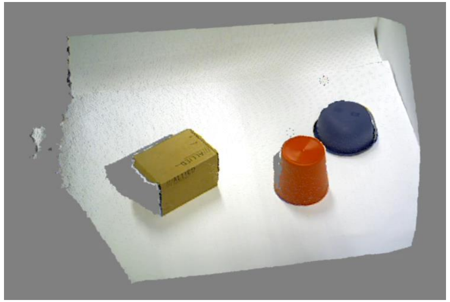
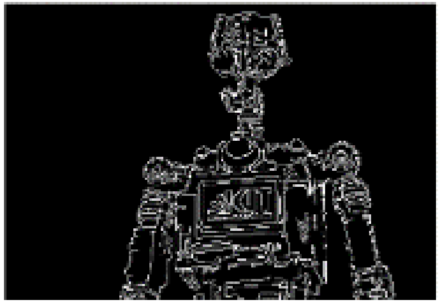
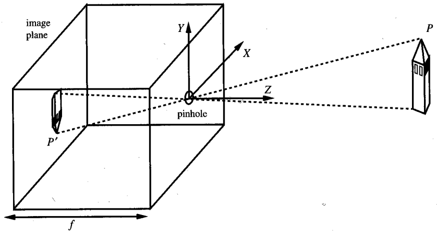
Karlsruhe Institute of Technology

High Performance Humanoid Technologies

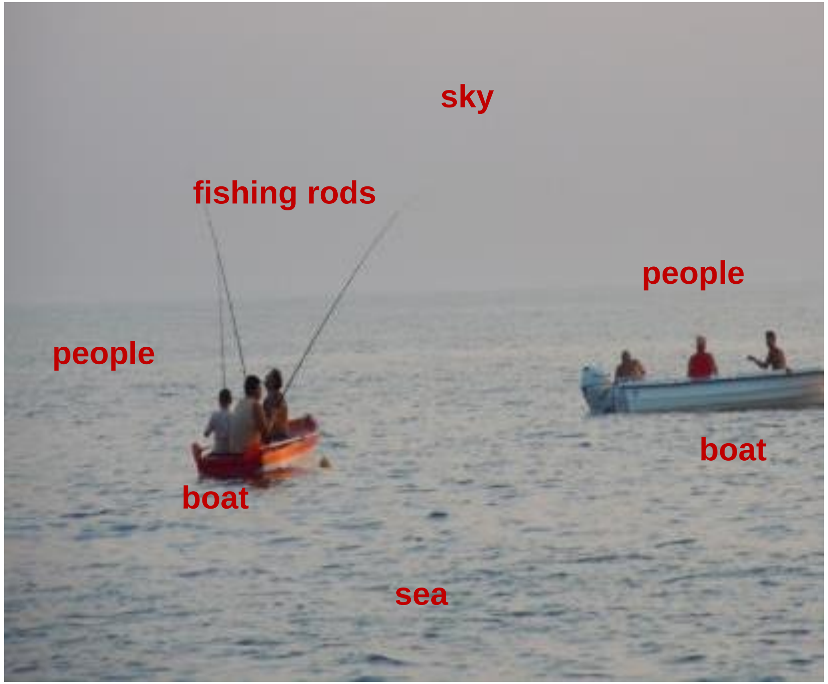
# Robotics I: Introduction to Robotics
## Chapter 9 – Introduction to Robot Vision

Tamim Asfour

http://www.humanoids.kit.edu

# Motivation - Image Understanding




What we see!

$u$ →

$v$ ↓

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 122 | 118 | 120 | 130 | 135 | 134 | 139 | 146 | 140 | 138 | 137 | 139 | 141 | 140 | 136 | 132 | 129 | 142 | 142 | 136 |
| 136 | 134 | 133 | 135 | 135 | 133 | 134 | 137 | 139 | 139 | 139 | 140 | 141 | 139 | 133 | 128 | 127 | 135 | 133 | 128 |
| 129 | 132 | 133 | 133 | 135 | 139 | 140 | 138 | 133 | 134 | 135 | 137 | 137 | 136 | 134 | 132 | 131 | 134 | 130 | 128 |
| 137 | 143 | 143 | 138 | 138 | 142 | 142 | 136 | 138 | 138 | 137 | 136 | 134 | 133 | 132 | 131 | 131 | 131 | 129 | 133 |
| 142 | 141 | 140 | 139 | 138 | 138 | 138 | 138 | 137 | 139 | 139 | 134 | 134 | 137 | 138 | 134 | 138 | 138 | 137 | 137 |
| 142 | 140 | 138 | 136 | 135 | 136 | 137 | 138 | 141 | 143 | 142 | 137 | 136 | 139 | 138 | 135 | 138 | 138 | 138 | 139 |
| 145 | 143 | 141 | 139 | 139 | 139 | 141 | 142 | 142 | 144 | 143 | 140 | 139 | 141 | 141 | 139 | 138 | 138 | 138 | 138 |
| 144 | 144 | 144 | 143 | 143 | 143 | 143 | 143 | 141 | 142 | 141 | 141 | 141 | 142 | 143 | 143 | 142 | 142 | 142 | 141 |
| 139 | 140 | 143 | 144 | 144 | 142 | 140 | 138 | 142 | 141 | 141 | 141 | 141 | 141 | 141 | 142 | 143 | 143 | 143 | 142 |
| 135 | 137 | 140 | 142 | 142 | 140 | 137 | 135 | 141 | 139 | 137 | 138 | 137 | 135 | 135 | 137 | 131 | 132 | 134 | 136 |
| 129 | 130 | 131 | 132 | 132 | 131 | 131 | 130 | 130 | 126 | 126 | 129 | 129 | 127 | 127 | 131 | 123 | 124 | 127 | 130 |
| 119 | 118 | 117 | 116 | 116 | 118 | 119 | 120 | 115 | 112 | 113 | 119 | 121 | 121 | 123 | 128 | 126 | 128 | 130 | 132 |
| 102 | 106 | 110 | 111 | 111 | 113 | 118 | 123 | 118 | 114 | 114 | 122 | 128 | 128 | 129 | 131 | 134 | 134 | 134 | 135 |
| 105 | 109 | 113 | 114 | 113 | 115 | 119 | 124 | 126 | 124 | 125 | 130 | 134 | 134 | 135 | 138 | 137 | 137 | 136 | 136 |
| 117 | 120 | 123 | 124 | 124 | 125 | 128 | 132 | 135 | 135 | 136 | 139 | 139 | 138 | 140 | 143 | 140 | 140 | 139 | 138 |
| 128 | 130 | 133 | 134 | 134 | 134 | 136 | 138 | 138 | 140 | 142 | 142 | 139 | 138 | 139 | 142 | 141 | 140 | 140 | 139 |
| 132 | 133 | 135 | 136 | 136 | 136 | 136 | 137 | 137 | 140 | 141 | 139 | 136 | 135 | 136 | 138 | 137 | 136 | 136 | 135 |
| 134 | 135 | 136 | 137 | 137 | 136 | 135 | 134 | 134 | 136 | 136 | 133 | 131 | 132 | 132 | 131 | 130 | 129 | 128 | 126 |
| 133 | 133 | 133 | 134 | 134 | 133 | 131 | 129 | 127 | 129 | 127 | 124 | 125 | 127 | 126 | 122 | 124 | 122 | 119 | 116 |
| 126 | 126 | 127 | 127 | 128 | 126 | 123 | 121 | 121 | 122 | 120 | 117 | 119 | 123 | 121 | 115 | 120 | 117 | 113 | 109 |
| 120 | 119 | 118 | 116 | 116 | 116 | 117 | 118 | 104 | 108 | 114 | 118 | 118 | 115 | 113 | 111 | 118 | 120 | 121 | 120 |
| 126 | 125 | 124 | 123 | 123 | 124 | 124 | 125 | 125 | 127 | 130 | 131 | 130 | 129 | 128 | 129 | 132 | 133 | 134 | 134 |
| 134 | 133 | 133 | 132 | 133 | 133 | 134 | 134 | 139 | 139 | 139 | 138 | 136 | 136 | 138 | 140 | 139 | 140 | 141 | 141 |
| 139 | 139 | 139 | 139 | 139 | 140 | 140 | 141 | 140 | 140 | 140 | 139 | 138 | 138 | 140 | 142 | 139 | 140 | 141 | 141 |
| 140 | 140 | 141 | 142 | 142 | 142 | 142 | 142 | 139 | 141 | 142 | 143 | 143 | 143 | 144 | 146 | 142 | 142 | 143 | 144 |
| 141 | 141 | 142 | 143 | 143 | 142 | 142 | 142 | 139 | 141 | 144 | 146 | 146 | 146 | 146 | 147 | 144 | 143 | 143 | 144 |
| 142 | 143 | 143 | 144 | 144 | 143 | 142 | 141 | 141 | 143 | 144 | 145 | 144 | 144 | 145 | 146 | 143 | 142 | 141 | 142 |
| 143 | 144 | 145 | 145 | 145 | 144 | 142 | 141 | 146 | 147 | 147 | 145 | 144 | 144 | 146 | 148 | 145 | 143 | 142 | 142 |
| 142 | 144 | 146 | 145 | 143 | 142 | 143 | 145 | 144 | 145 | 146 | 146 | 145 | 145 | 146 | 147 | 146 | 146 | 146 | 145 |
| 141 | 142 | 144 | 143 | 141 | 140 | 141 | 142 | 143 | 144 | 145 | 145 | 144 | 144 | 145 | 146 | 146 | 146 | 146 | 145 |

What a robot sees!

# What is Computer Vision?

Semantics

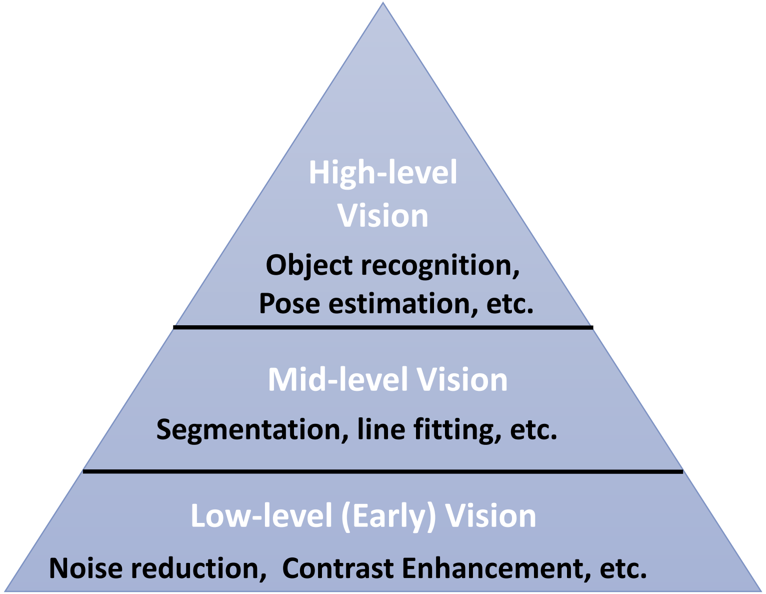
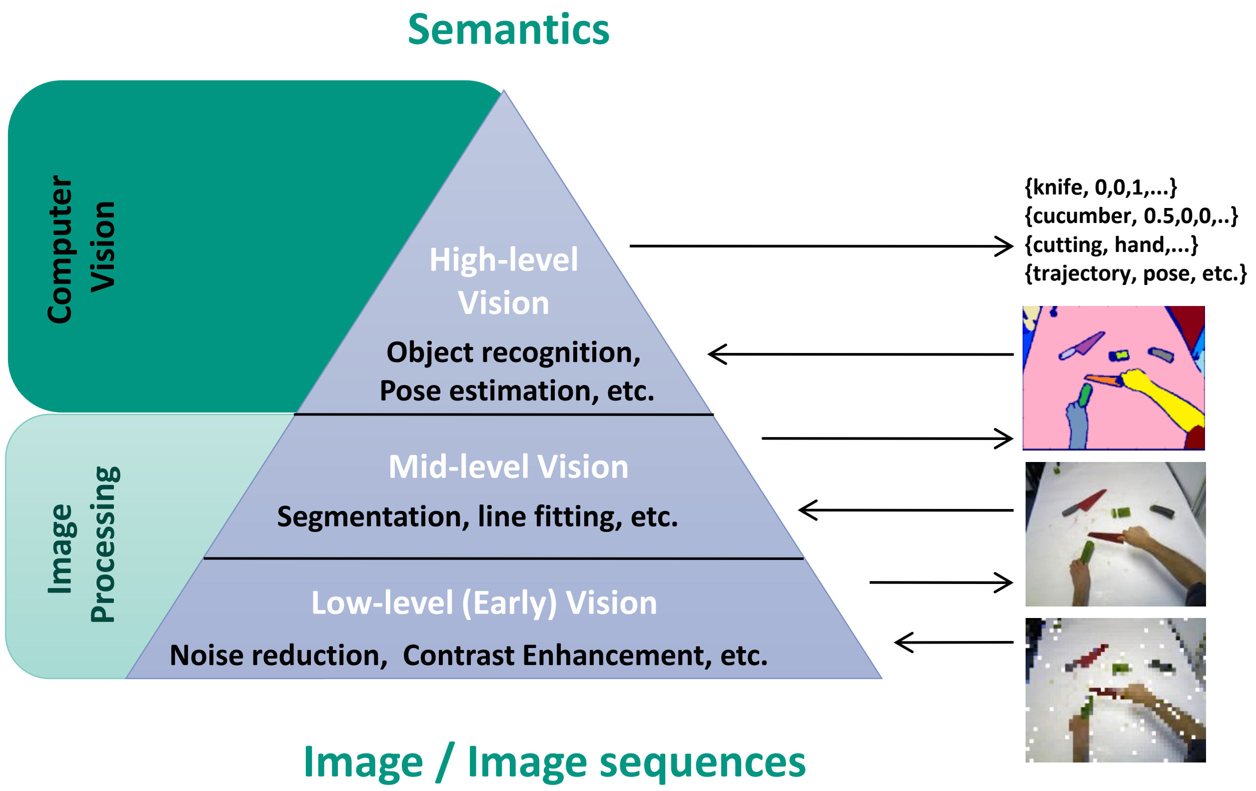
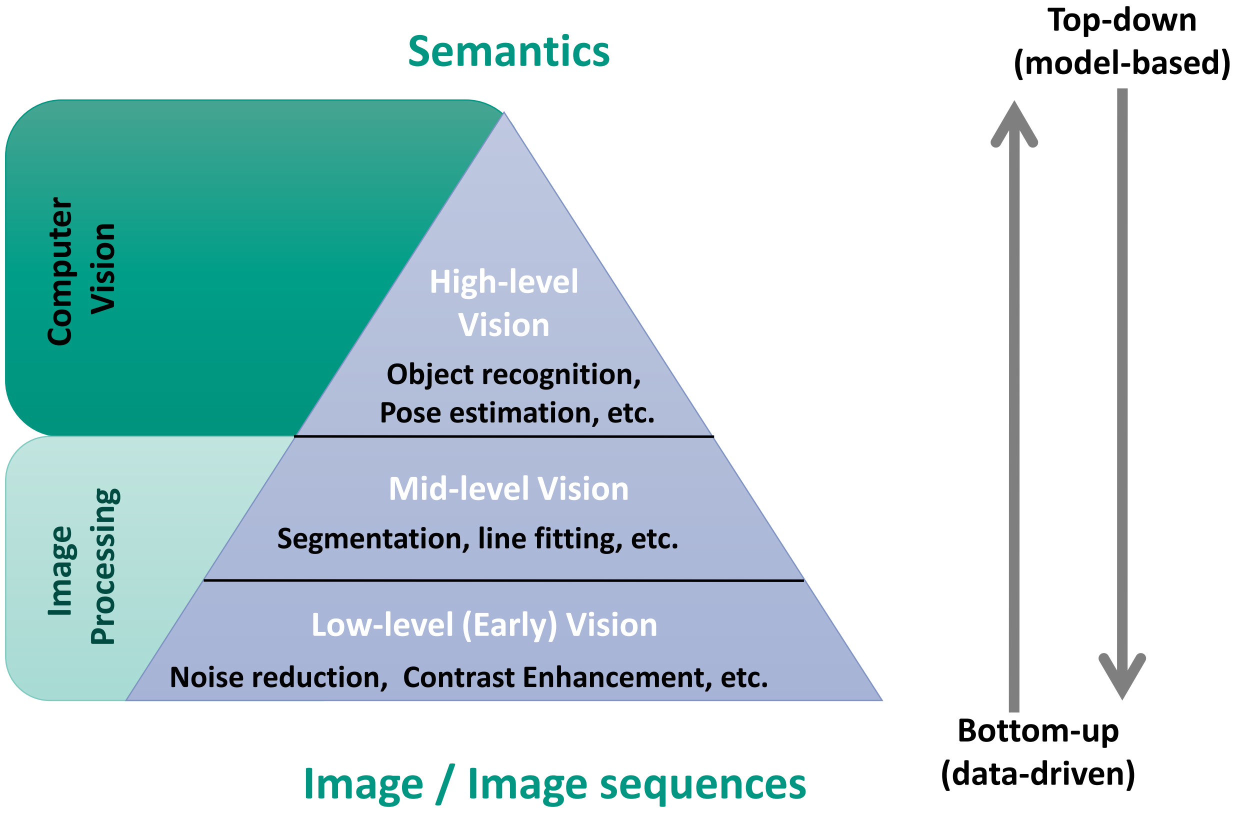



Image / Image sequences

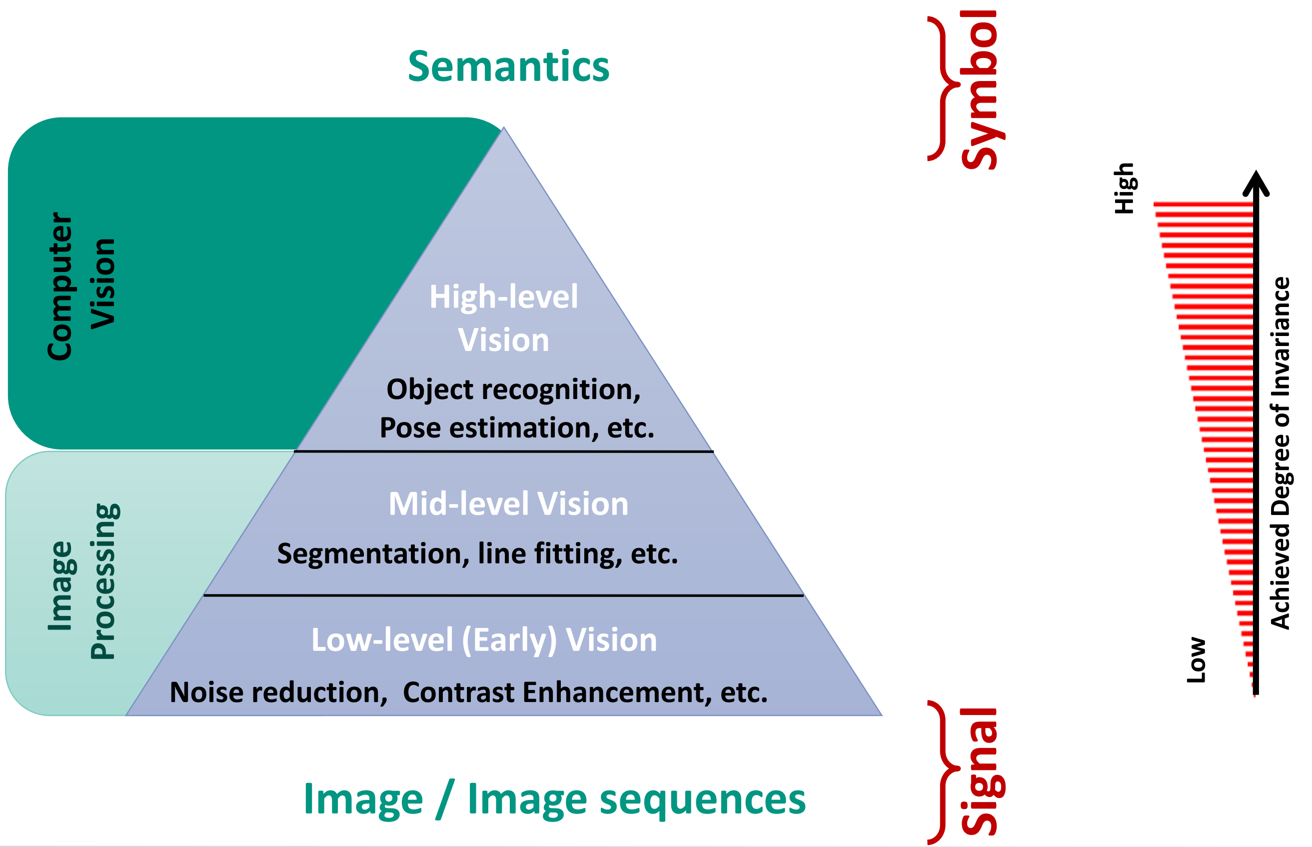
# What is Computer Vision?



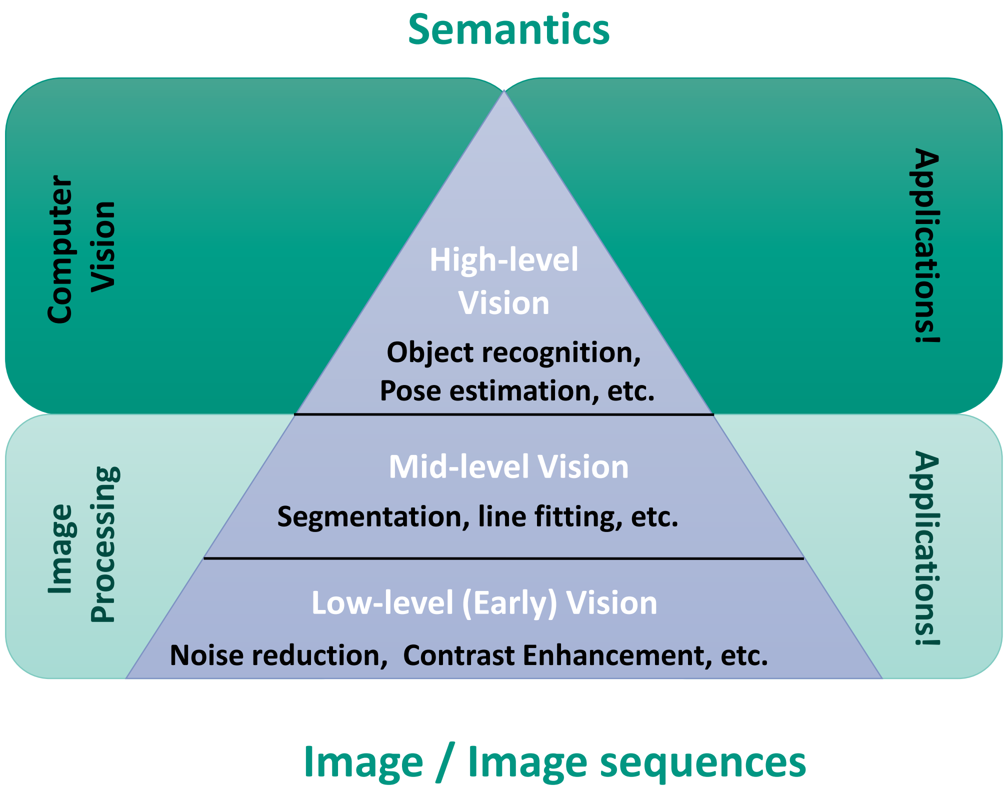
# What is Computer Vision?



# What is Computer Vision?



# What is Computer Vision?



# Robot Vision

## What is the Problem?

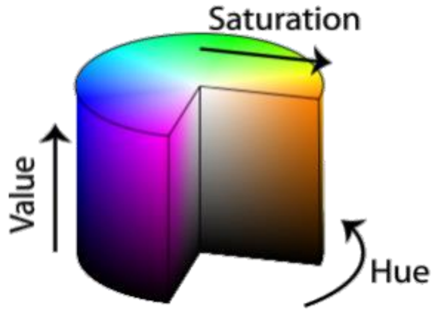
- **How can we enable a robot to 'see'?**

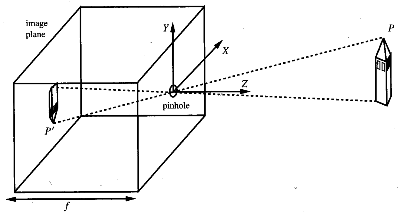
## Why is **Robot Vision** challenging?

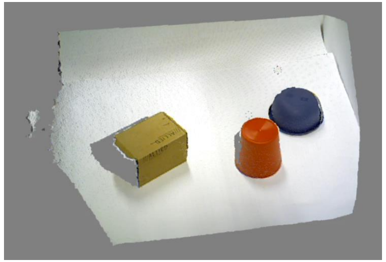
- Computer Vision is not just about transmitting images from a camera to a computer but also involves **processing** and **interpreting** them **(extraction of knowledge from images)**
- During transmission (3D world → 2D image), one dimension is lost
- **Robot Vision:**
  - **Dynamics:** Movement in the scene or camera
  - **Occlusions:** Objects may not be fully visible
  - Interpretation dependent on the current **task**
  - **Photometry:** Dependence on lighting and material properties

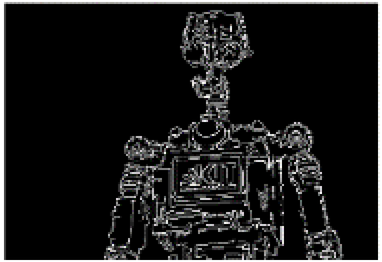
# Content

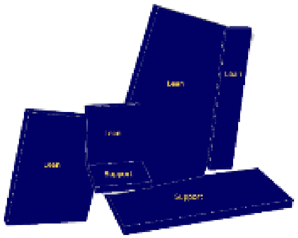
- Image Capture
  - 2D Image Representation
  - 3D Image Representation
  - Camera Model
- Operations on Images
  - Filtering Operations
  - Morphological Operators
- Feature Extraction and Pattern Recognition
  - Segmentation
  - Canny Edge Detector
  - Visual Servoing
  - Point Cloud Registration
- Example Applications from H²T



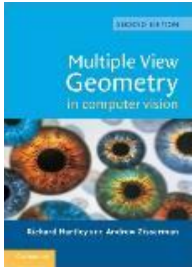



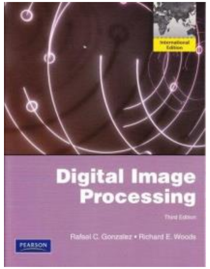


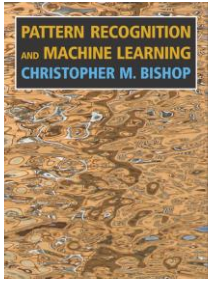


# Literature

- **Pattern Recognition and Machine Learning;** Christopher M. Bishop; Download
- **Multiple View Geometry in Computer Vision**; R. Hartley und A. Zisserman
- **Digital Image Processing**, Rafael C. Gonzalez and Richard E. Woods
- **Automatische Sichtprüfung**; J. Beyerer, F. Puente León und C. Frese
- **Computer Vision – Das Praxisbuch**; Pedram Azad, Tilo Gockel und Rüdiger Dillmann
- **Computer Vision: Algorithms and Applications**; Richard Szeliski ([http://szeliski.org/Book](http://szeliski.org/Book))



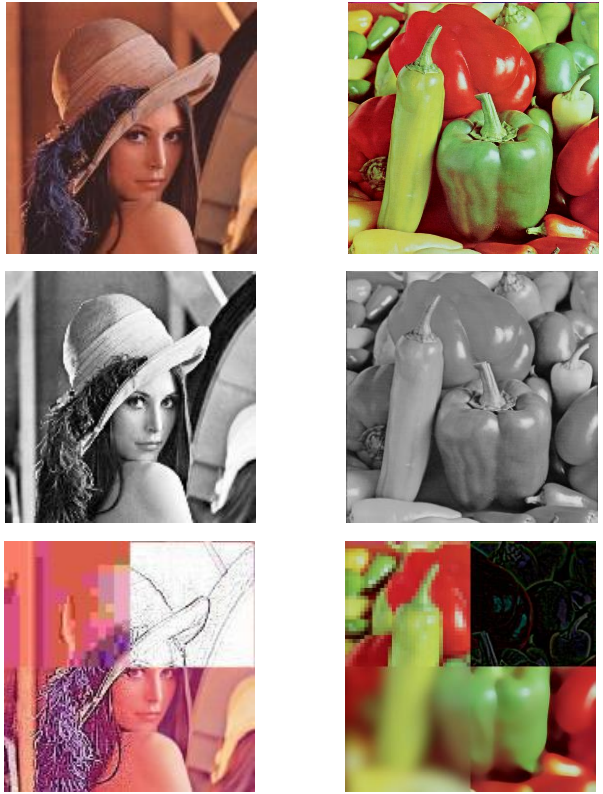


# Programming Libraries

- OpenCV
  - http://opencv.org
  - Facedetection, Optical Flow, GPU Computing
- Point Cloud Library (PCL)
  - http://pointclouds.org
  - Pointcloud processing, RANSAC primitive fitting, ICP
- Integrating Vision Toolkit (IVT)
  - http://ivt.sourceforge.net
  - Image processing, Feature matching

# Motivation

- Vision enables perception of the environment
- Usability in a technical system
  - Visual information must be captured
    - Good quality
    - Digital format
  - Relevant information must be extracted from the data
- **Image Capture:** Hardware
- **Image Processing:** mostly Software



# Test Image: Lena

The original image comes from the U.S. November issue of Playboy magazine from 1972.

It shows swedish Playmate **Lena Söderberg** (referred to by Playboy as 'Lenna Sjööblom').

# Test Image

The **USC-SIPI Image Database** was first released in 1977 to support research in image processing. The 'Peppers' image is one of the images in the database.

Early publications often used the 'Lena' image from Playboy magazine, which contradicts goals like equality and respect, so its use is discouraged.

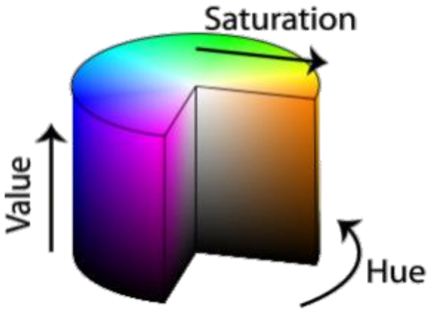
*[1] USC-SIPI image database, https://sipi.usc.edu/database/*
*[2] On alternatives to Lenna. Journal of Modern Optics, Jul. 2017*
*[3] A note on the Lena image. Nature Nanotech, vol. 13, no. 12, Art. no. 12, 2018. ("affects all Nature Research journals")*

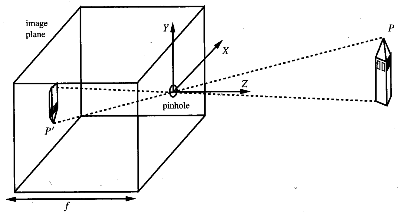
# Content

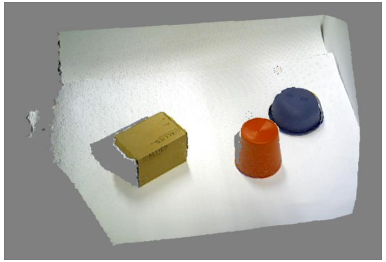
- Image Capture
  - **2D Image Representation**
  - 3D Image Representation
  - Camera Model

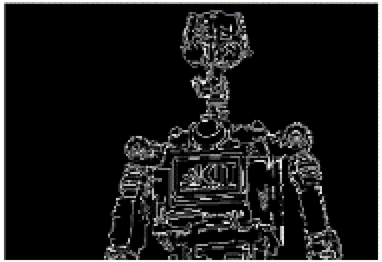
- Operations on Images

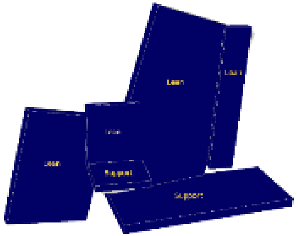
- Feature Extraction and Pattern Recognition







# Image Representation

- Images have to be represented on the computer
- An image is a 2D grid of discrete points (**pixels**)
- **Image Coordinates** (here):
  - $u$ (horizontal)
  - $v$ (vertical)
  - Origin is top left
  - Units: Pixel
- The **color** of a pixel can be represented in different ways
  - **Grayscale images**: A brightness value is stored for each pixel: usually **one byte per pixel**, generally values in $[0, 255]$
  - **Color images**: A separate value is stored for each pixel, e.g. **R** (red), **G** (green) and **B** (blue) → **3 bytes per pixel**

# Image Representation – Resolution

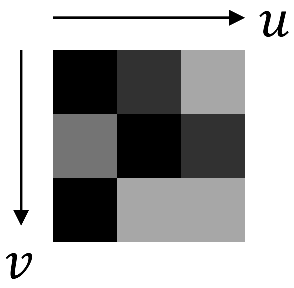
- The resolution indicates how many pixels are in an image

# Image representation – Monochrome Image

**Monochrome Image:** Discrete Function

$$Img: [0 \cdots n-1] \times [0 \cdots m-1] \rightarrow [0 \cdots q]$$

$$(u, v) \mapsto Img(u, v)$$




0 for black
255 for white

Usually: $q = 255$
$n = 640, \quad m = 480$ (VGA)
$n = 1920, \; m = 1080$ (1080p „Full HD“)
$n = 3840, \; m = 2160$ (2160p „Ultra HD“ or 4K UHD)

# Image Representation – Color Image

Karlsruhe Institute of Technology

- Different color models for different applications
- Classification by color space

**Example:**

- $RGB$-Model (**R**ed-**G**reen-**B**lue-Model): specifically for monitors (formerly: phosphorus crystals)

$$Img(u,v) \; = (r, g, b)^T \epsilon \; \mathbb{R}^3$$

- $HSI$ (Hue, Saturation, Intensity): suitable for color segmentation
- $CIE$: physical (wavelength)
- $CMYK$-Model (Cyan, Magenta, Yellow, black component “key”): Color printer (subtractive color mixing)
- $YIQ$: Analog television model

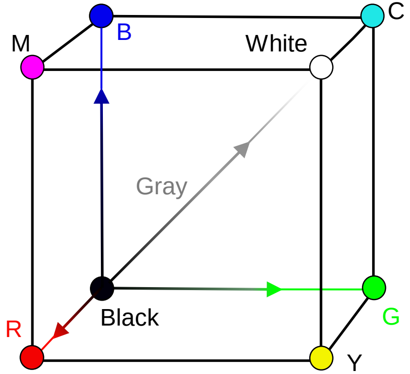
# Image Representation - RGB Color Spectrum

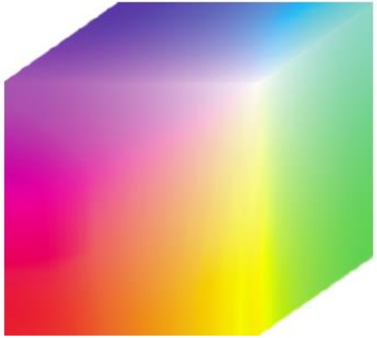


- Additive color mixing
- Three color values: *Red*, *Green*, *Blue*
- *RGB24*
  - One pixel is represented by **3 bytes** (red, green, blue)
  - Each byte corresponds to 8 bits
    → 256 values for each color
  - $2^8 \times 2^8 \times 2^8$ = 16,8 million colors representable

$$Img\colon [0 \cdots n-1] \times [0 \cdots m-1] \rightarrow [0 \cdots R] \times [0 \cdots G] \times [0 \cdots B]$$

$$(u, v) \mapsto Img(u, v) = (r, g, b)^T$$

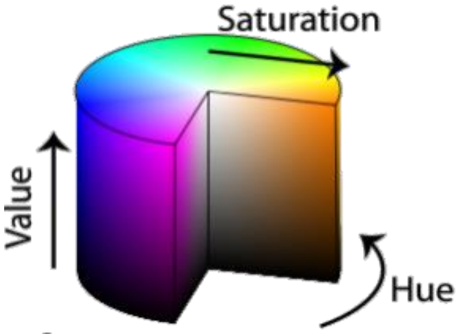


24-Bit $RGB$ color cube

# Image Representation – HSI (HSV) Color Spectrum

- *Hue, Saturation, Intensity/Value*
- Encodes the color information **separately** from the intensity and saturation of the color
  → not sensitive to changes in lighting
- Conversion from $RGB$ to $HSI$
  - $H$ undefined, if $R = G = B$
  - $S$ undefined, if $R = G = B = 0$




$$H = \begin{cases} \theta, & \text{falls } B < G \\ 360 - \theta, & \text{sonst} \end{cases}$$

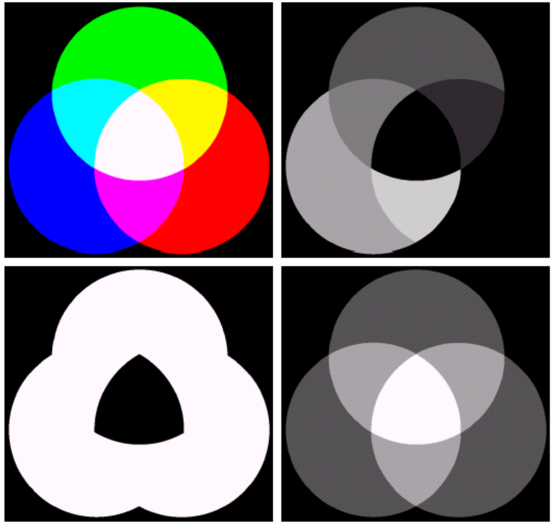
$$\theta = \arccos \frac{2R - G - B}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}$$

$$S = 1 - \frac{3}{R+G+B} \min(R, G, B)$$

$$I = \frac{1}{3}(R + G + B)$$

# Image Representation – Differences to RGB

$RGB$ | Hue



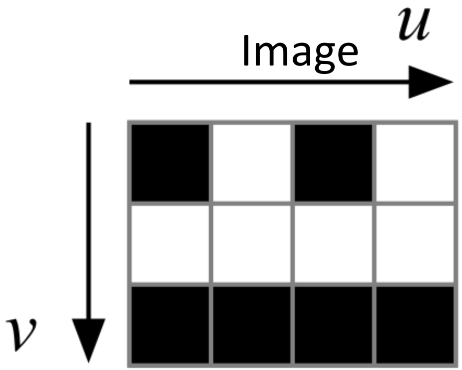
Saturation | Intensity

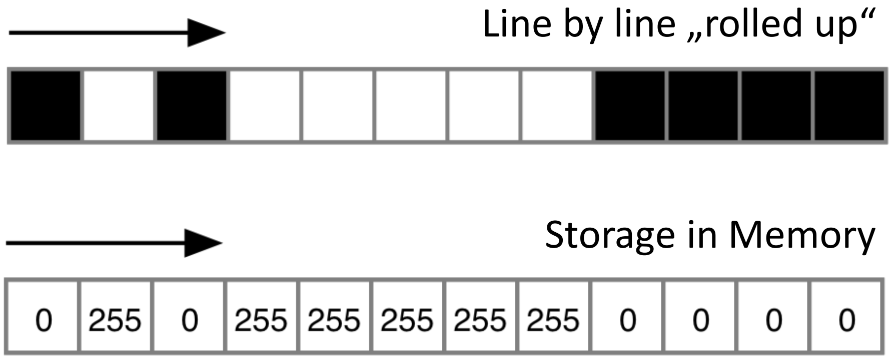
- The **main differences** compared to $RGB$ model: The $HSV$-color model **decouples color values** from **intensity values** and allows **independent changes** to hue, saturation and intensity values!

# Image Representation – Representation in Memory

Representation of an 8-bit grayscale image in memory

- Pixels are stored line by line, linear
  - from top left to bottom right
  - Careful: e.g. for bitmaps from bottom left to top right
- Grayscale coding:
  - One byte per pixel
  - 0 black, 255 white, grayscale in between

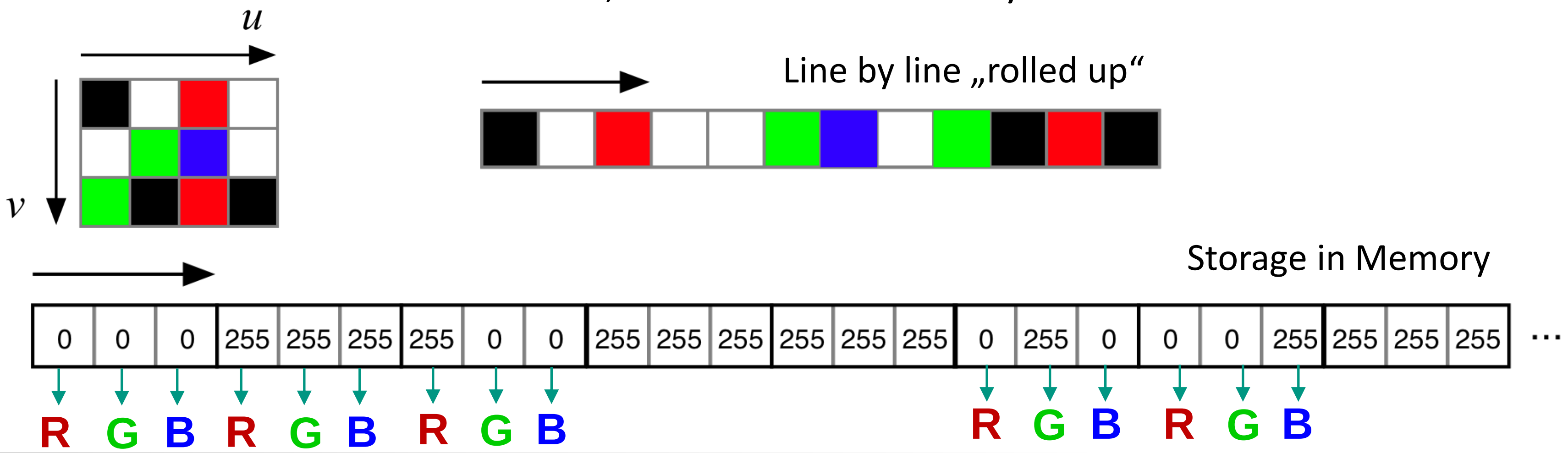


# Image Representation – Representation in Memory (2)

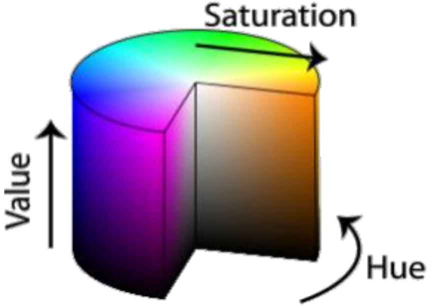
Representation of a $RGB24$ color image in memory

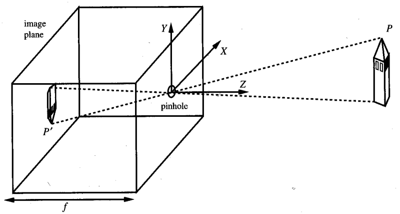
- Pixels are stored line by line, as with a grayscale image
- Color coding:
  - Three bytes per pixel (24 bit integers)
  - For each channel:  0 minimum, 255 maximum intensity

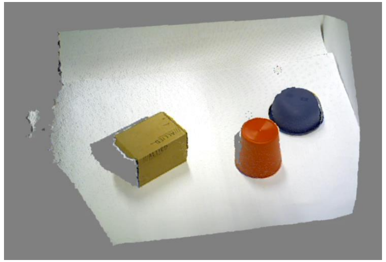


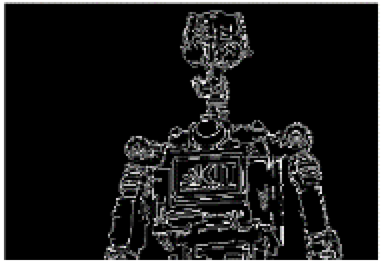
# Content

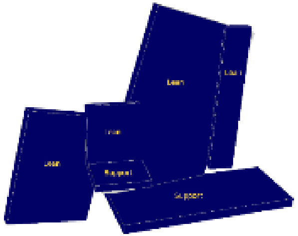
- Image Capture
  - 2D Image Representation
  - **3D Image Representation**
  - Camera Model
- Operations on Images
- Feature Extraction and Pattern Recognition



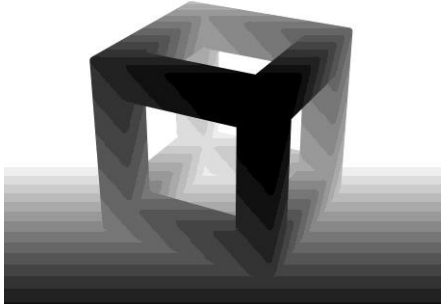




# Image Representation – Depth Images

## Depth Image:

- Monochrome and color images have no information about spatial or geometric relationships in the image
- In addition to the visual information, the distance to the sensor can be stored for each pixel



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

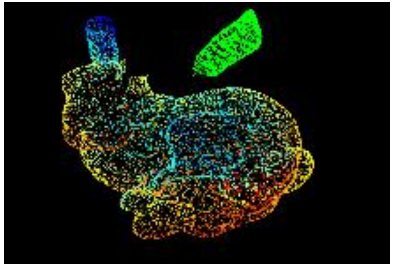
## Example:

- $RGBD$-Model (**R**ed-**G**reen-**B**lue-**D**epth Model) for depth cameras:

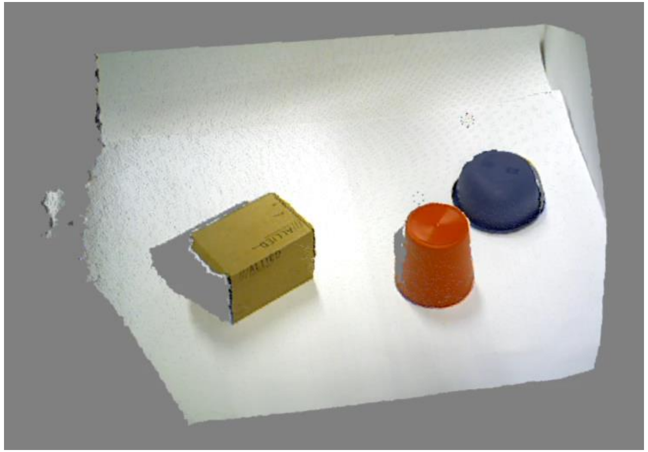
$$Img(u,v) = (r,g,b,d)^T \rightarrow \{d \in \mathbb{R}, (r,g,b) \in [0 \dots 255]^3 \subset \mathbb{N}_0^3\}$$

# Image Representation – Point Clouds

- A **Point Cloud** is a **discrete** set of **3D points** with a fixed coordinate system.



**2D Image**



**3D Point Cloud**

# Image Representation – Point Clouds (2)

- Point Cloud $P = \{(X, C) \big| X \in \mathbb{R}^3, C \in [0 \dots 255]^3 \subset \mathbb{N}_0^3\}$
  $X = (x, y, z)$ Location information
  $C = (r, g, b)$ Color information
  Additional (sensor) information can be stored (e.g. labels, normals, ...)

- Two different types of representation
  - **Organized/Ordered** point cloud (e.g. data from depth cameras)
    **2D array format**, size must be known in advance; **allocate memory for all possible positions**, even empty ones; **efficient access to neighboring points** since positions are known
  - **Unorganized/Unordered** point cloud (e.g. data from 3D laser scanner or LiDAR)
    **vector format**, points stored in a list **without predefined structure**/arrangement; only actual points are stored → **memory efficient** but it is hard to find neighboring points

# Example – Image Representation of the Roboception Camera

- Roboception camera (rc visard 160): stereo camera system with *160 mm* baseline (distance between the two cameras)

- **Camera Images**: Image resolution 1280 x 960 Pixel (≈1.3 MegaPixel).

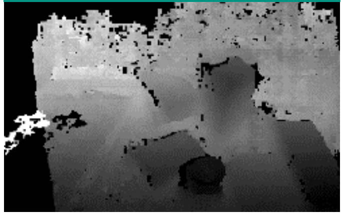
- **Depth Image:** Distance is calculated from the left and right camera image relative to the sensor (stereovision, later)

- **Confidence Image (special case):** Estimation of the confidence in the measured values of the depth image.

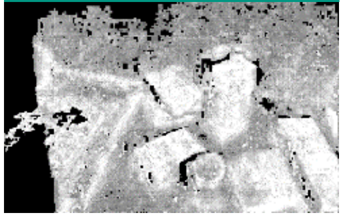
https://roboception.com

Left Camera (RGB)

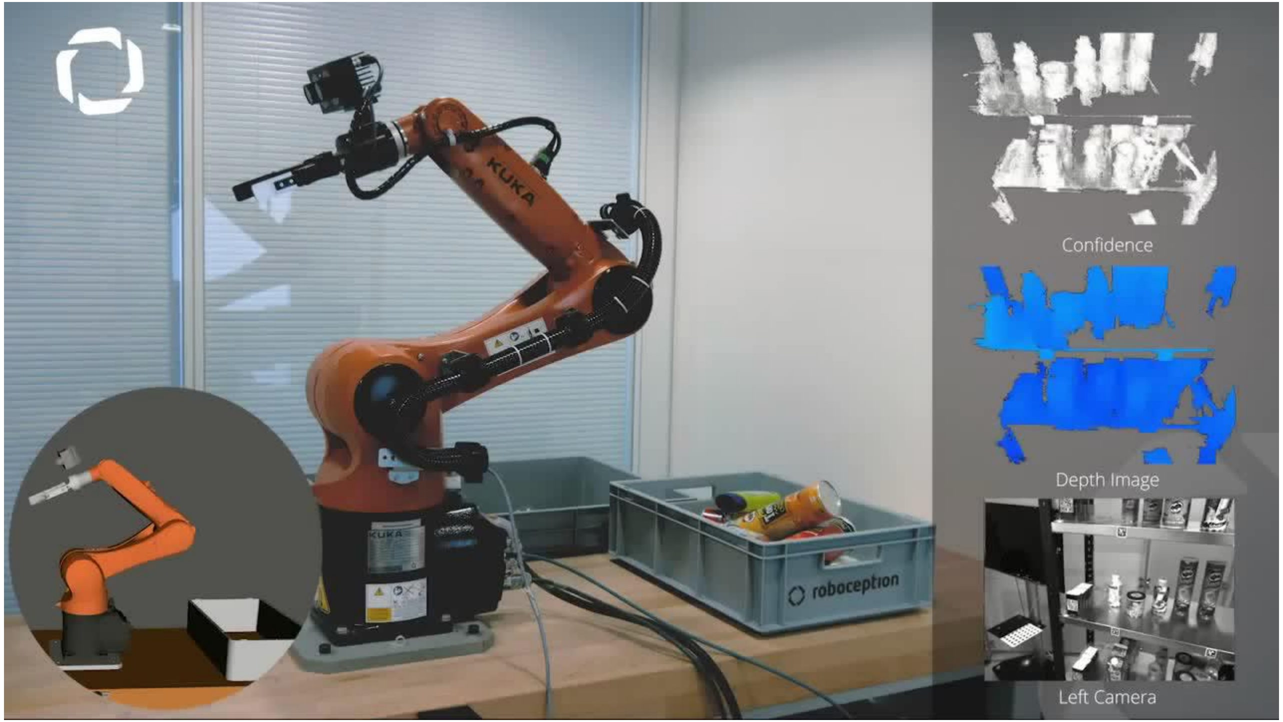
Depth Image



Confidence Image



# Roboception: rc visard 65 – Bin Picking Application



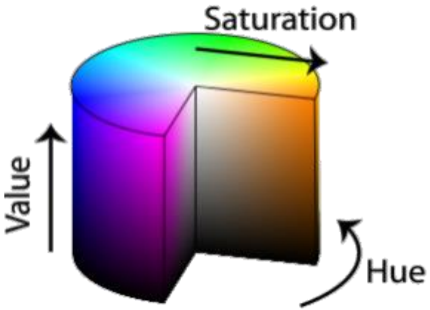

https://www.youtube.com/watch?v=-IeZbn1aA8Q

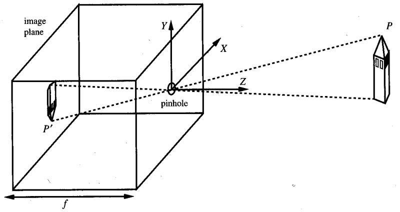
# Microsoft Azure Kinect

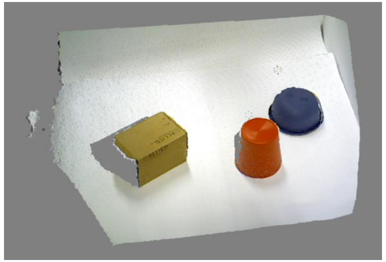
https://www.youtube.com/watch?v=jJglCYFiodI

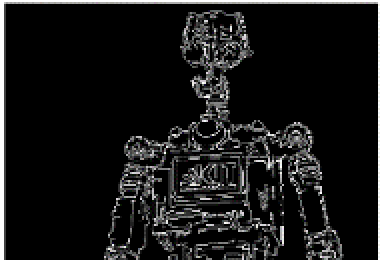
# Content

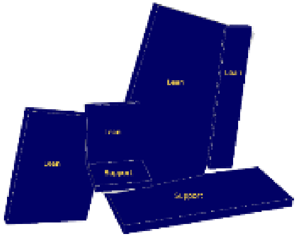
- Image Capture
  - 2D Image Representation
  - 3D Image Representation
  - **Camera Model**
- Operations on Images
- Feature Extraction and Pattern Recognition



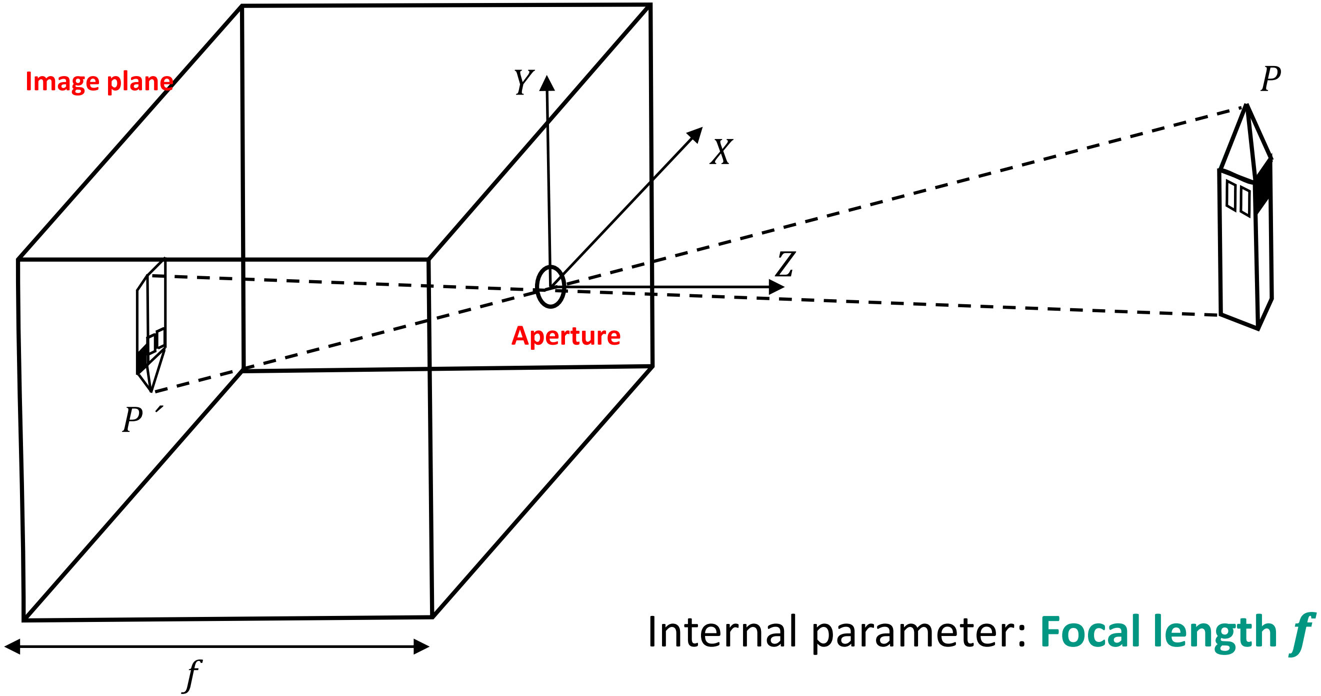







# Image Generation – Pinhole Camera

- Simplest model: **pinhole camera model**

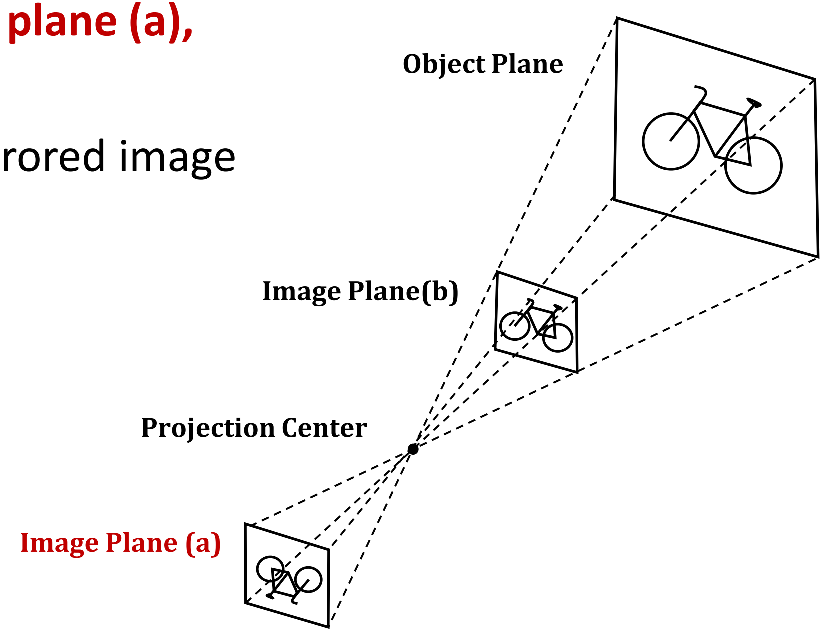



Describes the relationship between the coordinates of a point in 3D space and its projection onto the image plane

Internal parameter: **Focal length *f* ("focal distance")**

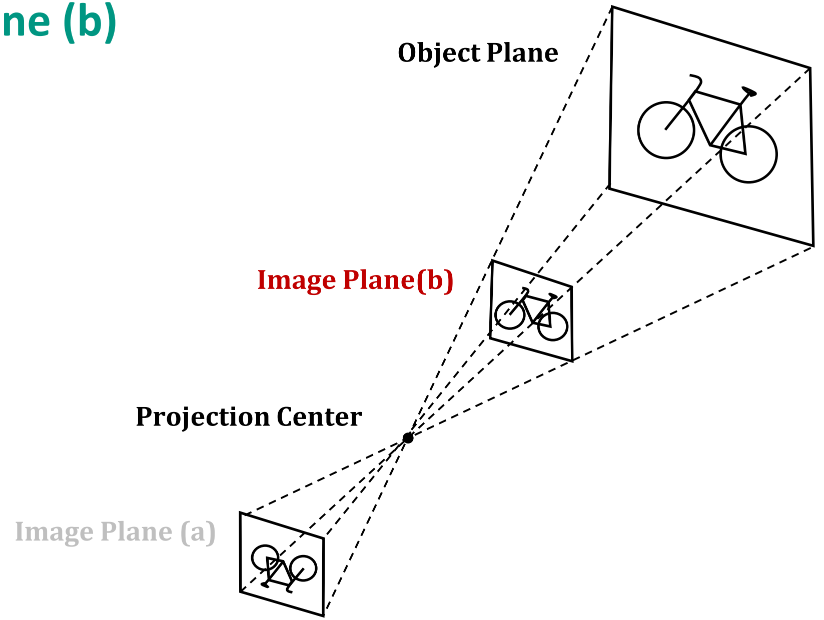
# Image Generation – Classic Pinhole Camera

- Classic pinhole camera model
  - **Projection center is in front of the image plane (a),** i.e. between scene and image plane
  - As a result: Horizontally and vertically mirrored image



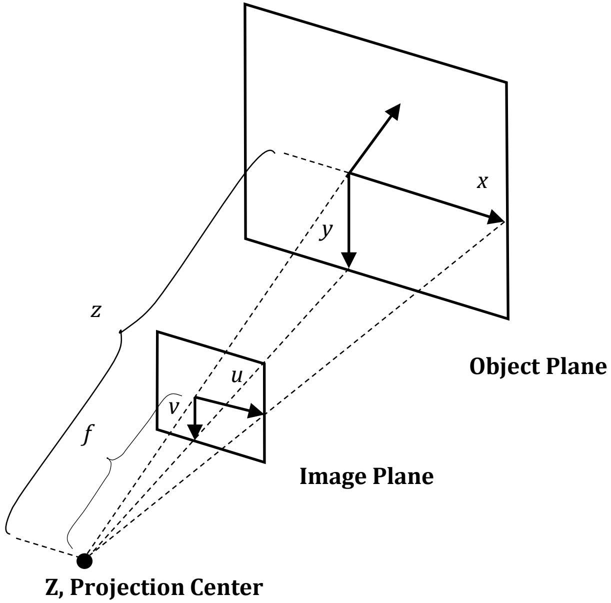
# Image Generation - Pinhole Camera in Positive Position

- Often used variant: pinhole camera model in **positive position**
  - **Projection center is behind the image plane (b)**
  - Therefore: no mirroring



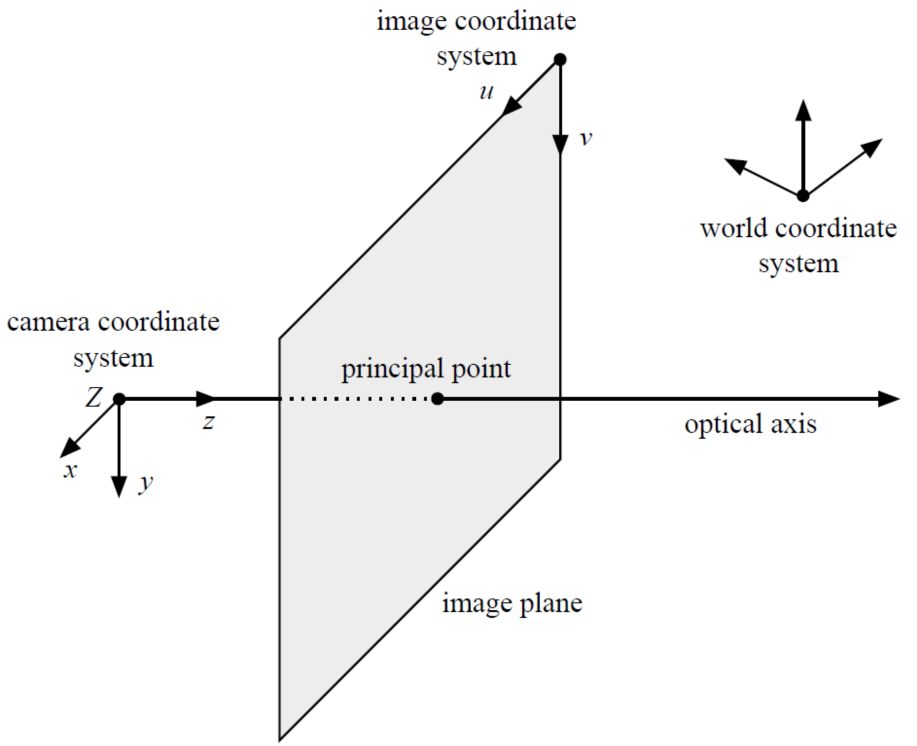
# Coordinate Systems

- **Principal axis:** Straight line through the projection center, perpendicular to the image plane
- **Principal point:** Intersection of the principal axis with the image plane
- **Image coordinates:** 2D coordinates $(u, v)$ of a point in the image. Unit: pixel
- **Camera coordinate system**: 3D coordinates $(x, y, z)$ of a point relative to the camera. The origin is in the projection center, the $x$- and $y$-axis are parallel to the $u$- and $v$-axis in the image plane. Unit: mm
- **World coordinate system:** 3D base coordinate system in the world.




Unit in this lecture: mm

# Coordinate Systems



# Image Generation – Pinhole Camera Projection

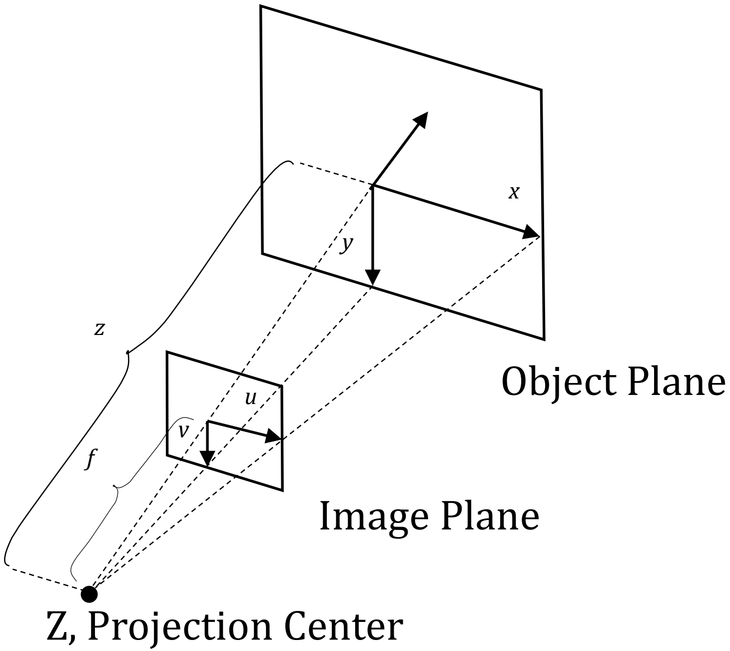
- **Second Intercept Theorem:** project a scene point $(x, y, z)$ onto an image point $(u, v)$:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x \\ y \end{pmatrix}$$

- The $z$ component is lost during projection!

- Back-projection:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{z}{f} \begin{pmatrix} u \\ v \end{pmatrix}$$



# Image Generation – From RGBD Image to Point Cloud

- With a pinhole camera it is not easily possible to determine correspondences from a pixel to a point in space

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{z}{f} \begin{pmatrix} u \\ v \end{pmatrix}$$

The key issue is that a pixel in an image corresponds to a ray in 3D space, not a unique point

- If the sensor is able to determine depth values $d$ for each pixel (e.g. stereo camera, LiDaR,, Time of Flight, ...), a simple correspondence can be established

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{d}{f} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

# Extended Camera Model

- Previous classic pinhole camera model is too simple for applications (lens distortion, imperfect calibration, or complex imaging conditions, …)
- **Extension of the model**
  - **Independent focal lengths $f_x$ and $f_y$** in $u$ and $v$ direction (**rectangular pixels**): $f = \begin{pmatrix} f_x \\ f_y \end{pmatrix}$
  - Main point $(c_x, c_y)$ is not identical to the origin of the camera coordinate system
  - Projection from camera to image coordinates can now be extended to:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \frac{1}{z} \begin{pmatrix} f_x \cdot x \\ f_y \cdot y \end{pmatrix}$$

Simple pinhole camera model:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x \\ y \end{pmatrix}$$

# Extended Camera Model (2)

- Using homogeneous coordinates, this can be written in the form of a matrix multiplication:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix} + \frac{1}{z} \begin{pmatrix} f_x \cdot x \\ f_y \cdot y \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f_x}{z} & 0 & \frac{c_x}{z} \\ 0 & \frac{f_y}{z} & \frac{c_y}{z} \\ 0 & 0 & \frac{1}{z} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

# Extended Camera Model (3)

- Using homogeneous coordinates, this can be written in the form of a matrix multiplication:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f_x}{z} & 0 & \frac{c_x}{z} \\ 0 & \frac{f_y}{z} & \frac{c_y}{z} \\ 0 & 0 & \frac{1}{z} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

# Extended Camera Model (4)

- Using homogeneous coordinates, this can be written in the form of a matrix multiplication:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f_x}{z} & 0 & \frac{c_x}{z} \\ 0 & \frac{f_y}{z} & \frac{c_y}{z} \\ 0 & 0 & \frac{1}{z} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix} = \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{K}} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- $K$ is the **calibration matrix of the camera**

$$\begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix} = K \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

# Extended Camera Model (5)

$$\begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix} = K \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = K^{-1} \begin{pmatrix} u \cdot z \\ v \cdot z \\ z \end{pmatrix}$$

$$K^{-1} = \begin{pmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{pmatrix}$$

# Intrinsic and Extrinsic Camera Parameters

- The parameters in the calibration matrix $K$ are referred to as **intrinsic parameters**.
- The relationship between the camera coordinate system and the world coordinate system is described by the **extrinsic parameters**: **a rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$ in space.**
- $\boldsymbol{R}$ and $\boldsymbol{t}$ specify the transformation from the world coordinate system to the camera coordinate system such that for a world point $\boldsymbol{x_w}$, the camera coordinates $\boldsymbol{x_c}$ are determined by:

$$\boldsymbol{x_c} = \boldsymbol{R}\,\boldsymbol{x_w} + \boldsymbol{t}$$

- By combining extrinsic and intrinsic parameters, the projection of a point from world coordinates to image coordinates is given by the **projection matrix $\boldsymbol{P}$**:

$$\boldsymbol{P} = \boldsymbol{K} \cdot (\boldsymbol{R} \mid \boldsymbol{t}) \qquad \boldsymbol{P} \in \boldsymbol{R}^{3\times 4}$$

# Camera Calibration

- **Camera calibration** means determining the extrinsic and intrinsic parameters of the camera
- This requires **at least 6 correspondences** between non-coplanar world points and their projections onto the image plane. For each correspondence, the following relationship holds true:

$$\begin{pmatrix} u \cdot w \\ v \cdot w \\ w \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \text{ mit } \; P = (K \cdot R \mid K \cdot t) = \begin{pmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{pmatrix}$$

- The unkown parameters $p_1$ to $p_{12}$ must be determined here

# Camera Calibration (2)

- The equation can be resolved as follows

$$P = (K \cdot R \mid K \cdot t) = \begin{pmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{pmatrix}$$

$$u = \frac{p_1 X + p_2 Y + p_3 Z + p_4}{p_9 X + p_{10} Y + p_{11} Z + p_{12}}$$

$$v = \frac{p_5 X + p_6 Y + p_7 Z + p_8}{p_9 X + p_{10} Y + p_{11} Z + p_{12}}$$

- As homogeneous coordinates are used, multiplying $P$ by a factor (not equal to zero) does not change anything. Therefore, the equation can be normalized and, for example, $p_{12} = 1$ can be set (i.e. multiplying P with $1/p_{12}$)

# Camera Calibration (3)

- The two equations can be further converted to

$$p_1X + p_2Y + p_3Z + p_4 = u \cdot p_9X + u \cdot p_{10}Y + u \cdot p_{11}Z + u$$
$$p_5X + p_6Y + p_7Z + p_8 = v \cdot p_9X + v \cdot p_{10}Y + v \cdot p_{11}Z + v$$

$$\Downarrow$$

$$u = p_1X + p_2Y + p_3Z + p_4 - u \cdot p_9X - u \cdot p_{10}Y - u \cdot p_{11}Z$$
$$v = p_5X + p_6Y + p_7Z + p_8 - v \cdot p_9X - v \cdot p_{10}Y - v \cdot p_{11}Z$$

# Camera Calibration (4)

- Each correspondence between world point and image point results in **2 linear equations**.
- With $n \geq 6$ correspondences, the linear system of equations $A\,x = b$ resolves to:

$$u = p_1X + p_2Y + p_3Z + p_4 - u \cdot p_9X - u \cdot p_{10}Y - u \cdot p_{11}Z$$
$$v = p_5X + p_6Y + p_7Z + p_8 - v \cdot p_9X - v \cdot p_{10}Y - v \cdot p_{11}Z$$

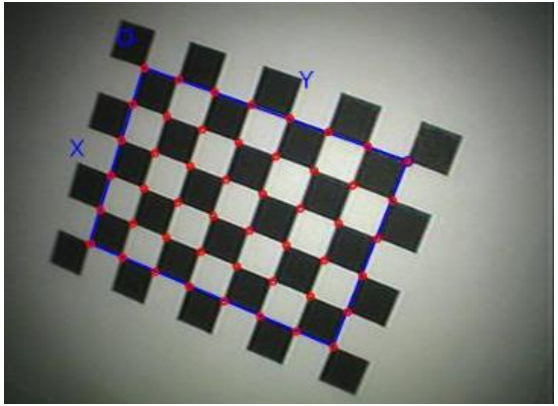
$$\underbrace{\begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_{10} \\ p_{11} \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} u_1 \\ v_1 \\ \cdots \\ u_n \\ v_n \end{pmatrix}}_{b}$$

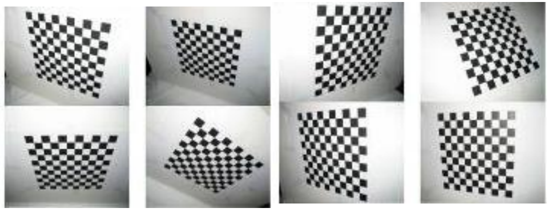
# Camera Calibration (5)

- Each correspondence between world point and image point results in **2 linear equations**.
- With $n \geq 6$ correspondences, the linear system of equations $A\,x = b$ resolves to:

$$A = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n \end{pmatrix}$$

$$x = \begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_{10} \\ p_{11} \end{pmatrix} \qquad b = \begin{pmatrix} u_1 \\ v_1 \\ \cdots \\ u_n \\ v_n \end{pmatrix}$$



# Camera Calibration (6)

- The optimal solution $x^*$ using the least squares method for such an **overdetermined linear system** of equations $(Ax \; = \; b)$ results from the solution of

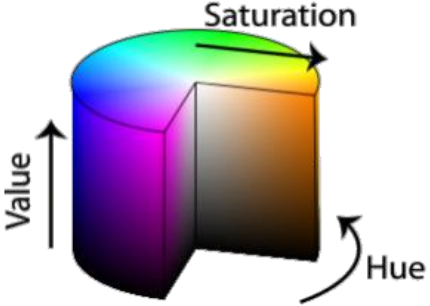
$$A^T A \, x^* = A^T \, b$$

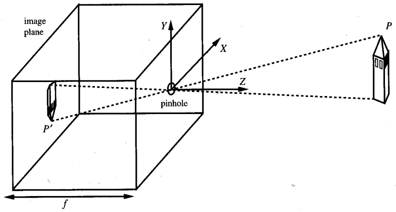
$$x^* = (A^T A)^{-1} A^T \, b$$

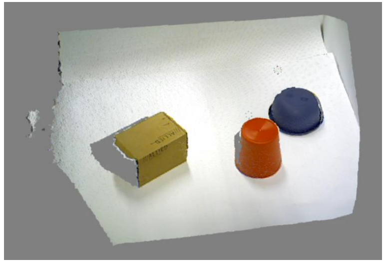
$(A^T A)^{-1} A^T$: **Moore-Penrose Pseudo Inverse**

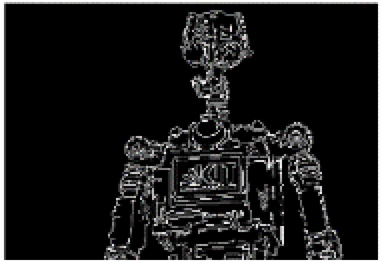
# Content

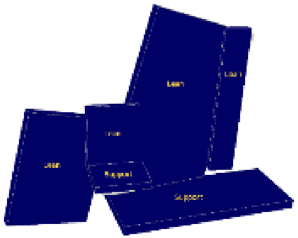
- Image Generation
- Operations on Images
  - **Filter Operations**
  - Morphological Operators
- Feature Extraction and Pattern Recognition







# Filter Operations

- **Filters** in image processing
  also *spatial filters, spatial masks, kernels, windows*

- A filter is an **operation** on a set of neighboring pixels, i.e.
  - predefined operations
  - Neighborhood (usually a small rectangle)

- A filter is applied to all pixels of the image
  - Calculation of a new pixel value by applying the filter operation taking into account the neighborhood pixels

- Properties of a **linear filter**

| | |
|---|---|
| $f(x+y) = f(x) + f(y)$ | additive |
| $f(\alpha x) = \alpha f(x)$ | homogenous |

# The Filter

- A filter is applied to an image element by placing the filter mask on this image element, multiplying the mask values by the underlying pixel values and adding up the result.

- Example:

$w(x, y)$

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

Filter

$*$

$f(x, y)$

| | | | | |
|---|---|---|---|---|
| a | b | c | d | e |
| f | g | h | i | j |
| k | l | m | n | o |
| p | q | r | s | t |
| u | v | w | x | y |

Input Image

$=$

$g(x, y)$

| | | | | |
|---|---|---|---|---|
| | | | | |
| | g´ | h´ | i´ | |
| | l´ | m´ | n´ | |
| | q´ | r´ | s´ | |
| | | | | |

Resulting Image

# The Filter (2)

$w(x,y)$

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

$*$

$f(x,y)$

| | | | | |
|---|---|---|---|---|
| a | b | c | d | e |
| f | g | h | i | j |
| k | l | m | n | o |
| p | q | r | s | t |
| u | v | w | x | y |

$=$

$g(x,y)$

| | | | | |
|---|---|---|---|---|
| | | | | |
| | g´ | h´ | i´ | |
| | l´ | m´ | n´ | |
| | q´ | r´ | s´ | |
| | | | | |

$$m' = -1 \cdot g + 0 \cdot h + 1 \cdot i - 2 \cdot l + 0 \cdot m + 2 \cdot n - 1 \cdot q + 0 \cdot r + 1 \cdot s$$

- Example:

| | | | | |
|---|---|---|---|---|
| -1 a | 0 b | 1 c | d | e |
| -2 f | 0 g | 2 h | i | j |
| -1 k | 0 l | 1 m | n | o |
| p | q | r | s | t |
| u | v | w | x | y |

$g' = -a + c - 2f + 2h - k + m$

| | | | | |
|---|---|---|---|---|
| a | -1 b | 0 c | 1 d | e |
| f | -2 g | 0 h | 2 i | j |
| k | -1 l | 0 m | 1 n | o |
| p | q | r | s | t |
| u | v | w | x | y |

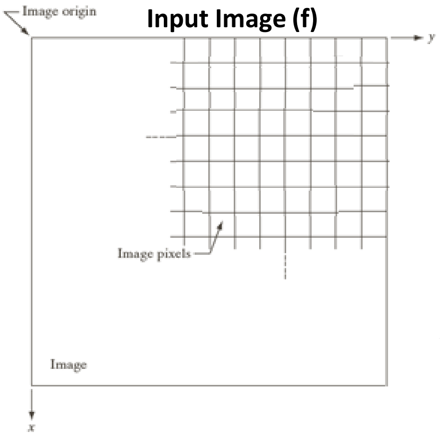
$h' = (d + 2i + n) - (b + 2g + l)$

| | | | | |
|---|---|---|---|---|
| a | b | -1 c | 0 d | 1 e |
| f | g | -2 h | 0 i | 2 j |
| k | l | -1 m | 0 n | 1 o |
| p | q | r | s | t |
| u | v | w | x | y |

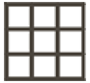
$i' = (e + 2j + o) - (c + 2h + m)$

# Filter over every Pixel

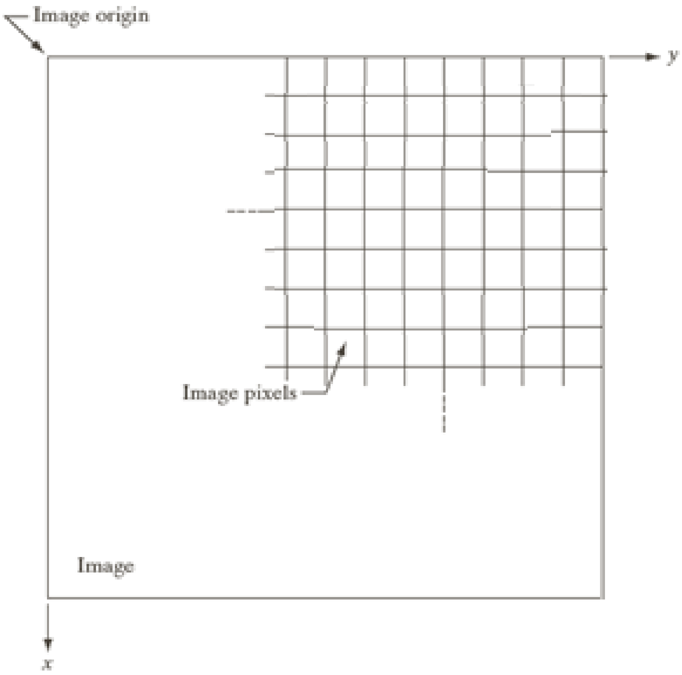
Input Image (f)



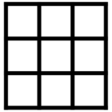

Filter



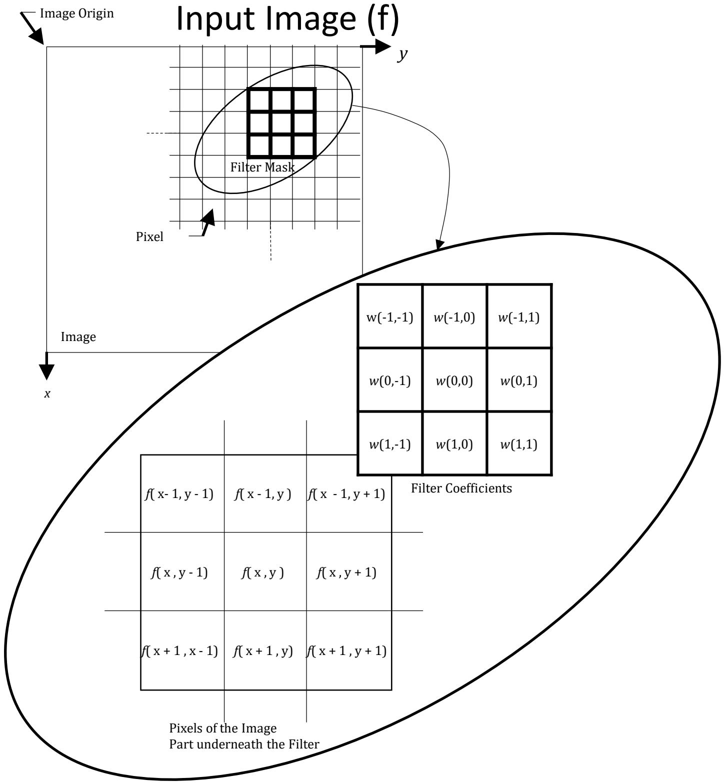
# Filter Operations



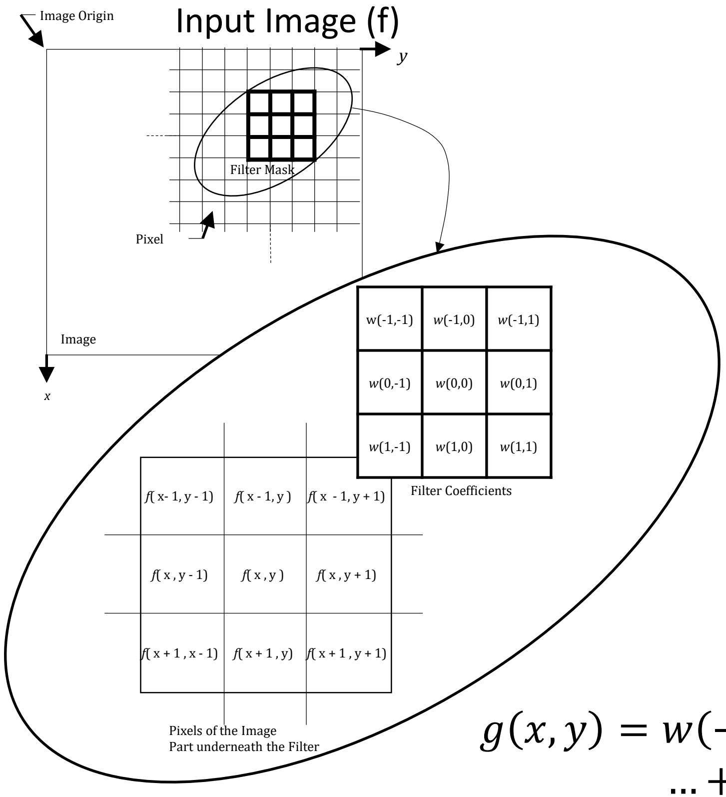

Filter

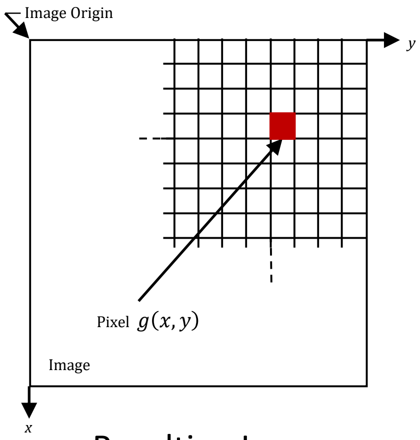


# Filter Operations



# Filter Operation – Correlation



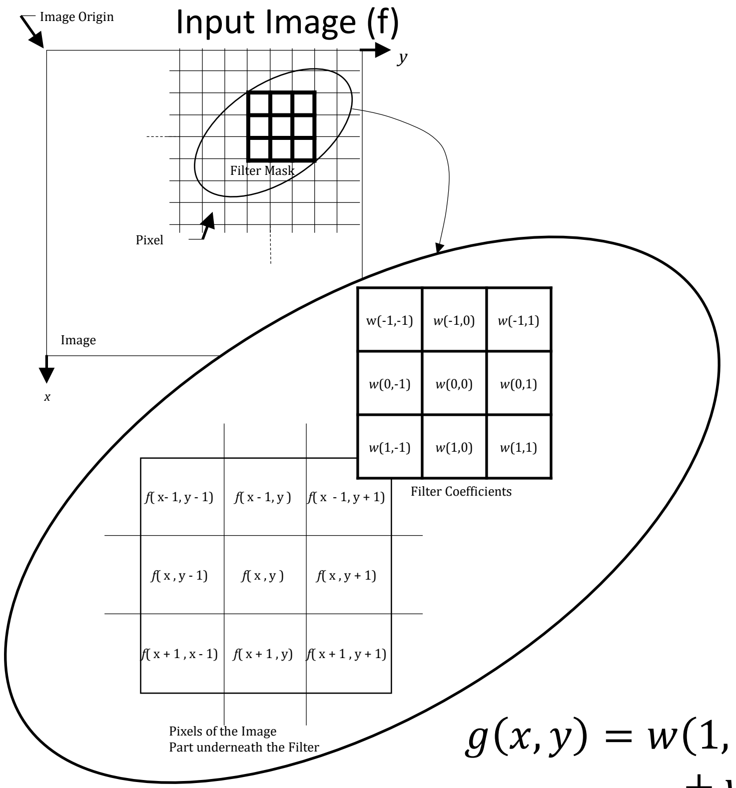



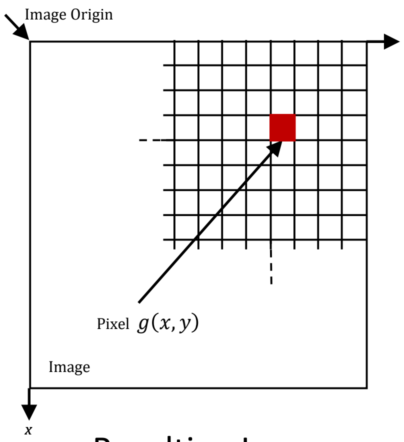

Resulting Image:

$$\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x+s, y+t)$$

$$g(x,y) = w(-1,-1) \cdot f(x-1, y-1) + w(-1,0) \cdot f(x-1, y) + \\ \ldots + w(0,0) \cdot f(x,y) + \cdots + w(1,1) \cdot f(x+1, y+1)$$

# Filter Operation – Convolution




Resulting Image:

$$\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x-s, y-t)$$

$$g(x,y) = w(1,1) \cdot f(x-1, y-1) + w(1,0) \cdot f(x-1, y) + \\ \ldots + w(0,0) \cdot f(x,y) + \cdots + w(-1,-1) \cdot f(x+1, y+1)$$

# Filter Operations – Basics

- **Correlation** is a filter operation in which a filter mask is moved across the image and the sum of the products at each pixel is calculated. Correlation is a function of the shift of the filter

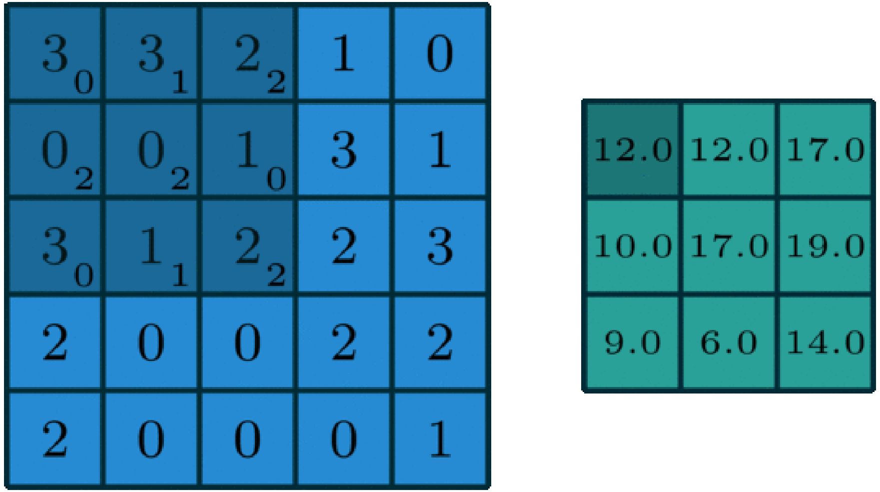
$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x+s, y+t)$$

- **Convolution** is a filter operation in which the filter is first rotated by 180°

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x-s, y-t)$$

# Filter Operations – Image Area

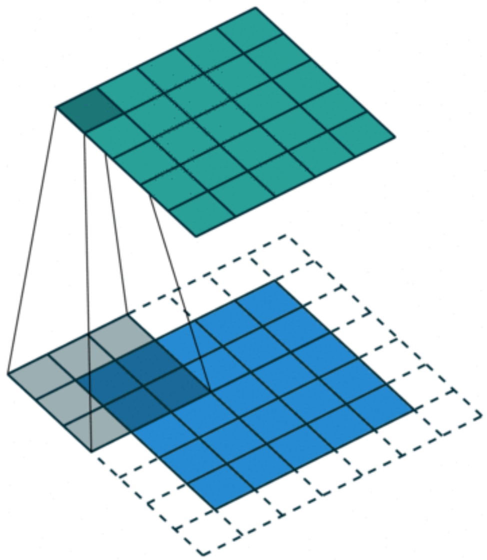
- Standard filter operations can change the resolution of the image. If only valid pixels are considered, **the image is reduced in size** - the edges are cut off, as the edge pixels are never in the center of a filter.




https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

# Filter Operations – Edges

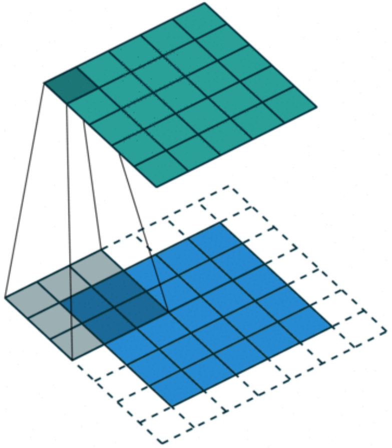
- In order to maintain the size of the original image, the edges are often "filled" with artificial pixels → This is referred to as **padding**



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

# Filter Operations – Edges (2)

- What happens at the edges?
  - **Constant** value, e.g. zero:
    Pixels at the edges are set to **zero**
  - **Wrap**:
    Image is “continued”

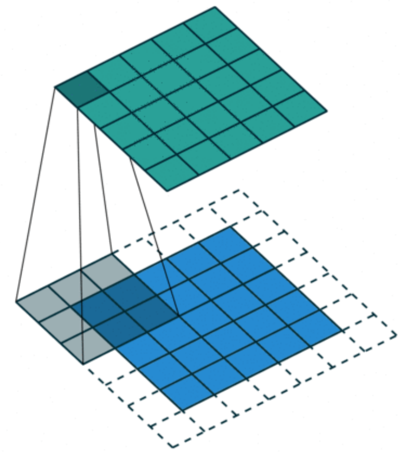


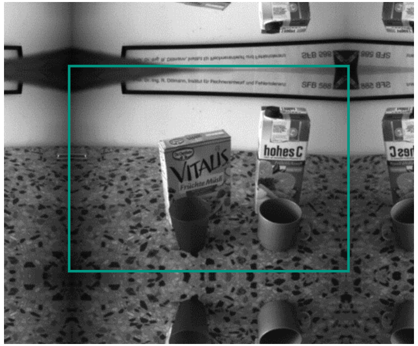
Constant

Wrap

# Filter Operations – Edges (3)

- What happens at the edges?
  - **Mirror/ Reflect**:
    Image is mirrored at the edges
  - **Clamp/ Replicate**:
    Take last value





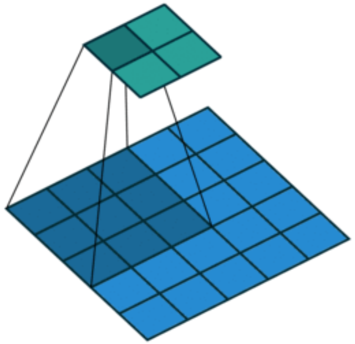

Mirror

Clamp

# Filter Operations – Step Size

- It is often useful to reduce the resolution of the original image in order to change the information content. This is achieved by changing the **step size of the filter.**
- **Example:**

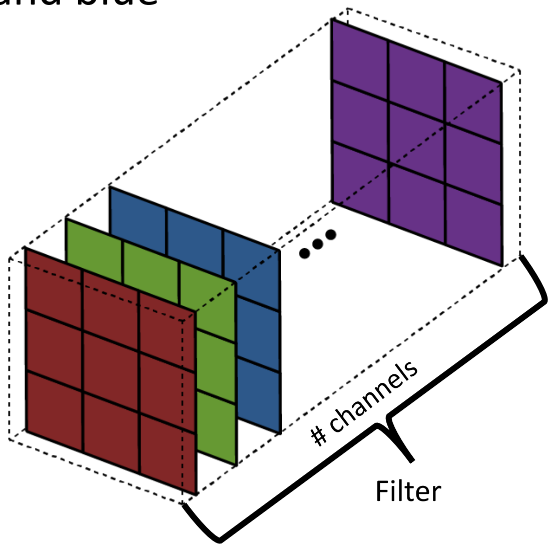
  The height and width of the image is approximately halved if only every **second pixel** is considered for the filter operation.



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

# Filter Operations - Application to Color Images

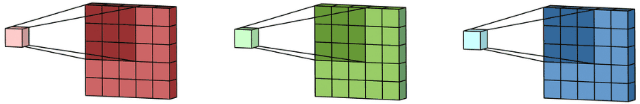
- The application of filters to grayscale images is the trivial case
  - The image has only one channel (0...255 as gray value)
- RGB images are often of greater interest than grayscale images
  - The image has three channels - one each for red, green and blue
- In general:
  Each filter has one filter matrix (**kernel**)
  per input channel



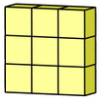

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

# Filter Operations - Application to Color Images (2)

- Each kernel is applied to the corresponding channel



- The results of each kernel are added
- One output channel is created per filter:
  - A bias term is often added to the output value (CNNs)



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

# Filter Operations

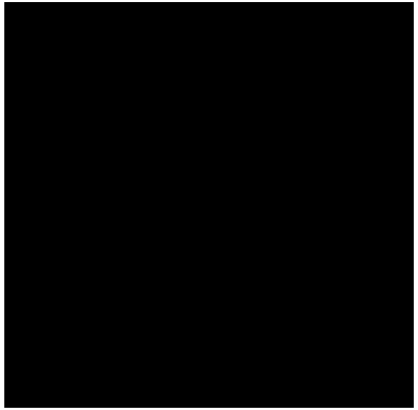
| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

How are **mask coefficients** defined?

- Depends on what the filter is supposed to do!

Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |



Result (Deletion)

# Filter Operations

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

How are **mask coefficients** defined?

- Depends on what the filter is supposed to do!

Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Results (Identity)

# Filter Operations

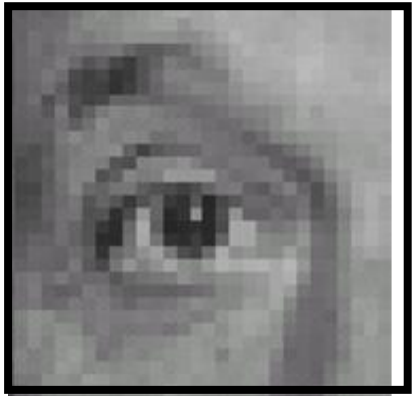
| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

How are **mask coefficients** defined?

- Depends on what the filter is supposed to do!

Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |



*Shifted* to the *left* by 1 pixel

# Filter Operations

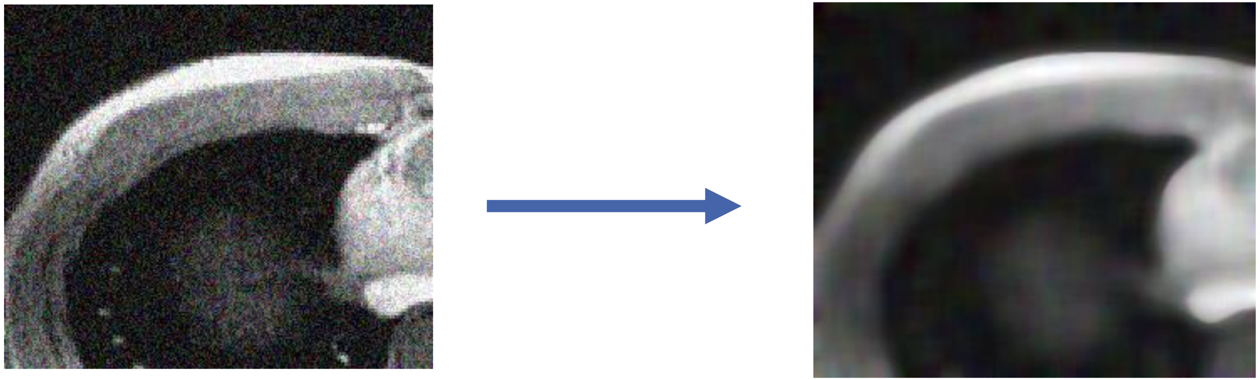
- **Low-pass filter:** smoothing, noise elimination
  - Median filter
  - Mean filter
  - Gaussian filter

- **High-pass filter:** Edge detection
  - Prewitt
  - Sobel
  - Laplace

- **Combined** operators
  - Laplacian of Gaussian

# Median Filter

- **Non-linear** filter for noise suppression
  - The filter response is based on the order (ranking) of the pixels contained in the image area enclosed by the filter.
- Steps:
  - Select the kernel size (mask)
  - Sort all gray values in the area of the kernel
  - Determine the average gray value from the sorted pixels
  - Select the pixel as the new value



# Median Filter - Example

| | | |
|---|---|---|
| 2 | 4 | 3 |
| 5 | 9 | 2 |
| 7 | 2 | 4 |

2 , 4 , 3 , 5 , 9 , 2 , 7 , 2 , 4

**Sort:** 2 , 2 , 2 , 3 , 4 , 4 , 5 , 7 , 9

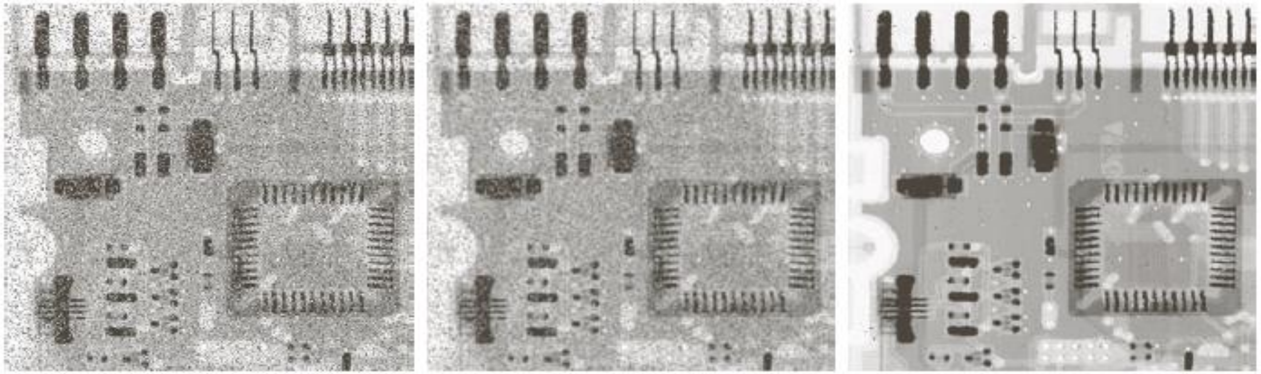
**New:**

| | | |
|---|---|---|
| 2 | 4 | 3 |
| 5 | 4 | 2 |
| 7 | 2 | 4 |

Instead of the median value, you can select the maximum value (**max filter**) to find the brightest points in an image. The **min filter** can also be used for the darkest points. Both are non-linear filters.

# Median Filter – Example (2)

- For certain types of noise, such as **salt and pepper noise**, the median filter has very good noise reduction properties with less blurring than linear filters of similar size!
  - Suitable for salt and pepper noise
  - Not suitable for Gaussian noise
- Preserves edges and removes noise in the image



Original image damaged by *salt and pepper noise* | 3 X 3 Mean filter | 3 X 3 Median filter

# Mean Filter

- **Goal:** Noise reduction
  - Average of one pixel and its 8 neighbors
  - Size can be chosen arbitrarily
  - 3x3 mean value filter

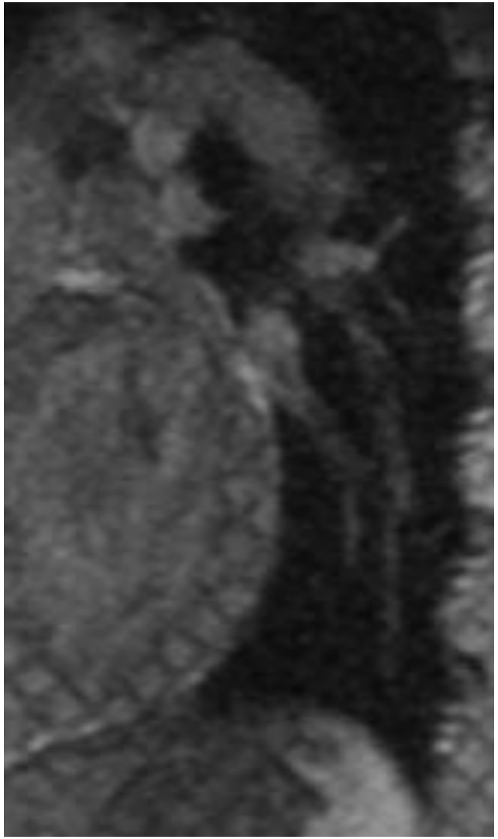
| | | |
|---|---|---|
| $\frac{1}{9}$ | $\frac{1}{9}$ | $\frac{1}{9}$ |
| $\frac{1}{9}$ | $\frac{1}{9}$ | $\frac{1}{9}$ |
| $\frac{1}{9}$ | $\frac{1}{9}$ | $\frac{1}{9}$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 13 | 25 | 30 | 34 | 40 | 46 | 60 | 76 |
| 45 | (1/9)*50 | (1/9)*52 | (1/9)*55 | 65 | 67 | 87 | 77 |
| 34 | (1/9)*45 | (1/9)*55 => 52 | (1/9)*60 | 54 | 45 | 56 | 65 |
| 45 | (1/9)*50 | (1/9)*52 | (1/9)*48 | 45 | 65 | 65 | 45 |
| 34 | 34 | 54 | 56 | 57 | 58 | 67 | 70 |
| 45 | 46 | 46 | 46 | 45 | 53 | 52 | 60 |
| 50 | 68 | 69 | 60 | 70 | 70 | 78 | 79 |
| 68 | 70 | 78 | 78 | 80 | 80 | 80 | 90 |

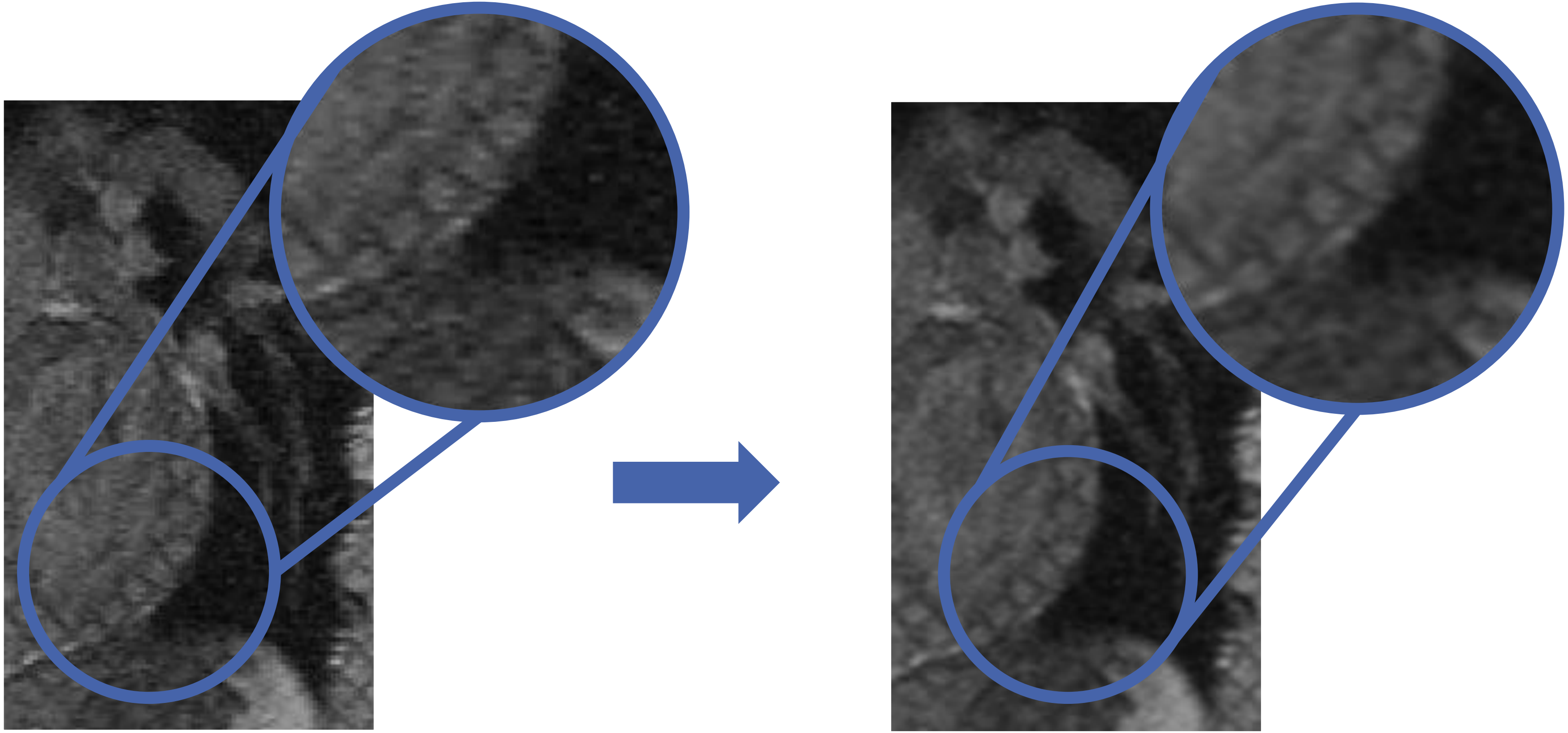
$$W_{Mittelwert} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$g(x,y) = \sum_{s=-1}^{1}\sum_{t=-1}^{1} w(s,t) \cdot f(x+s, y+t)$$

# Mean Filter - Example



# Mean Filter – Example (2)



# Gaussian Filter

- **Objective:** Noise suppression, smoothing
- Defined by two-dimensional Gaussian function

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

- Approximation of $w(x, y)$ using a $3 \times 3$ filter for $\sigma = 0.85$:

$$W_{Gauß} = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

| | | |
|---|---|---|
| $w(-1,-1)$ | $w(-1,0)$ | $w(-1,1)$ |
| $w(0,-1)$ | $w(0,0)$ | $w(0,1)$ |
| $w(1,-1)$ | $w(1,0)$ | $w(1,1)$ |

# Gaussian Filter (2)

- The degree of the smoothing is determined by the parameter $\sigma$ :
  **The larger $\sigma$, the stronger the smoothing.**
- The size $n \times n$ of the filter mask influences the quality of the approximation of the filter

Original Image

$\sigma^2 = 4$

$\sigma^2 = 16$

# Filter Operations

- **Low-pass filter:** smoothing, noise elimination
  - Median filter
  - Mean filter
  - Gaussian filter

- **High-pass filter:** Edge detection
  - Prewitt
  - Sobel
  - Laplace

- **Combined** operators
  - Laplacian of Gaussian

# Filter – Prewitt

- **Prewitt-X Filter** $P_x = \dfrac{\partial f(x,y)}{\partial x}$
  - Gradient in horizontal direction
  - Approximated by

$$p_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

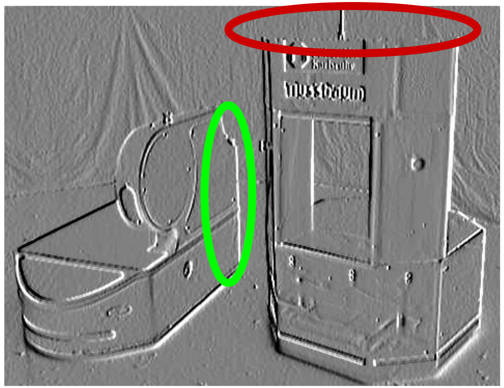
- **Prewitt-Y Filter** $P_y = \dfrac{\partial f(x,y)}{\partial y}$
  - Gradient in vertical direction
  - Approximated by

$$p_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

# Filter – Prewitt (2)

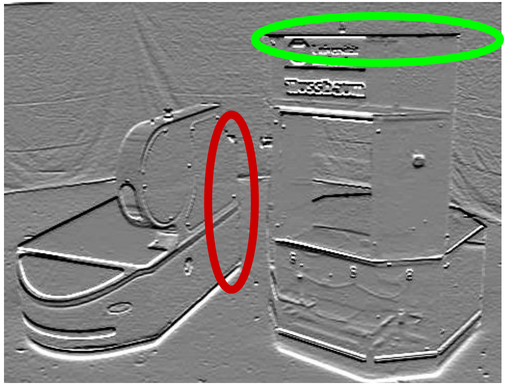
- **Prewitt-X Filter** $P_x = \frac{\partial f(x,y)}{\partial x}$
  - Approximated by

$$p_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$




- **Prewitt-Y Filter** $P_y = \frac{\partial f(x,y)}{\partial y}$
  - Approximated by

$$p_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$



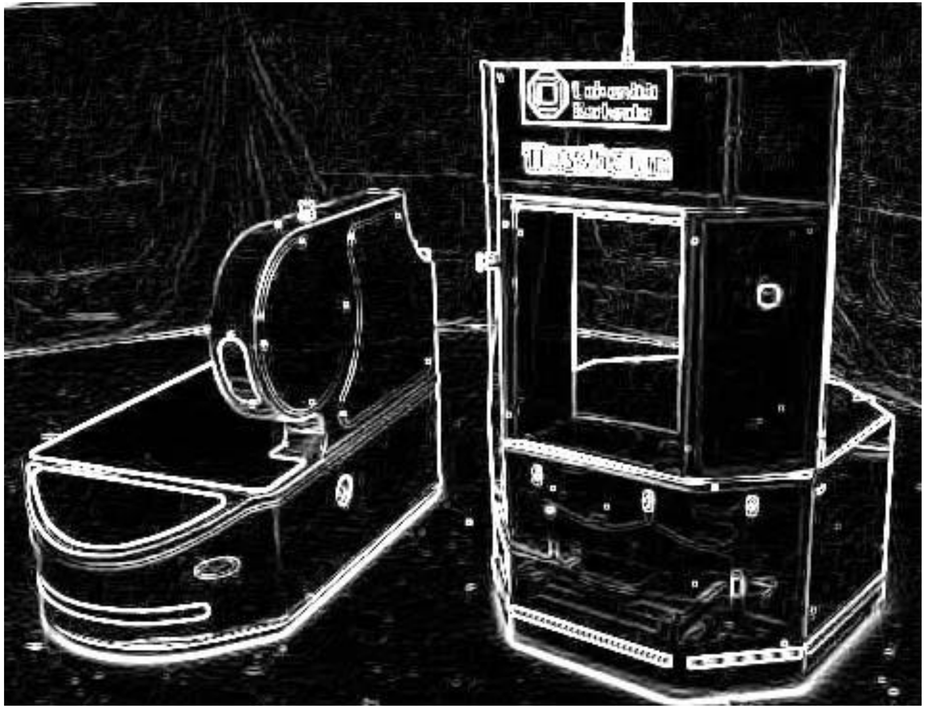
- **Features**:
  - Good results when detecting vertical (prewitt-x) or horizontal (prewitt-y) edges

# Filter – Prewitt (3)

- Combination of the Prewitt filters to determine the gradient amount $M$

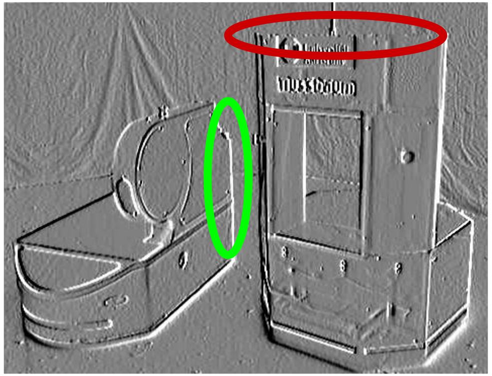
$$M \approx \sqrt{P_x^2 + P_y^2}$$

- Afterwards: Threshold filtering



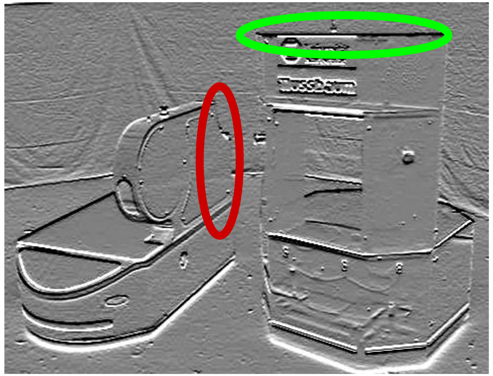
# Filter – Sobel

- Similar to Prewitt, but gives more weight to the center
- Suppresses noise; but retains the edges
- Sobel filter is the combination of a Gaussian filter with differentiation and difference quotients.



- **Sobel-X Filter**
  Approximated by

$$s_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

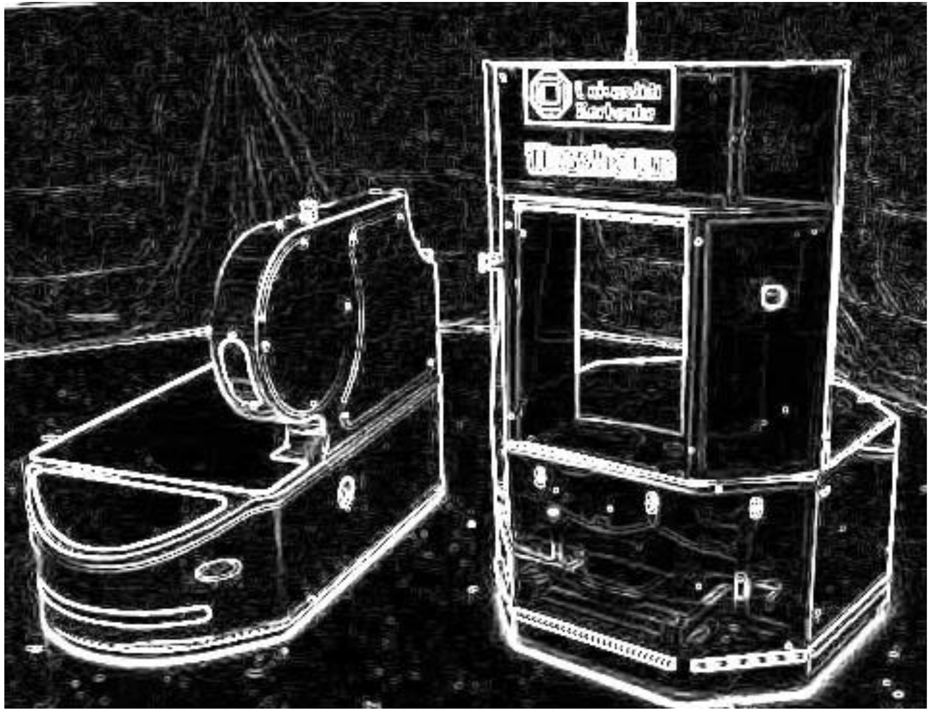


- **Sobel-Y Filter**
  Approximated by

$$s_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

# Filter – Sobel (2)

- Similar to Prewitt
- Combination of the Sobel filters to determine the gradient amount $M$
- Afterwards: Threshold filtering



# Filter – Laplace (1)

- The Laplace operator is a linear and **rotation invariant** second-order operator.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \begin{cases} \dfrac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \\ \dfrac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \end{cases}$$

$$\Delta \ \widehat{=} \ \nabla^2$$

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

| | | |
|---|---|---|
| $f(x-1,y-1)$ | $f(x-1,y)$ | $f(x-1,y+1)$ |
| $f(x,y-1)$ | $f(x,y)$ | $f(x,y+1)$ |
| $f(x+1,y-1)$ | $f(x+1,y)$ | $f(x+1,y+1)$ |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | −4 | 1 |
| 0 | 1 | 0 |

**Filter Mask**
**(90° rotation invariant)**

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |

**Extended Mask**
**(45° rotation invariant)**

The sum of the mask coefficients is zero, indicating that the response in the areas of constant intensity is zero.

# Filter – Laplace (2)

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

$$\begin{cases} \dfrac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \\ \dfrac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \end{cases}$$

| | | |
|---|---|---|
| $f(x-1,y-1)$ | $f(x-1,y)$ | $f(x-1,y+1)$ |
| $f(x,y-1)$ | $f(x,y)$ | $f(x,y+1)$ |
| $f(x+1,y-1)$ | $f(x+1,y)$ | $f(x+1,y+1)$ |

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} &= f(x+1,y) - f(x,y) - (f(x,y) - f(x-1,y)) \\ &= f(x+1,y) + f(x-1,y) - 2f(x,y) \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 f}{\partial y^2} &= f(x,y+1) - f(x,y) - (f(x,y) - f(x,y-1)) \\ &= f(x,y+1) + f(x,y-1) - 2f(x,y) \end{aligned}$$

# Filter – Laplace (3)

## Laplace-Operator:

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

$$\nabla^2 \approx \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Zero crossings define edges

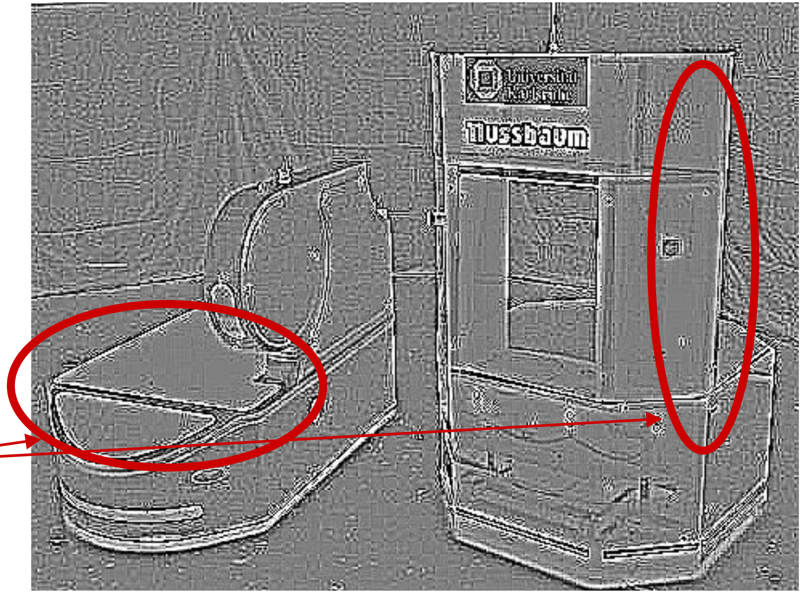
The edges are thinner than with Prewitt or Sobel.

# Filter – Laplace (4)

Variation of the **Laplace-Operator**:

$$\nabla^2 \approx \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- Stronger, but more unwanted edges
- Noise-sensitive, therefore smoothing before applying the Laplace filter so that artefacts are not recognized as edges



# Filter Operations

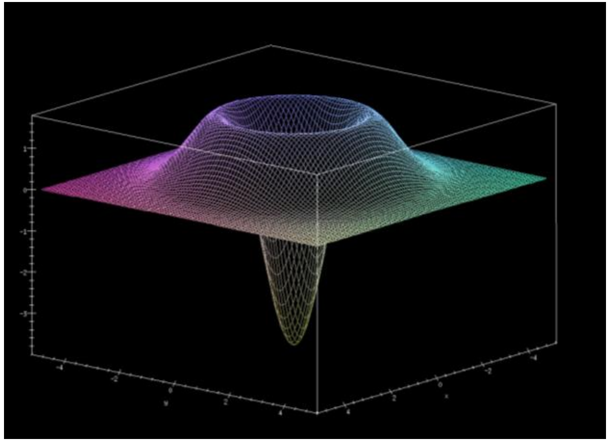
- **Low-pass filter:** smoothing, noise elimination
  - Median filter
  - Mean filter
  - Gaussian filter

- **High-pass filter:** Edge detection
  - Prewitt
  - Sobel
  - Laplace

- **Combined** operators
  - Laplacian of Gaussian

# Filter – Laplacian of Gaussian (LoG)

- The Laplacian operator is very sensitive to noise
- Significantly better results are achieved by smoothing the image with a Gaussian filter and then using the Laplacian of Gaussian (LoG) operator:



$$LoG\big(f(x,y)\big) = \nabla^2\big(f(x,y) * g(x,y)\big) = \Delta\,\big(f(x,y) * g(x,y)\big) \qquad \Delta \;\widehat{=}\; \nabla^2$$

$g$ denotes the filter function of a Gaussian filter.

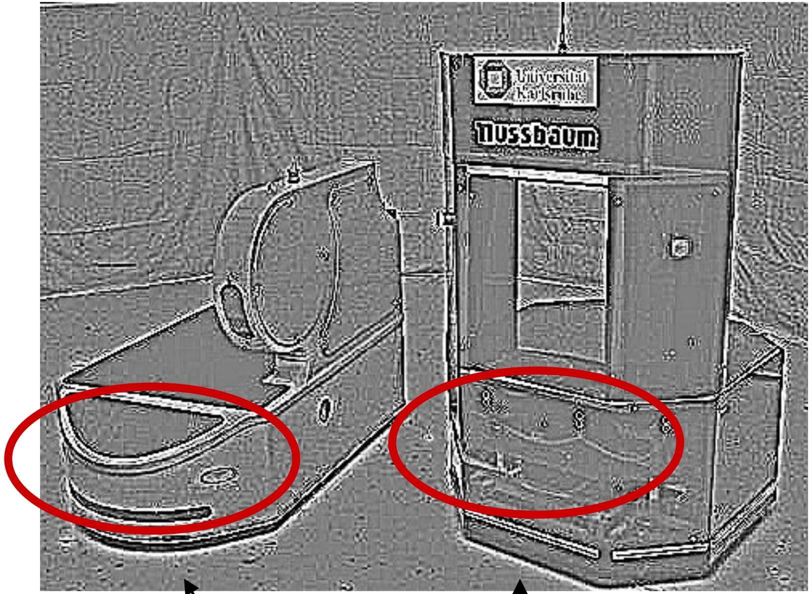
# Main Features of the LoG Operator

- The **Gaussian part** of the operator is a **low-pass filter** that smoothes (blurs) the image and thus reduces the intensity of structures (e.g. noise) to scales much smaller than $\sigma$.
- In contrast to the mean filter, the Gaussian filter is unlikely to produce artifacts such as “*staircase*”-effects that are not present in the original image.
- The **Laplace operator** (the second derivative) is rotation-invariant and thus responds equally to intensity changes **in any mask direction**. This way, we can avoid using multiple masks to calculate the strongest response at each point of the image.
- Zero crossings of the Laplacian correspond to the edges.

# Filter - Laplacian of Gaussian (LoG) (2)

- Approximation of LoG by convolution with the matrix

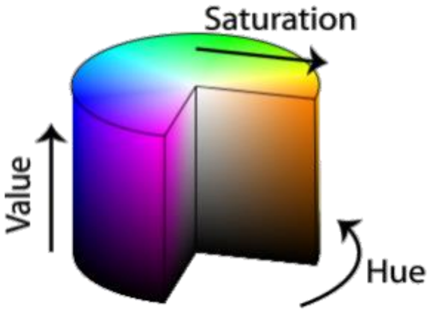
$$\Delta F(x,y) = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

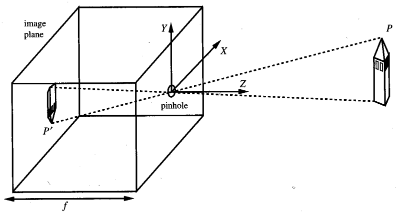


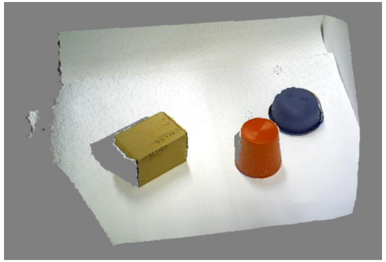

Stronger edges,
Less noise

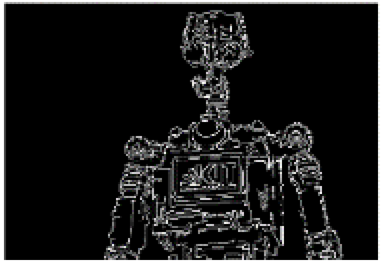
# Content

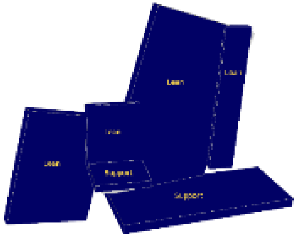
- Image Generation
- Operations on Images
  - Filter Operations
  - **Morphological Operators**
- Feature Extraction and Pattern Recognition



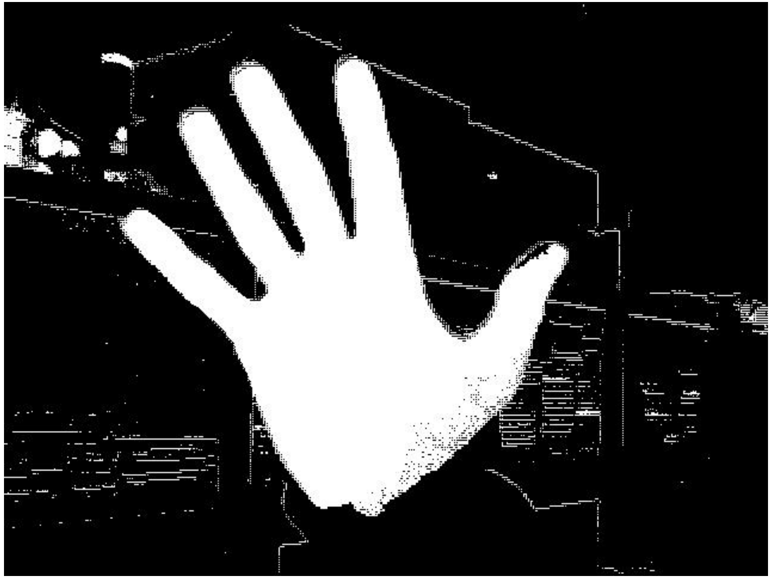




# Motivation: Color Segmentation

- Segmentation problem: artifacts, incomplete segmentation

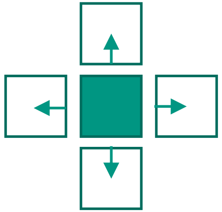
Input Image



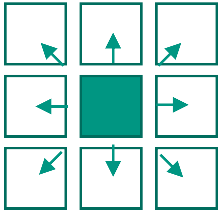
Resulting Segmentation

# Morphological Operators

- Morphological operators are often used for post-processing of binary images (e.g. result of color segmentation)
- Common morphological operators:
  - **Dilation:** Dilation enlarges pixels into larger areas; connects structures; **OR operator**
  - **Erosion:** Erosion removes isolated pixels and weakly connected pixel groups; dissolves structures; **AND operator**
- The effect of a morphological operator depends on the size and shape of the pixel neighborhood **(structural element)**.



4-neighborhood



8-neighborhood

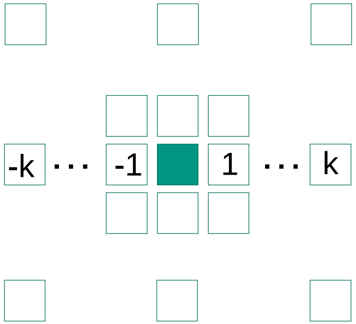
# Morphological Operators – Dilatation (OR)

Karlsruhe Institute of Technology

**Algorithmus 19** Dilatation$(I, n) \rightarrow I'$

$k := \mathrm{div}((n-1), 2)$
**for all** pixels $(u, v)$ in $I'$ **do**
    $I'(u, v) := 0$
**end for**
**for** $v := k$ **to** $h - k - 1$ **do**
    **for** $u := k$ **to** $w - k - 1$ **do**
        **if** $I(u, v) = q$ **then**
            **for** $i := -k$ **to** k **do**
                **for** $j := -k$ **to** k **do**
                    $I'(u + j, v + i) = q$
                **end for**
            **end for**
        **end if**
    **end for**
**end for**

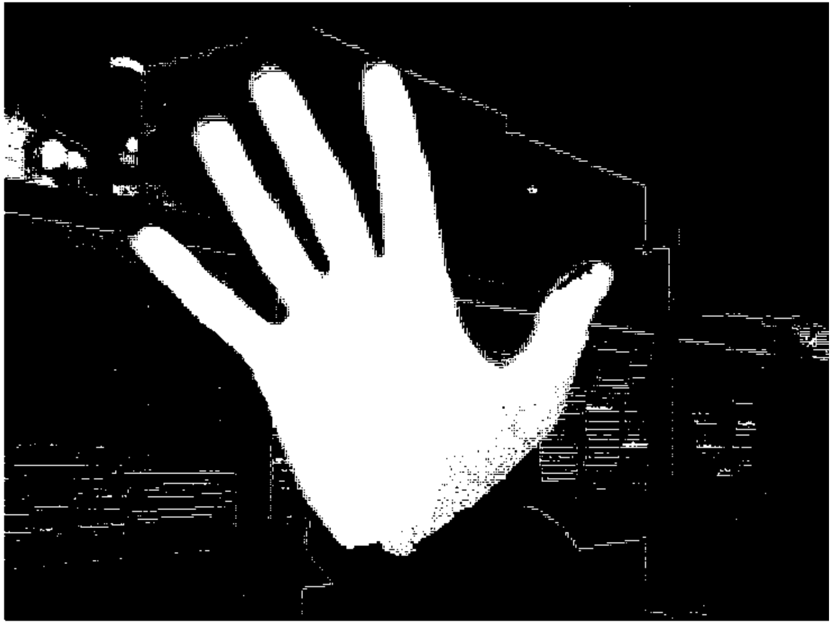
Image height $h$,
Image width $w$




**OR between structural element and image**

# Example: Dilatation

- Connects structures; OR



Input image

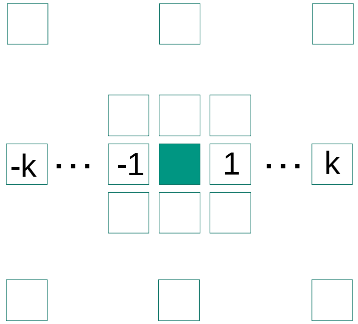
Output image

# Morphological Operators – Erosion (AND)

**Algorithmus 20** Erosion$(I, n) \rightarrow I'$

```
k := div((n − 1), 2)
for all pixels (u, v) in I' do
  I'(u, v) := 0
end for
for v := k to h − k − 1 do
  for u := k to w − k − 1 do
    if I(u, v) = q then
      for i := −k to k do
        for j := −k to k do
          if I(u + j, v + i) ≠ q then
            goto NEXT
          end if
        end for
      end for
      I'(u, v) = q
      NEXT:
    end if
  end for
end for
```

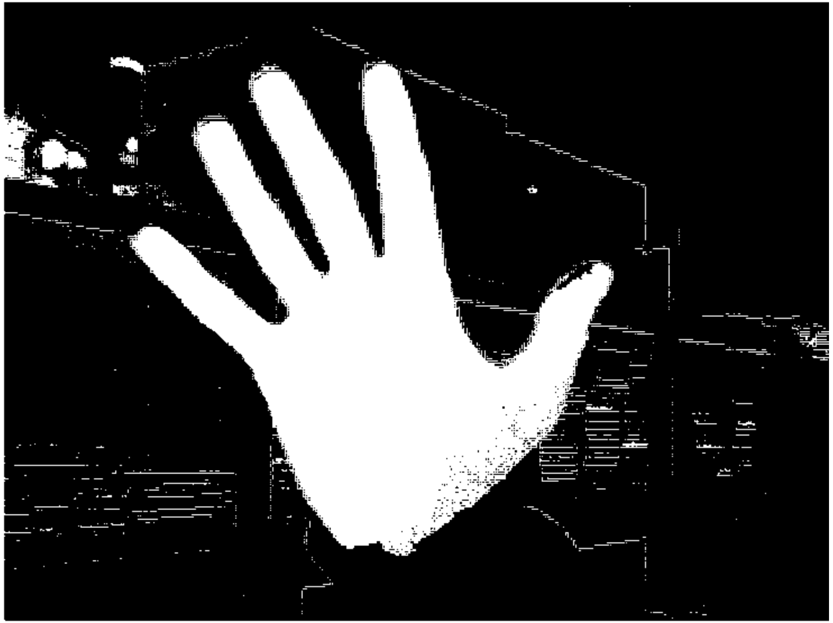
Image height $h$,
Image width $w$



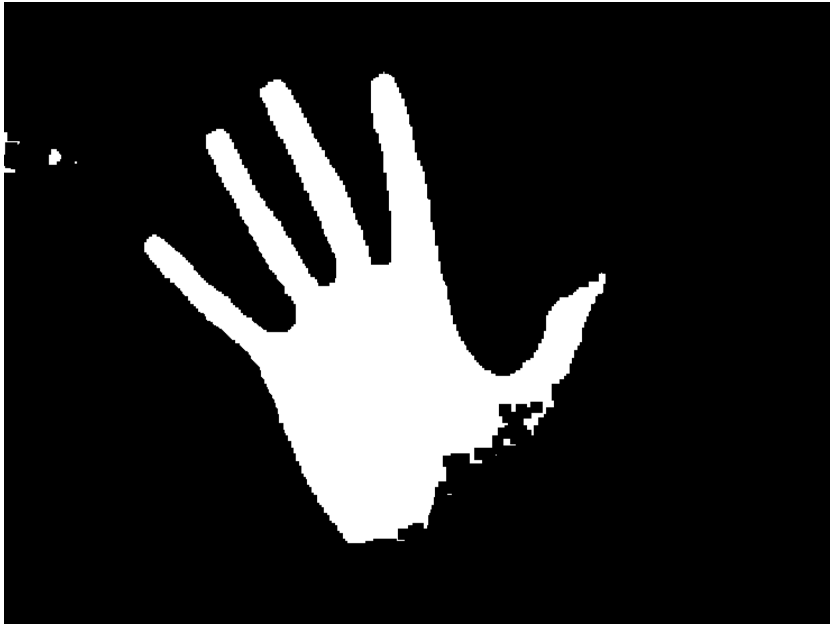

**AND between structure element and image**

# Example Erosion

- Dissolves structures; AND



Input image



Output image

# Morphological Operators – Opening & Closing

**Opening**:

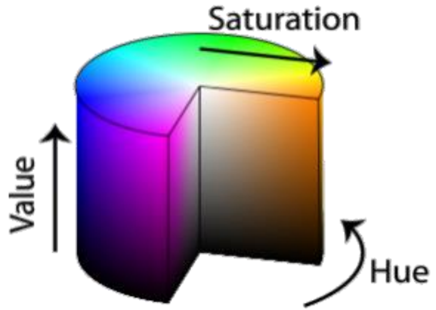
- Application of erosion, then dilation
- Removes thin lines or small external areas

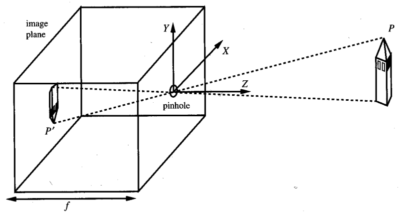
**Closing**:

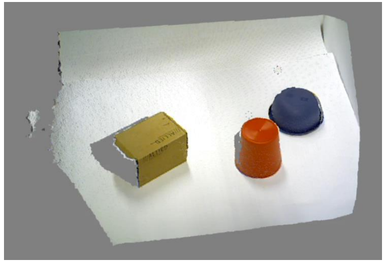
- Application of dilation, then erosion
- Bridging small distances and closing inner holes

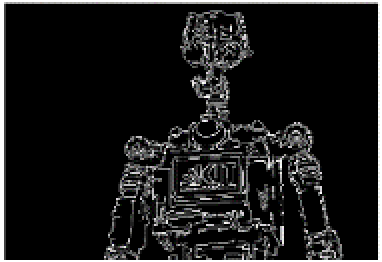
# Content

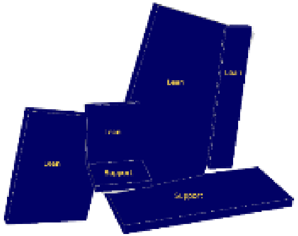
- Image Generation
- Operations on Images
- Feature Extraction and Pattern Recognition
  - **Segmentation**
  - Canny Edge Detection
  - Visual Servoing
  - Registration of Point Clouds
  - Example applications H²T



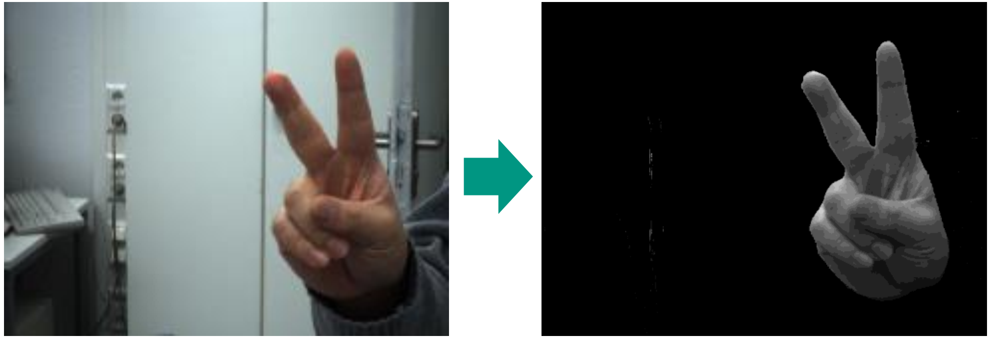




# Segmentation

- Segmentation is the division of a set into meaningful segments
  - Possible sets: images, point clouds, shapes, ...
  - Each pixel is assigned to at least one segment
  - Identification of interesting image regions for analysis, recognition and classification

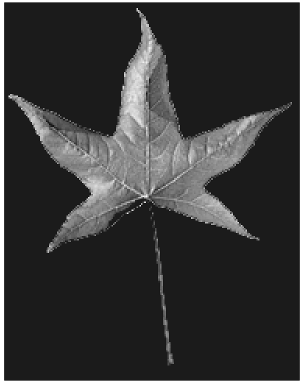
- Possible methods:
  - Threshold filtering
  - Clustering
  - Edge extraction
  - Region growing

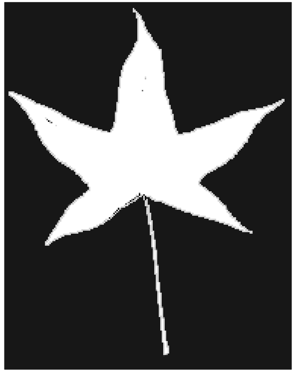


# Segmentation – Threshold Filtering

- Threshold filtering to convert a grayscale image into a binary image
- Intensity of each pixel $(u,v)$ is compared to a predefined threshold $T$

$$Img'(u,v) = \begin{cases} 255, & \text{if } Img(u,v) > T \\ 0, & \text{else} \end{cases}$$





# Segmentation – Threshold Filtering (2)

- Objects can often be segmented by their **color** (skin color, single-colored objects, ..)

- **Example:**
  - Interval bounds in the $HSV$ color space:

$$Img'(u,v) = \begin{cases} 255, & \text{if} \quad \begin{array}{l} H_{max} \geq Img_H(u,v) \geq H_{min}, \\ S_{max} \geq Img_S(u,v) \geq S_{min}, \\ V_{max} \geq Img_V(u,v) \geq V_{min} \end{array} \\ 0, & \text{else} \end{cases}$$

- **Problem:**
  - Changing lighting conditions
  - Reflections, shadows

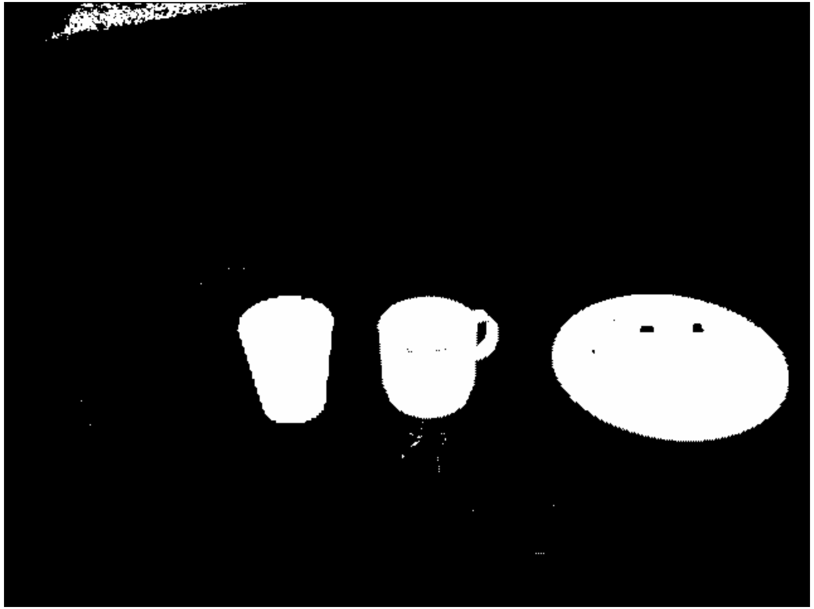
# Problems of Lighting Conditions

Is this dress white & gold or blue & black?

# Segmentation – Example Application

- Example application: object detection and localization

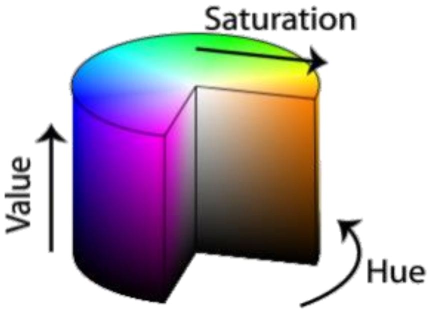
Input image

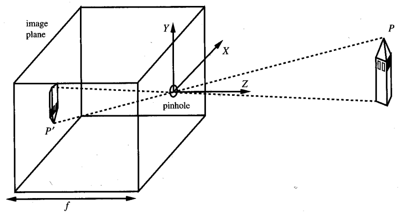


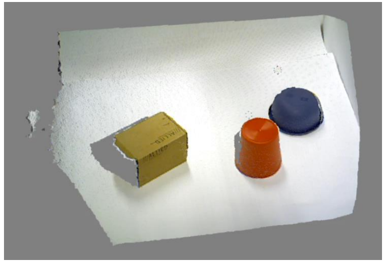
Output image

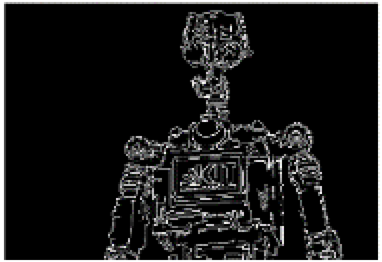
# Content

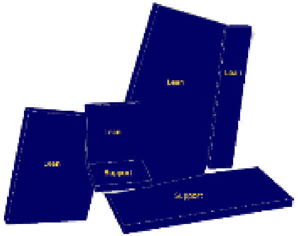
- Image Generation
- Operations on Images
- Feature Extraction and Pattern Recognition
  - Segmentation
  - **Canny Edge Detection**
  - Visual Servoing
  - Registration of Point Clouds
  - Example applications H²T







# Canny Edge Detector

- According to John F. Canny from 1986
- The Canny edge detector is widely used and superior to other edge detectors in terms of performance.
- The goal was to find the "**optimal**" edge detector:
  - Good detection
  - Good localization
  - Minimal response ("thin lines")
- Canny edge detector calculates binary response (usually 0: no edge, 255: edge)
- Subpixel accuracy possible through extension

# Canny Edge Detector (2)

The principle is based on three principles:

1. **Low error rate:** All edges should be found and detected edges should be as close as possible to the real edges.

2. **Edge points should be well detected:** The distance between detected edge points and the center of the real edges should be minimal.

3. **Uniqueness:** The detector should only provide one point, not several edge points, for a real edge point.

# Canny Edge Detector – Algorithm

1. Gaussian filter
   → noise suppression

2. Calculation of intensity gradients

3. Non-maximum suppression
   → suppression of non-maximum “edge pixels”

4. Double threshold
   → determination of possible edges

5. Edge tracking with hysteresis
   → removal of weakly connected edge pixels

# Canny Edge Detector – Intensity Gradients

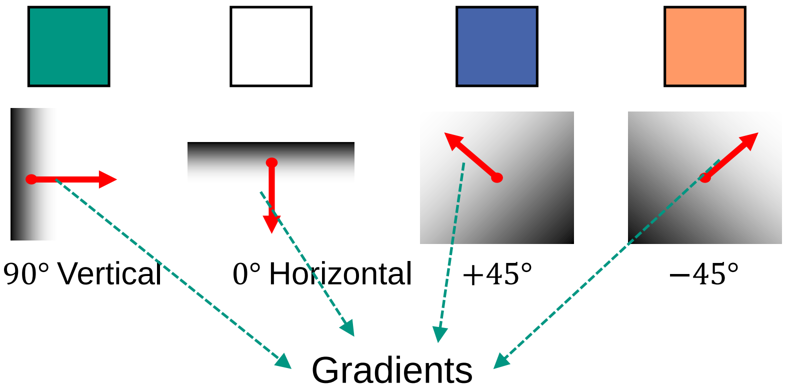
- Calculation of gradients in horizontal and vertical direction using Prewitt or Sobel filters

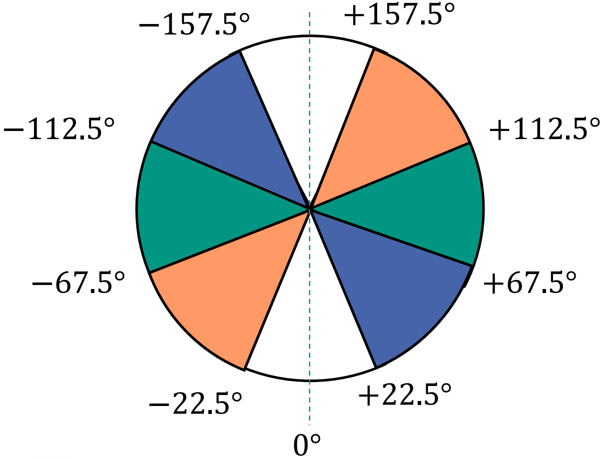
$$g_x = f * s_x, \qquad g_y = f * s_y$$

- Determination of the orientation $\theta$ and the magnitude $M$ of the gradient

$$\theta = \operatorname{atan2}(g_y, g_x), \qquad \theta \in (-180°, 180°], \qquad M = \sqrt{g_x^2 + g_y^2}$$

**Discretization** of the angle $\theta$: Division into **four areas**

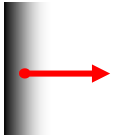


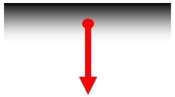
# Canny Edge Detector – Non-Maximum Suppression

**Non-Maximum Suppression:** i.e. edge thinning. All pixels that do not represent a maximum are suppressed.

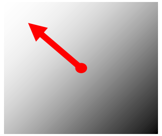
- Gradient must be a local maximum
- Consideration of the two direct neighbors along the gradient direction
- Checking is carried out according to the discrete gradient direction
  → 3x3 filter determines neighboring pixels for comparison



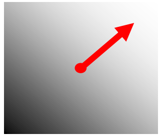
Vertical



Horizontal



$+45°$



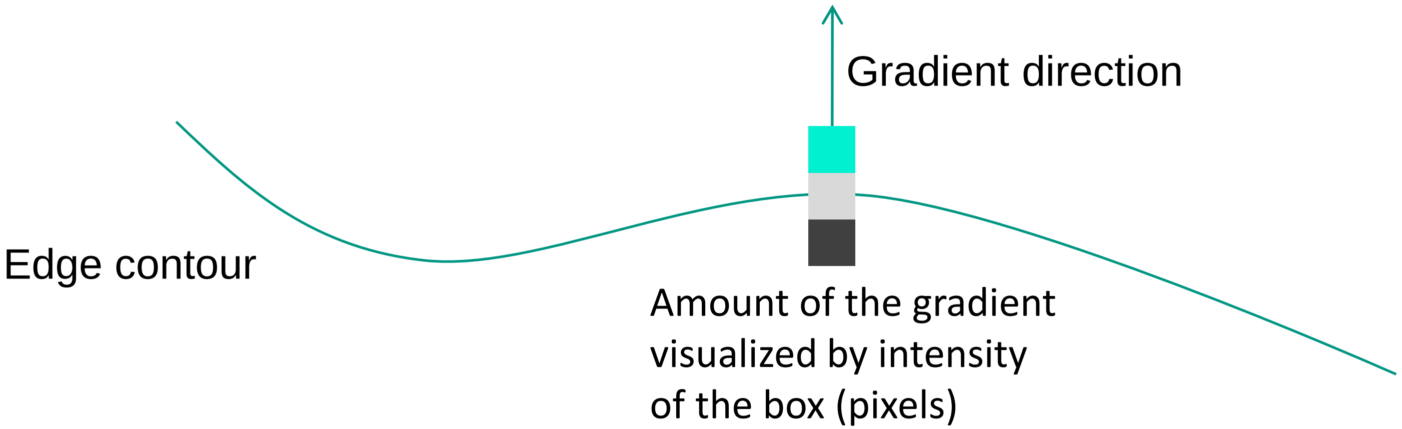
$-45°$

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

# Canny Edge Detector – Non-Maximum Suppression (2)

**Non-Maximum Suppression:** i.e. edge thinning. All pixels that do not represent a maximum are suppressed.

- Gradient must be a local maximum
- Consideration of the two direct neighbors along the gradient direction
- Checking is carried out according to the discrete gradient direction
  → 3x3 filter determines neighboring pixels for comparison



# Canny Edge Detector – Double Threshold

**Double Threshold** to remove weakly connected edge pixels

- Classification of pixels into **strong**, **weak** and **no edges**
- Two thresholds are defined

$$M_{weak}, \qquad M_{strong}$$

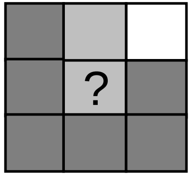
- Three intervals for the magnitude $M$ of the gradient

| | | |
|---|---|---|
| ■ (dark grey) | $0 \leq M \leq M_{weak}$ | No Edge |
| ■ (light grey) | $M_{weak} \leq M \leq M_{strong}$ | Weak Edge |
| □ (white) | $M > M_{strong}$ | Strong Edge |

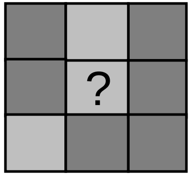
# Canny Edge Detector – Edge Tracking with Hysteresis

The Canny edge detector uses **edge tracking with hysteresis**

- Strong edge pixels are always preserved
  - $M > M_{strong}$: strong edge
- Which weak edge pixels are retained or discarded?
  - $M_{weak} \leq M \leq M_{strong}$: weak edge
- Consider the 8 neighboring pixels of each weak edge pixel
  - If there is a strong edge pixel in the neighborhood: Edge pixel
  - If there is no strong edge pixel in the neighborhood: No edge pixel




→ edge pixel

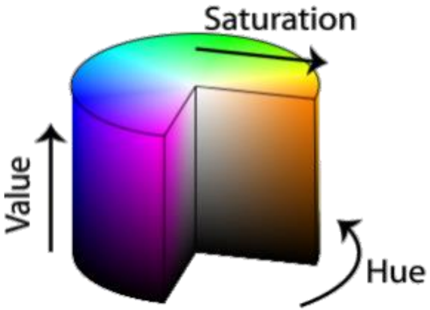


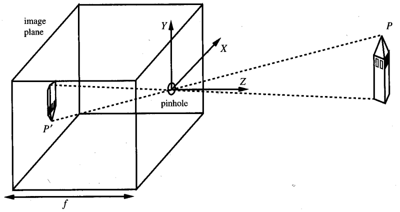

→ no edge pixel

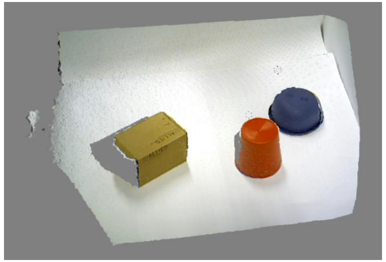
# Canny Edge Detector – Example

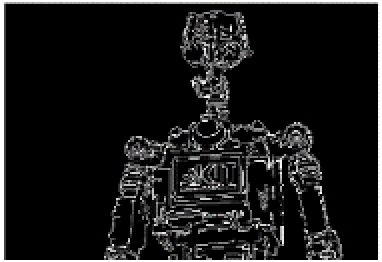

# Inhalt

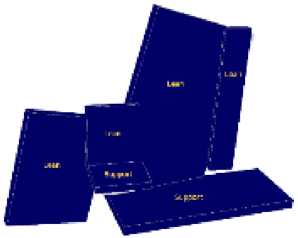
- Image Generation
- Operations on Images
- Feature Extraction and Pattern Recognition
  - Segmentation
  - Canny Edge Detection
  - **Visual Servoing**
  - Registration of Point Clouds
  - Example applications H²T



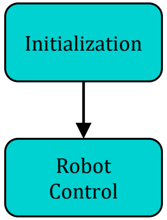




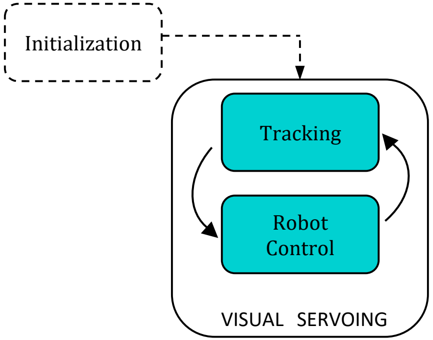
# Visual Servoing – Motiviation

- The term visual **servoing describes** methods in which visual input data is used to control the movement of a robot (**image-guided movement**).

- **Motivation**
  - Model errors
    e.g. kinematics
  - Execution errors
    e.g. incorrect positioning of the robot joints
  - Monitoring of the scene
    e.g. reaction to collisions




open-loop control

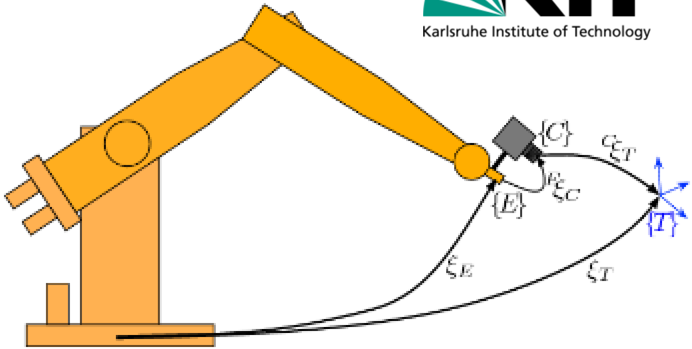



Visual Servoing

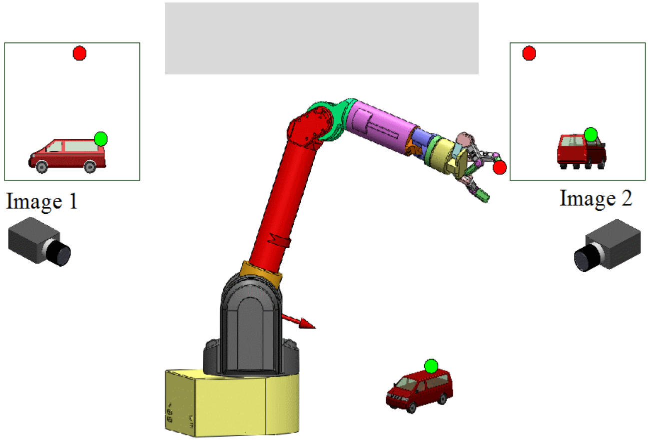
- François Chaumette, Seth Hutchinson, *Visual servo control - Part I - Basic approaches,* IEEE Robotics & Automation Magazine 13 (4), 82-90, 2006
- Danica Kragic and Henrik I Christensen, *Survey on Visual Servoing for Manipulation*, Techn. Report, KTH, 2002

# Visual Servoing

- **Eye-in-hand**
  - Camera is attached to the manipulator
  - Movements of the manipulator influence the pose of the camera




- **Eye-to-hand**
  - External camera system is used to observe the movement

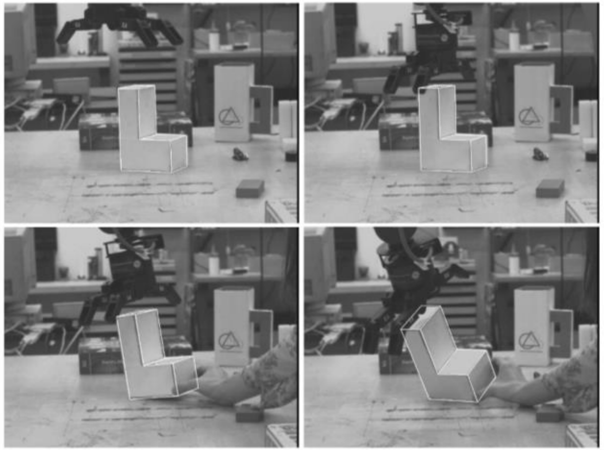


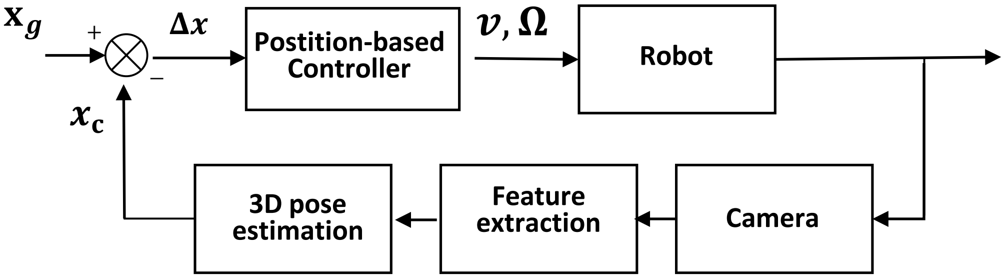
# Position-based Visual Servoing

- Target pose $\boldsymbol{x}_g$ is specified
- Control loop sequence
  - **3D position estimation:**
    Current hand pose $\boldsymbol{x}_c$ is extracted from image features
  - Calculate **control difference** (Cartesian):

    $$\Delta\boldsymbol{x} = \boldsymbol{x}_g - \boldsymbol{x}_c$$

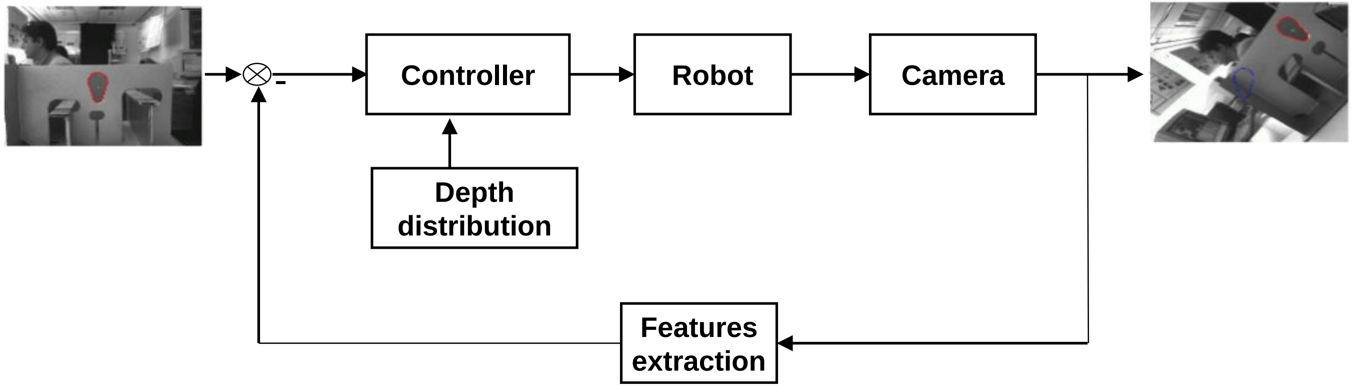
  - **Cartesian controller** to compensate for $\boldsymbol{\Delta x}$
  - Target is reached when distance $\boldsymbol{\Delta x}$ is less than threshold





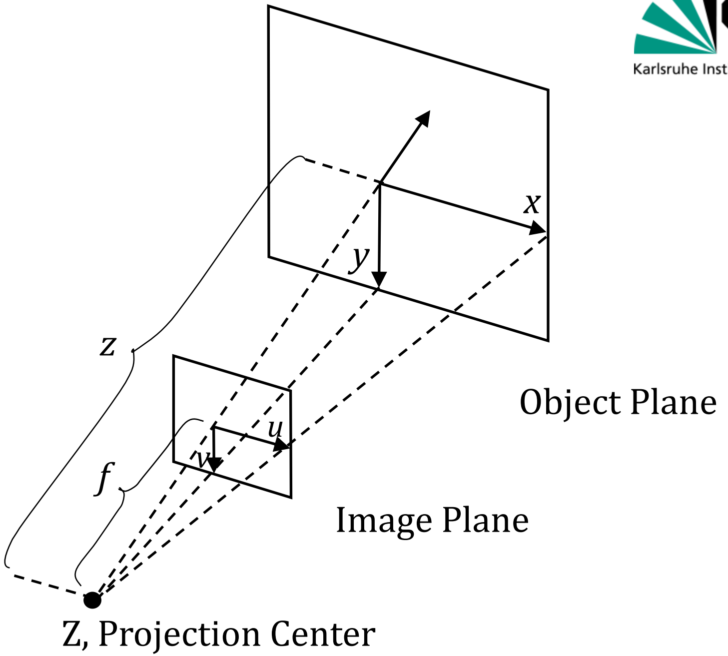
# Image Based Visual Servoing

- **Approach:** The movement of the robot (arm) results from the current and desired position of **image features**
- **Image features**: extracted using image processing methods
- **Control**: Velocity specifications are generated directly from the current and desired position of the image features



# Image Based Visual Servoing (2)

- **Image Feature:**
  - An image feature $s = (u, v)^T$ is the projection of a 3D point $\mathrm{p} = (x, y, z)^T$ in the camera image.




- **Error function**
  - Current position of the features in the camera image (time t): $s(t)$
  - Desired target position of the features in the camera image (constant): $s$*
  - Error: $e(t) = s(t) - s^*$

# Image Based Visual Servoing (3)

- **Interaction Matrix / Image Jacobian $(L)$** describes the relationship between the velocity of an image feature and the camera

$$\boxed{\dot{s} = L(u, v, z) \cdot \dot{p}}$$

$\dot{s}$ velocity of a pixel $(\dot{u}, \dot{v})$

$\dot{p}$ velocity of the camera $(v, \omega) = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$

$z$ depth of a point

$L \in \mathbb{R}^{2\times 6}$

# Image Based Visual Servoing (4)

- **Interaction Matrix / Image Jacobian** $(L)$ describes the relationship between the velocity of an image feature and the camera

$$\dot{s} = L(u, v, z) \cdot \dot{p}$$

- The projection rules (pinhole camera model) result in

$$L = \begin{pmatrix} \frac{f}{z} & 0 & -\frac{u}{z} & -\frac{u \cdot v}{f} & \frac{f^2 + u^2}{f} & -v \\ 0 & \frac{f}{z} & -\frac{v}{z} & -\frac{f^2 + v^2}{f} & \frac{uv}{f} & u \end{pmatrix}$$

# Image Based Visual Servoing (5)

- **Inversion of the interaction matrix**
  - Distance $z$ is estimated
  - Several (**at least 3**) features are observed
  - Assumption: Camera-in-hand system
    Movement of the 3D points corresponds to movement of the camera $v_C$
  - This results in:

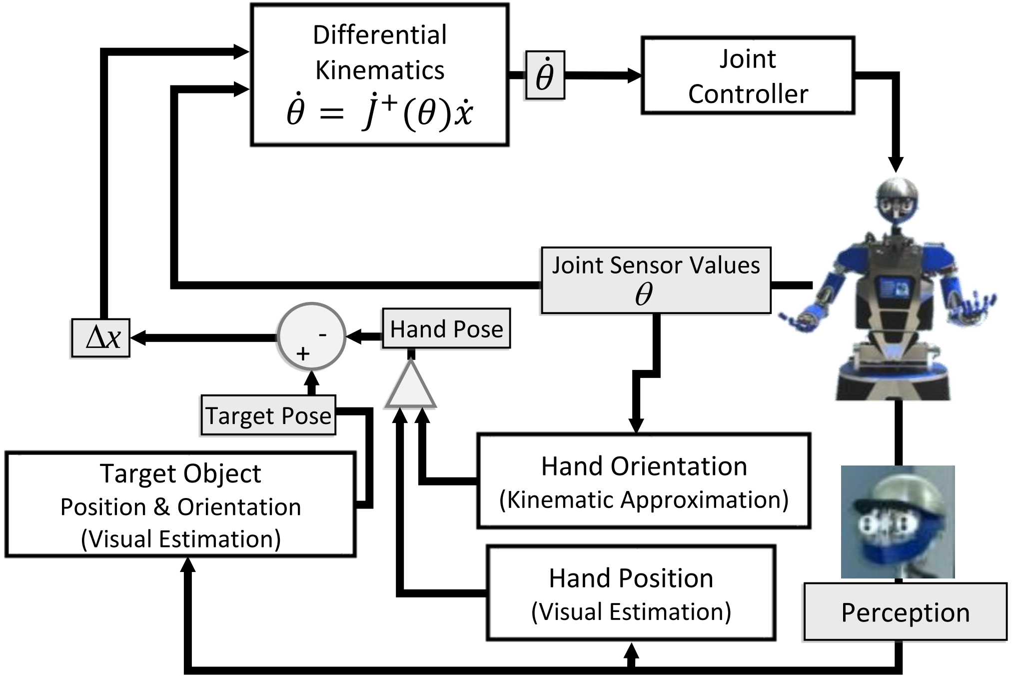
$$\dot{e} = L\, v_C$$

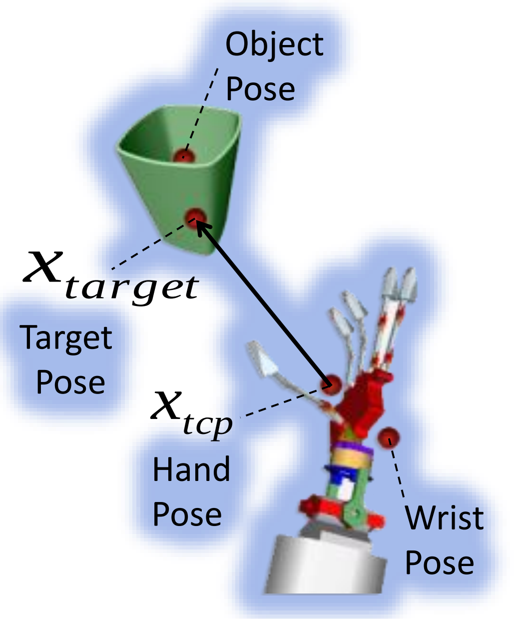
  - With $\dot{e} = -\lambda\, e$ (error should approach zero) we get

$-\lambda\, e = L\, v_C \quad \rightarrow \quad v_C = -\lambda\, L^{+}\, e$ $\qquad$ $L^{+}$ is the pseudo inverse of $L$

  - Control input $v_C$ is calculated from the position of the image features

# Position-based Visual Servoing on ARMAR-III



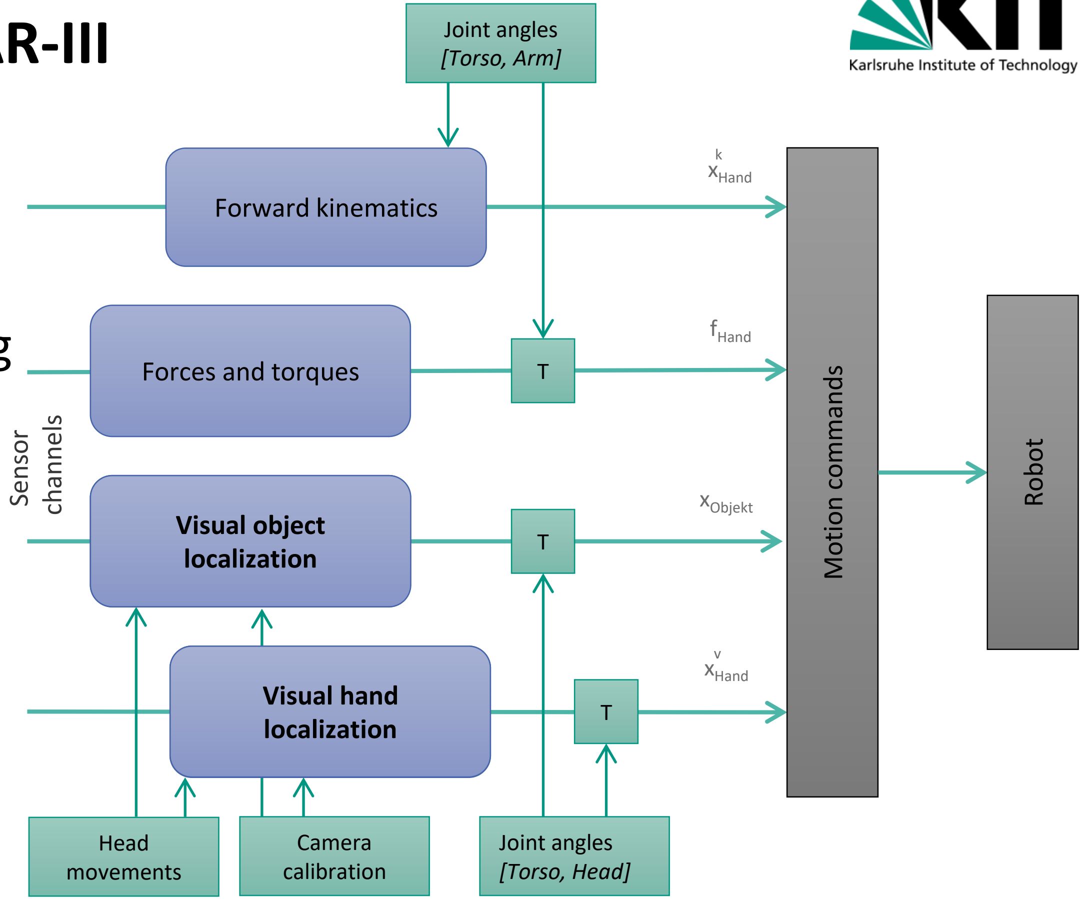

$$\dot{\theta} = J^{+}(\theta)\dot{x}$$

$$\delta^{t} = x^{t}_{vision} - x^{t}_{kinematic}$$

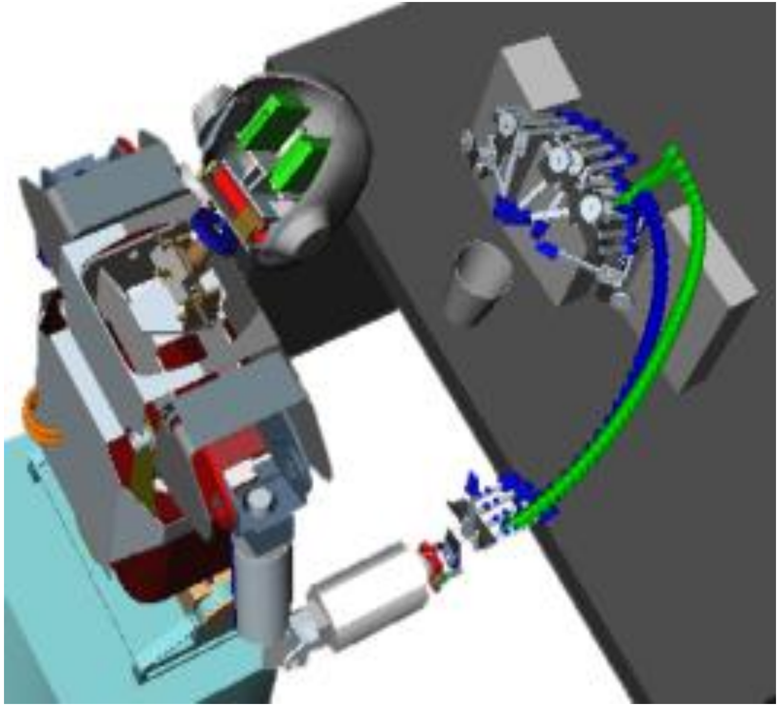
$$x^{t+1}_{tcp} = x^{t+1}_{kinematic} + \delta^{t}_{tcp}$$

# Visual Servoing for ARMAR-III

- Execution of grasping and manipulation tasks
- Position based visual servoing
  - Model knowledge
  - Object localization
  - Hand localization
- Sensors
  - Force/contact
  - Cameras
  - Internal sensors



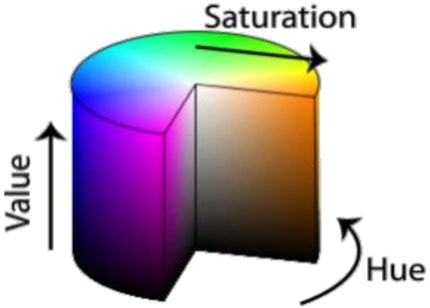
# Position-based Visual Servoing – ARMAR-III (1)

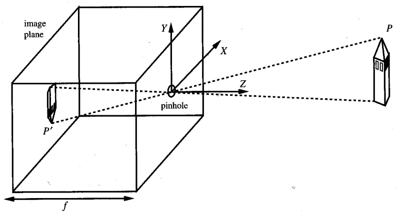


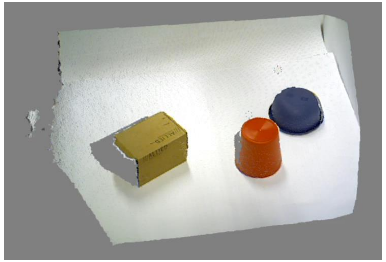
# Position-based Visual Servoing – ARMAR-III (2)

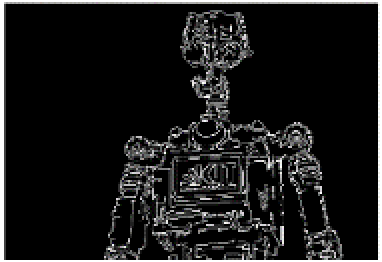
# Content

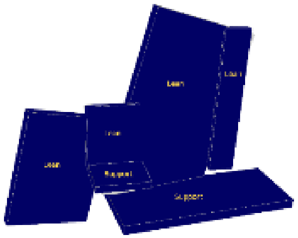
- Image Generation
- Operations on Images
- Feature Extraction and Pattern Recognition
  - Segmentation
  - Canny Edge Detection
  - Visual Servoing
  - **Registration of Point Clouds**
  - Example applications H²T



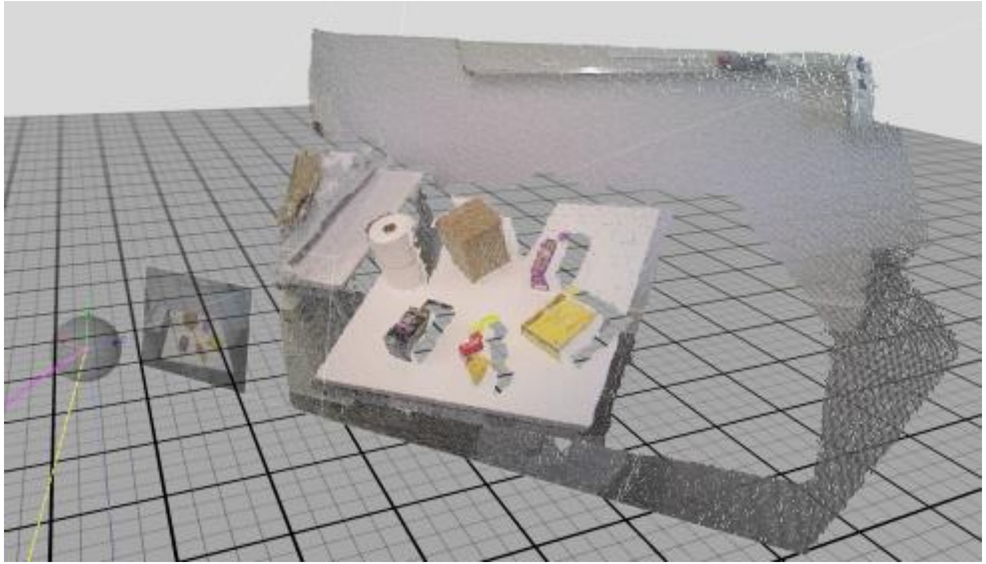




# Normal Estimation in 3D Point Clouds

- Goal:
  - Additional surface information by including local neighboring points

- Basis for further algorithms:
  - Segmentation
  - Descriptors
  - Object recognition
  - Surface modeling



# Normal Estimation in 3D Point Clouds (2)

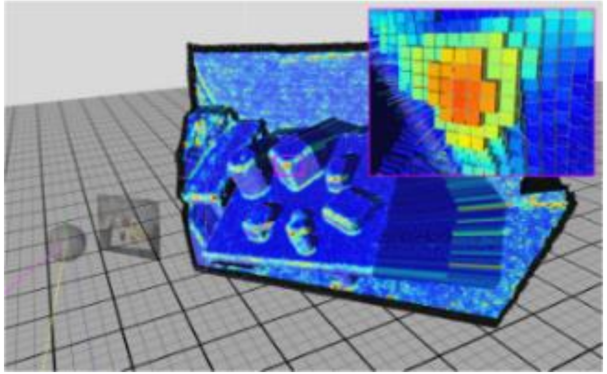
- PCA-based approach
- Create the covariance matrix $C$ of the $k$-point neighborhood for each point $p$

$$C = \frac{1}{k}\sum_{1}^{k}(p_i - \bar{p}) \cdot (p_i - \bar{p})^T$$

$p_i$: $k$ neighboring points

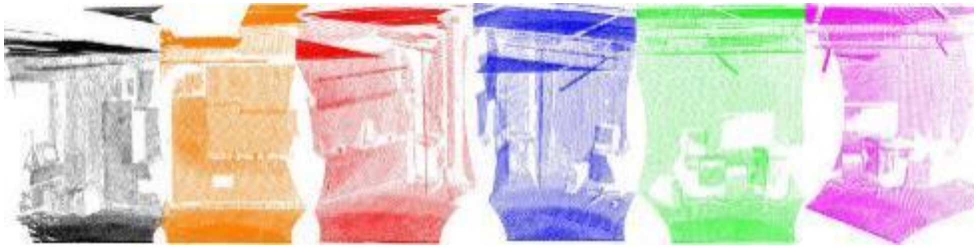
$\bar{p}$: center of all $k$ neighbors

- Determine the eigenvalues and eigenvectors of $C$
  - Principal Component Analysis (PCA)
  - **Eigenvector to smallest eigenvalue corresponds to the normal**

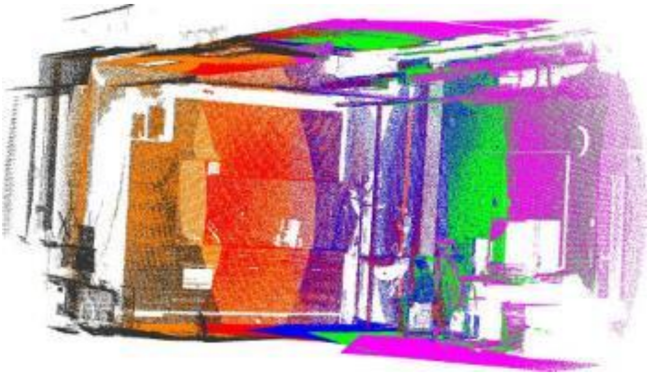


# Registration of Point Clouds

- **Registration:** Alignment of point clouds that describe the same object from different views
- Transfermation to a “higher-level” coordinate system (e.g. world coordinate system)
- Extrinsic calibration of the cameras required



Point clouds of a room from different perspectives



Registered point cloud

# Iterative Closest Point

- **Iterative Closest Point** (ICP) is a common algorithm for registering two sets $A$, $B$ with a priori unknown assignment (Besl and McKay, 1992)

- **Example:** Registration of two 3D point clouds
  - For each iteration k:
    - For each point $a_i$ from $A$, find point $b_i$ from $B$ that is closest to $a_i$
    - Calculate a transformation $T_k$ such that $D_k$ is minimal, e.g. with (Horn, 1987):

$$D_k = \sum_i \| a_i - T_k \cdot b_i \|^2$$

  - $D_k$ combines translation and rotation
  - Apply transformation $T_k$ to all points from $B$ (**update**)
  - Termination:
    - Threshold for the difference $(D_{k-1} - D_k)$ or
    - Maximum number of iterations reached

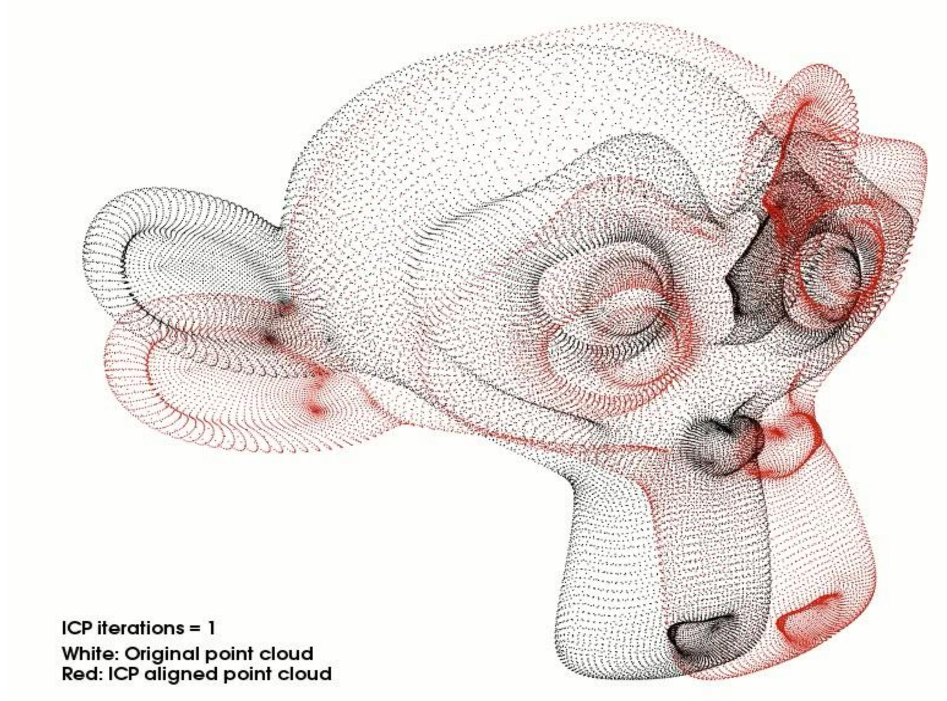
P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239 256, Feb. 1992, DOI: 10.1109/34.121791

Berthold K. P. Horn, Closed-Form Solution of Absolute Orientation Using Unit Quaternions, Journal of the Optical Society of America A 4(4):629-642; April 1987, DOI: 10.1364/JOSAA.4.000629

# Iterative Closest Point (2)

- Minimizes the "distance" between two point clouds
- Very well suited for reconstruction in 2D and 3D
- Advantages
  - Algorithm can be used for points, normal vectors and other forms of representation
  - Only simple mathematical operations necessary
  - Fast registration result
- Disadvantages
  - Overlapping of point clouds required
  - Symmetrical objects cannot be easily registered
  - Convergence to a local minimum is possible

# Iterative Closest Point – Visualization



# RANSAC – Random Sample Consensus

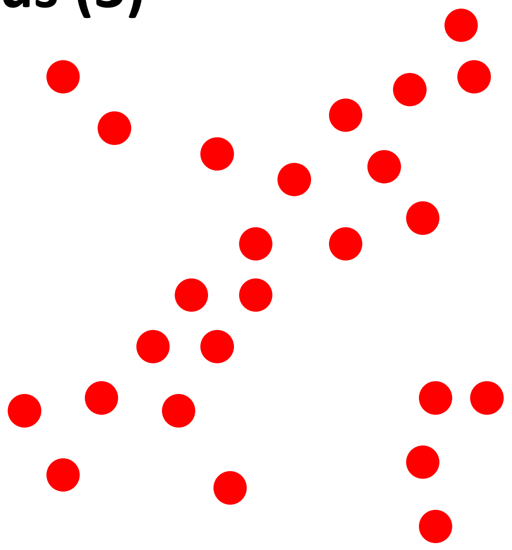
- **RANSAC** is an **iterative method** for estimating model parameters from data points
- RANSAC is a **non-deterministic** algorithm, i.e. it does not always produce the same result
- Robust against outliers and missing data points
- Application in image processing
  - Estimation of lines in 2D images
  - Estimation of planes and other primitives in 3D point clouds

# RANSAC – Random Sample Consensus (2)

- The RANSAC algorithm:
  1. Randomly select the minimum number of points needed to calculate the model parameters, i.e. **2 points** for lines in 2D **and 3 points** for planes in 3D
  2. Estimate a model from the selected dataset
  3. Evaluate the model estimation: Calculate the subset of data points (*inliers*) whose distance to the model is smaller than a predefined threshold
  4. Repeat 1–3 until the model with the most inliers is found

# RANSAC – Random Sample Consensus (3)

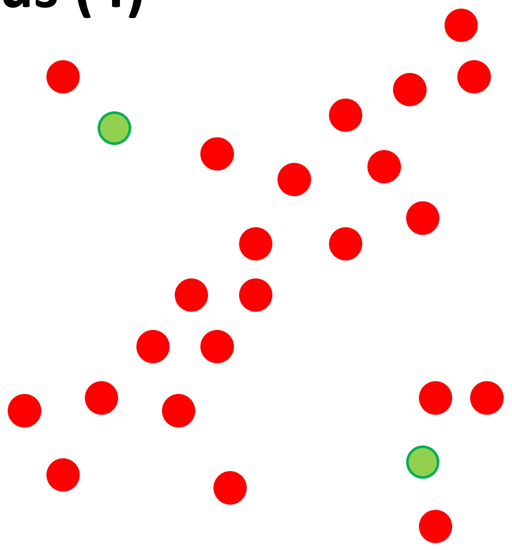
- Example:
  - Line fitting in 2D data points



- RANSAC Algorithm:
  1. Randomly select the minimum number of points needed to calculate the model parameters
  2. Estimate the model from the selected dataset
  3. Evaluate the model estimate
  4. Repeat 1–3 until the model with the most inliers is found

# RANSAC – Random Sample Consensus (4)

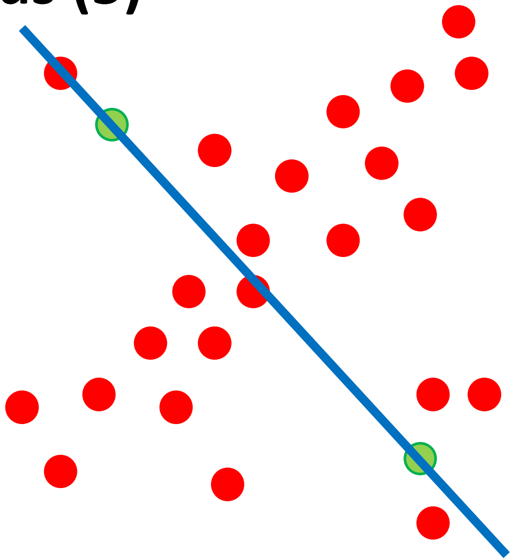
- Example:
  - Line fitting in 2D data points



- RANSAC Algorithm:
  1. Randomly select the minimum number of points needed to calculate the model parameters
  2. Estimate the model from the selected dataset
  3. Evaluate the model estimate
  4. Repeat 1–3 until the model with the most inliers is found

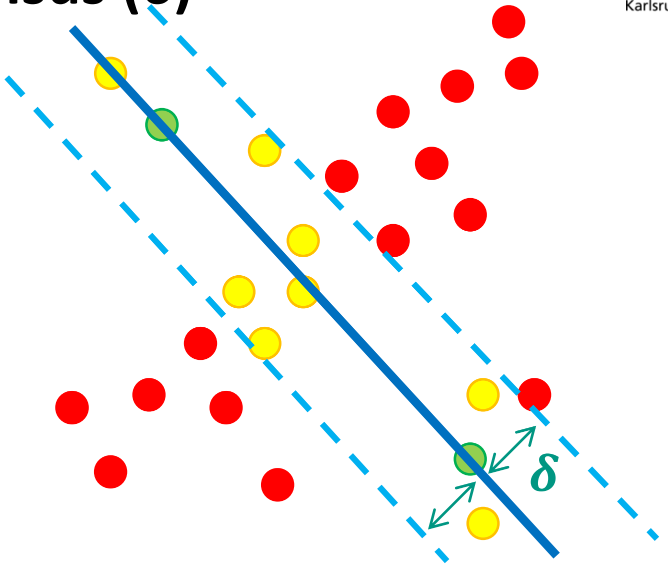
# RANSAC – Random Sample Consensus (5)

- Example:
  - Line fitting in 2D data points



- RANSAC Algorithm:
  1. Randomly select the minimum number of points needed to calculate the model parameters
  2. Estimate the model from the selected dataset
  3. Evaluate the model estimate
  4. Repeat 1–3 until the model with the most inliers is found

# RANSAC – Random Sample Consensus (6)




- Example:
  - Line fitting in 2D data points

    **Number of inliers = 10**

    **Number of outliers = 15**

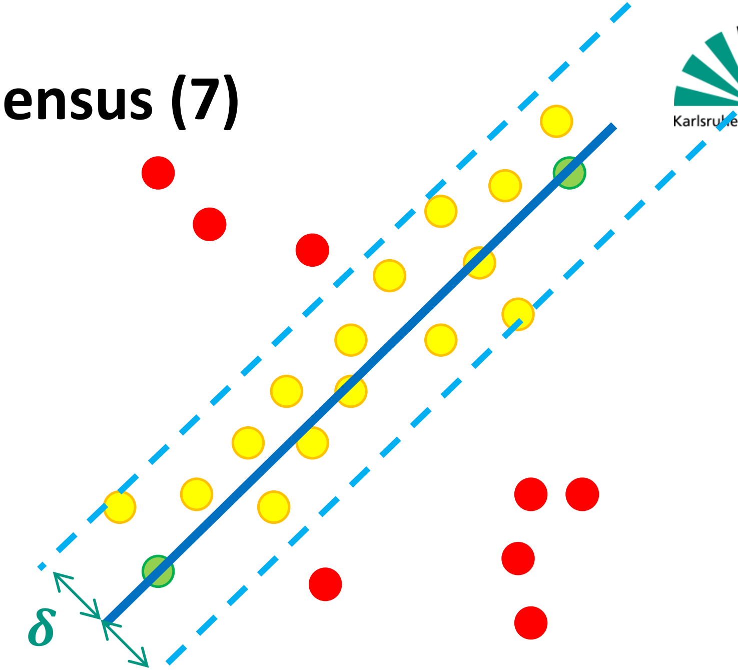
- RANSAC Algorithm:
  1. Randomly select the minimum number of points needed to calculate the model parameters
  2. Estimate the model from the selected dataset
  3. Evaluate the model estimate
  4. Repeat 1–3 until the model with the most inliers is found

# RANSAC – Random Sample Consensus (7)

- Example:
  - Line fitting in 2D data points

**Number of inliers = 10**

**Number of outliers = 15**




- RANSAC Algorithm:
  1. Randomly select the minimum number of points needed to calculate the model parameters
  2. Estimate the model from the selected dataset
  3. Evaluate the model estimate
  4. Repeat 1–3 until the model with the most inliers is found

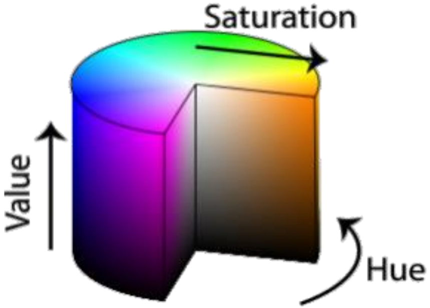
# RANSAC – Random Sample Consensus (5)

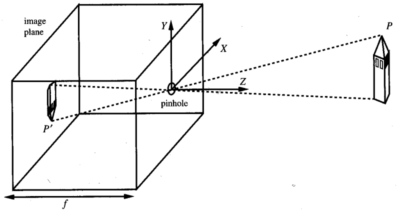
- Advantages:
  - General and easy to implement
  - Robust model estimation for data with few outliers
  - Versatile

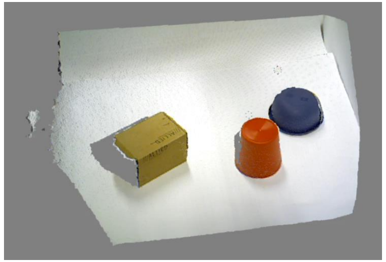
- Disadvantages:
  - Non-deterministic
  - Trade-off between accuracy and runtime (requires many iterations)
  - Not applicable if the ratio of inliers/outliers is too small

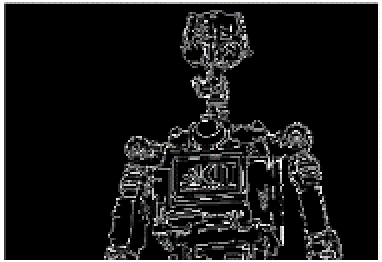
# Inhalt

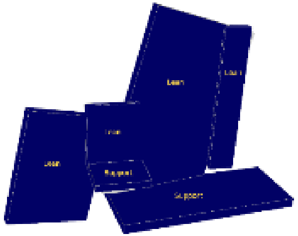
- Image Generation
- Operations on Images
- Feature Extraction and Pattern Recognition
  - Segmentation
  - Canny Edge Detection
  - Visual Servoing
  - Registration of Point Clouds
  - **Example applications H²T**



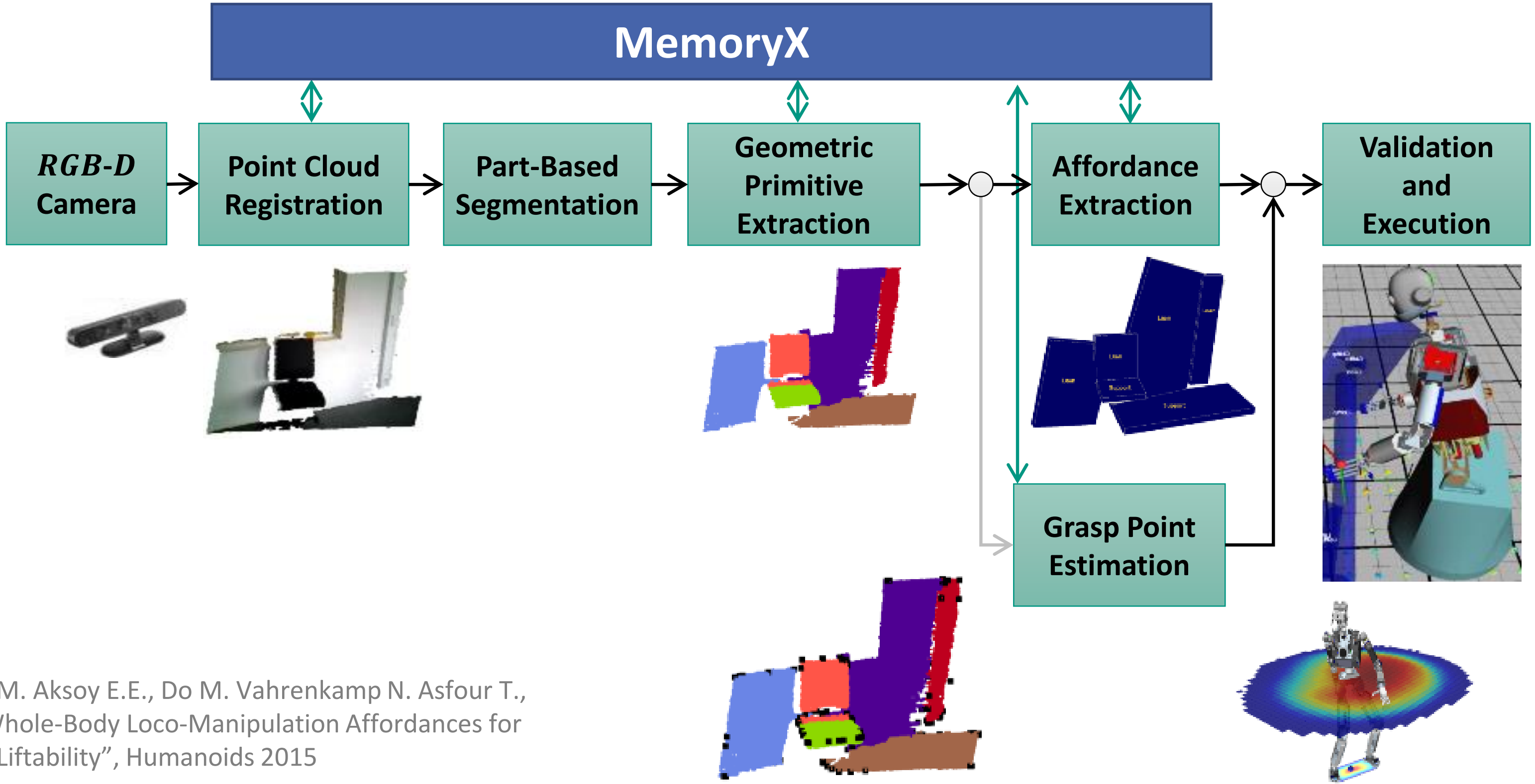





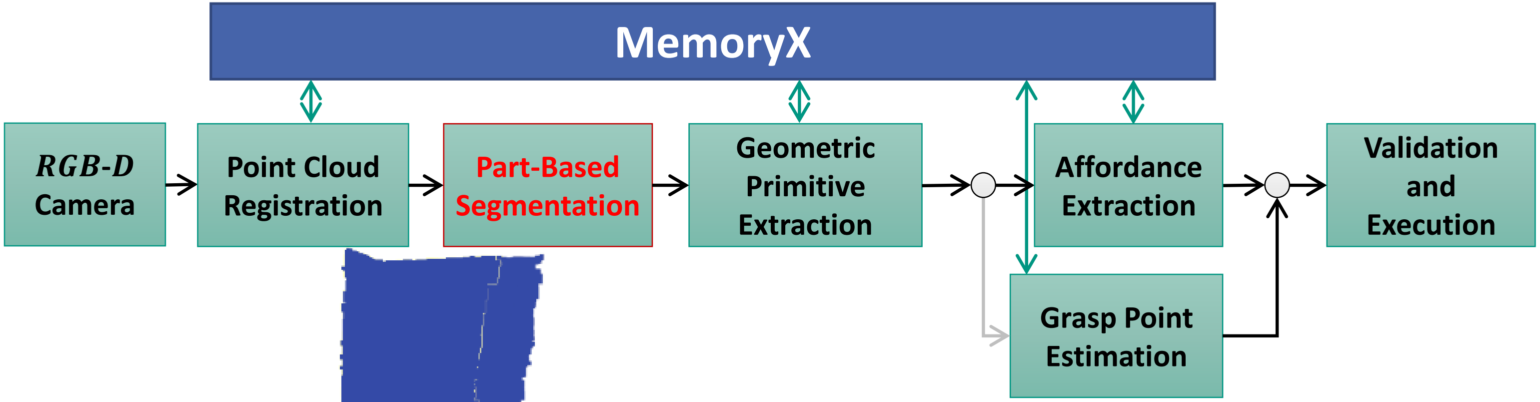


# Loco-Manipulation Affordances




Kaiser P., Grotz M. Aksoy E.E., Do M. Vahrenkamp N. Asfour T., "Validation of Whole-Body Loco-Manipulation Affordances for Pushability and Liftability", Humanoids 2015

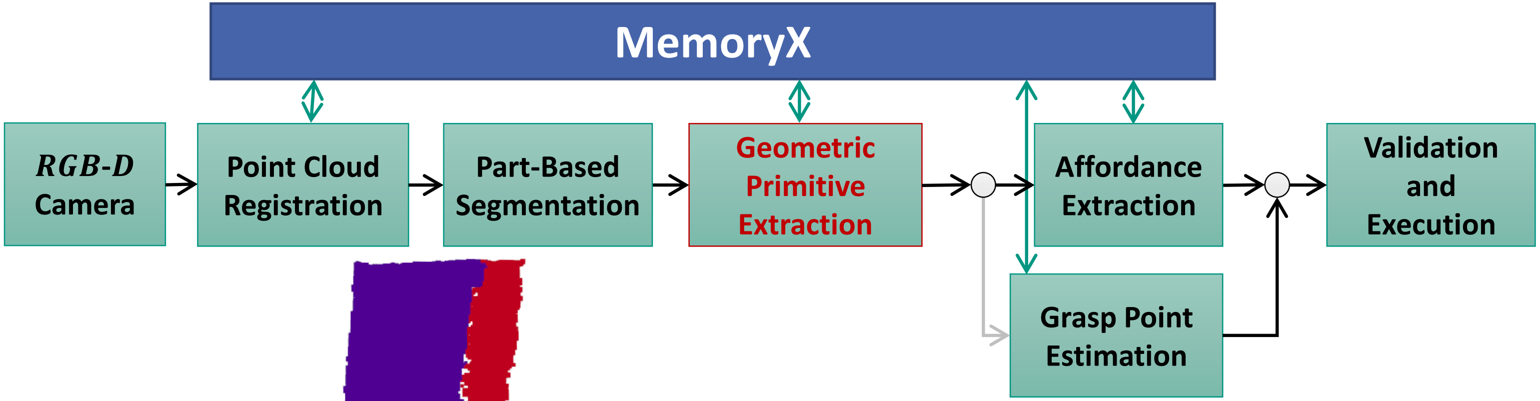
# Loco-Manipulation Affordances (2)




- Depth-based segmentation: required for scene decomposition
- We use Locally Convex Connected Patches (LCCP) for segmentation
- Advantage: Simple and fast
- Disadvantage: Under-segmentation (e.g. chair)

**Segmentation**: S. Stein, F. Wörgötter, M. Schoeler, J. Papon, and T. Kulvicius, "Convexity based object partitioning for robot applications," ICRA 2014.

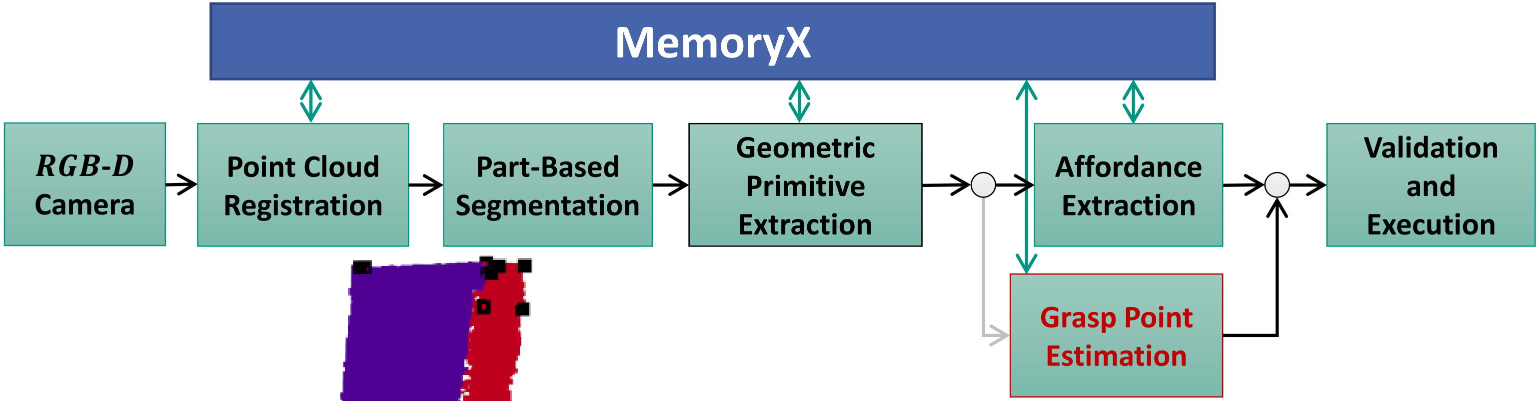
# Loco-Manipulation Affordances (3)

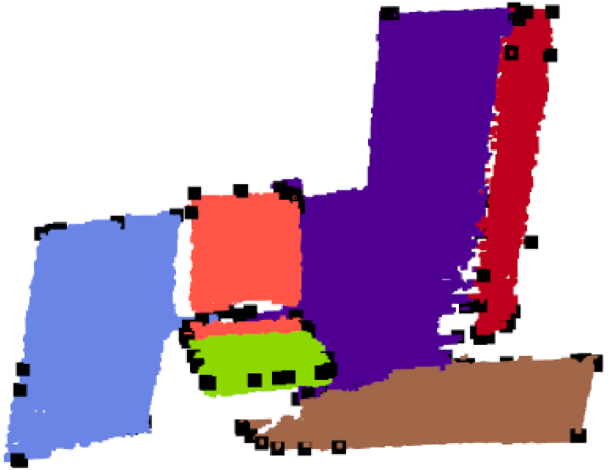



- Fitting geometric models (e.g. planes and cylinders) into segments with RANSAC
- Due to subsegmentation, models are iteratively fitted to the remaining outliers (e.g. chair)

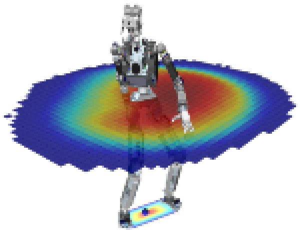
**PCL**: R. B. Rusu and S. Cousins, "3d is here: Point Cloud Library (PCL)," ICRA 2011

# Loco-Manipulation Affordances (4)

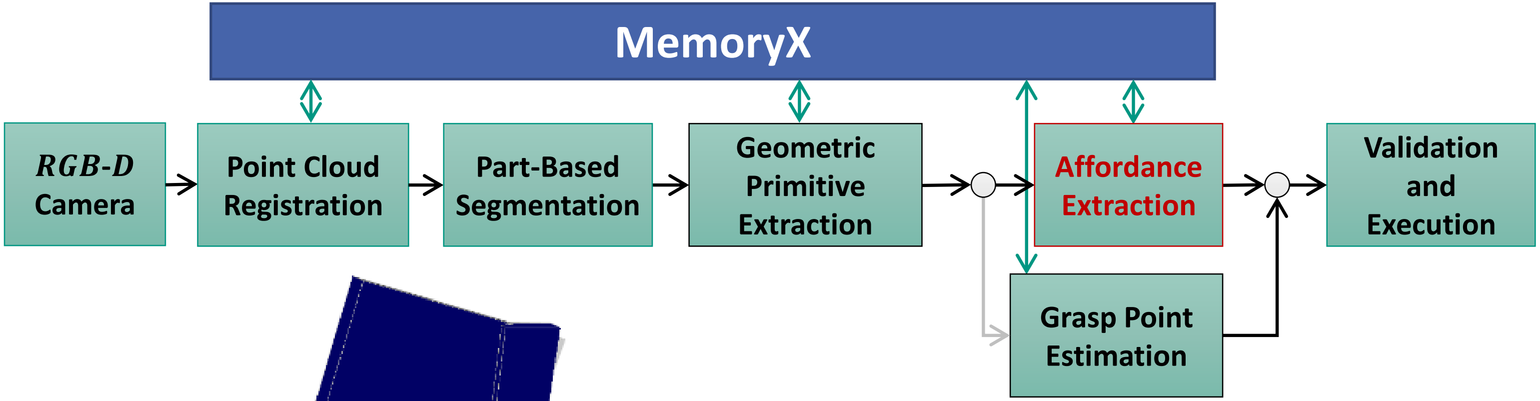


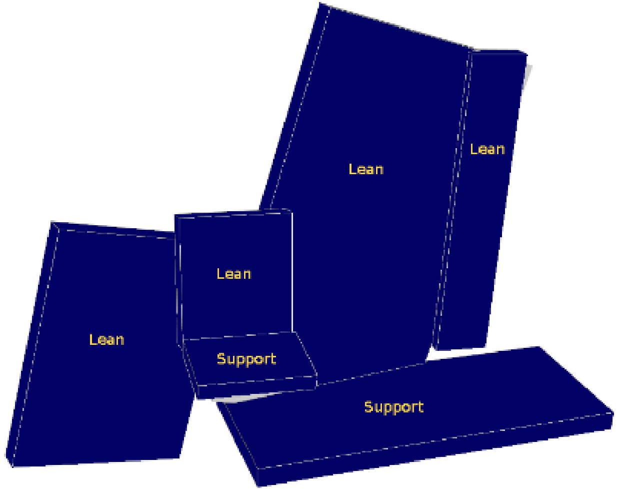



- Compute convex hull for the projected inlier points to estimate *grasp points*. See black points on the chair.
- Inverted reachability map to filter points.



# Loco-Manipulation Affordances (5)



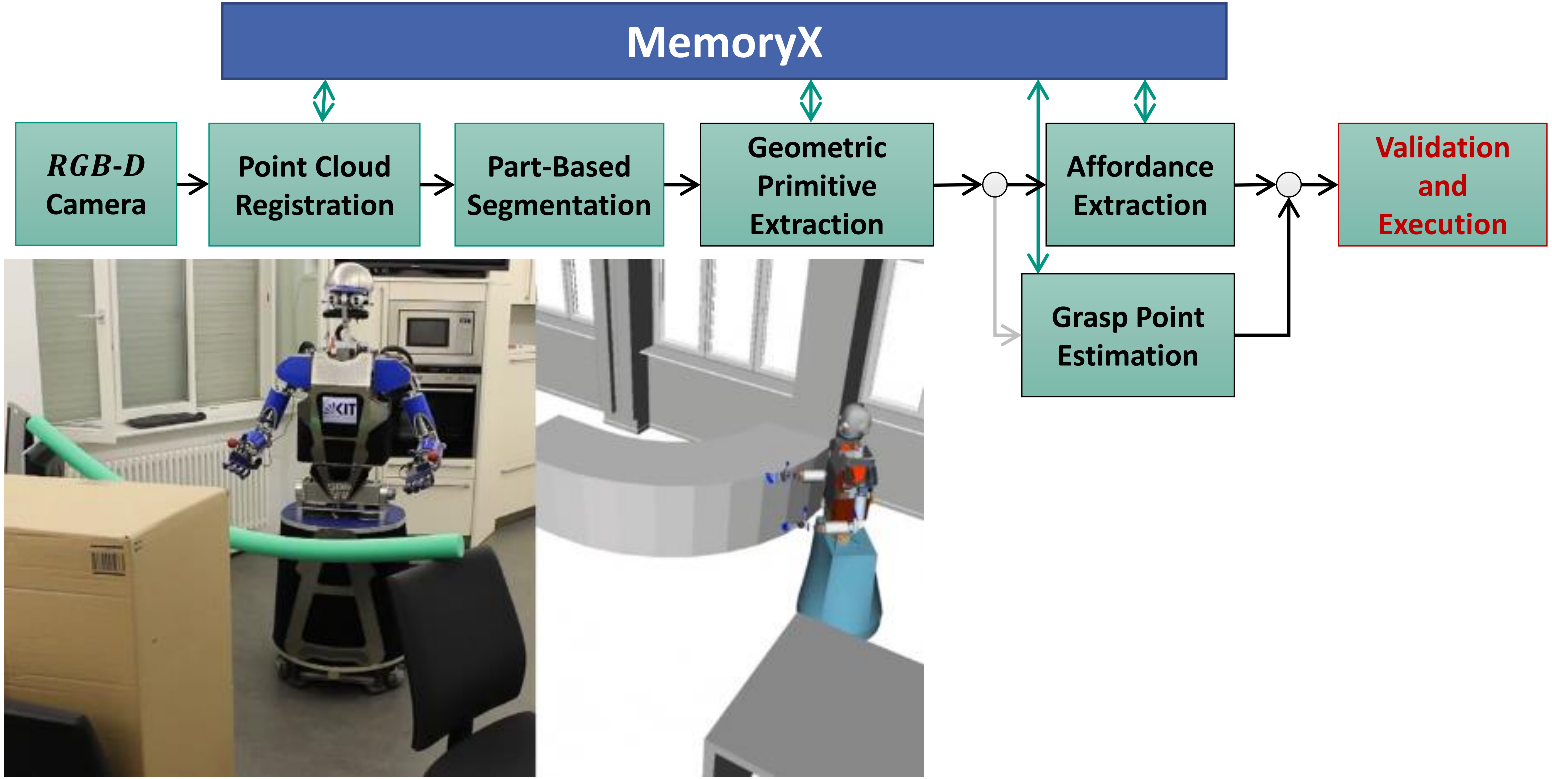

Look-up table!

| Affordance | Shape | Parameters | Conditions | Valid. |
|---|---|---|---|---|
| **Support** | Planar | Normal $\boldsymbol{n}$<br>Area $a$ | $\boldsymbol{n} \uparrow \boldsymbol{z}_{world}$<br>$a \geq \lambda_1$ | (1a) |
| **Lean** | Planar | Normal $\boldsymbol{n}$<br>Area $a$ | $\boldsymbol{n} \perp \boldsymbol{z}_{world}$<br>$a \geq \lambda_2$ | (1a) |
| **Grasp** | Planar | Normal $\boldsymbol{n}$<br>Area $a$ | $a \in [\lambda_3, \lambda_4]$ | (3) |
| | Cylindrical | Radius $r$<br>Direction $\boldsymbol{d}$ | $r \in [\lambda_5, \lambda_6]$<br>$\|\boldsymbol{d}\| \leq \lambda_7$ | |
| | Spherical | Radius $r$ | $r \in [\lambda_8, \lambda_9]$ | |
| **Hold** | Cylindrical | Radius $r$<br>Direction $\boldsymbol{d}$ | $r \in [\lambda_{10}, \lambda_{11}]$<br>$\|\boldsymbol{d}\| \geq \lambda_{12}$ | (2a) |
| **Push** | Planar | Normal $\boldsymbol{n}$<br>Area $a$ | $\boldsymbol{n} \perp \boldsymbol{z}_{world}$<br>$a \leq \lambda_{13}$ | (1b) |

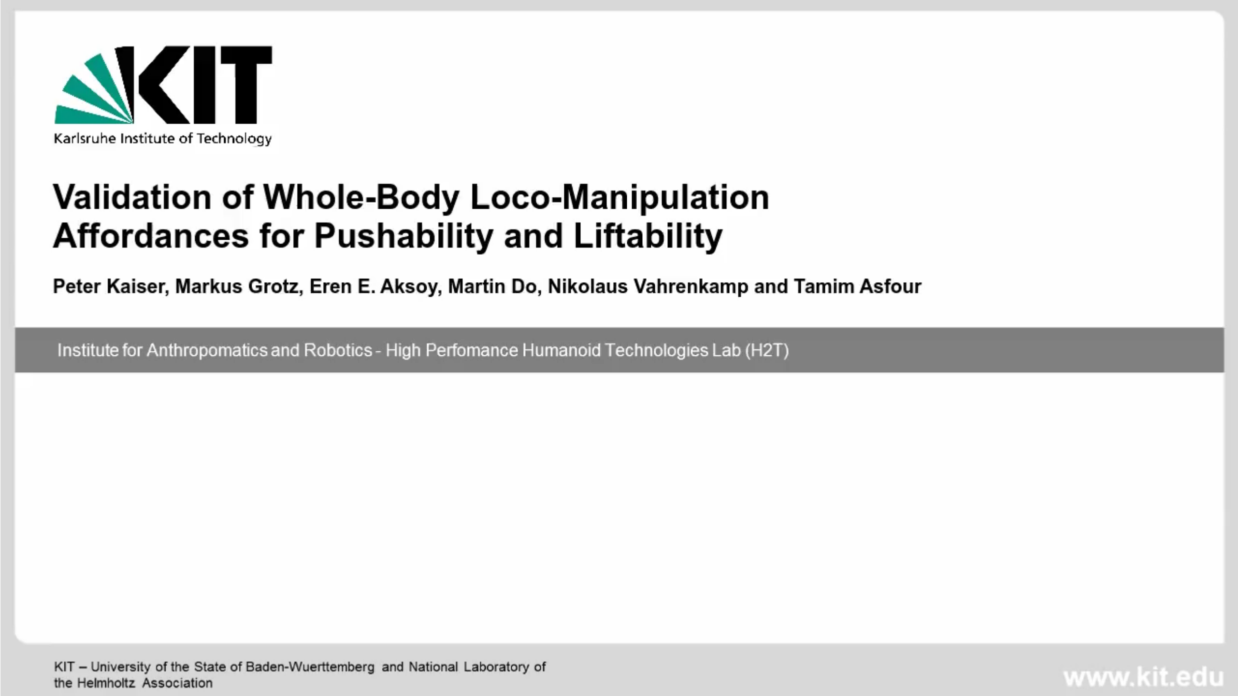
# Loco-Manipulation Affordances (6)

Registered Point Clouds | Segmented Point Cloud | Primitive & Grasp Points | Affordances

# Loco-Manipulation Affordances (7)



# Loco-Manipulation Affordances (8)




Kaiser P., Grotz M. Aksoy E.E., Do M. Vahrenkamp N. Asfour T., "Validation of Whole-Body Loco-Manipulation Affordances for Pushability and Liftability", Humanoids 2015

# German Terms

| Deutsch | Englisch |
| --- | --- |
| Farbnuance | hue |
| Sättigung | saturation |
| Helligkeit | intensity/value |
| Hauptachse | principal axis |
| Hauptpunkt | principal point |
| Bildkoordinaten | image coordinates |
| Kamerakoordinaten | camera coordinates |
| Weltkoordinatensystem | world coordinate system |
| Schwellenwertverfahren | thresholding |
| Punktwolken | point clouds |
| Kartierung | mapping |
| Orientierungspunkt | landmark |