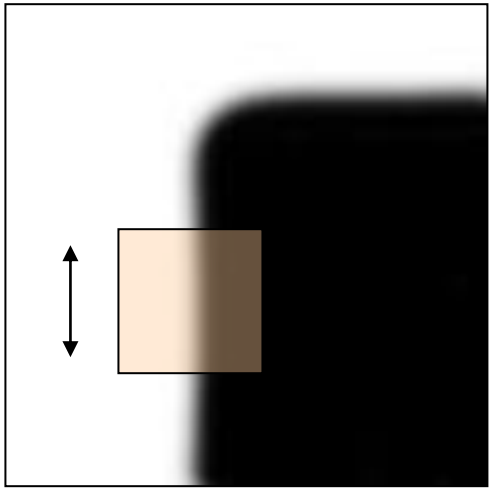
05_hough_transform.py

Feature Extraction: Corner Detection

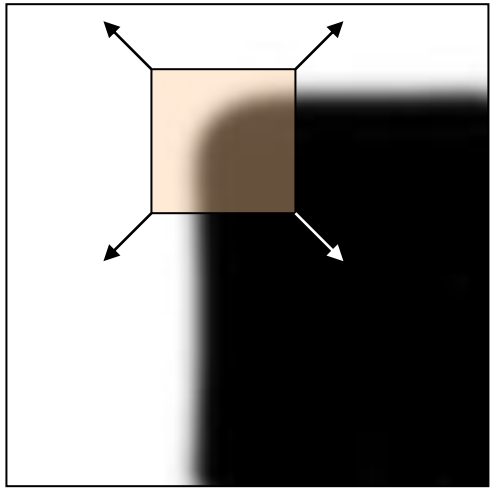
Shifting a window in **any direction** should result in **large changes**



"flat" region:
no change in all directions



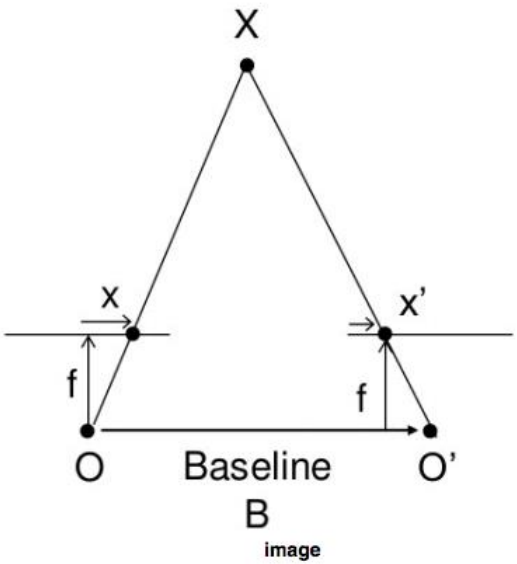
"edge":
no change along the edge direction

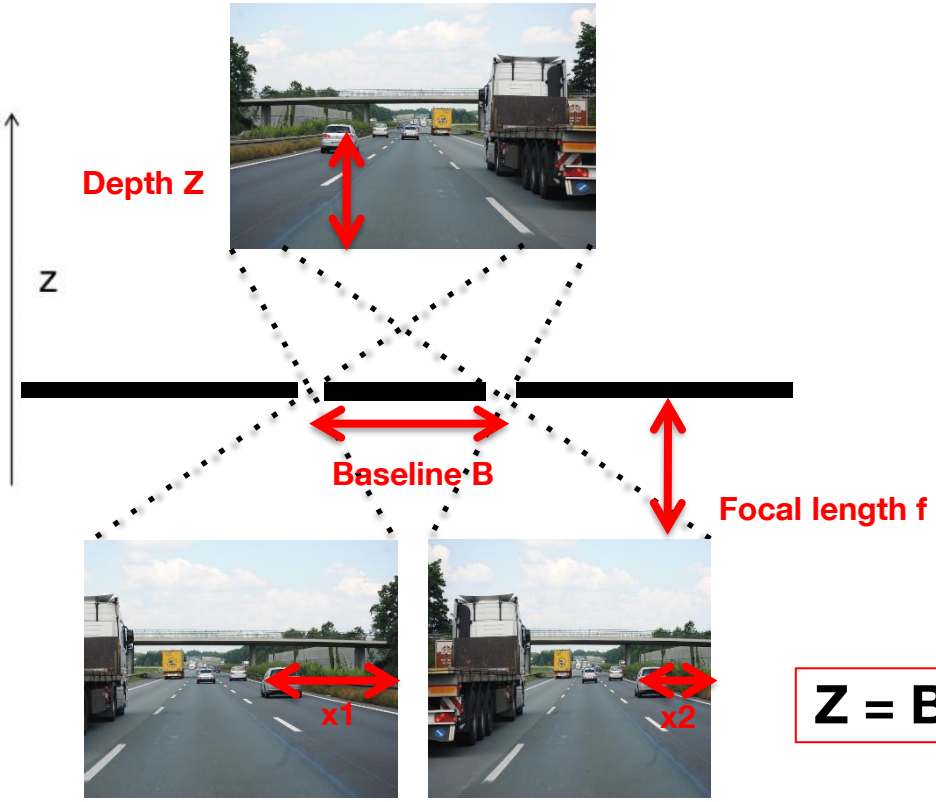


"corner":
significant change in all directions

Feature Extraction: Depth Detection

Disparity between 2 images → disparity map

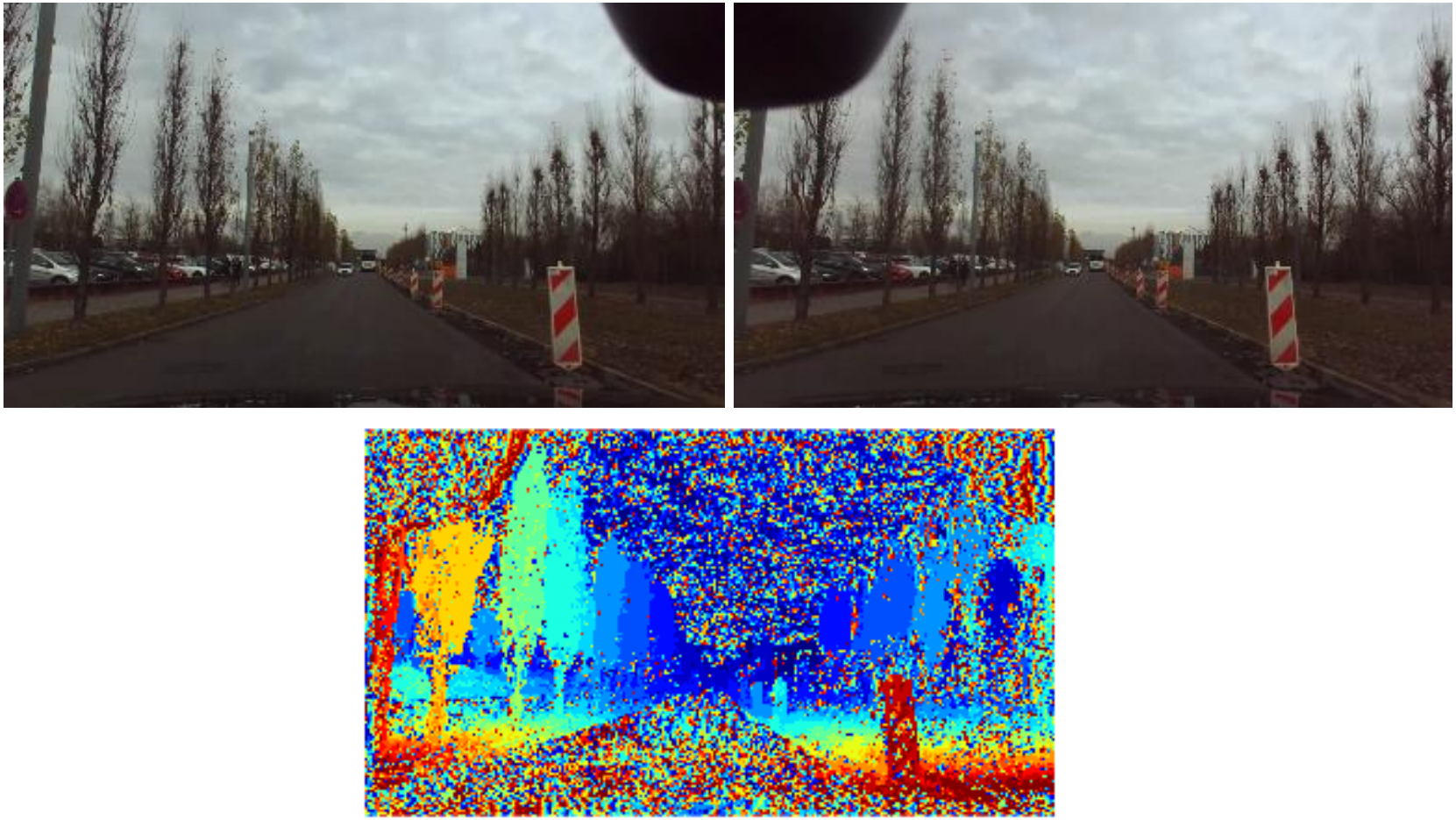




$$Z = B*f/d$$

Disparity d = x1-x2

Feature Extraction: Depth Detection



Feature Extraction – Dilation, Erosion, Opening, Closing

- Morphology-based operations to connect points and edges and build regions/areas
- A set of operations that process images based on shapes. Morphological operations apply a **structuring element** to an input image and generate an output image.
- Dilation and Erosion are **minkowski addition and subtraction**
- Opening and Closing are **combined** Dilation and Erosion

Feature Extraction – Dilation, Erosion, Opening, Closing

Dilation:

- The operations consists of convolving an image A with some kernel B, which can have any shape or size, usually a square or circle.
- Kernel B has a defined anchor point, usually being the center of the kernel.
- As kernel B is scanned over the image, we compute the maximal pixel value overlapped by B and replace the image pixel in the anchor point position with that maximal value. As you can deduce, this maximizing operation causes bright regions within an image to "grow" (therefore the name dilation). Take as an example the image below. Applying dilation we can get:

Feature Extraction – Dilation, Erosion, Opening, Closing

Erosion:

- This operation is the sister of dilation. What this does is to compute a local minimum over the area of the kernel.
- As kernel B is scanned over the image, we compute the minimal pixel value overlapped by B and replace the image pixel under the anchor point with that minimal value.
- Analogously to the example for dilation, we can apply the erosion operator to the original image. You can see in the result below that the bright areas of the image get thinner, whereas the dark zones (the background) gets bigger.

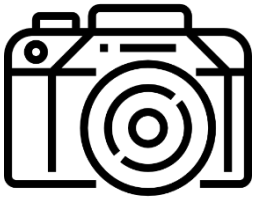
Perception

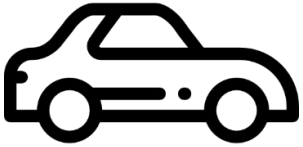
Prof. Dr.-Ing. Markus Lienkamp

Sebastian Huber

Agenda

1. Introduction
2. Computer Vision
3. Image Processing
4. Feature Extraction
5. **Image Classification**
6. Video Analysis
7. Summary





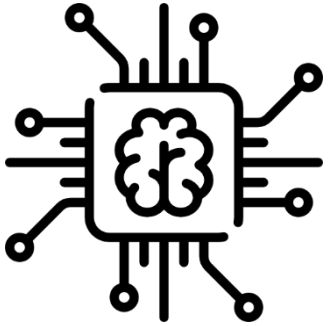
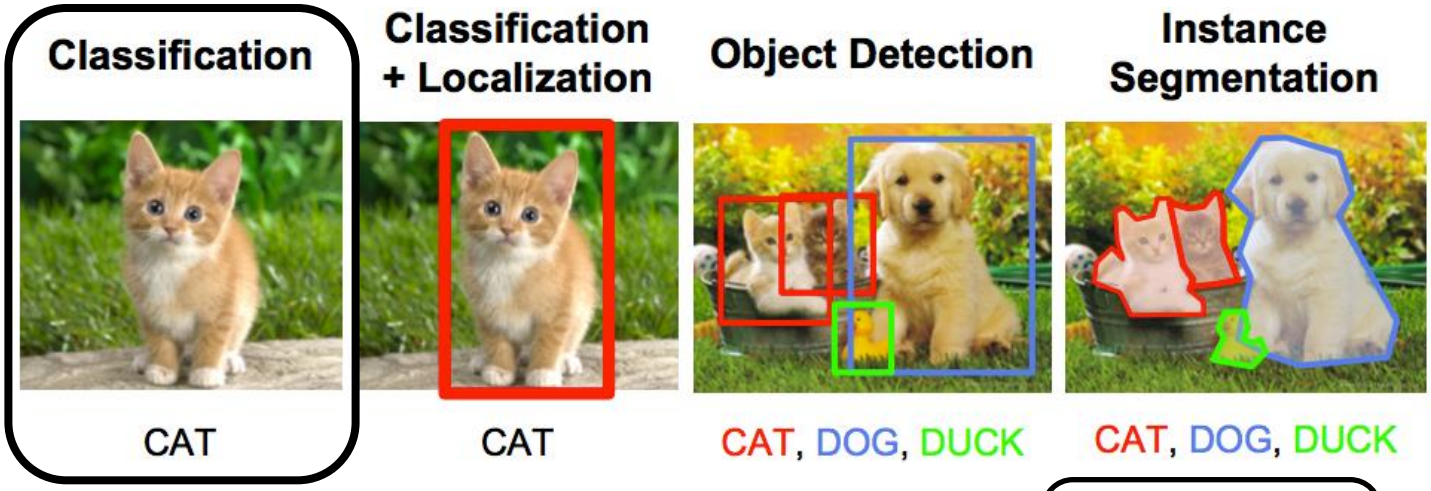
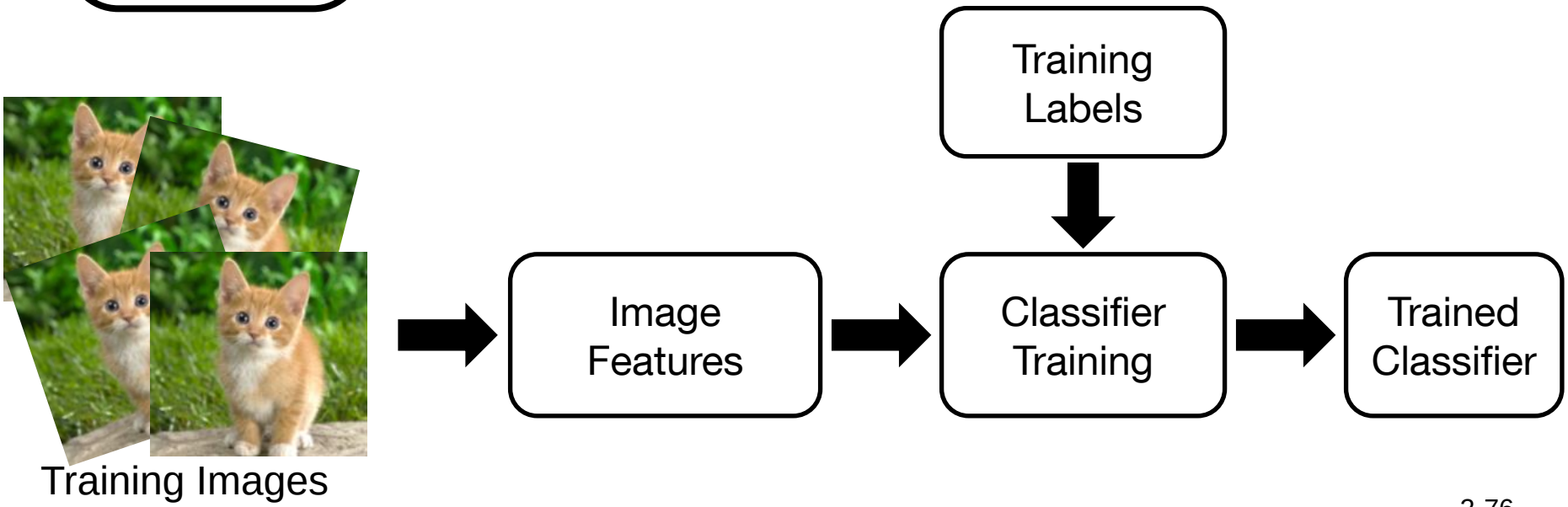


Image Classification



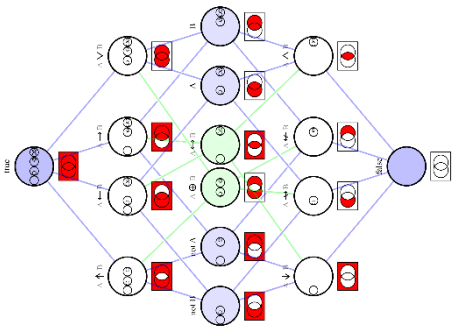


https://www.datacamp.com/community/tutorials/object-detection-guide

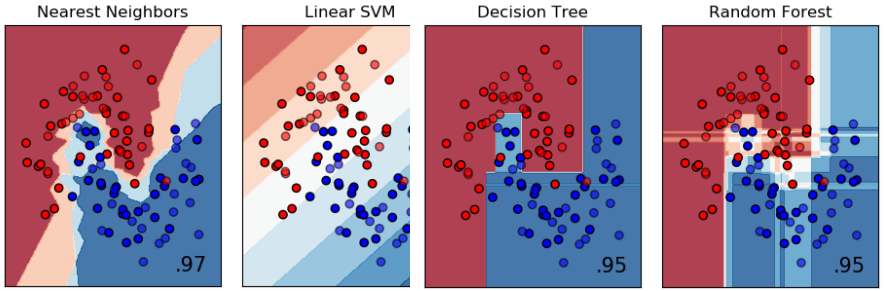
Image Classification

Methods:

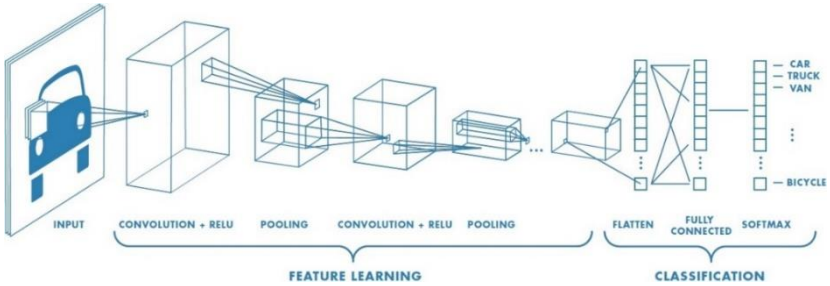
- **Rule based**
  manual feature extraction &
  manual classification logic



- **Machine Learning**
  manual feature extraction,
  classifier learned automatically



- **Deep Learning**
  relevant features & classifier
  learned jointly



https://de.mathworks.com/solutions/deep-learning/convolutional-neural-network.html?requestedDomain
https://commons.wikimedia.org/wiki/Logic_diagram

Image Classification: Manual Features

Person

Bag of features: Gradients, Edges, Pixels

Image Classification: Machine Learning Example

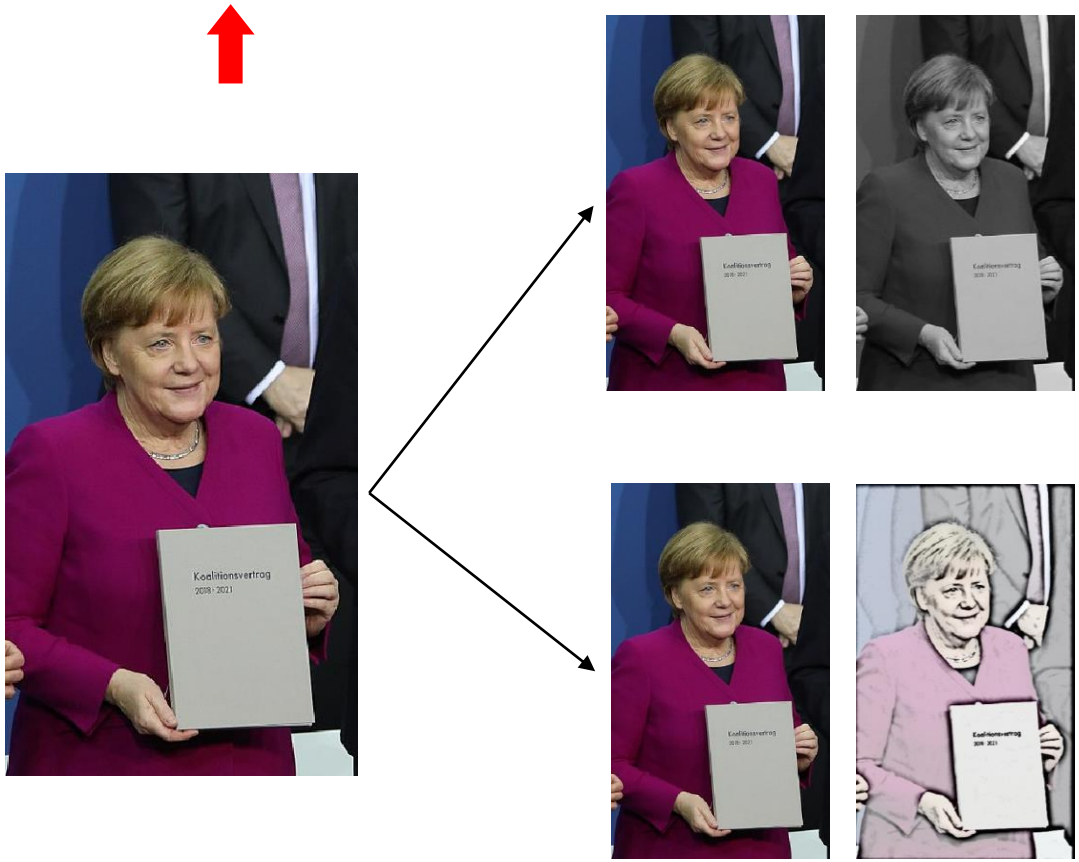
Input Image → Colour & Contrast → Compute Gradients → Compute the histogram of oriented gradients → Normalize contrast over overlapping spatial blocks → Collect HOGs over detection window → Apply Machine Learning → Person / non-person classification

Adjust image resolution and/or crop image

N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

Image Classification: Machine Learning Example

Input Image → Colour & Contrast → Compute Gradients → Compute the histogram of oriented gradients → Normalize contrast over overlapping spatial blocks → Collect HOGs over detection window → Apply Machine Learning → Person / non-person classification



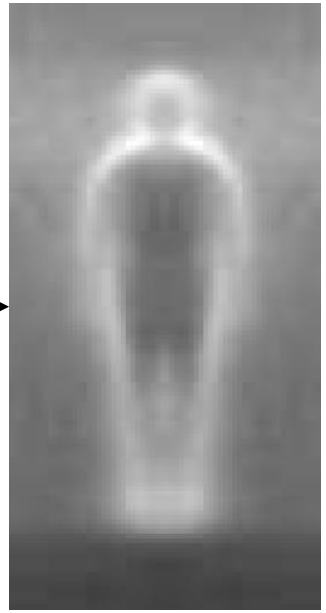
Change color space
→ e.g., HSV offers more features due to different color separation

Contrast Enhancement
→ Offers more features because of better edge detection

N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

Image Classification: Machine Learning Example

Input Image → Colour & Contrast → Compute Gradients → Compute the histogram of oriented gradients → Normalize contrast over overlapping spatial blocks → Collect HOGs over detection window → Apply Machine Learning → Person / non-person classification



Compute Gradients by selecting the best kernel:

0	1
-1	0

diagonal

-1	0	1

centered

-1	1

uncentered

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

Sobel

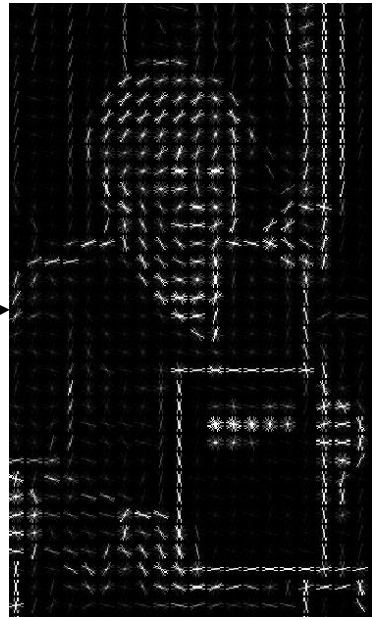
1	-8	0	8	-1

cubic-corrected

N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

Image Classification: Machine Learning Example

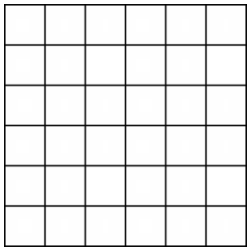
Input Image → Colour & Contrast → Compute Gradients → Compute the histogram of oriented gradients → Normalize contrast over overlapping spatial blocks → Collect HOGs over detection window → Apply Machine Learning → Person / non-person classification



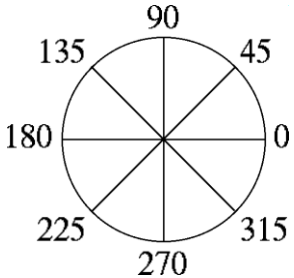
Compute the Histogram of Oriented Gradients

- Votes weighted by magnitude
- Bilinear interpolation between cells

Histograms in k*k pixel cells



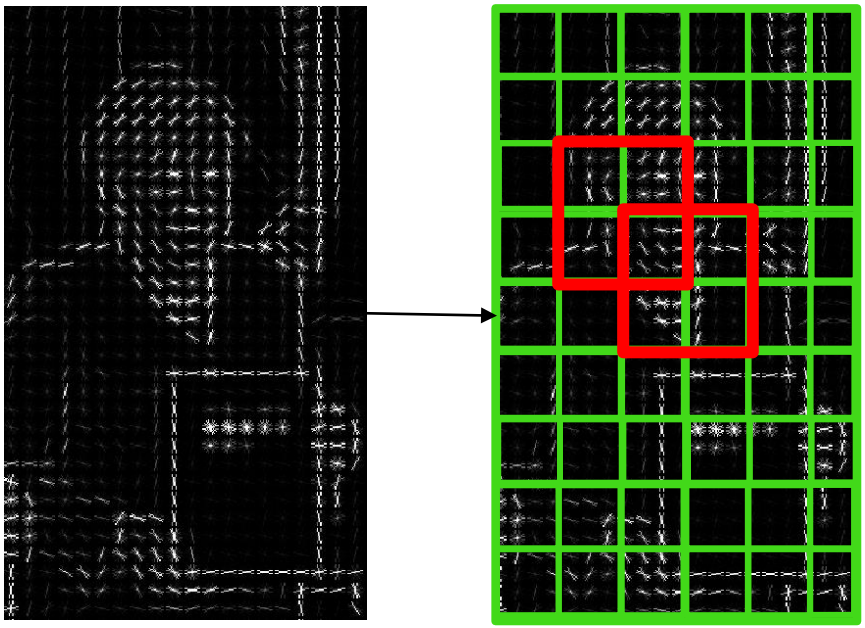
Orientation: 9 bins (for unsigned angles 0 -180)



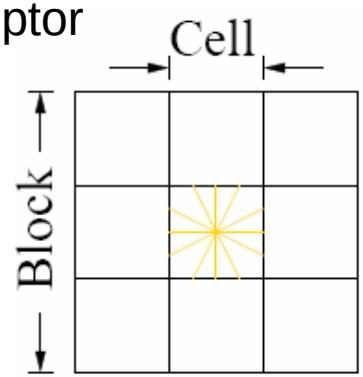
N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

Image Classification: Machine Learning Example

Input Image → Colour & Contrast → Compute Gradients → Compute the histogram of oriented gradients → Normalize contrast over overlapping spatial blocks → Collect HOGs over detection window → Apply Machine Learning → Person / non-person classification



- Normalization of the blocks by super-imposition
- Normalization of cells is independent of blocks
- HOGs for overlapping 2x2 blocks of cells are normalized and collected to form the final descriptor

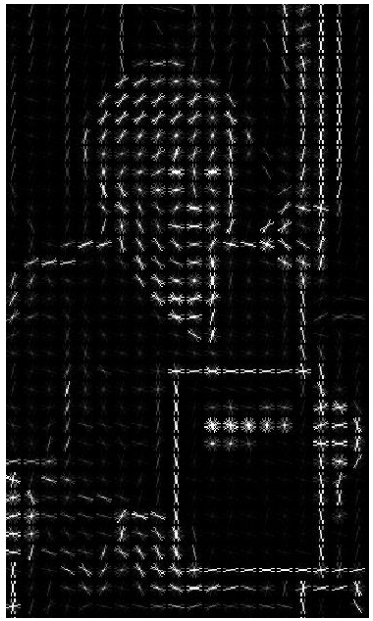


N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

Image Classification: Machine Learning Example

Input Image → Colour & Contrast → Compute Gradients → Compute the histogram of oriented gradients → Normalize contrast over overlapping spatial blocks → Collect HOGs over detection window → Apply Machine Learning → Person / non-person classification

X=



- HOGs of all blocks are written to feature output vector
- Vector is used in window classifier
- Feature vector f = [..., ..., ...]

features = 15 x 7 x 9 x 4 = 3780

orientations

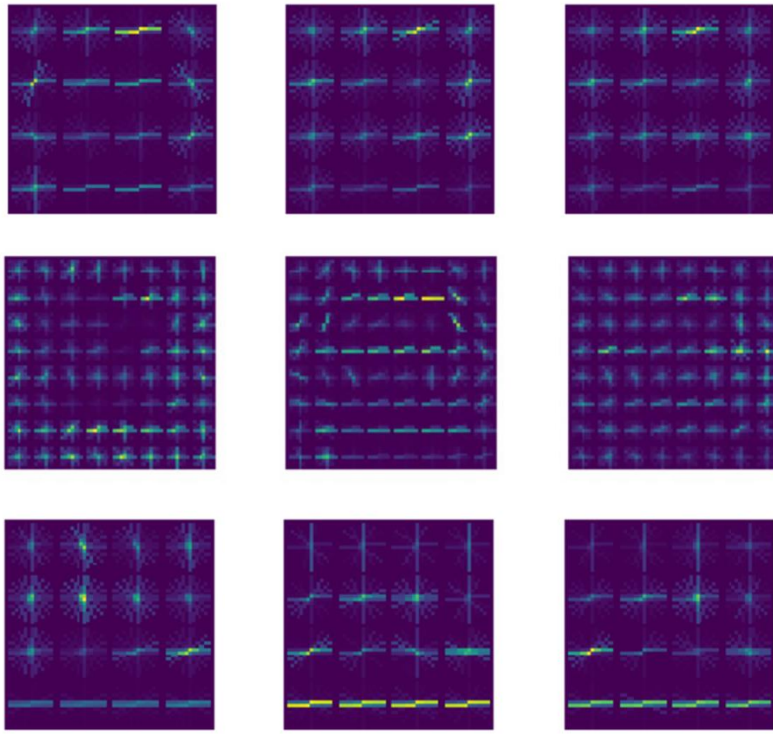
cells

normalizations by neighboring cells

N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

Image Classification: Machine Learning Example

Input Image → Colour & Contrast → Compute Gradients → Compute the histogram of oriented gradients → Normalize contrast over overlapping spatial blocks → Collect HOGs over detection window → Apply Machine Learning → Person / non-person classification



HOG features

Machine Learning Algorithm for Classification

N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

Image Classification: Machine Learning Example

Input Image → Colour & Contrast → Compute Gradients → Compute the histogram of oriented gradients → Normalize contrast over overlapping spatial blocks → Collect HOGs over detection window → Apply Machine Learning → Person / non-person classification

Pedestrian: 95%

Classifiers:

- Support Vector Machine (SVM)
- Decision Trees / Random Forests
- k-Nearest Neighbors

Other methods for feature extraction:

- Scale-invariant feature transform (SIFT)
- Speeded up robust features (SURF)

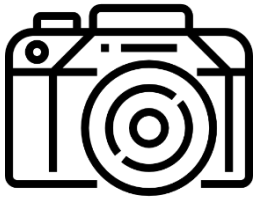
N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

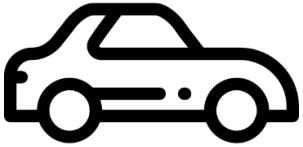
Perception
Prof. Dr.-Ing. Markus Lienkamp

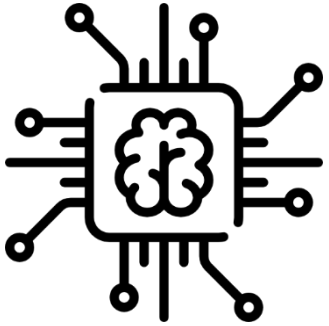
Sebastian Huber

Agenda

1. Introduction
2. Computer Vision
3. Image Processing
4. Feature Extraction
5. Image Classification
6. **Video Analysis**
7. Summary

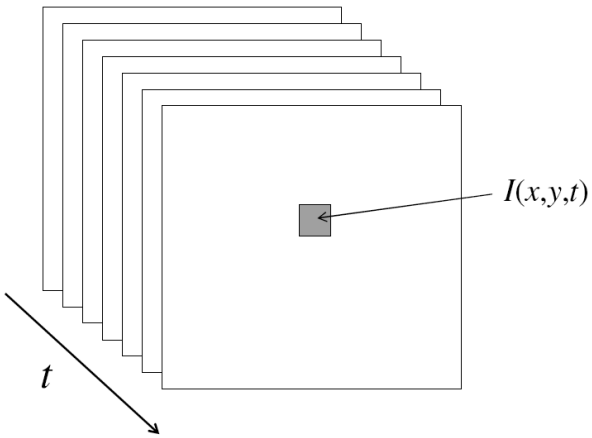




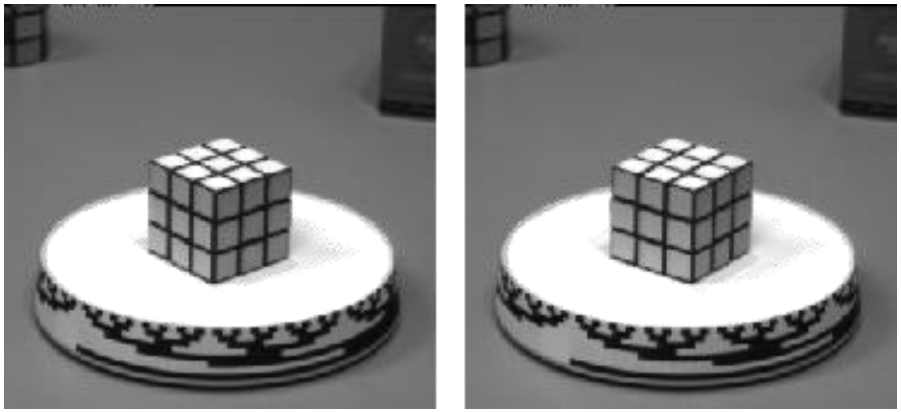


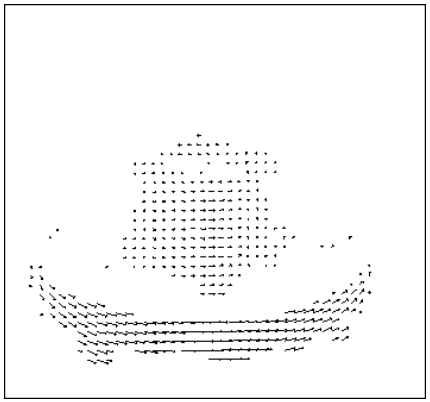
Video Analysis: Optical Flow

- A **video** is a sequence of frames captured over time
- Now our image data is a function of space (x, y) and **time (t)**



- **Optical flow** is the **pattern of apparent motion** of objects, surfaces, and edges in a visual scene caused by the **relative motion between an observer and a scene**





https://web.stanford.edu/class/cs231m/lectures/lecture-7-optical-flow.pdf

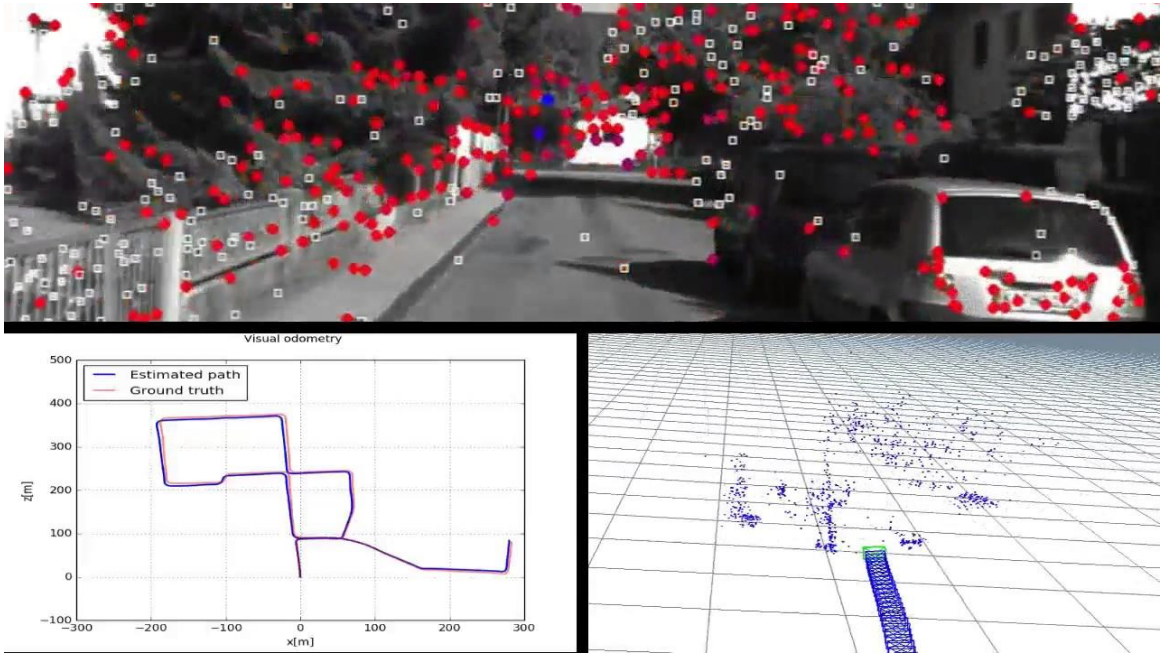
Video Analysis: Optical Flow

https://www.youtube.com/watch?v=LNLPNK0pc2U

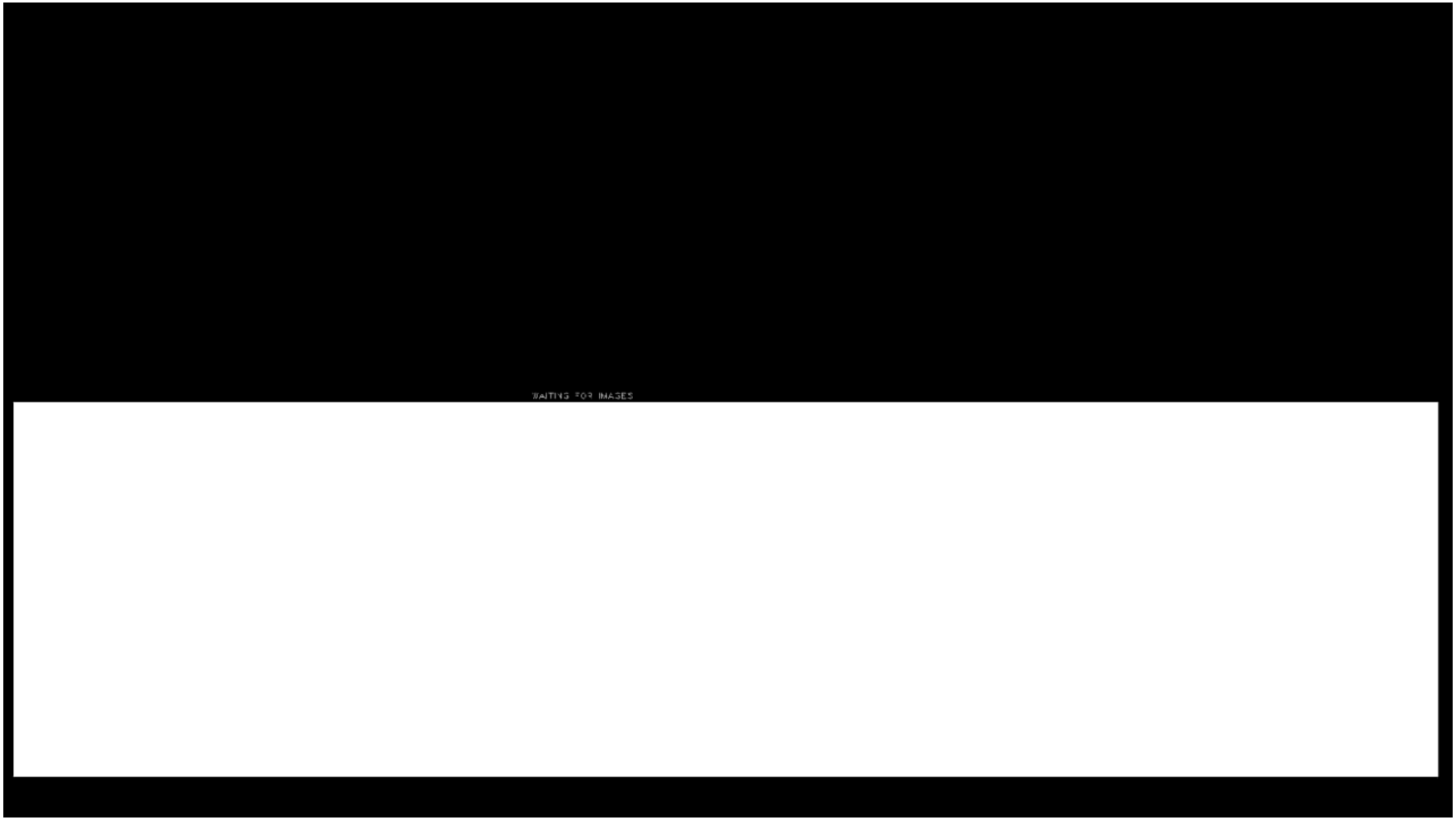
Video Analysis: Egomotion Estimation and SLAM

In robotics and computer vision, **visual odometry** is the process of determining the position and orientation of a robot by analyzing the associated camera images.

SLAM = Simultaneous Localization and Mapping



Information Analysis – Egomotion Estimation and SLAM



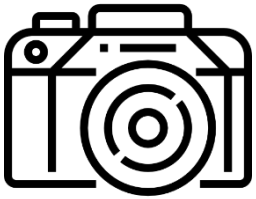
https://www.youtube.com/watch?v=8DISRmsO2YQ

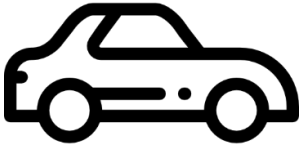
Perception
Prof. Dr.-Ing. Markus Lienkamp

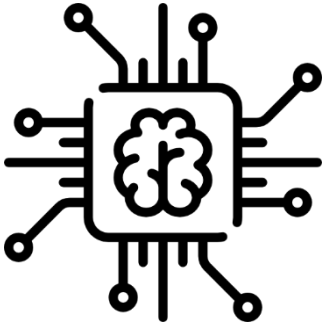
Sebastian Huber

Agenda

1. Introduction
2. Computer Vision
3. Image Processing
4. Feature Extraction
5. Image Classification
6. Video Analysis
7. **Summary**







Summary

- Computer Vision (CV) is a subfield of machine perception

- We focused on the image processing pipeline for camera-based CV
  1. Connect camera to computer and acquire images
  2. Preprocess image: change color space, apply filters, adjust contrast, …
  3. Extract features: edge detection, hough transform, disparity map, …
  4. Classify image, e.g. using Support Vector Machine (SVM)

- Videos provide additional temporal information
  → egomotion estimation, SLAM

- Next up: Supervised Machine Learning (Regression/Classification)