64-424
Intelligent Robotics

https://tams.informatik.uni-hamburg.de/lectures/2019ws/vorlesung/ir

Marc Bestmann / Michael Görner / Jianwei Zhang

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

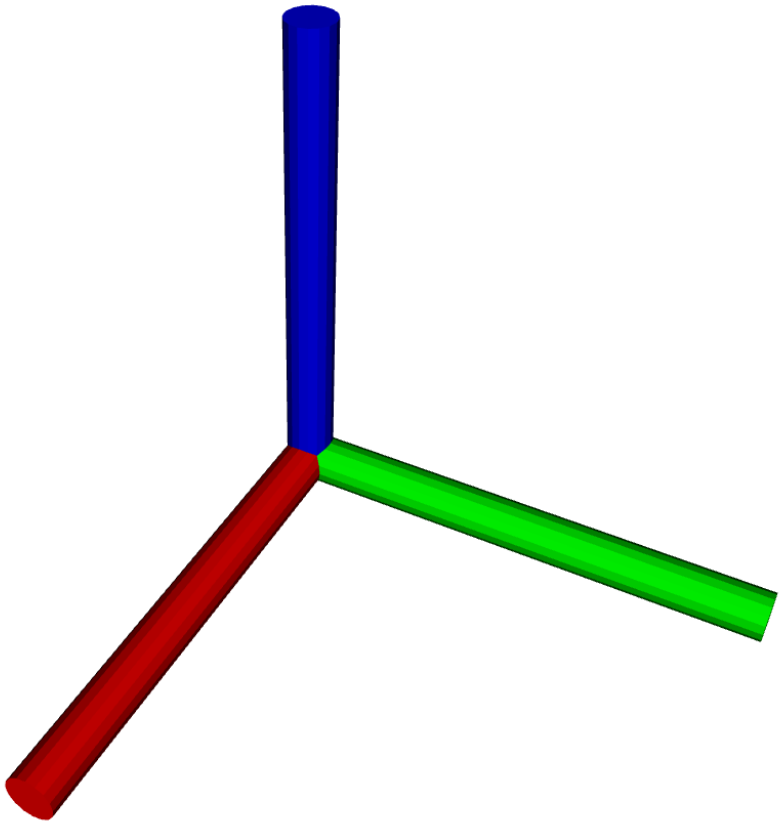
Winterterm 2019/2020

Outline

1. Transformations
2. Vision systems

Outline

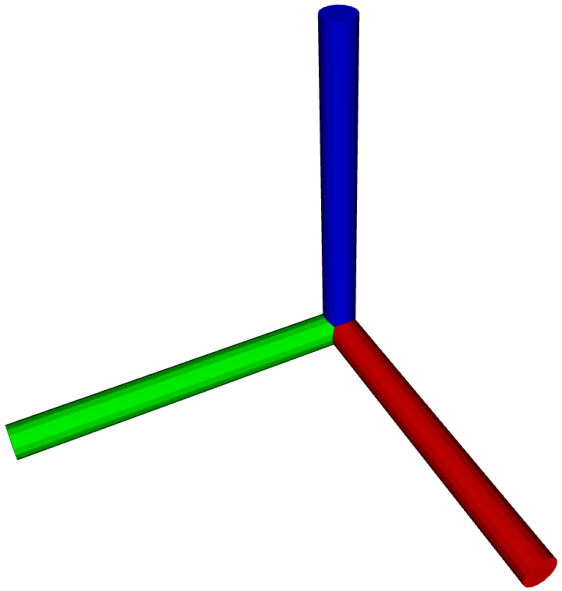
Coordinate Systems



- Standard orthonormal basis for 3D Cartesian space
- RGB $\rightarrow$ XYZ for visualization
- When multiple CSs are relevant, usually called ***frame*** and associated with a *name*
- *There are many ways to specify frames w.r.t. each other*

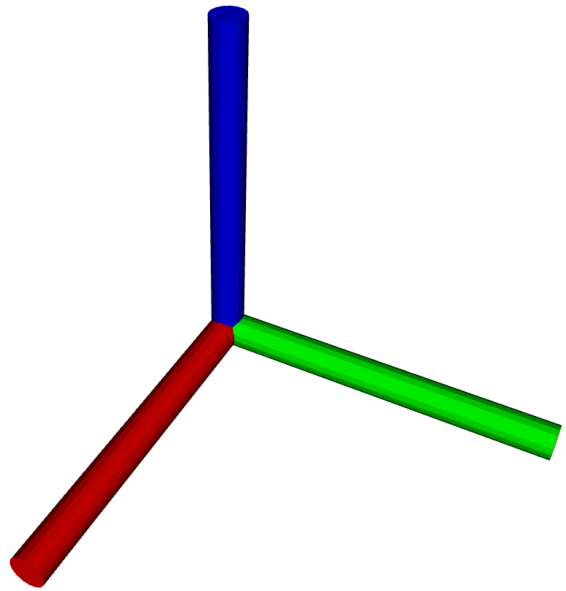
Coordinate Systems - Handedness

Left-Handed



- DirectX, POV-Ray, Unity, ...

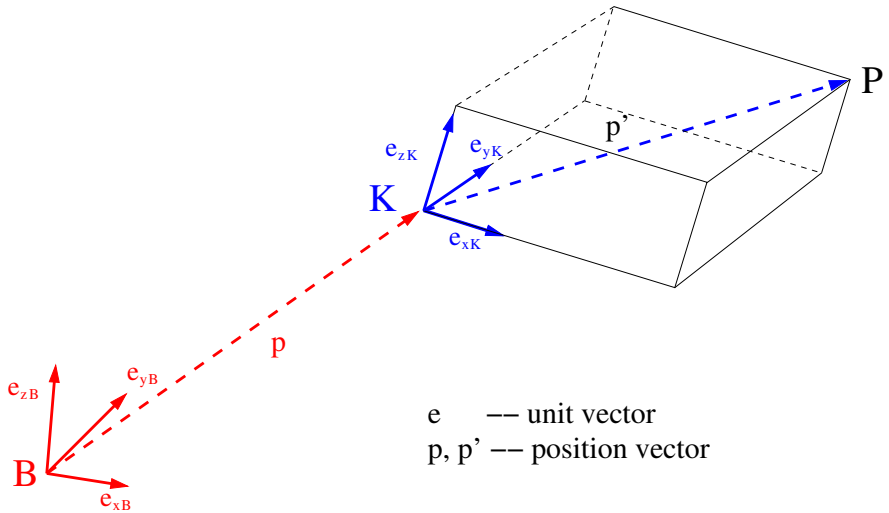
Right-Handed



- OpenGL, OpenCV, **ROS**, ...

Coordinate Systems - Relative Pose

- The **pose** of a rigid object comprises its **position** and **orientation** w.r.t. some CS
- It can be represented by
  - its Cartesian coordinate system (CS) **K** (frame) and
  - a transformation between CS **K** and e. g. the global CS **B** (**B** $\rightarrow$ **K**)



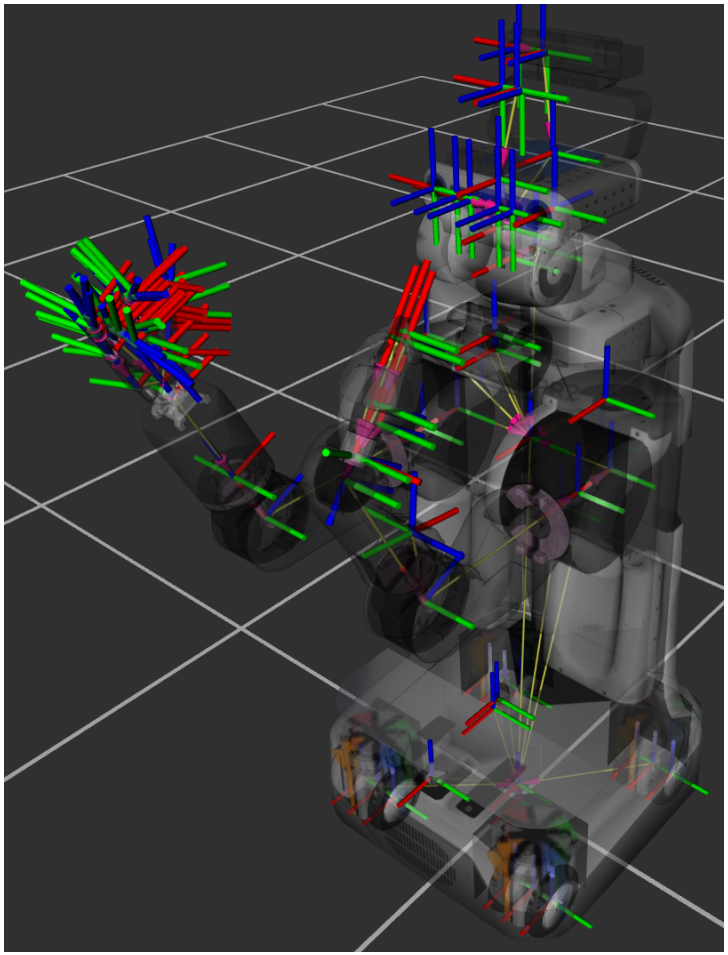
Coordinate transformation

- Transition between coordinate systems (frames)

  Typical reference frames:

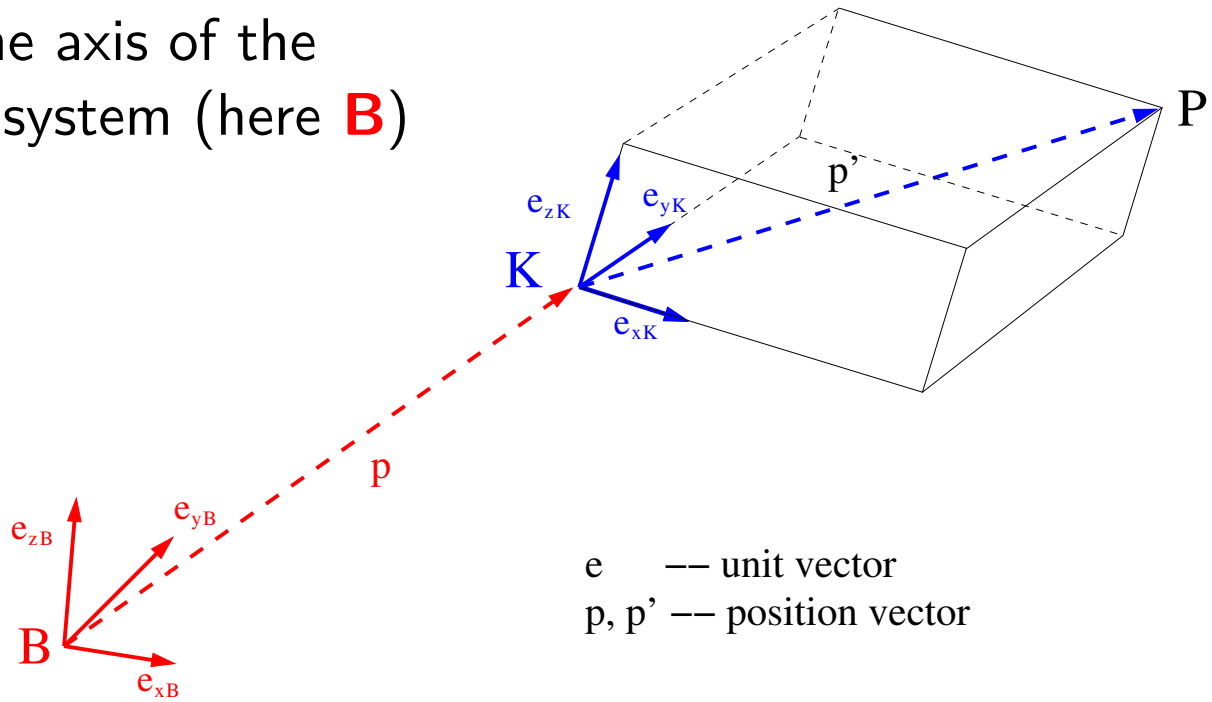
  - Robot base
  - End-effector (tool)
  - Table (world)
  - Object
  - Camera
  - Screen
  - ...

Frame transformations convert one frame into another.



Relative Pose - Position

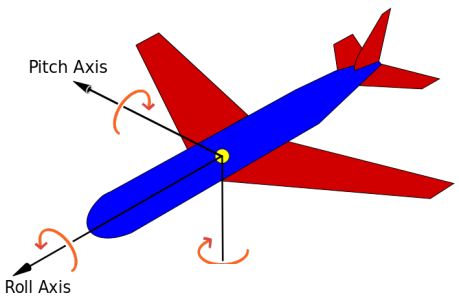
- Coordinates of the Origin
- Translations along the axis of the reference coordinate system (here **B**)

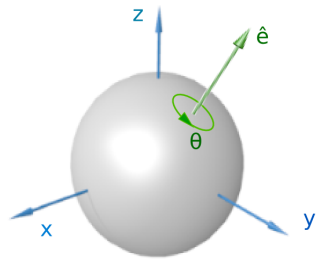


- Defined by: $\mathbf{p} = [p_x, p_y, p_z]^T \in \mathcal{R}^3$ w.r.t. e_{xB}, e_{yB}, e_{zB}

Relative Pose - Orientation (Spacial Alignment)

- Euler-angles ϕ, θ, ψ
  - Rotations performed in sequence around the axes of a coordinate system
  - e.g. Roll-Pitch-Yaw, $ZY'Z''$, ...
- Axis-Angle $\hat{e}, \theta$
  - Rotation around axis $\hat{e}$ by angle θ
  - Elegant algebraic version: Unit Quaternions
- Rotation matrix $R \in \mathcal{R}^{3\times3}$
  - 9 parameters for 3 DOF
  - For plain rotations: $det(R) = 1$





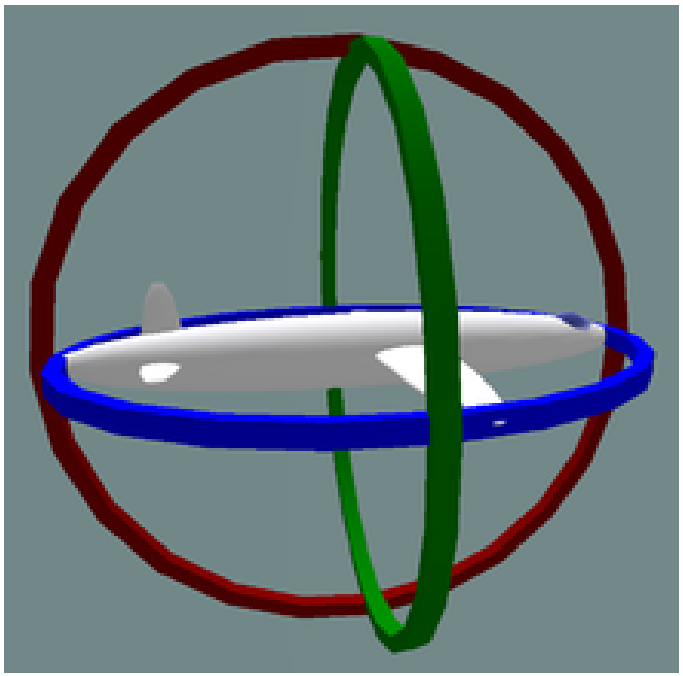
$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

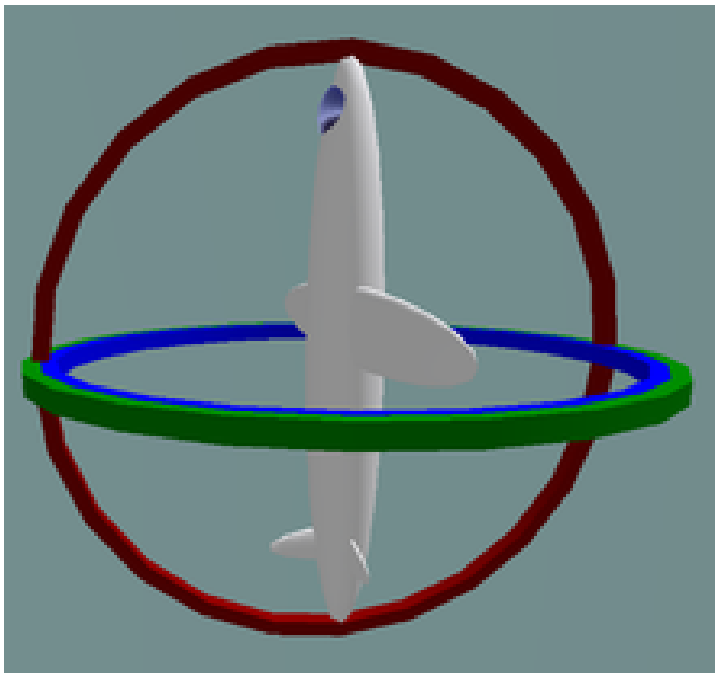
Euler Angles

- Represented as rotations $\phi, \theta, \psi \in [0; 2\pi)$ around axes
- It remains undefined **around which axes!**
- *Proper Euler angles* rotate around:
  - $z - x - z$
  - $y - z - y$
  - $x - z - x$
  - $x - y - x$
  - $z - y - z$
  - $y - x - y$
- *Tait-Bryan angles* rotate around:
  - $x - y - z$
  - $z - x - y$
  - $z - y - x$
  - $y - z - x$
  - $x - z - y$
  - $y - x - z$
- All sequences can be interpreted w.r.t. the reference frame (**extrinsic**) or the partially rotated frame (**intrinsic**)

There are 24 possible interpretations of Euler angles!
A gimbal lock can happen!

Gimbal Lock





Axis-Angle / Unit Quaternions

Euler's Rotation Theorem Any rotation can be expressed as an elemental rotation around a single axis (The *Euler axis* $\hat{e}$).

Axis-Angle

- Represented as $\hat{e} = [x, y, z] \in \mathcal{R}^3$ and $\theta \in [0; 2\pi)$
- Often encoded as $\hat{e}$ only, where $\|\hat{e}\| = \theta$

Unit Quaternions

- Represented as $q = w + x \cdot \mathbf{i} + y \cdot \mathbf{j} + z \cdot \mathbf{k}$ with unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$ and $\|q\| = 1$

$$q = \cos\frac{\theta}{2} + (\hat{e}_x\mathbf{i} + \hat{e}_y\mathbf{j} + \hat{e}_z\mathbf{k})\sin\frac{\theta}{2}$$

- Usually encoded as $[x, y, z, w]$ *or* $[w, x, y, z]$

Coordinate transformation

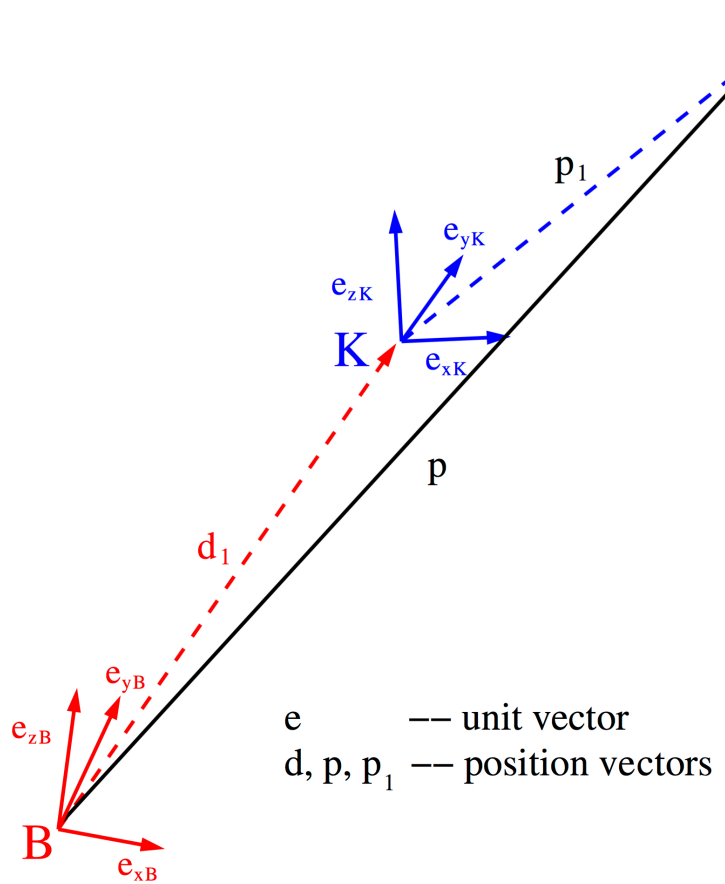
Points in space referenced in a local frame can be transformed into a global frame through:

- Translations
- Rotations

$$
\begin{aligned}
{}^{B}p &= {}^{B}d_1 + {}^{B}p_1 \\
&= {}^{B}d_1 + {}^{B}R_K \; {}^{K}p_1
\end{aligned}
$$

Please note:

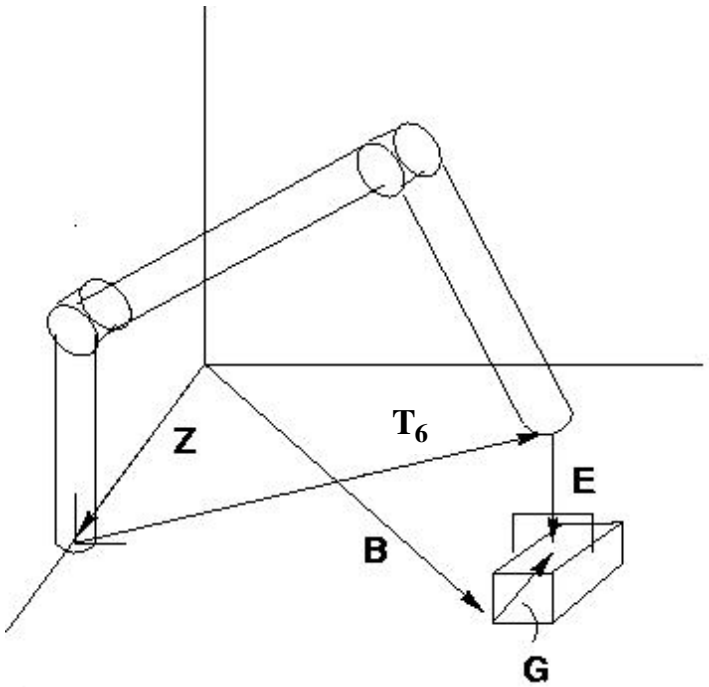
- ${}^{B}R_K$: rotation matrix R, that describes rotations to generate frame K from frame B
- ${}^{K}p$: vector p based on frame K.



Application: Relative transformations

There are the following transformations:

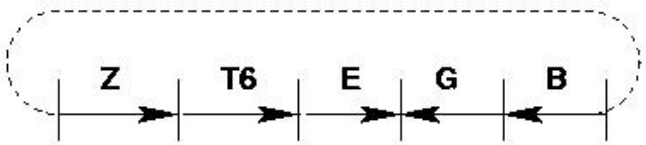
- Z: World $\rightarrow$ Base of the manipulator
- T_6: Base of the manipulator $\rightarrow$ End of the manipulator
- E: End of the manipulator $\rightarrow$ End effector
- B: World $\rightarrow$ Object
- G: Object $\rightarrow$ Grasp Pose



Application: Transformation chain

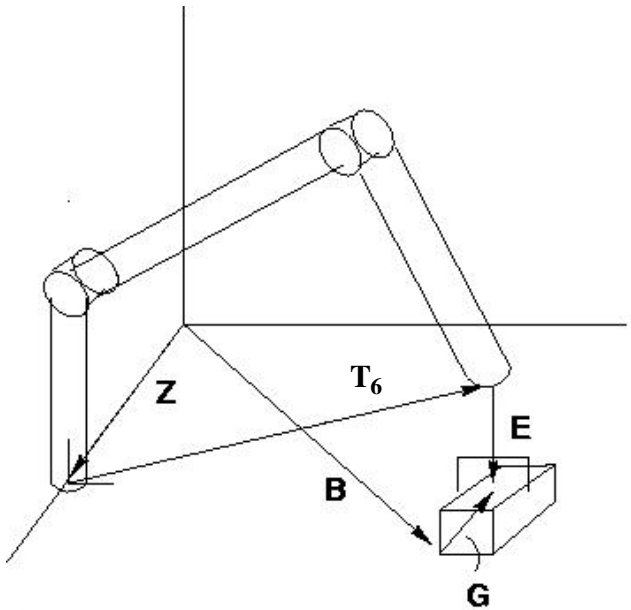
There are two descriptions of the end effector's frame, one in relation to the object and the other in relation to the manipulator. Both descriptions are equal:

$$ZT_6E = BG$$



To find the manipulator transformation:

$$T_6 = Z^{-1}BGE^{-1}$$



To determine the coordinate frame of the object:

$$B = ZT_6EG^{-1}$$

The transformation chain is also called the *kinematic chain*.

In Reality

If you are programming anything robotic related just use a library which does the transformations for you. You will save time and have less bugs. If you use Euler Angles you do have to be aware of their downsides and the order that this library uses.
Example libraries:

- tf2 (ROS, C++, Python)
- Robotics Library (C++)
- Robotic System Toolbox (Matlab)

Want to Know More About Transformations

For a more in depth lecture about transformations in the robotic domain, please visit "Introduction to Robotics" in the summer term.
You can find the last years slides here:
https://tams.informatik.uni-hamburg.de/lectures/2019ss/vorlesung/itr/index.php?content=03-documents

Outline

1. Transformations
2. Vision systems

Introduction
Camera calibration
Lens distortion

Introduction

Various vision systems are used in robotic applications as a fundamental tool for environmental perception

- Much of the information obtained by the human brain includes vision as a dominating/contributing modality
- Roughly 60% of the brain are said to be dealing with visual data - mostly in combination with other modalities

Some of the questions computer/machine vision research is trying to answer are:

- *Where am I?* (Scene modeling, classification, recognition, etc.)
- *What is around?* (Object detection/recognition, etc.)
- *How can I interact?* (Structure estimation, tracking, etc.)

Introduction (cont.)

Vision sensor technology is evolving steadily, with many (quite) different options available

- Linear camera sensor
- Analog CCD camera (black/white or color)
- Standard CMOS camera (e.g. web cam)
- High-Dynamic-Range (HDR) CMOS camera
- Structured light camera (Infrared, RGB)
- Stereo vision system
- Omnidirectional vision system

Introduction (cont.)

Vision systems are widely used in industrial automation applications

- Object grasping tasks
  - Objects with predetermined position (e.g. production line)
  - Randomly positioned objects (e.g. 'bin-picking')
- Object handling tasks
  - Cutting, tying, wrapping, sealing, etc.
  - Inspection during assembly
- Assembly tasks
  - Welding, gluing, attaching, etc.

Introduction (cont.)

Automated visual inspection at the production line.

Introduction (cont.)

Vision systems in robotics are used for a wide range of applications

- Perception of objects
  - *Static:* Recognition, searching, indexing, ...
  - *Dynamic:* Tracking, manipulation, ...
- Perception of humans
  - Face recognition
  - Gaze tracking
  - Gesture recognition
  - ...

Introduction (cont.)

Navigation and modeling of the environment

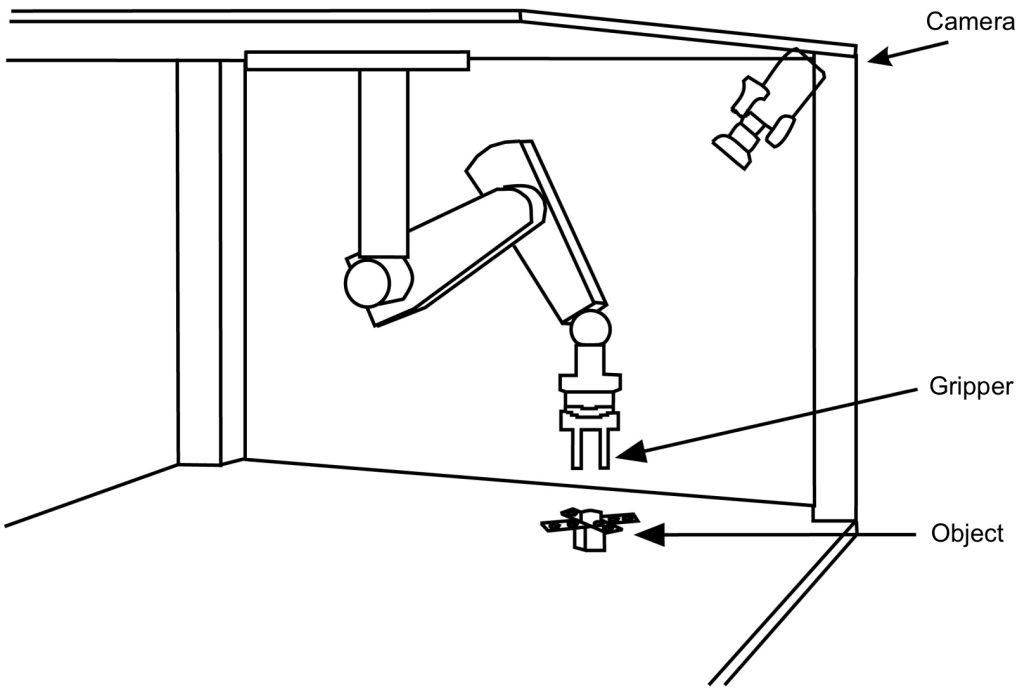
- Robot localization:
  - Relative
  - Absolute
  - In reference to various coordinate frames
- Object recognition and localization
- 3D scene reconstruction
- Allocation of the environment

Introduction (cont.)

Vision-based motion control

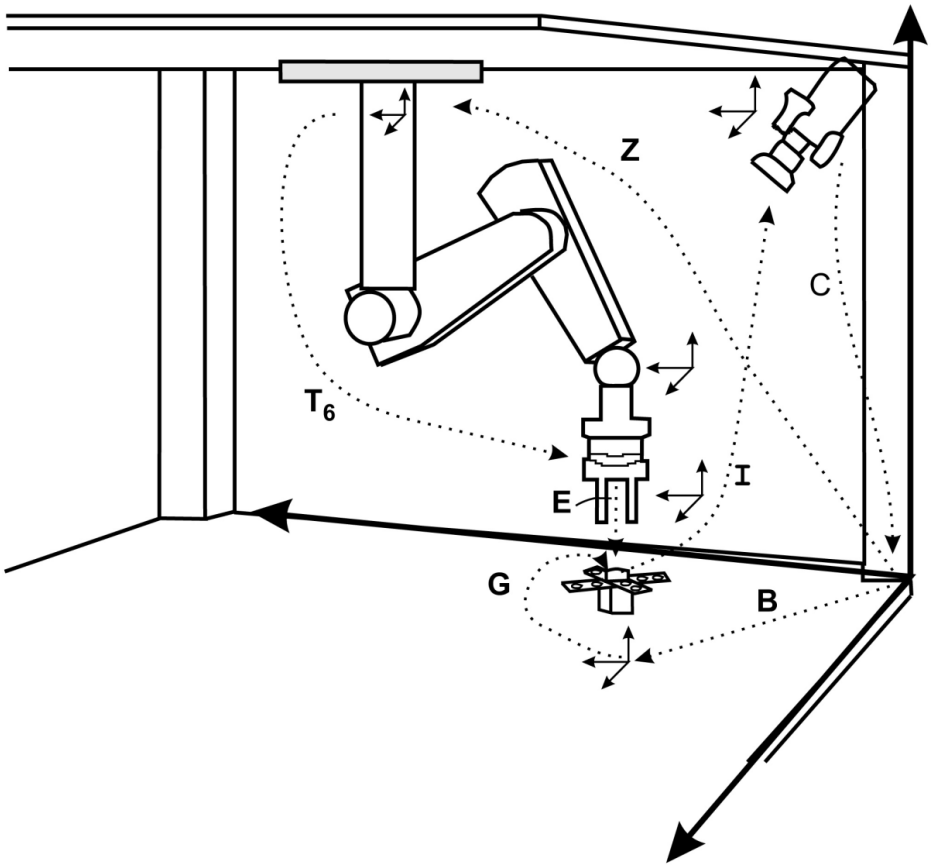
- Visual-Servoing
  - Coarse and fine positioning
  - Tracking of movable objects
- Collision avoidance
  - Depth map based distance measurement
- Coordination with other robots and/or humans
  - Motion estimation
  - Intention recognition

Application scenario



Very common application scenario: Vision-based manipulator control.

Application scenario (cont.)



Chain of transformations of the common application scenario.

Kinematic chain

Data (points) can be easily transformed between coordinate frames using a suitable transformation sequence of the *kinematic chain*

Z: Transformation from world coordinates to manipulator base coordinates

T_6: Compound transformation from the base of the manipulator to the end of the manipulator

E: Transformation from the end of the manipulator to the gripper

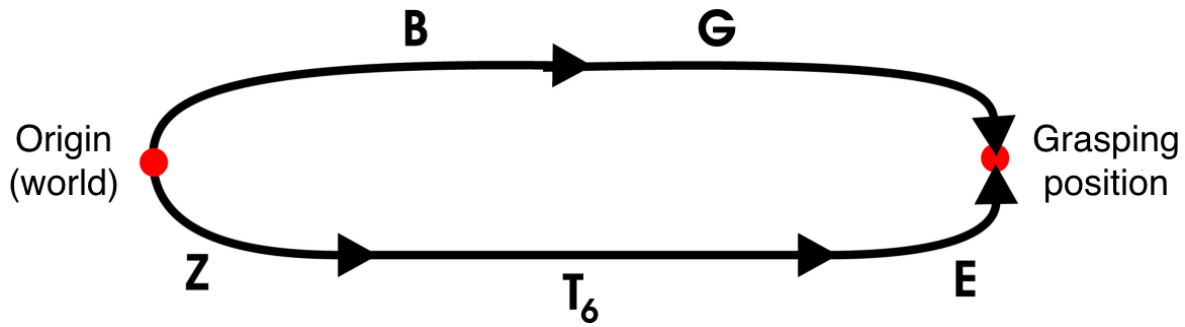
B: Transformation from world coordinates to object coordinates

G: Specification of the grasp coordinate frame in reference to the object coordinate frame

Kinematic chain (cont.)

During grasping and manipulation of the object, the coordinates of the grasping point can be determined in two ways:

$$ZT_6E = BG$$



Kinematic chain (cont.)

To determine the transformation of the manipulator, the following equation needs to be solved:

$$T_6 = Z^{-1}BGE^{-1}$$

In order to determine the location of the object after manipulation, one has to solve:

$$B = ZT_6EG^{-1}$$

Kinematic chain (cont.)

A camera included in the scenario introduces two additional transformation matrices:

- C: Transformation of camera coordinates into world coordinates
  (*Off-line* determination of the transformation through camera calibration)
- I: Transformation of grasping point coordinates into the camera coordinate frame
  (Grasping point is determined using image processing techniques)

Kinematic chain (cont.)

The transformation P from the grasping point to the world coordinate system is given as:

$$P = I\ C$$

The camera to world transformation is determined through the following equation:

$$C = I^{-1}\ P$$

[t]Outline

Camera calibration

Camera calibration in the context of three-dimensional image processing is the determination of the intrinsic and/or extrinsic camera parameters

Intrinsic parameters

- Internal geometrical structure and optical features of the camera

Extrinsic parameters

- Three-dimensional position and orientation of the camera's coordinate frame in relation to a world coordinate frame

Camera calibration (cont.)

What information does one get?

In order to reconstruct 3D information of the environment from two or more images, it is necessary to know the relation between the coordinate frames of the 2D image and the objects of the 3D environment

- The relation between 2D and 3D can be described using two transformations
  - Perspective projection (3D $\rightarrow$ 2D)
  - Backprojection (2D $\rightarrow$ 3D)

Camera calibration (cont.)

Perspective projection (3D point → 2D point)

- Knowing the camera projection matrix the 2D projection of a 3D point can be determined (approximated)

Backprojection (2D point → 3D beam)

- If a 2D image point is known, there is a ray in 3D space on which the corresponding 3D point lies
- If there are two or more views of the 3D point, its coordinates can be determined using *triangulation*

Camera calibration (cont.)

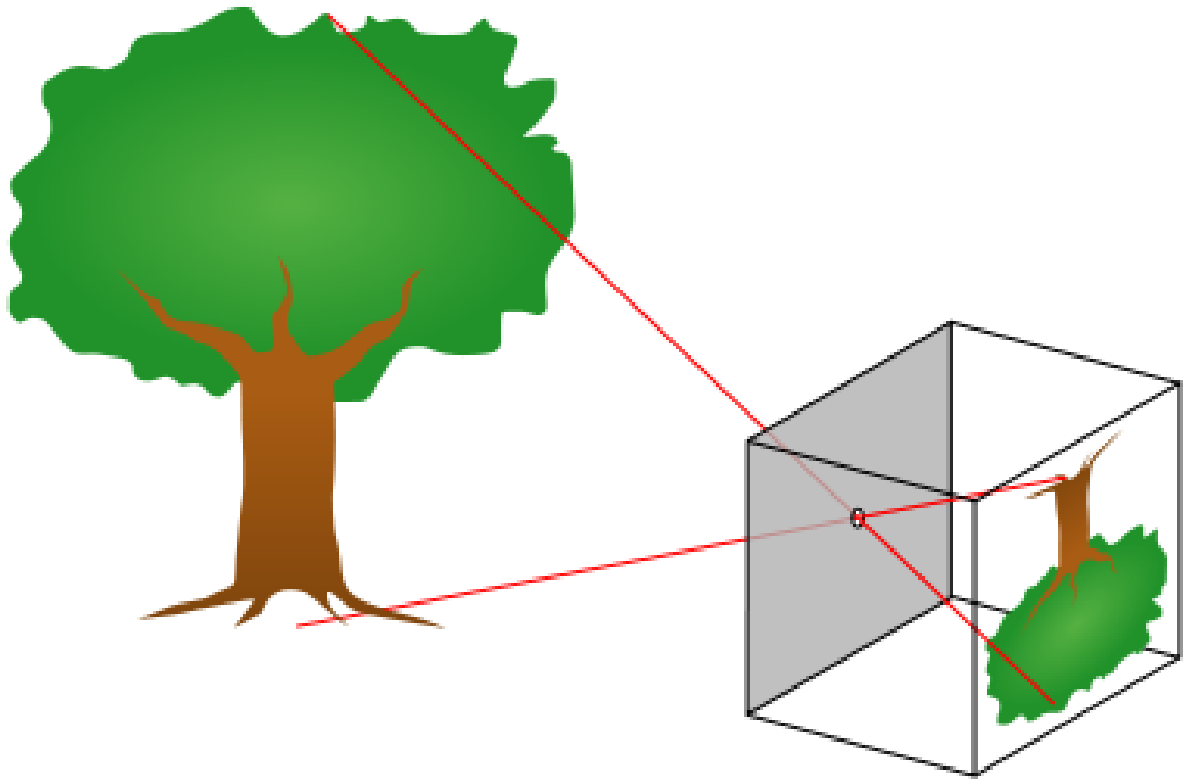
- The perspective projection is useful in order to reduce the search space during scene analysis
- The backprojection is helpful for deriving 3D information based on features in 2D images
- These transformations are used in various applications:
  - Automatic assembly
  - Robot calibration
  - Tracking
  - Trajectory analysis
  - 3D metrology
  - ...

Camera calibration (cont.)

Several calibration techniques have been established

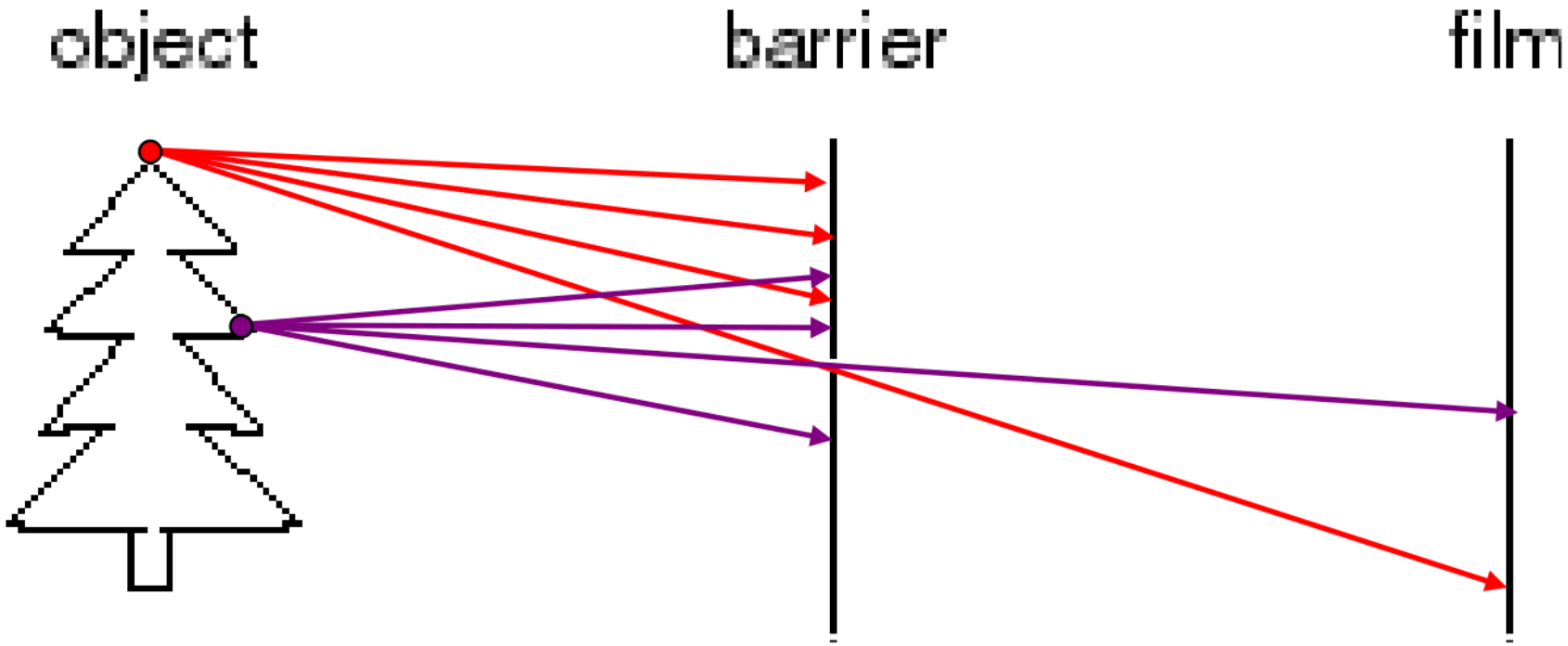
- Camera calibration can be done *on-line* or *off-line*
- Using a calibration object:
  - Identification of the camera parameters
  - Determination of coordinate transformation between camera coordinates and world coordinates
- Using self-calibration approaches
- Using machine learning methods

Pinhole camera model

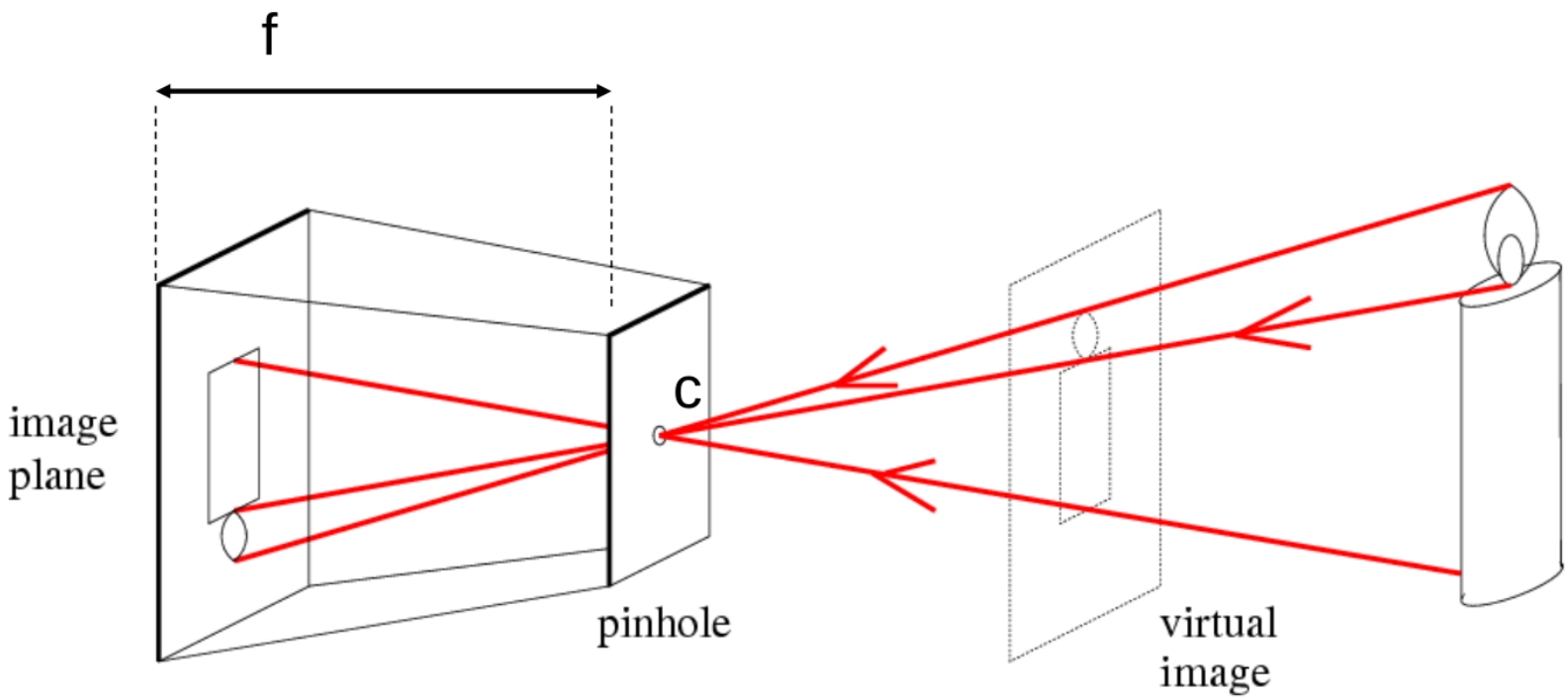


Pinhole camera without lens distortion

Pinhole camera model

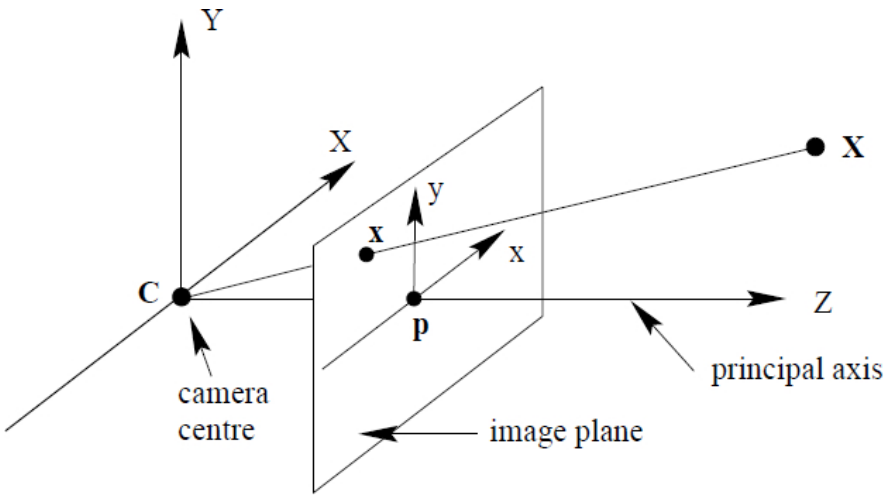


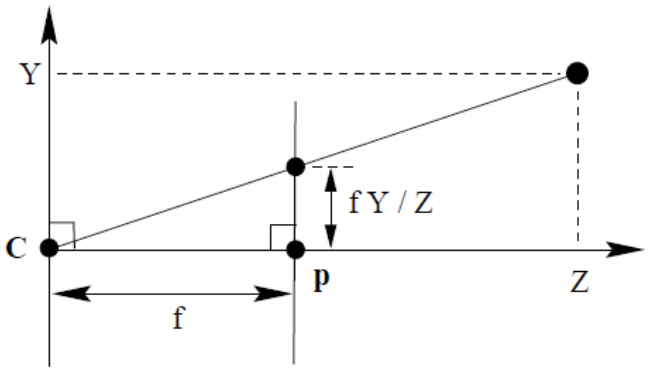
Pinhole camera model



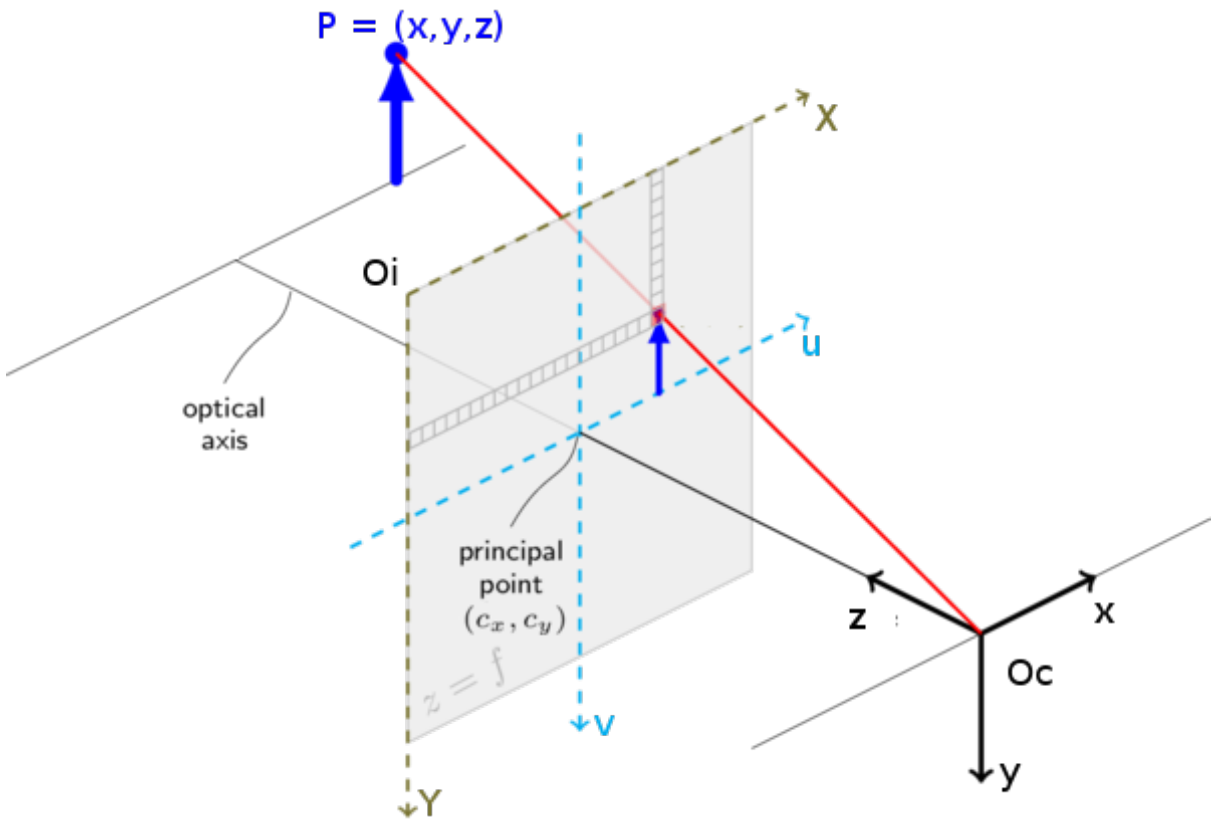
f = focal length
c = center of the camera

Pinhole camera model 1D





Pinhole camera model 2D



Pinhole camera without lens distortion [OpenCV]

Pinhole camera model (cont.)

- (x_w, y_w, z_w): 3D world coordinate system with the origin O_w [m]
- (x, y, z): 3D coordinate system of the camera with the origin O_c (optical center) [m]
- (X, Y): 2D image coordinate system with the origin O_i [pixels]
- f: Focal length of the camera [m]

Pinhole camera model (cont.)

Projection of camera coordinates onto image coordinates

- Point P is projected onto the corresponding (ideal) image coordinate (u, v)
- *Perspective projection* with focal length f:

$$u = f\frac{x}{z} \qquad v = f\frac{y}{z}$$

- The image coordinates (X, Y) are calculated from (u, v) as follows:

$$X = s_x u + C_x \qquad Y = s_y v + C_y$$

- The scaling factors s_x and s_y are used to convert the image coordinates from meters to pixels [pixels/m]
- C_x, C_y are the coordinates of the principal point in the image

Pinhole camera model - Assumptions

- Intrinsic Assumptions (typical error source in braces)
  - Pixels are squared (sensor manufacturing)
  - Optical center in the center (sensor misplacement)
  - No skew (inaccurate synchronization of the pixel-sampling process)
- Extrinsic Assumptions
  - No rotation
  - Camera at (0,0,0) - no translation

Intrinsic Parameters

Transformation from camera to image coordinates

- Let P(x,y,z) be a point in the camera coordinate system
- Let $f_x = s_x * f$ and $f_y = s_y * f$
- Let θ be the skew angle between x and y axis
- Let C_x, C_y be the principal points coordinates in image plane
- Its projection into the camera coordinate system can be determined as follows:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ with } \quad K = \begin{bmatrix} f_x & -f_x cot(\theta) & c_x \\ 0 & f_y / sin(\theta) & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- K is the camera matrix consisting of the intrinsic parameters

Extrinsic Parameters

Transformation from world to camera coordinates

- Let $P(x_w, y_w, z_w)$ be a point in the world coordinate system
- Its projection into the camera coordinate system can be determined as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + t$$

$$\text{with} \quad R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \quad \text{and} \quad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

- The parameters R and t are the extrinsic parameters

Calibration parameters

The pinhole camera model contains the following calibration parameters:

- The three independent extrinsic parameters of R
- The three independent extrinsic parameters of t
- The intrinsic parameters f_x, f_y, θ, C_x and C_y

Lens distortion

Real cameras have lenses and produce a variety of imaging errors and do not satisfy constraints of the pinhole camera model

The main error sources are:

- Low spatial resolution due to low resolution of the camera device being used
- Most (cheap) lenses are asymmetrical and generate distortions
- Imprecision during assembly (pose of the lens)

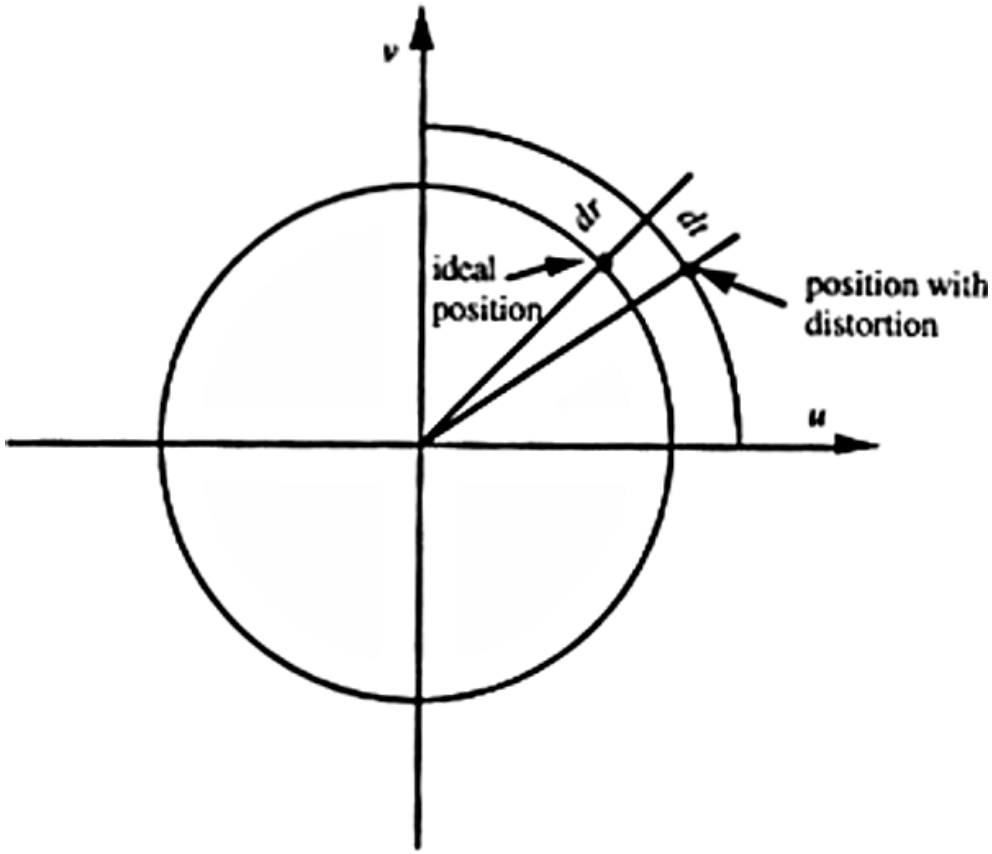
Lens distortion (cont.)

- Distortion by the lens system results in a changed position of the image pixels on the image plane
- The pinhole camera model is no longer sufficient
- It is replaced by the following model:

$$u' = u + D_u(u, v)$$
$$v' = v + D_v(u, v)$$

where u and v are the non-observable, distortion-free image coordinates, and u' and v' the corresponding distorted coordinates

Lens distortion (cont.)

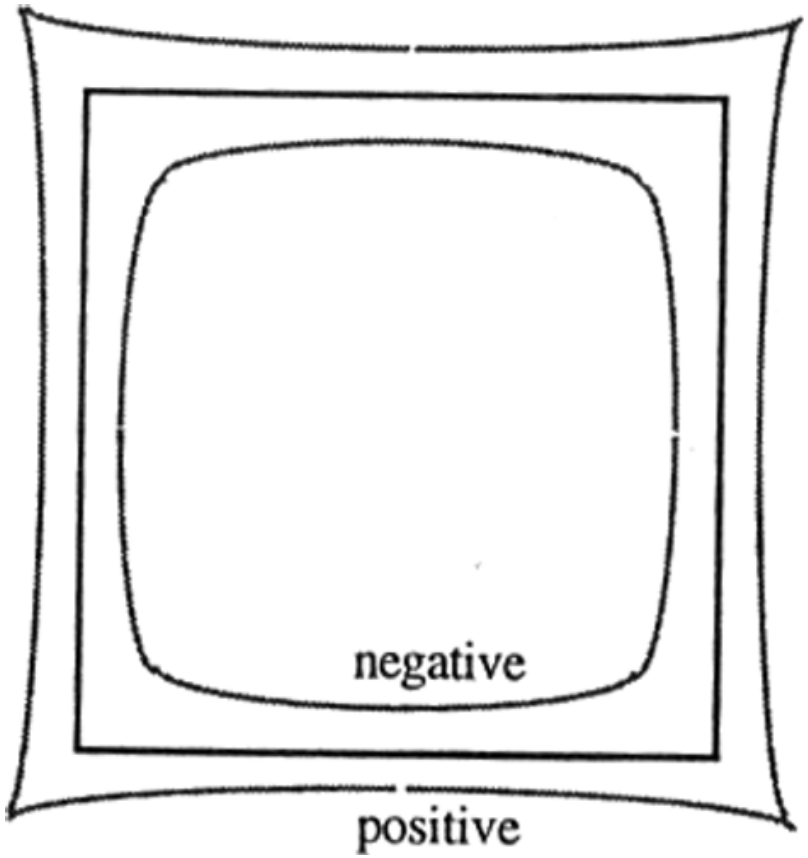


Lens distortion (cont.)

Types of distortion

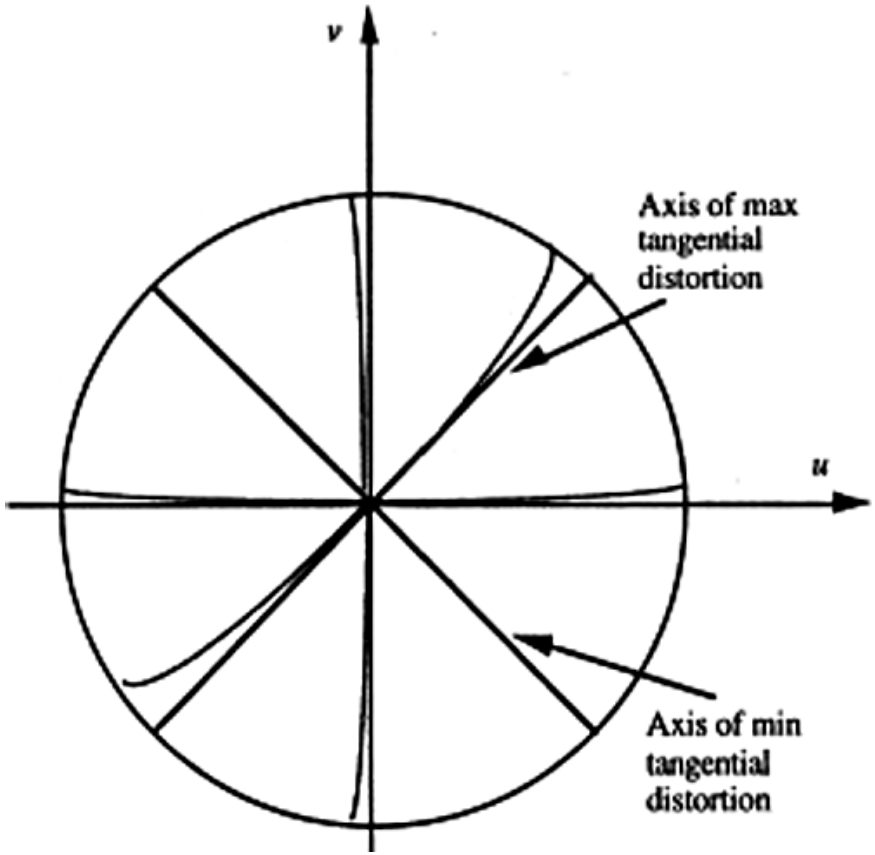
- There are **two** primary types of distortions:
  - *Radial*
  - *Tangential*
- Radial distortion causes an offset of the ideal position inwards (barrel distortion) or outwards (pincushion distortion)
  Possible cause: Flawed radial bend of the lens
- Tangential distortion shifts the ideal position along a tangential curve
  Possible cause: Non-parallel sensor/lens

Lens distortion



Radial distortion: Straight lines $\rightarrow$ no distortion

Lens distortion (cont.)



Tangential distortion: Straight lines $\rightarrow$ no distortion

Camera model

Since radial lens distortion is the dominant type, the following equations can be used to establish a simplified, yet more correct camera model:

$$X_d = X(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$Y_d = Y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$\text{with} \quad r^2 = x^2 + y^2$$

Where X_d, Y_d are the distorted image coordinates.

Calibration

The problem camera calibration procedures are trying to solve is the identification of the unknown parameters of the camera model

- The computation of these parameters for the distortion-free camera model yields the position of the camera in world coordinates

Calibration requires a set of m object points, which:

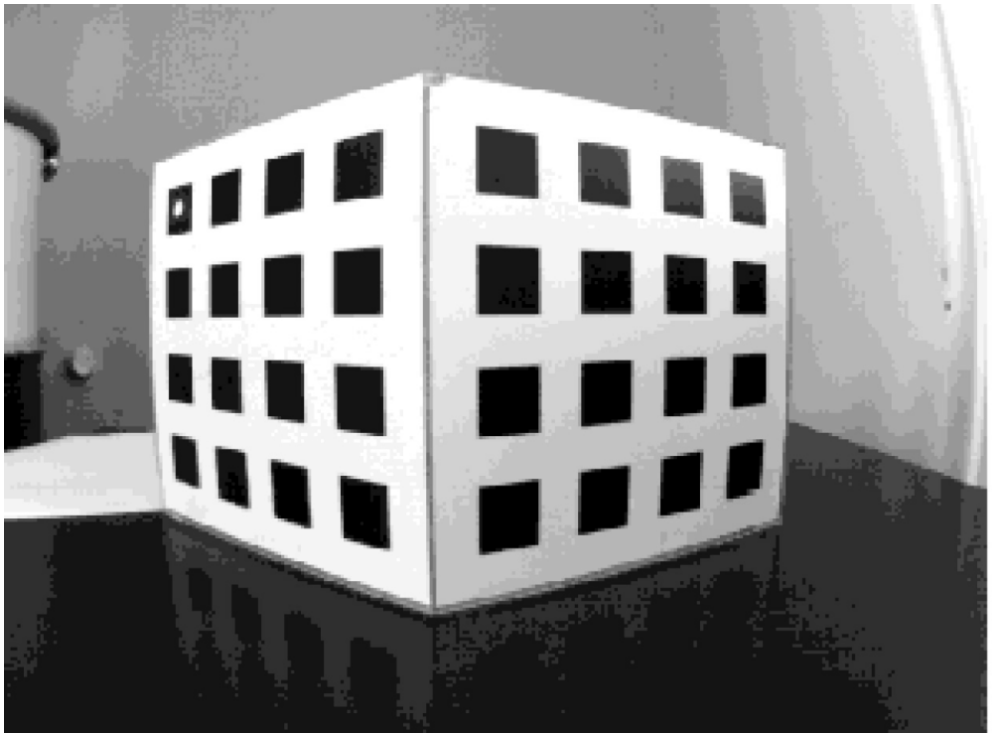
1. Have known world coordinates $\{x_{w,i},\ y_{w,i},\ z_{w,i}\},\ i = 1, \ldots, m$ with sufficiently accurate precision
2. Lie within the camera's field of view

These *calibration points* are detected in the camera image with their respective image coordinates $\{X_i, Y_i\}$

Calibration (cont.)

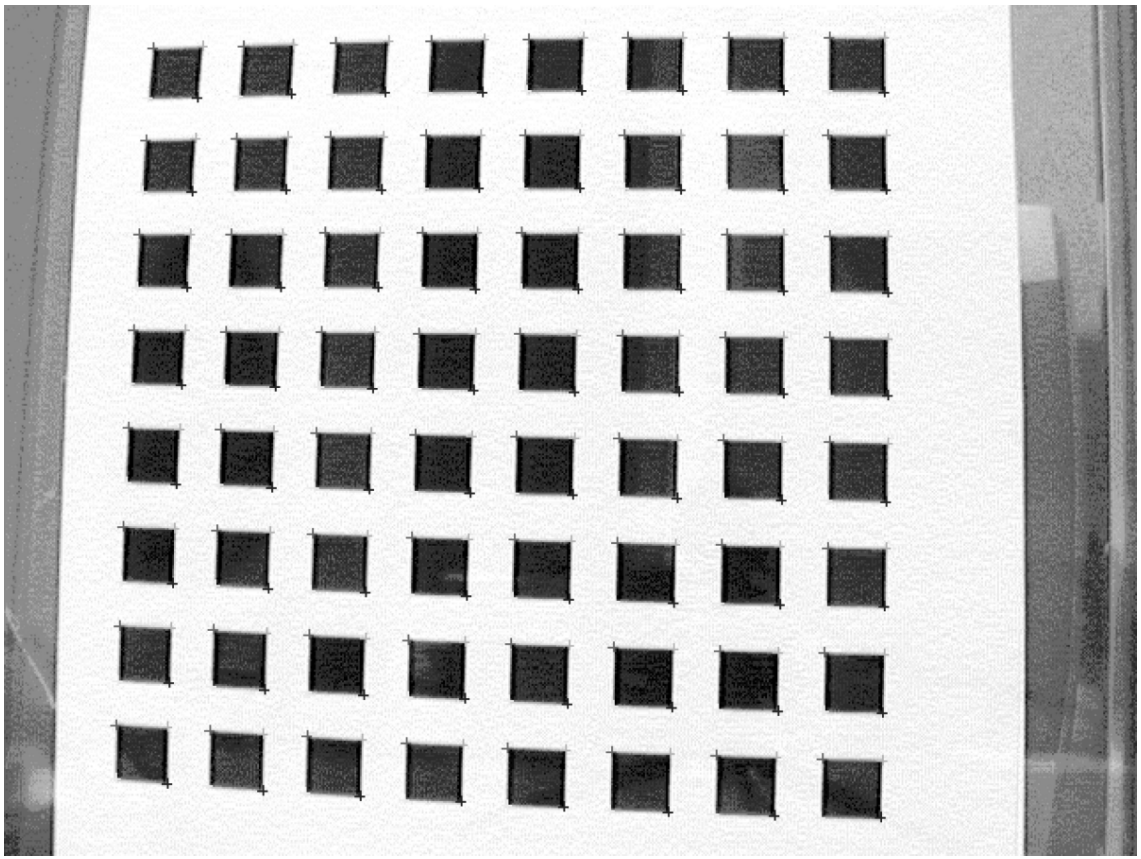
- Having these known world coordinates can be tricky
- Only calibrating intrinsic parameters is easier, since we only need known points in camera frame
- A checkerboard with known square sizes is enough
- Dependent on the use case, intrinsic parameters may be enough

Intrinsic Calibration Objects



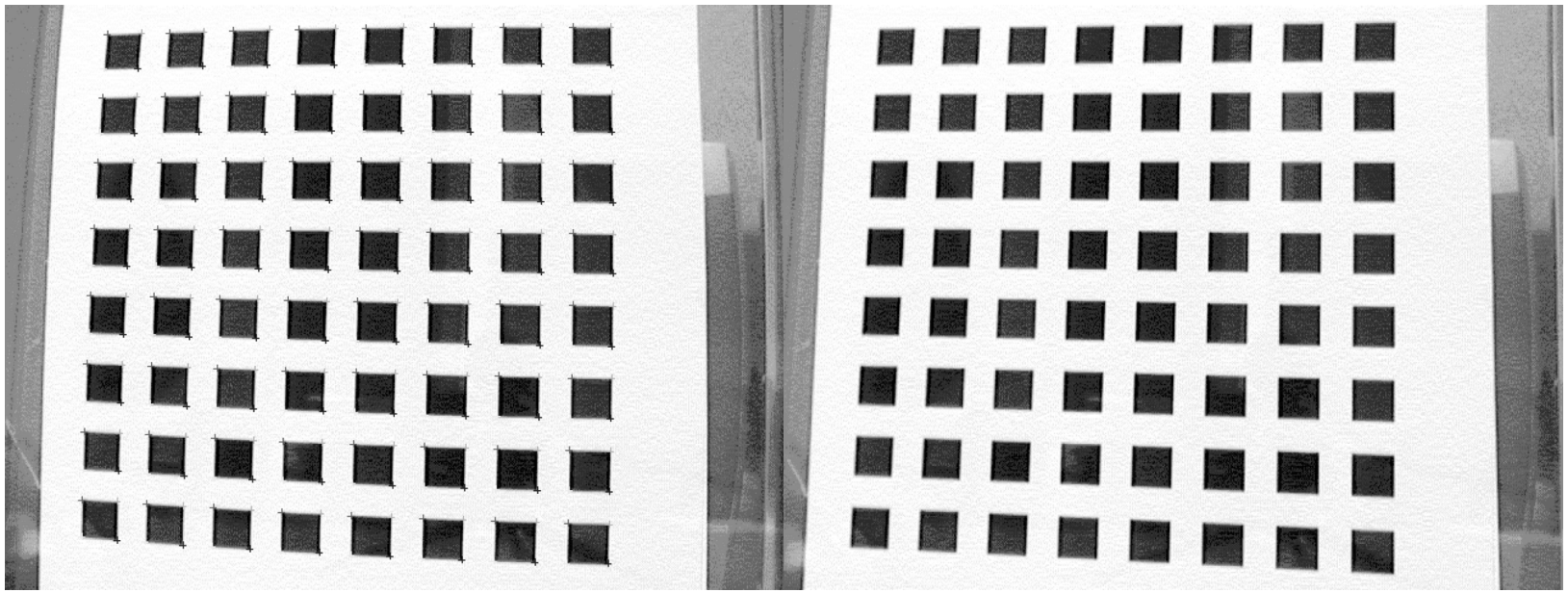
A typical 3D calibration object

Intrinsic Calibration Objects



Typical calibration pattern

Use of Intrinsic Calibration Parameters

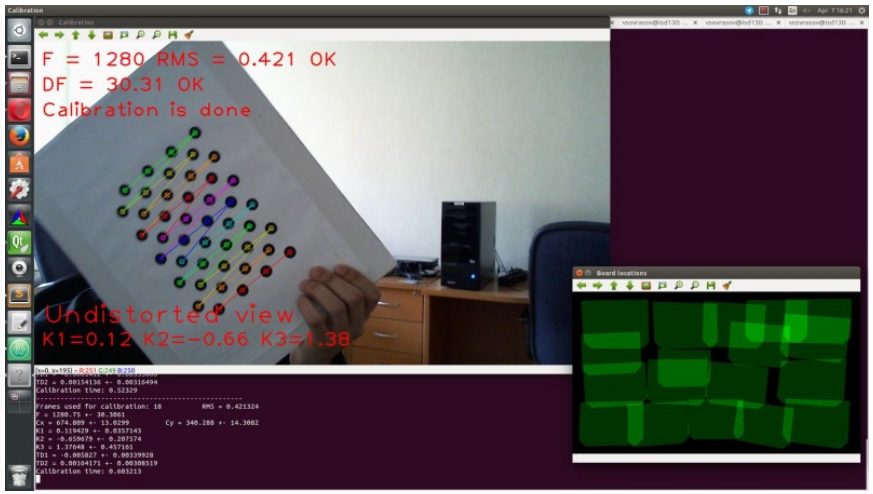


Using the determined camera parameters, the image can be undistorted

In the Real World

Different open source implementations exists that do the camera calibration for you.

- camera_calibration (ROS package)
- Interactive camera calibration application (OpenCV)
- Camera Calibrator (Matlab)



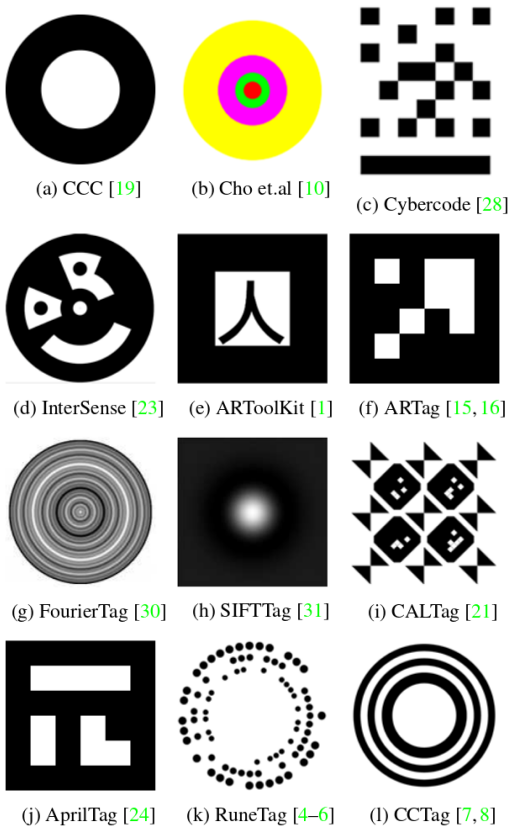
Live Demo

Live demo

Fiducial Markers

- In robotics, we are often interested in getting transforms
  - Robot -> Object
  - Robot -> World
- Getting these directly from camera images is difficult
- Computing poses of known, well-structured elements is easy
- Those are called *fiducial markers* or short *tags*
- Tags can be used for extrinsic calibration
- Also widely used in augmented reality

Fiducal Markers

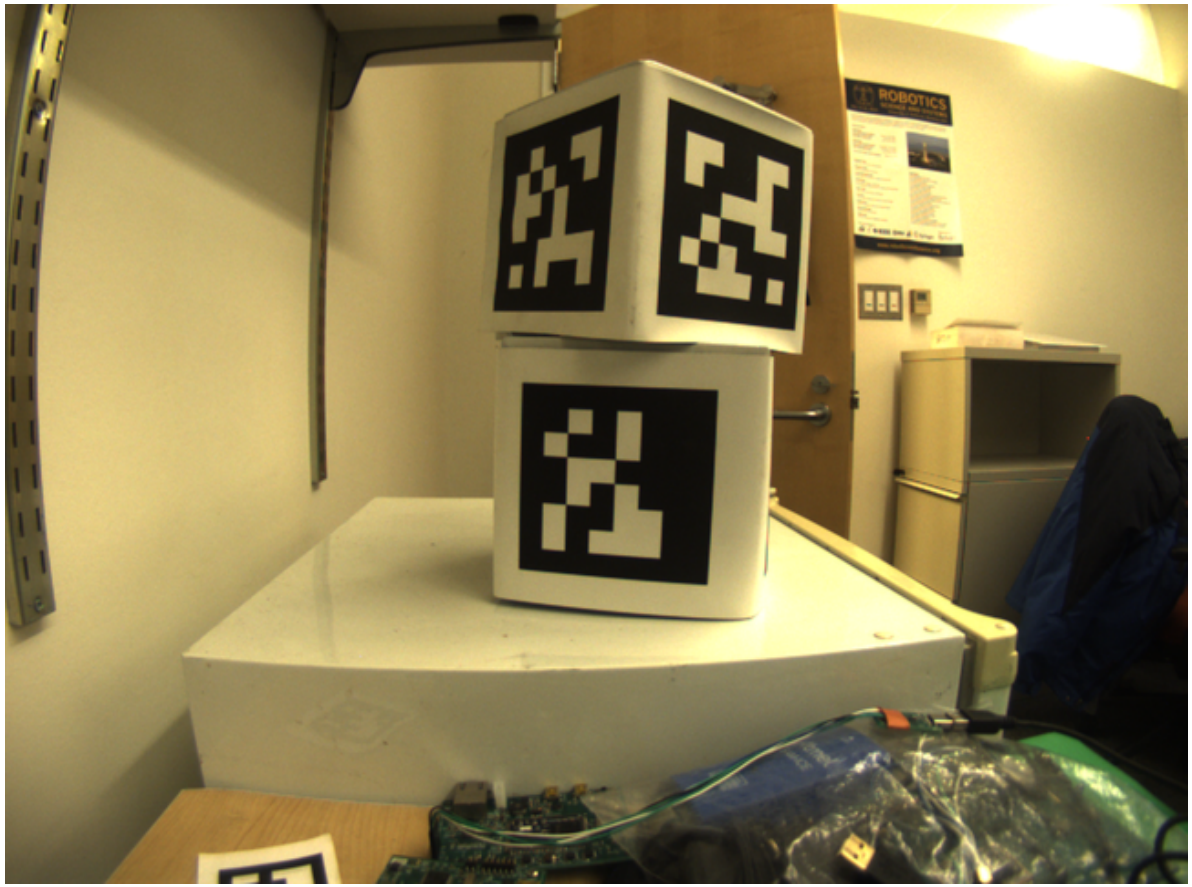


From *DeGol et a., ChromaTag: A Colored Marker and Fast Detection Algorithm*

AprilTag

- One of the most used marker types in robotics is the AprilTag
- There are different families available
- The widely used Tag36h11 family offers 587 tags with minimal Hamming distance of 11 bit
- For each tag you get the ID and the pose
- Very low false-positive detection rates
- False-negative detection highly depends on image quality
- Multiple can be grouped as bundle to get better measurements

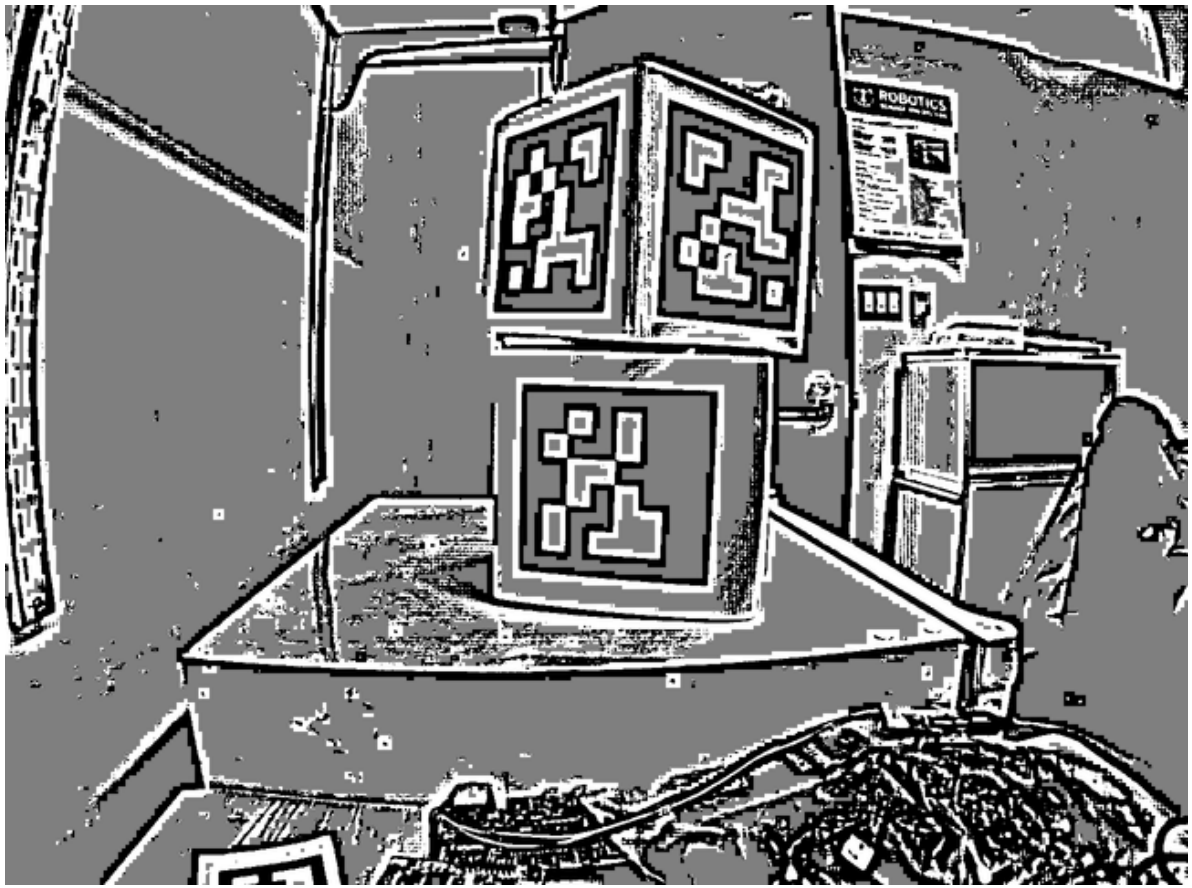
AprilTag Detection



Original image

From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

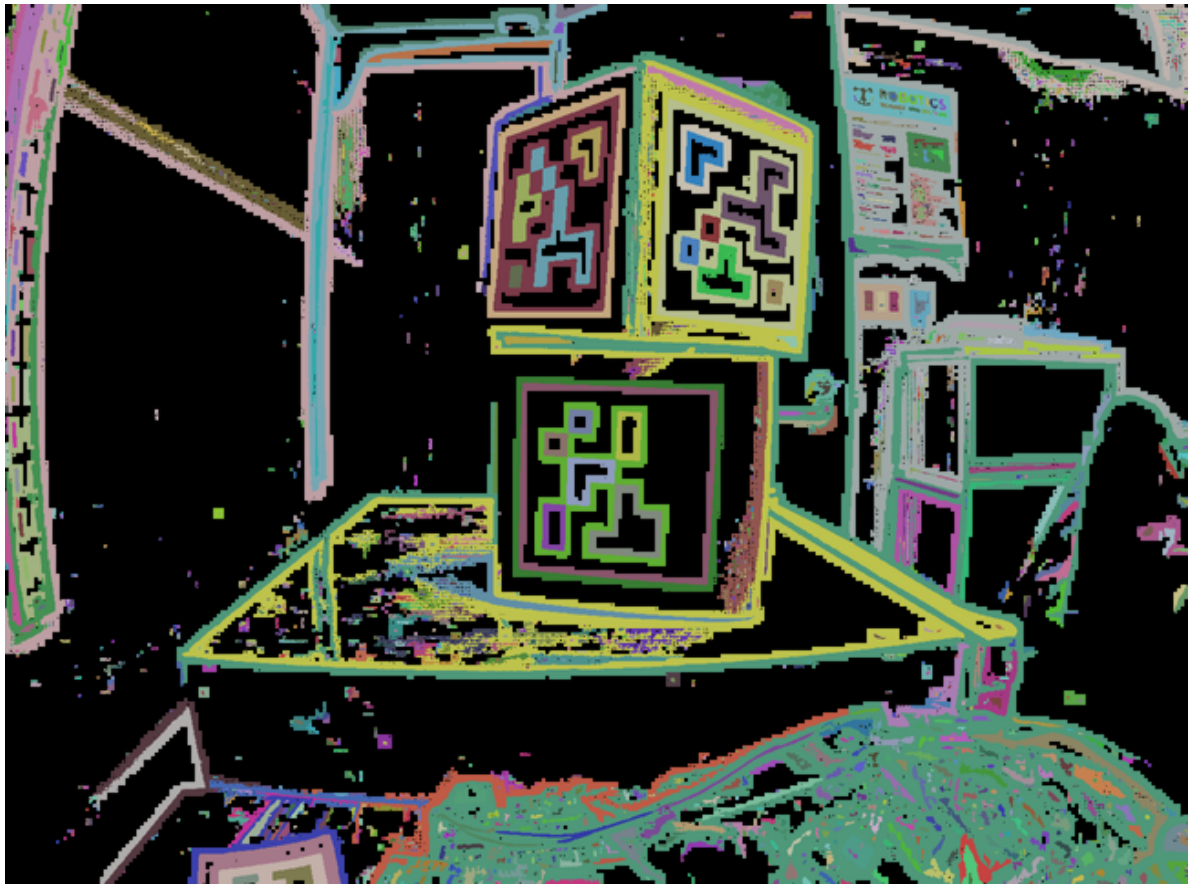
AprilTag Detection (cont.)



Adaptive thresholding

From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

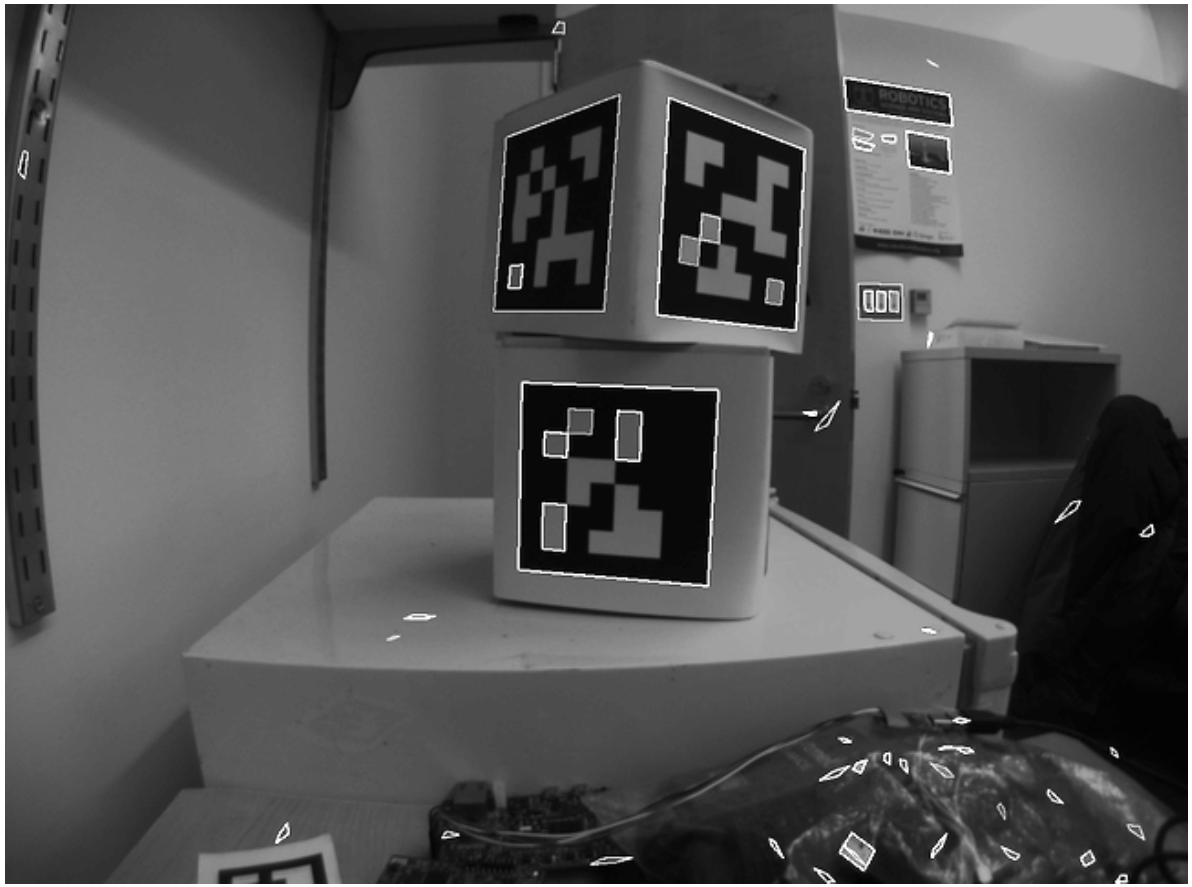
AprilTag Detection (cont.)



Segmentation

From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

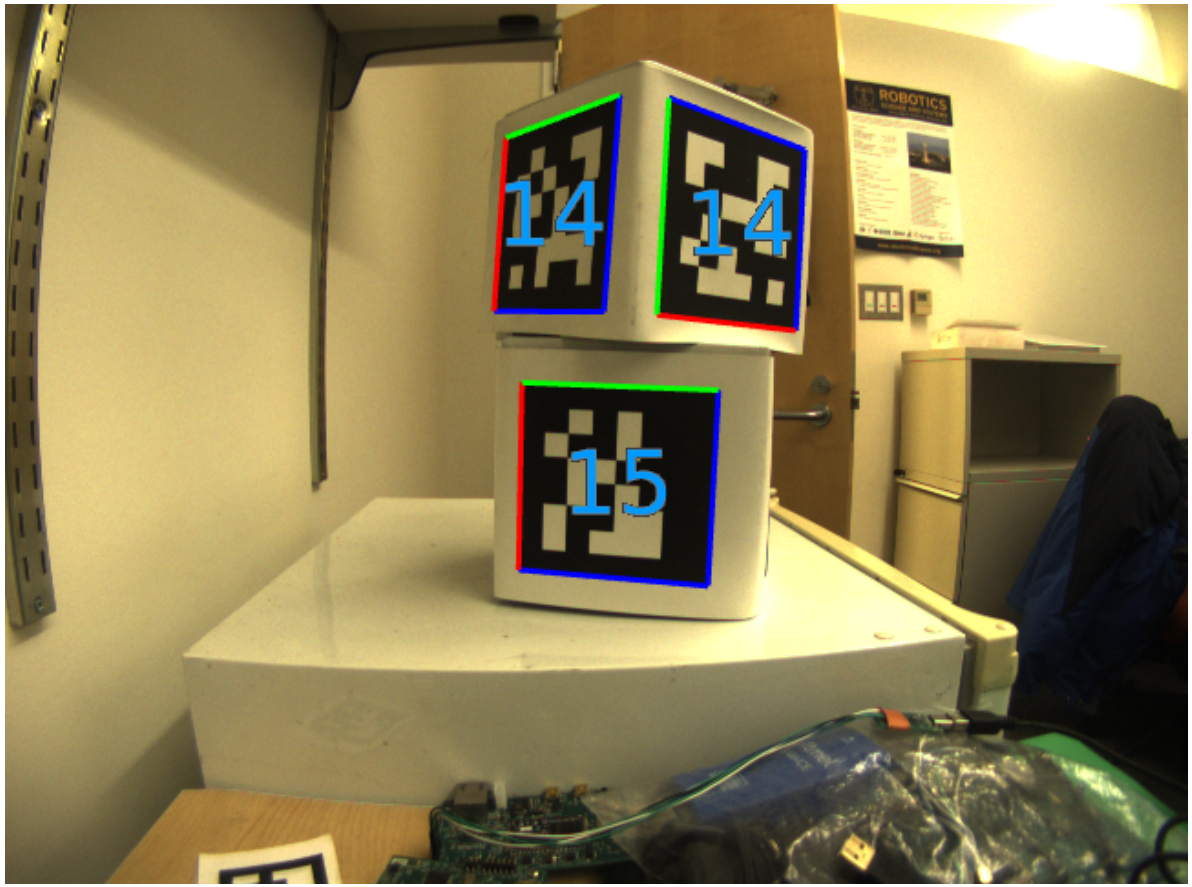
AprilTag Detection (cont.)



Detected quads

From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

AprilTag Detection (cont.)



Tag detections

From *John Wang and Edwin Olson, AprilTag 2: Efficient and robust fiducial detection*

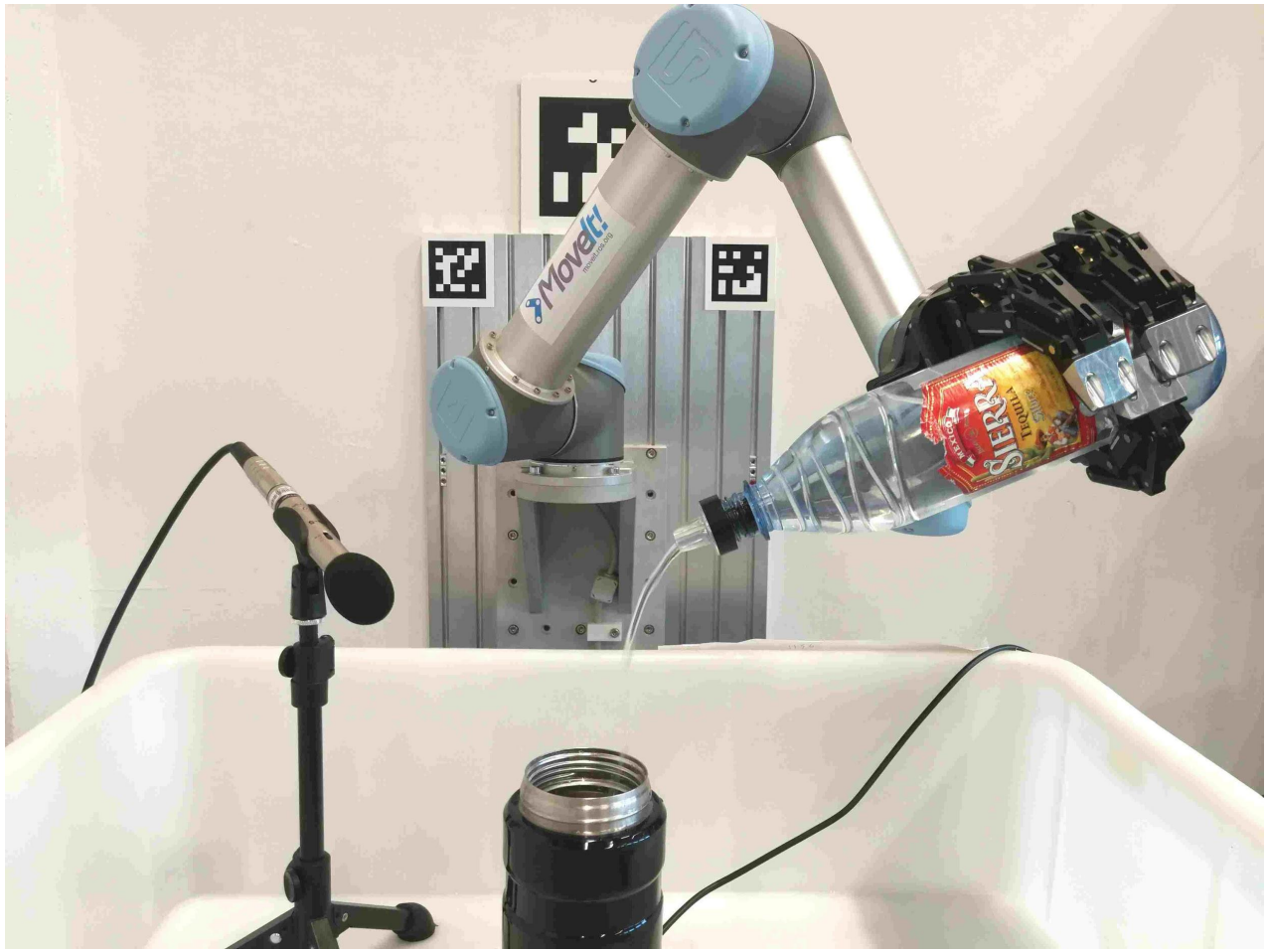
Live Demo

Live demo

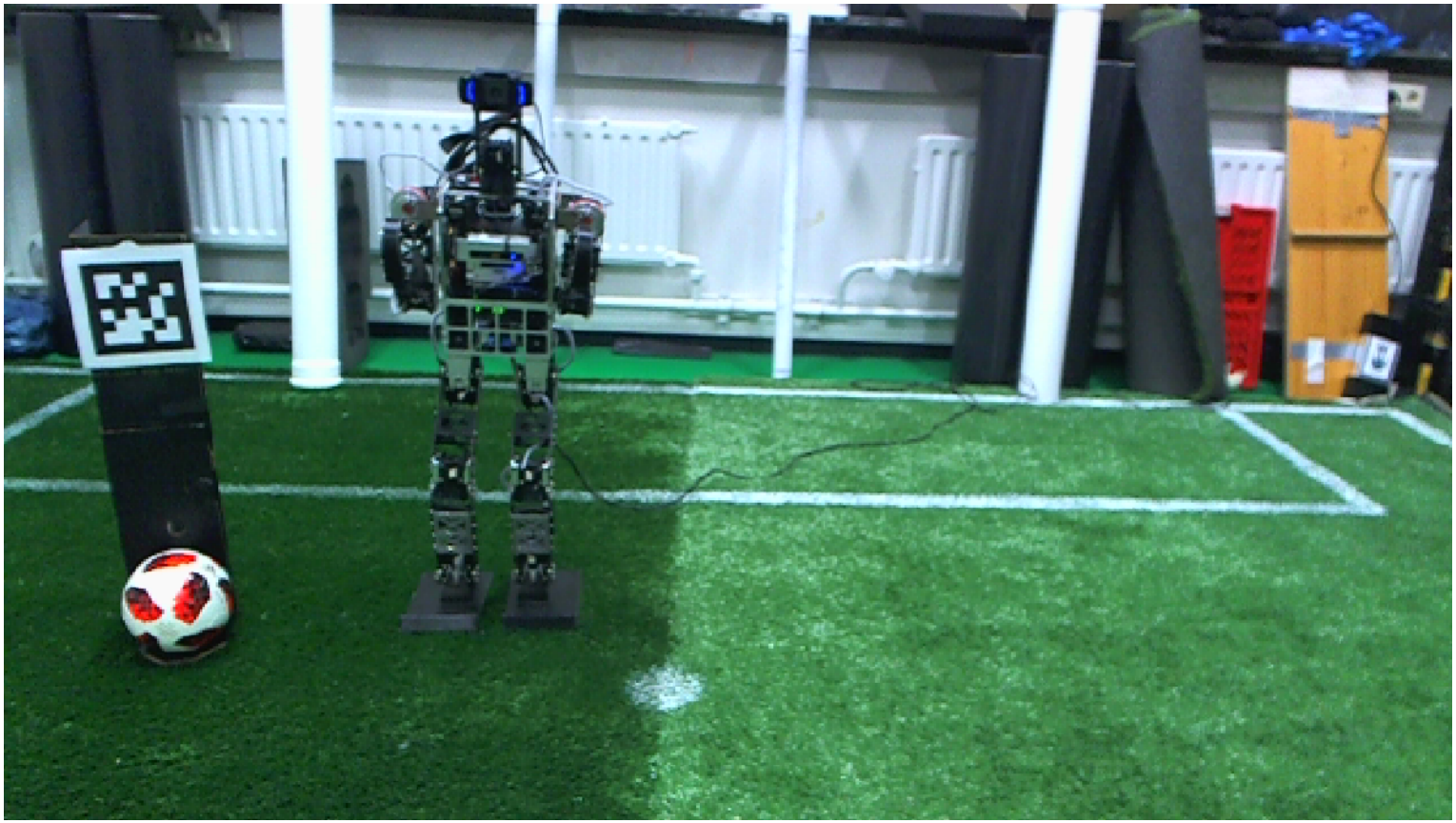
Tag Application Examples

- Extrinsic calibration
- Getting "ground truth" for training data
- Removing the object pose estimation sub task from a larger task, to concentrate on the other aspects

Tag Application Examples



Tag Application Examples



Discussion of Tags

- Pro
  - Simplifies task
  - Production costs very low
  - Computational cost comparably low
  - Normally quiet precise
  - Ease to employ in artificial environments, e.g. factory
- Contra
  - Difficult to use on large distances
  - Difficult to use on very small scales, e.g. getting finger tip positions
  - Not usable for natural environments
  - Tags in training data lead to learning tag detection
  - Rotation is not so precise (can be improved by bundles)

Application Examples

Visually controlled grasping

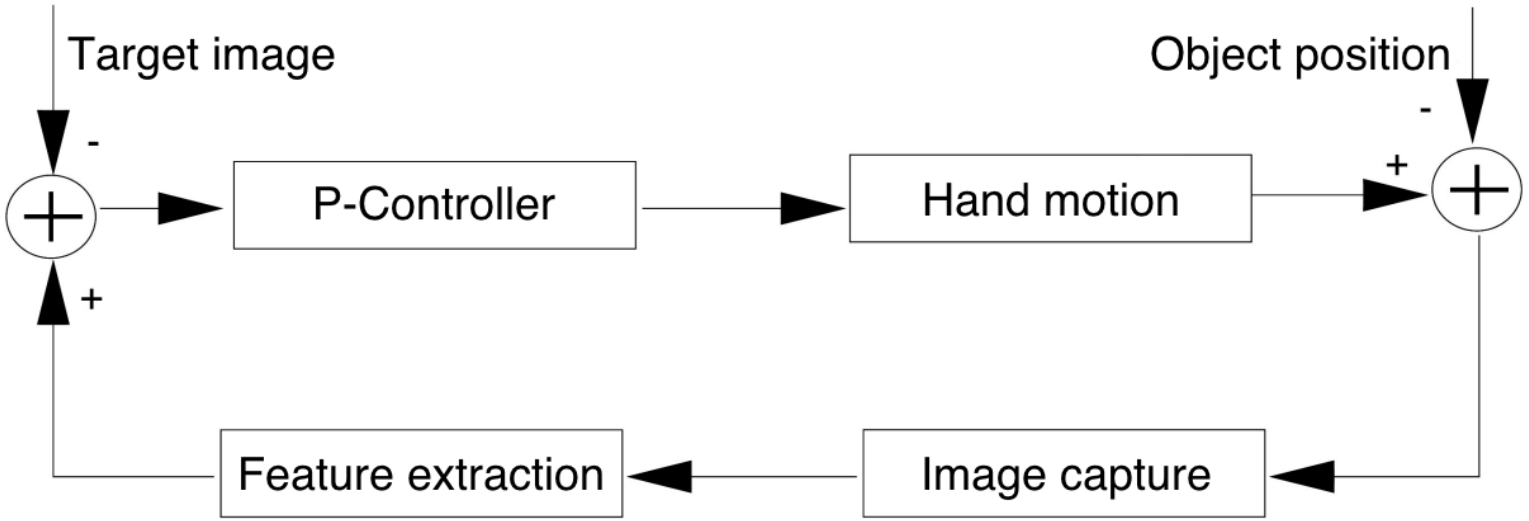
Task:

Two-dimensional fine positioning of a robot hand or a gripper in relation to the object that is to be grasped

Procedure:

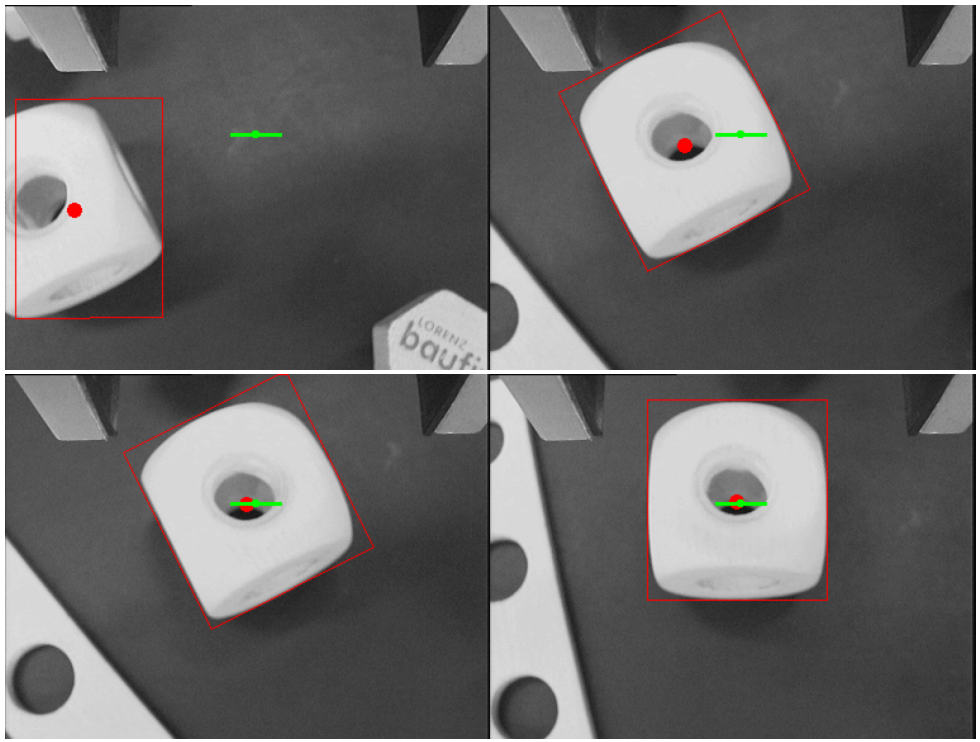
1. Offline-specification of the target position (e.g. object features from stereo image processing)
2. Online transformation of the current difference in relation to the target position (e.g. with a hand camera)

Visually controlled grasping (cont.)



Basic control loop used for visual servoing

Visually controlled grasping (cont.)



Example of visually controlled gripper reaching a suitable position and orientation for grasping of the object

Summary

- Relies on computing transformation sequences between world, object, reference, camera (...) frames
- Camera calibration requires a set of free parameters to be tuned (can be achieved by a suitable optimization technique)
  - Essential for retrieving accurate sensor measurements
  - Is typically done by using a geometric object with structured visual pattern
  - Lens distortion (radial / tangential) needs to be incorporated
- Usage of tags can largely simplify the task