64-424
Intelligent Robotics

https://tams.informatik.uni-hamburg.de/lectures/2018ws/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 2018/2019

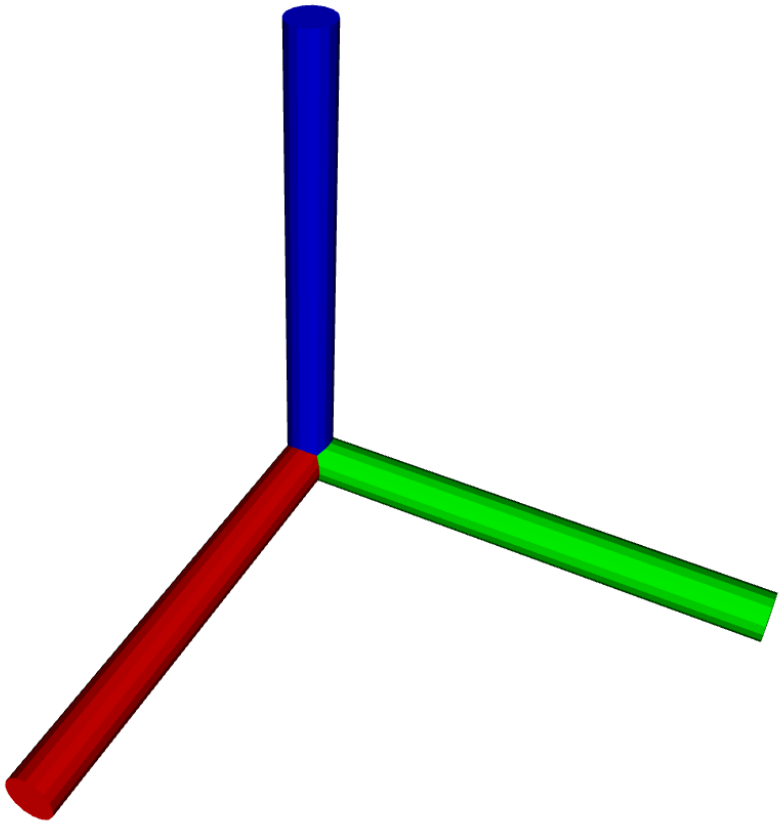
Outline

1. Transformations
2. Vision systems

Outline

1. Transformations
   Coordinate systems
   Further Reading
2. Vision systems

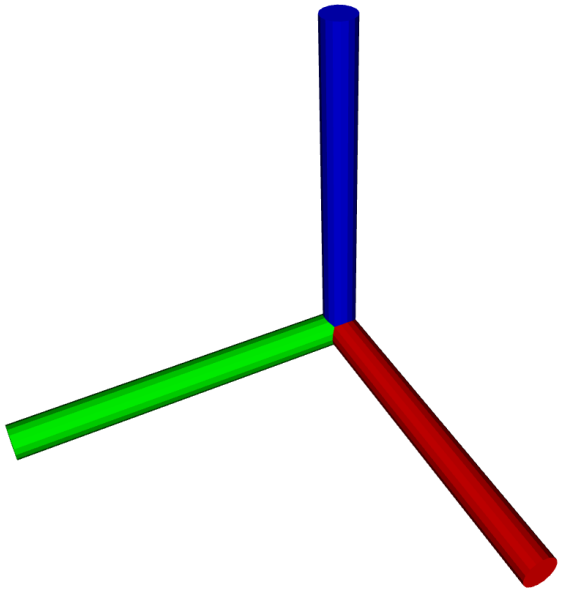
Coordinate Systems



- 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*

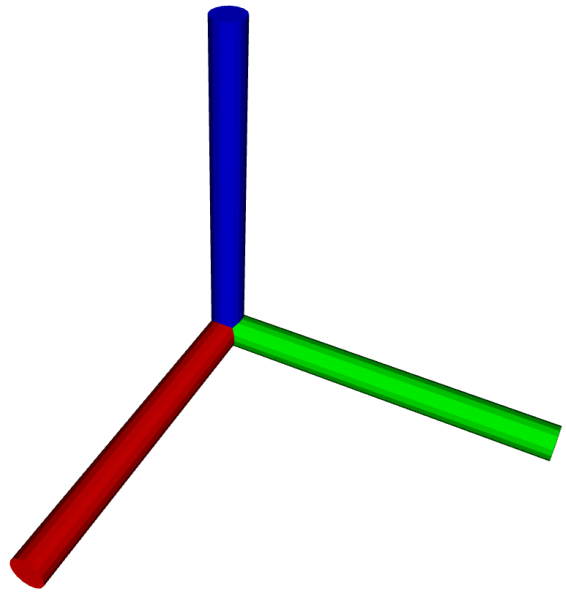
Coordinate Systems - Handedness

Left-Handed



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

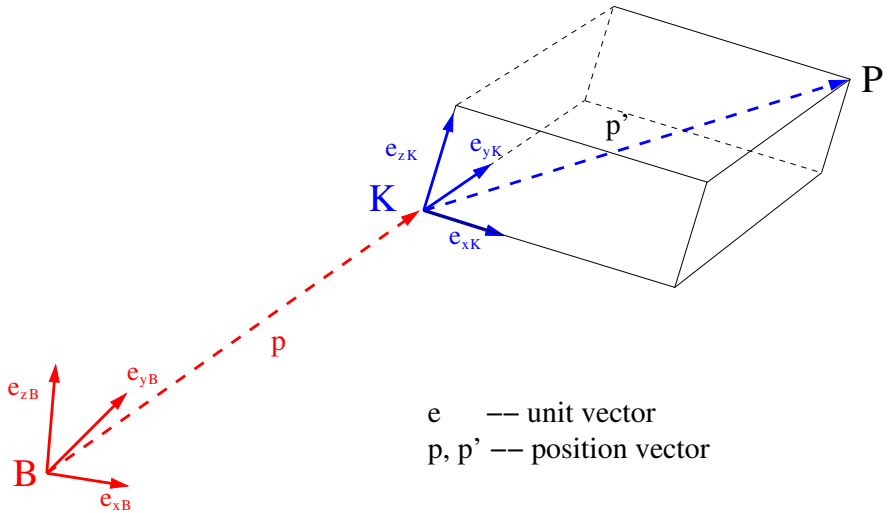
Right-Handed



- 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**)

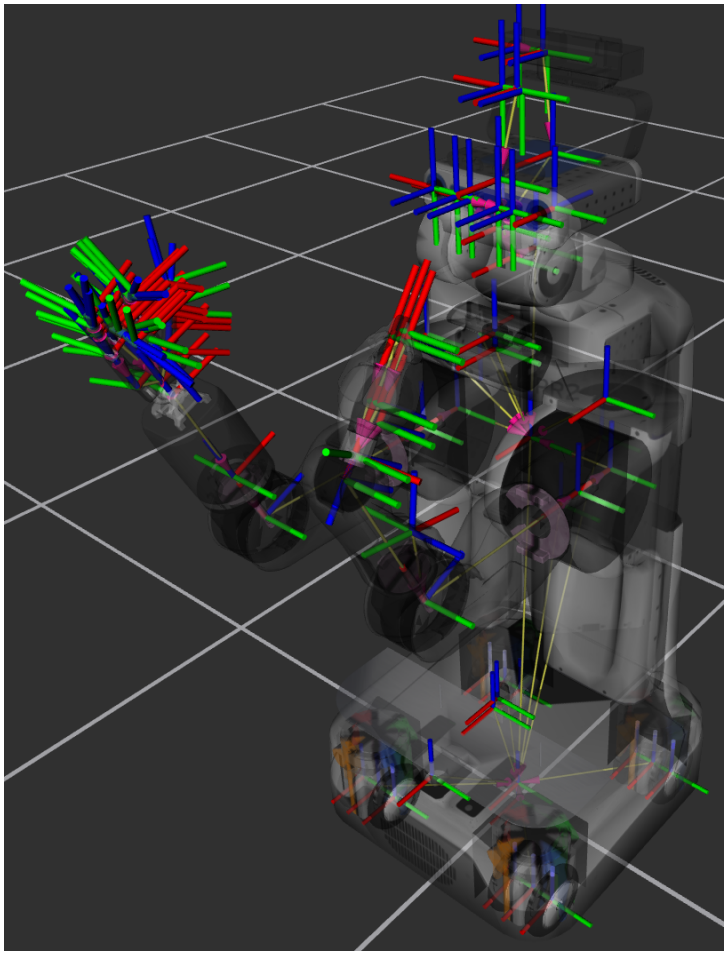


Coordinate transformation

- Transition between coordinate systems (frames)

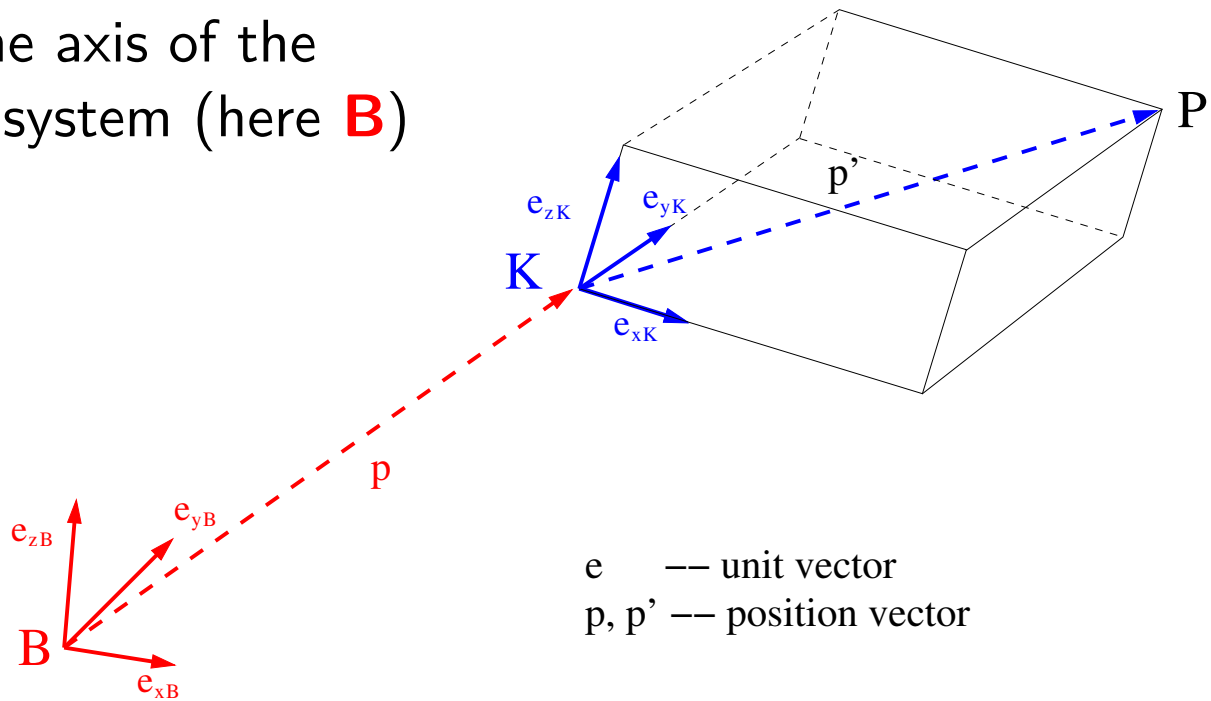
  Typical reference frames:
  - Robot base
  - End-effector (tool)
  - Table (world)
  - Object
  - Camera
  - Screen
  - ...

Frame transformations convert one frame into another.



Relative Pose - Position

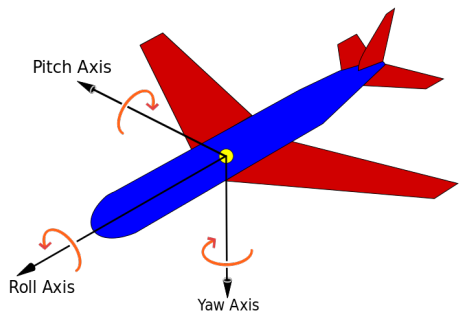
- Coordinates of the Origin
- Translations along the axis of the reference coordinate system (here **B**)

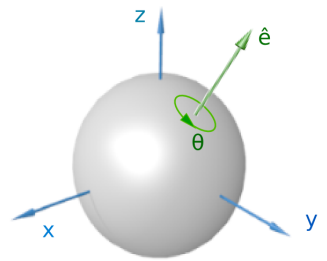


- Defined by: $\mathbf{p} = [p_x, p_y, p_z]^T \in \mathcal{R}^3$ w.r.t. e_{xB}, e_{yB}, e_{zB}

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 $\vec{u}, \phi$
  - 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: $det(R) = 1$





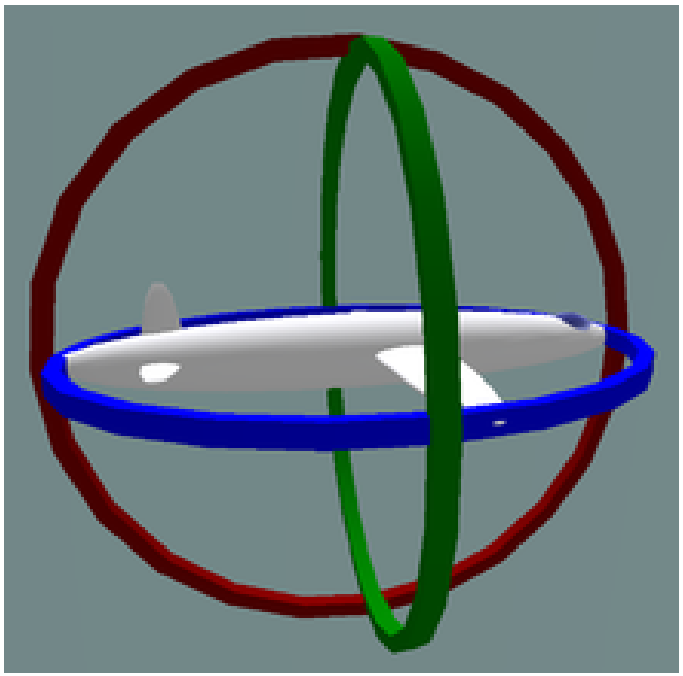
$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

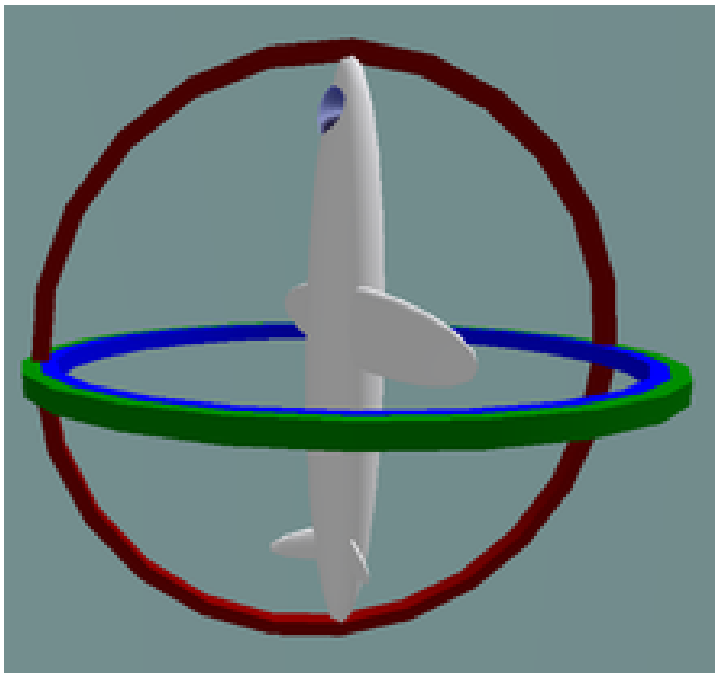
Euler Angles

- 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

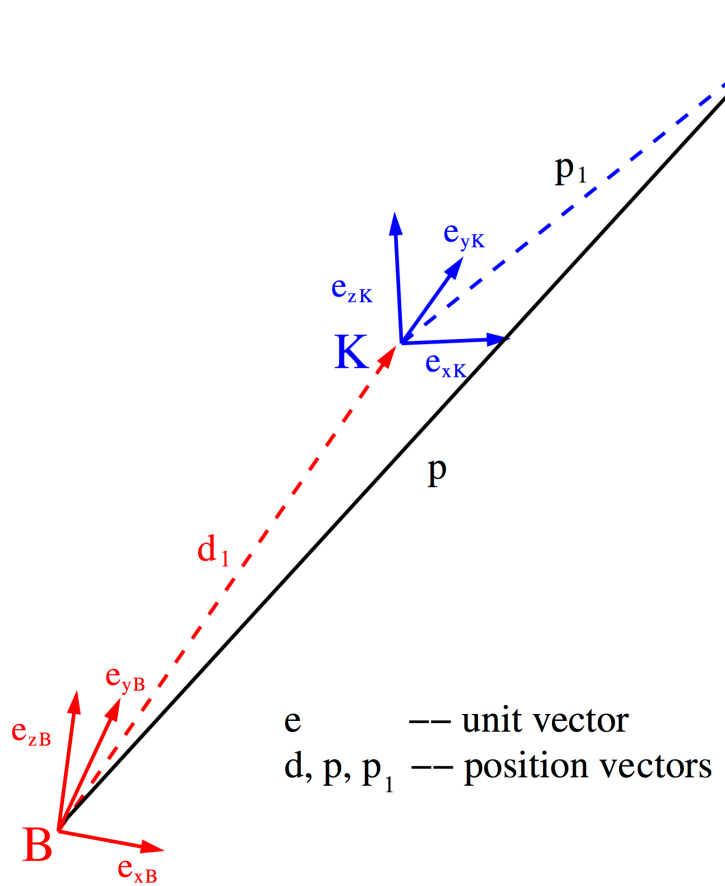
Points in space referenced in a local frame can be transformed into a global frame through:

- 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:

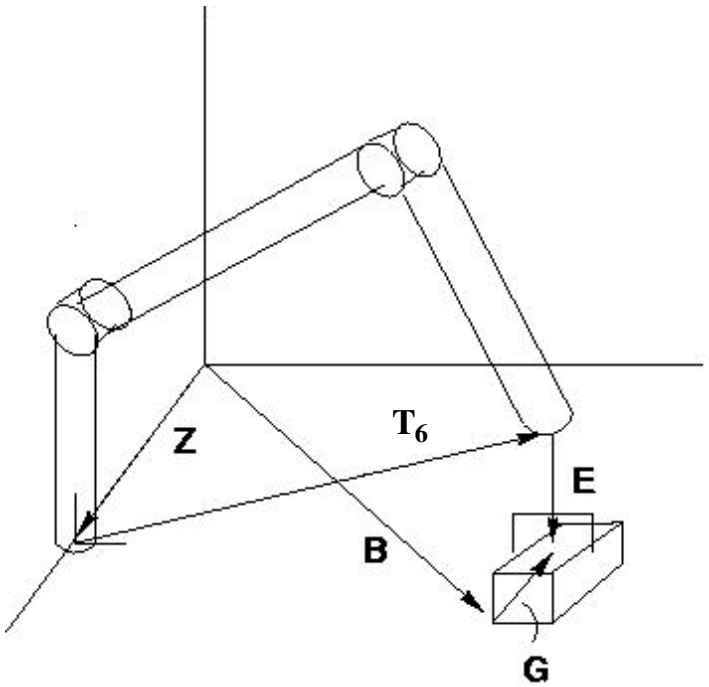
- ${}^{B}R_K$: rotation matrix R, that describes rotations to generate frame K from frame B
- ${}^{K}p$: vector p based on frame K.



Application: Relative transformations

There are the following transformations:

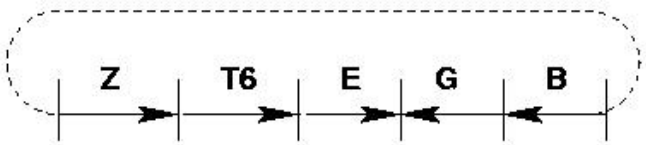
- Z: World $\rightarrow$ Base of the manipulator
- T_6: Base of the manipulator $\rightarrow$ End of the manipulator
- E: End of the manipulator $\rightarrow$ End effector
- B: World $\rightarrow$ Object
- G: Object $\rightarrow$ Grasp Pose



Application: Transformation chain

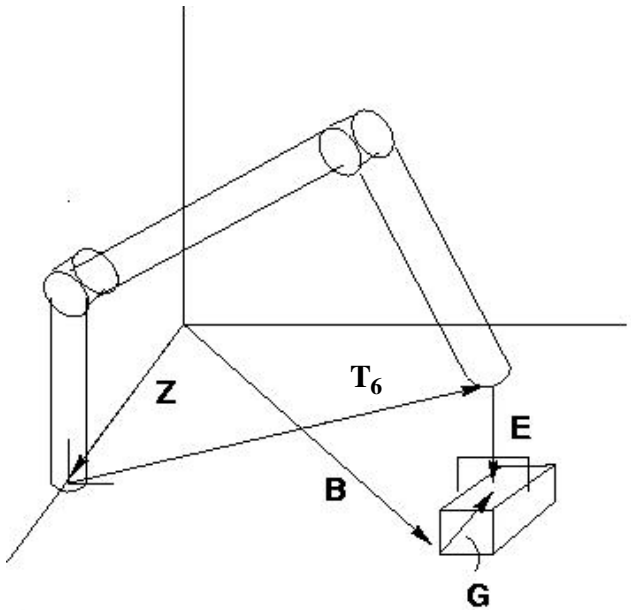
There are two descriptions of the end effector's frame, one in relation to the object and the other in relation to the manipulator. Both descriptions are equal:

$$ZT_6E = BG$$



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:

- tf (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.

Further Reading

— ADDITIONAL SLIDES FOR FURTHER READING —

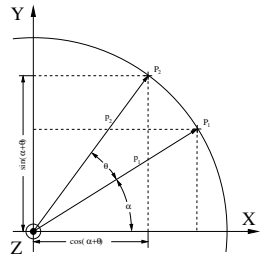
Rotations in Euclidean plane $\mathcal{R}^2$

- Counterclockwise rotation of a vector $\vec{p_1}$ in $\mathcal{R}^2$ on the unit circle by angle θ

$$\vec{p_1} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad \begin{aligned} x_1 &= \cos(\alpha) \\ y_1 &= \sin(\alpha) \end{aligned} \quad \vec{p_2} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \quad \vec{p_2} = R(\vec{p_1}, \theta)$$

$$\begin{aligned} x_2 &= \cos(\alpha + \theta) = \cos\alpha \cdot \cos\theta - \sin\alpha \cdot \sin\theta = x_1 \cos\theta - y_1 sin\theta \\ y_2 &= \sin(\alpha + \theta) = \sin\alpha \cdot \cos\theta + \cos\alpha \cdot \sin\theta = y_1 \cos\theta + x_1 sin\theta \end{aligned}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$



- Consequently

$$\vec{p_2} = R_\theta \vec{p_1} \qquad with \qquad R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

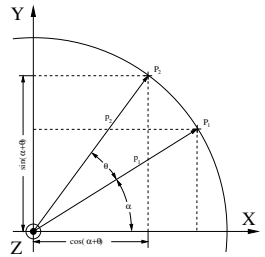
Rotations in Euclidean plane $\mathcal{R}^2$

- Counterclockwise rotation of a vector $\vec{p_1}$ in $\mathcal{R}^2$ on the unit circle by angle θ

$$\vec{p_1} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad \begin{matrix} x_1 = \cos(\alpha) \\ y_1 = \sin(\alpha) \end{matrix} \quad \vec{p_2} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \quad \vec{p_2} = R(\vec{p_1}, \theta)$$

$$\begin{aligned} x_2 &= \cos(\alpha + \theta) = \cos\alpha \cdot \cos\theta - \sin\alpha \cdot \sin\theta = x_1 \cos\theta - y_1 sin\theta \\ y_2 &= \sin(\alpha + \theta) = \sin\alpha \cdot \cos\theta + \cos\alpha \cdot \sin\theta = y_1 \cos\theta + x_1 sin\theta \end{aligned}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$



- Consequently

$$\vec{p_2} = R_\theta \vec{p_1} \qquad with \qquad R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

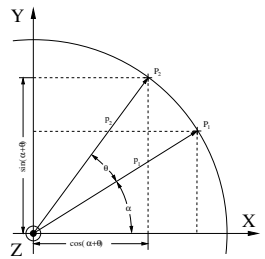
Rotations in Euclidean plane $\mathcal{R}^2$

- Counterclockwise rotation of a vector $\vec{p_1}$ in $\mathcal{R}^2$ on the unit circle by angle θ

$$\vec{p_1} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad \begin{matrix} x_1 = \cos(\alpha) \\ y_1 = \sin(\alpha) \end{matrix} \quad \vec{p_2} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \quad \vec{p_2} = R(\vec{p_1}, \theta)$$

$$\begin{aligned} x_2 &= \cos(\alpha + \theta) = \cos\alpha \cdot \cos\theta - \sin\alpha \cdot \sin\theta = x_1 \cos\theta - y_1 sin\theta \\ y_2 &= \sin(\alpha + \theta) = \sin\alpha \cdot \cos\theta + \cos\alpha \cdot \sin\theta = y_1 \cos\theta + x_1 sin\theta \end{aligned}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$



- Consequently

$$\vec{p_2} = R_\theta \vec{p_1} \qquad with \qquad R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Elemental Rotation around z-axis

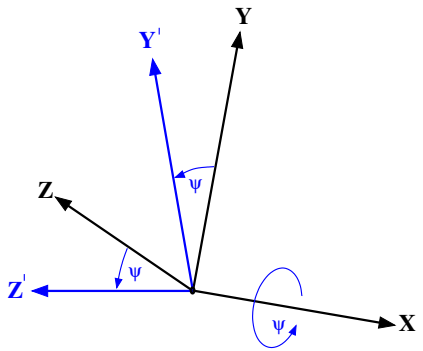
- As before, adding an unchanged third dimension
- Rotation around z-axis by angle ϕ:

$$R_{z,\phi} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Elemental Rotation around x-axis

Rotation around x-axis by angle ψ:

$$R_{x,\psi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}$$



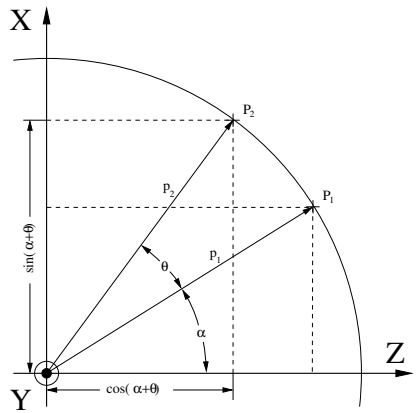
Example:

$$\begin{aligned} \vec{a}'^{\mathrm{T}} &= R_{x,\psi} \cdot \vec{a}^{\mathrm{T}} \\ &= \begin{pmatrix} a_x \\ a_y \cdot \cos(\psi) - a_z \cdot \sin(\psi) \\ a_y \cdot \sin(\psi) + a_z \cdot \cos(\psi) \end{pmatrix} \end{aligned}$$

Elemental Rotation around y-axis

Rotation around y-axis by angle θ:

$$R_{y,\theta} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$



$$\begin{aligned} z_2 &= \cos(\alpha+\theta) = \cos\alpha \cdot \cos\theta - \sin\alpha \cdot \sin\theta = z_1 \cos\theta - x_1 sin\theta \\ x_2 &= \sin(\alpha+\theta) = \sin\alpha \cdot \cos\theta + \cos\alpha \cdot \sin\theta = x_1 \cos\theta + z_1 sin\theta \end{aligned}$$

$$\begin{aligned} x_2 &= x_1 \cos\theta + z_1 sin\theta \\ y_2 &= y_1 \\ z_2 &= -x_1 sin\theta + z_1 \cos\theta \end{aligned} \qquad \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$$

Concatenation of rotations

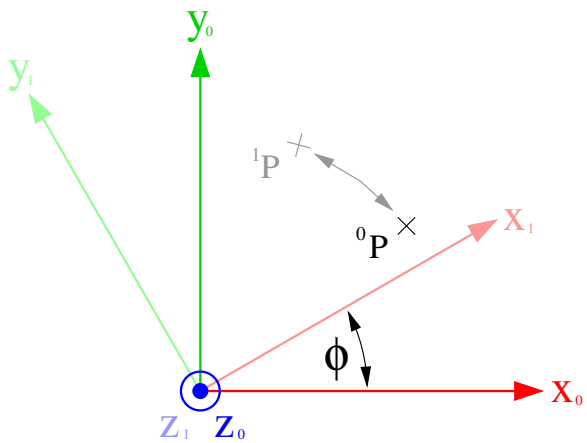
- Multiple rotations can be concatenated
- With rotation matrices, this is done by *matrix multiplication*
- Thus, the result is another rotation matrix $R \in \mathcal{R}^{3\times3}$

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

Sequence of rotations

- Sequence of rotations described by a single matrix ${}^{m}R_{n}$ (Rotation matrix R for frame ${}_{n}$ with reference to frame ^{m})
- Example:
  Frame 1 is rotated in reference to *Frame 0* by angle ϕ around the z-axis

$$ {}^{0}R_{1} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

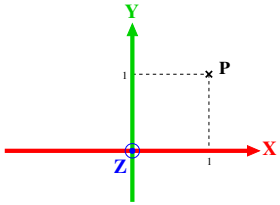


- Transformation of point P:
  - Multiplication from the left: ${p'}^{T} = {}^{0}R_{1} \cdot p^{T}$
  - Multiplication from the right: $p'' = p \cdot {}^{0}R_{1}$

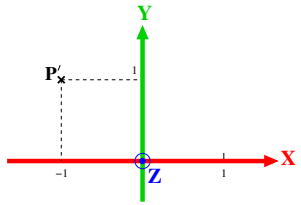
Left side multiplication (${p'}^T = {}^0R_1 \cdot p^T$)

$$\begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} p_x \cos(\phi) - p_y \sin(\phi) \\ p_x \sin(\phi) + p_y \cos(\phi) \\ p_z \end{bmatrix}$$

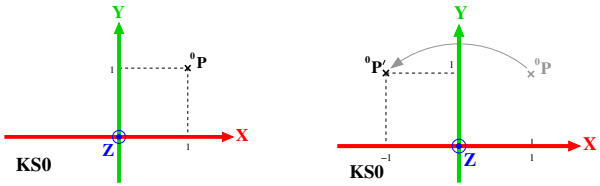
- Illustrated with $\phi = 90°$ and $p = \begin{bmatrix} 1 & 1 & * \end{bmatrix}^T$



$${}^0R_1 \cdot \begin{bmatrix} 1 & 1 & * \end{bmatrix}^T = \begin{bmatrix} -1 & 1 & * \end{bmatrix}^T$$



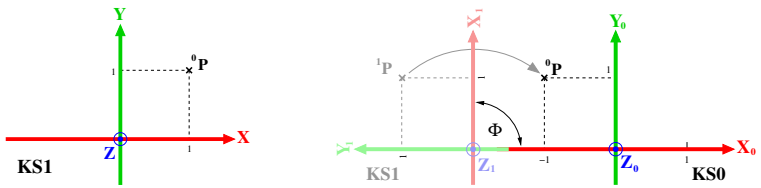
- p in reference to *Frame 0*



p rotated by θ in *Frame 0*

$$({}^0p')^T = {}^0R_1 \cdot ({}^0p)^T$$

- p in reference to *Frame 1*



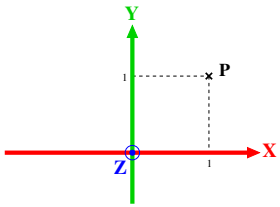
$$({}^0p)^T = {}^0R_1 \cdot ({}^1p)^T$$

Transformation of ${}^1p$ to ${}^0p$

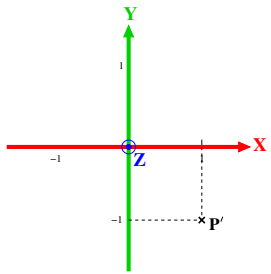
Right side multiplication ($p' = p \cdot {}^0R_1$)

$$\begin{bmatrix} p_x & p_y & p_z \end{bmatrix} \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} p_x \cos(\phi) - p_y \sin(\phi) & -p_x \sin(\phi) + p_y \cos(\phi) & p_z \end{bmatrix}$$

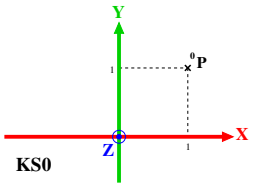
- Illustrated with $\phi = 90°$ und $p = \begin{bmatrix} 1 & 1 & * \end{bmatrix}$

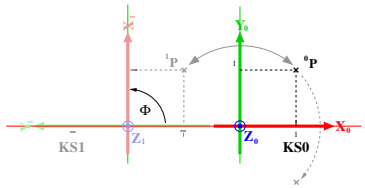


$$\begin{bmatrix} 1 & 1 & * \end{bmatrix} \cdot {}^0R_1 = \begin{bmatrix} 1 & -1 & * \end{bmatrix}$$



- p in reference to *Frame 0*

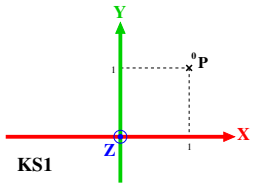


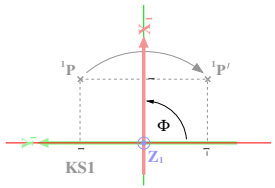


Transformation of ${}^0p$ to ${}^1p$

$${}^1p = {}^0p \cdot {}^0R_1$$

- p in reference to *Frame 1*





p rotated by $-\phi$ in *Frame 1*

$${}^1p' = {}^1p \cdot {}^0R_1$$

Left vs. right side multiplication

- ${}^0R_1 \cdot p$ (left side multiplication)

$$({}^0p)^T = {}^0R_1 \cdot ({}^1p)^T \quad \text{Transformation of coordinates from } F_1 \longrightarrow F_0 \tag{1}$$

$$({}^0p')^T = {}^0R_1 \cdot ({}^0p)^T \quad \text{Point of } F_0 \text{ rotated by } \phi$$

- $p \cdot {}^0R_1$ (right side multiplication)

$${}^1p = {}^0p \cdot {}^0R_1 \quad \text{Transformation of coordinates from } F_0 \longrightarrow F_1 \tag{2}$$

$${}^1p' = {}^1p \cdot {}^0R_1 \quad \text{Point of } F_1 \text{ rotated by } -\phi$$

Left vs. right side multiplication (cont.)

- ${}^0R_1 \cdot p$ (multiplication from left)

$$({}^0p)^T = {}^0R_1 \cdot ({}^1p)^T \quad \text{Transformation of coordinates from } F_1 \longrightarrow F_0$$

$$({}^1p)^T = {}^1R_0 \cdot ({}^0p)^T \quad \text{Transformation of coordinates from } F_0 \longrightarrow F_1 \qquad (3)$$

- $p \cdot {}^0R_1$ (multiplication from right)

$${}^1p = {}^0p \cdot {}^0R_1 \quad \text{Transformation of coordinates from } F_0 \longrightarrow F_1$$

$${}^0p = {}^1p \cdot {}^1R_0 \quad \text{Transformation of coordinates } F_1 \longrightarrow F_0 \qquad (4)$$

Concatenation of rotations - extrinsic X-Y-Z

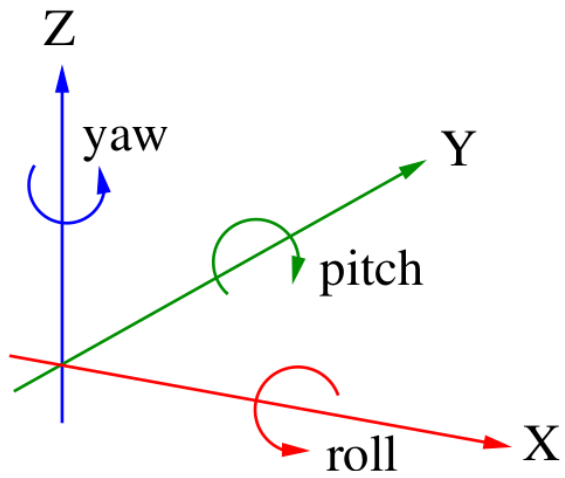
Rotations apply to the non-rotated axes of the original frame:

Sequential left side multiplications of the transformation matrices by sequence of rotation

Example:
Consecutive rotation around X-Y-Z (RPY):

1. Rotation by ψ around x-axis $R_{x,\psi}$ (roll)
2. Rotation by θ around y-axis $R_{y,\theta}$ (pitch)
3. Rotation by ϕ around z-axis $R_{z,\phi}$ (yaw)

$$a'^{\mathrm{T}} = R_{z,\phi} \cdot R_{y,\theta} \cdot R_{x,\psi} \cdot a^{\mathrm{T}}$$



Concatenation of rotations - extrinsic X-Y-Z

$$X_\psi\text{-}Y_\theta\text{-}Z_\phi: \qquad R_{\phi,\theta,\psi} = R_{z,\phi} R_{y,\theta} R_{x,\psi}$$

$$= \begin{bmatrix} C\phi & -S\phi & 0 \\ S\phi & C\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\psi & -S\psi \\ 0 & S\psi & C\psi \end{bmatrix}$$

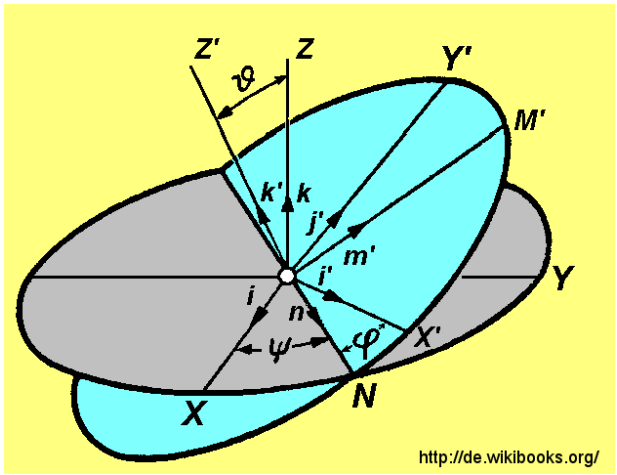
$$= \begin{bmatrix} C\phi C\theta & C\phi S\theta S\psi - S\phi C\psi & C\phi S\theta C\psi + S\phi S\psi \\ S\phi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\phi S\theta C\psi - C\phi S\psi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix}$$

Concatenation of rotations - intrinsic Z-X'-Z''

Rotations apply to the new, currently transformed axes:

Example: rotation around Z-X'-Z'':

1. Rotation ψ around the z-axis $R_{z,\psi}$
2. Rotation θ around the new x-axis $R_{x',\theta}$
3. Rotation ϕ around the new z-axis $R_{z'',\phi}$



Note:
The rotation sequence corresponds to rotations around the fixed axes in reverse order

1. Rotation around Z-axis by ϕ
2. Rotation around X-axis by θ
3. Rotation around Z-axis by ψ

$$a' = R_{z,\psi} \cdot R_{x,\theta} \cdot R_{z,\phi} \cdot a$$

Concatenation of rotations - intrinsic Z-X'-Z''

$$Z_\phi\text{-}X'_\theta\text{-}Z''_\psi: \qquad R_{\psi,\theta,\phi} = R_{z,\psi} R_{x,\theta} R_{z,\phi}$$

Around rotated axes — Around fixed axes

$$= \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta & -S\theta \\ 0 & S\theta & C\theta \end{bmatrix} \begin{bmatrix} C\phi & -S\phi & 0 \\ S\phi & C\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\psi C\phi - S\psi C\theta S\phi & -C\psi S\phi - S\psi C\theta C\phi & S\psi S\theta \\ S\psi C\phi + C\psi C\theta S\phi & -S\psi S\phi + C\psi C\theta C\phi & -C\psi S\theta \\ S\theta S\phi & S\theta C\phi & C\theta \end{bmatrix}$$

Note: Matrix multiplication is non-commutative:

$$AB \neq BA$$

Notation of transformation

- Matrix multiplication using a 3×3 matrix can specify rotation, scaling or shear
  - **But:** Translation requires vector addition
- This can be handled by adding an additional column to the matrix
  - **But:** The matrix is no longer invertible
- Transition to $\mathcal{R}^4$ (4×4 matrix), to specify rotation, translation, shear, scaling and projection
- Use of homogeneous coordinates

Homogeneous coordinates

- Homogeneous coordinates are known from the field of computer graphics, to circumvent problems in matrix computation
- The points of an n-dimensional space are illustrated in an $n+1$-dimensional space
- $p = (x, y, z) \in \mathcal{R}^3$ becomes $p' = (hx, hy, hz, h) \in \mathcal{R}^4$, with $h \neq 0 \in \mathcal{R}$

  Example:

  $(2, 5, 4) \in \mathcal{R}^3 \longrightarrow .., (1, 2.5, 2, 0.5), .., (2, 5, 4, 1), .., (4, 10, 8, 2), .. \in \mathcal{R}^4$
- In robotics, $h = 1$, which is equivalent to a direct projection between n-dimensional and $(n+1)$-dimensional space
- h is a scaling factor

Homogeneous transformations

$$H = \left[\begin{array}{c|c} \begin{array}{c} \textit{Rotation} \\ \textit{Shear} \\ \textit{Localscaling} \end{array} & \textit{Translation} \\ \hline \textit{Projection} & \textit{Scaling} \end{array}\right]$$

$$= \left[\begin{array}{c|c} \begin{array}{ccc} 3 & & \\ & \times & \\ & & 3 \end{array} & \begin{array}{c} 3 \\ \times \\ 1 \end{array} \\ \hline 1 \times 3 & 1 \times 1 \end{array}\right]$$

Homogeneous transformations (cont.)

- In robotics, only rotation and translation are important. Therefore the homogeneous coordinates are specified as:

$$H = \left[\begin{array}{ccc|c} & \textit{Rotation} & & \textit{Translation} \\ - & - & - & - \\ 0 & 0 & 0 & 1 \end{array}\right]$$

- Using $\vec{p}$ and R the result is: $H = \begin{bmatrix} & R & & \vec{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathcal{R}^{4\times 4}$
- Concatenation of several H through matrix multiplication
- Not commutative, in other words $A \cdot B \neq B \cdot A$

Translation

Relocation by a vector $p = [p_x, p_y, p_z]^T$:

$$T_{(p_x,p_y,p_z)} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} a'^{\mathrm{T}} &= T_{(p_x,p_y,p_z)} \cdot a^{\mathrm{T}} \qquad \textit{with } a=[a_x,a_y,a_z,1] \\ &= (a_x + p_x,\ a_y + p_y,\ a_z + p_z,\ 1)^{\mathrm{T}} \end{aligned}$$

Rotation around the x-axis

Rotation around the x-axis by the angle ψ:

$$R_{x(\psi)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) & 0 \\ 0 & \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} a'^{\mathrm{T}} &= R_{x(\psi)} \cdot a^{\mathrm{T}} \\ &= \begin{pmatrix} a_x \\ a_y \cdot \cos(\psi) - a_z \cdot \sin(\psi) \\ a_y \cdot \sin(\psi) + a_z \cdot \cos(\psi) \\ 1 \end{pmatrix} \end{aligned}$$

Rotation around y-axis

Rotation around the y-axis by the angle θ:

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around the z-axis

Rotation around the z-axis by the angle ϕ:

$$R_z(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Compound rotation (X_ψ, Y_θ, Z_ϕ)

$$R_{\phi,\theta,\psi} = R_{z,\phi} R_{y,\theta} R_{x,\psi}$$

$$= \begin{bmatrix} C\phi & -S\phi & 0 & 0 \\ S\phi & C\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\psi & -S\psi & 0 \\ 0 & S\psi & C\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\phi C\theta & C\phi S\theta S\psi - S\phi C\psi & C\phi S\theta C\psi + S\phi S\psi & 0 \\ S\phi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\phi S\theta C\psi - C\phi S\psi & 0 \\ -S\theta & C\theta S\psi & C\theta C\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Compound rotation (Z_ψ, X_θ,$'$ Z''_ϕ)

$$R_{\psi,\theta,\phi} = R_{z,\psi} R_{x,\theta} R_{z,\phi}$$

$$= \begin{bmatrix} C\psi & -S\psi & 0 & 0 \\ S\psi & C\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\phi & -S\phi & 0 & 0 \\ S\phi & C\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\psi C\phi - S\psi C\theta S\phi & -C\psi S\phi - S\psi C\theta C\phi & S\psi S\theta & 0 \\ S\psi C\phi + C\psi C\theta S\phi & -S\psi S\phi + C\psi C\theta C\phi & -C\psi S\theta & 0 \\ S\theta S\phi & S\theta C\phi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

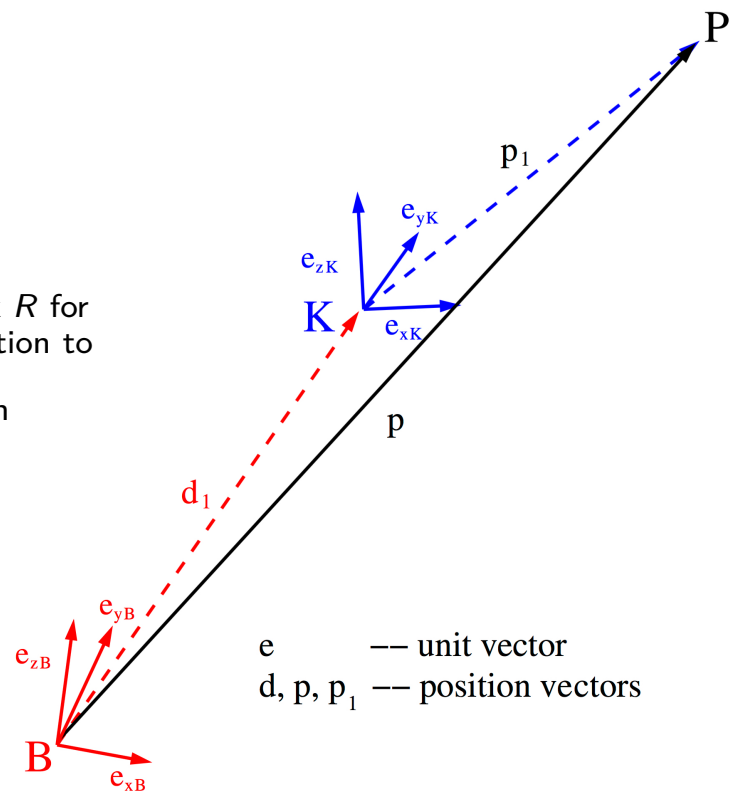
Example: Homogeneous coordinates

Points in space can be described by:

- Position vectors
- Rotation matrix

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

Please note::
${}^{B}R_K$: Rotation matrix R for frame K in relation to frame B
${}^{K}p$: Vector p based on frame K



In homogeneous coordinates:

$$
\begin{aligned}
{}^{B}p_{(H)} &= \begin{pmatrix} {}^{B}d_1 \\ 1 \end{pmatrix} + \begin{bmatrix} {}^{B}R_K & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{K}p_1 \\ 1 \end{pmatrix} \\
&= \begin{bmatrix} {}^{B}R_K & {}^{B}d_1 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{K}p_1 \\ 1 \end{pmatrix} \\
&= {}^{B}H_K \quad {}^{K}p_{(H)} \qquad (5)
\end{aligned}
$$

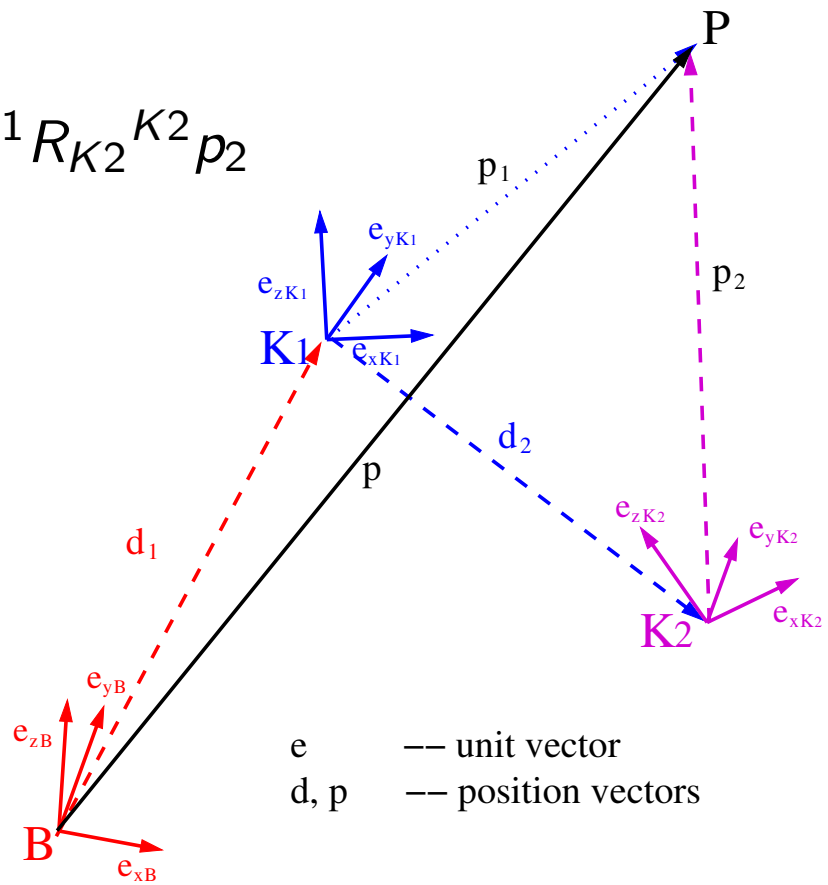
Please note:
${}^{B}H_K$: Homogeneous transf. matrix H for frame K related to frame B;
${}^{K}p_{(H)}$: Point p in homogeneous coordinates based on frame K

Example: Homogeneous coordinates (cont.)

$$
\begin{aligned}
{}^{B}p &= {}^{B}d_1 + {}^{B}d_2 + {}^{B}p_2 \\
&= {}^{B}d_1 + {}^{B}R_{K1} \cdot {}^{K1}d_2 + {}^{B}R_{K1}\,{}^{K1}R_{K2}\,{}^{K2}p_2
\end{aligned}
$$

With homogeneous coordinates:

$$
{}^{B}p_{(H)} = {}^{B}H_{K1} \quad {}^{K1}H_{K2} \quad {}^{K2}p_{2(H)}
$$



Coordinate frames

Coordinate frames are represented by four vectors of a homogeneous transformation

$$H = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Inverse transformations

The inverse of a rotation matrix is simply its transpose:

$$R^{-1} = R^T \text{ und } RR^T = I$$

where I is the indentity matrix.
The inverse of (6) is:

$$H^{-1} = \begin{bmatrix} r_{11} & r_{21} & r_{31} & -\mathbf{p^T} \cdot \mathbf{r}_1 \\ r_{12} & r_{22} & r_{32} & -\mathbf{p^T} \cdot \mathbf{r}_2 \\ r_{13} & r_{23} & r_{33} & -\mathbf{p^T} \cdot \mathbf{r}_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$ and $\mathbf{p}$ are the four column vectors of (6) and $\cdot$ the scalar product of vectors.

Homogeneous transformations

- A homogeneous transformation describes the position and orientation of a coordinate frame in space
- When defining a *coordinate frame* using a solid object, the coordinate frame also explicitly specifies the position and orientation of the solid object
- A homogeneous transformation can be decomposed into a rotation and a translation.
- In order to reverse a homogeneous transformation its inverse needs to be used.
- Several translations and rotations can be accumulated with respect to the sequence of transformations

Literature list

[1] Reza N. Jazar.
Theory of Applied Robotics: Kinematics, Dynamics and Control, chapter 2-4, pages 33–231.
Springer New York, 2. edition, 2010.

[2] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar.
Robot Modeling and Control, chapter 2, pages 35–72.
John Wiley & Sons, 2006.

Outline

1. Transformations
2. Vision systems
   - Introduction
   - Camera calibration
   - Lens distortion
   - Further Reading

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

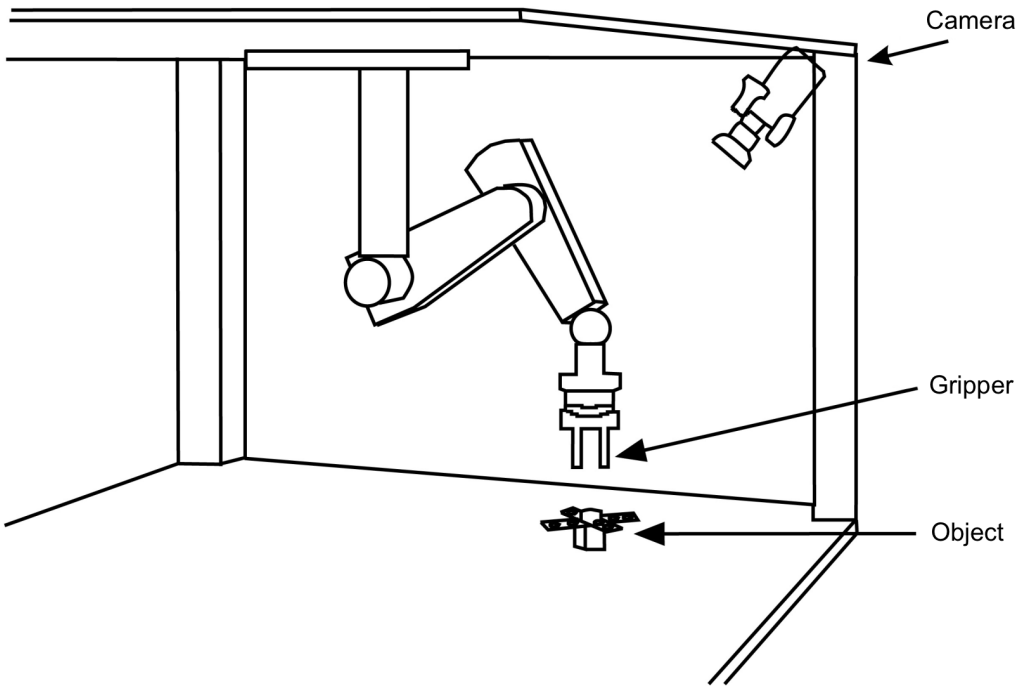
- Robot localization:
  - Relative
  - Absolute
  - In reference to various coordinate frames
- Object recognition and localization
- 3D scene reconstruction
- Allocation of the environment

Introduction (cont.)

Vision-based motion control

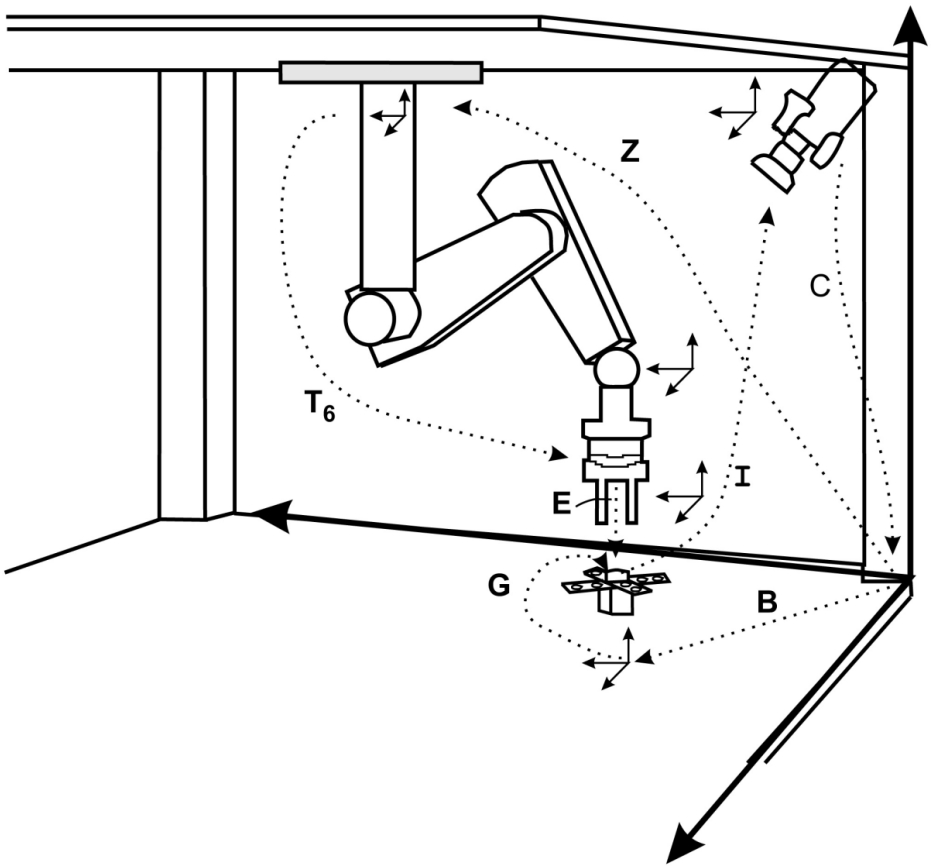
- Visual-Servoing
  - Coarse and fine positioning
  - Tracking of movable objects
- Collision avoidance
  - Depth map based distance measurement
- Coordination with other robots and/or humans
  - Motion estimation
  - Intention recognition

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

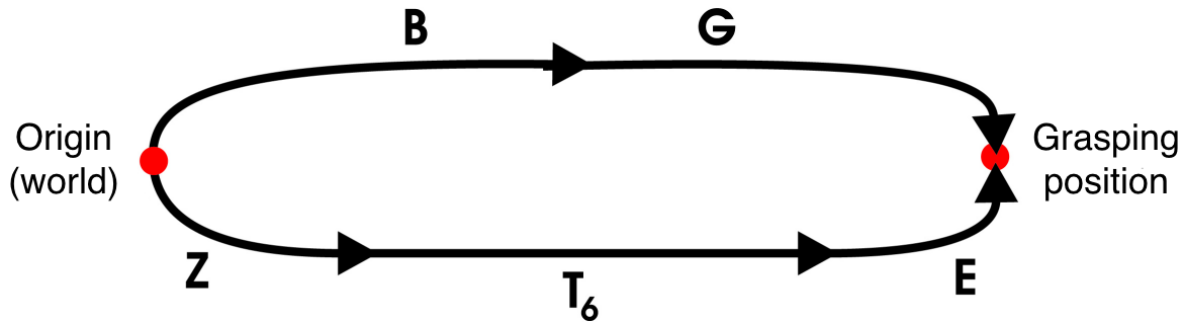
B: Transformation from world coordinates to object coordinates

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:

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

Outline

2. Vision systems
- Introduction
- Camera calibration
- Lens distortion
- Further Reading

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

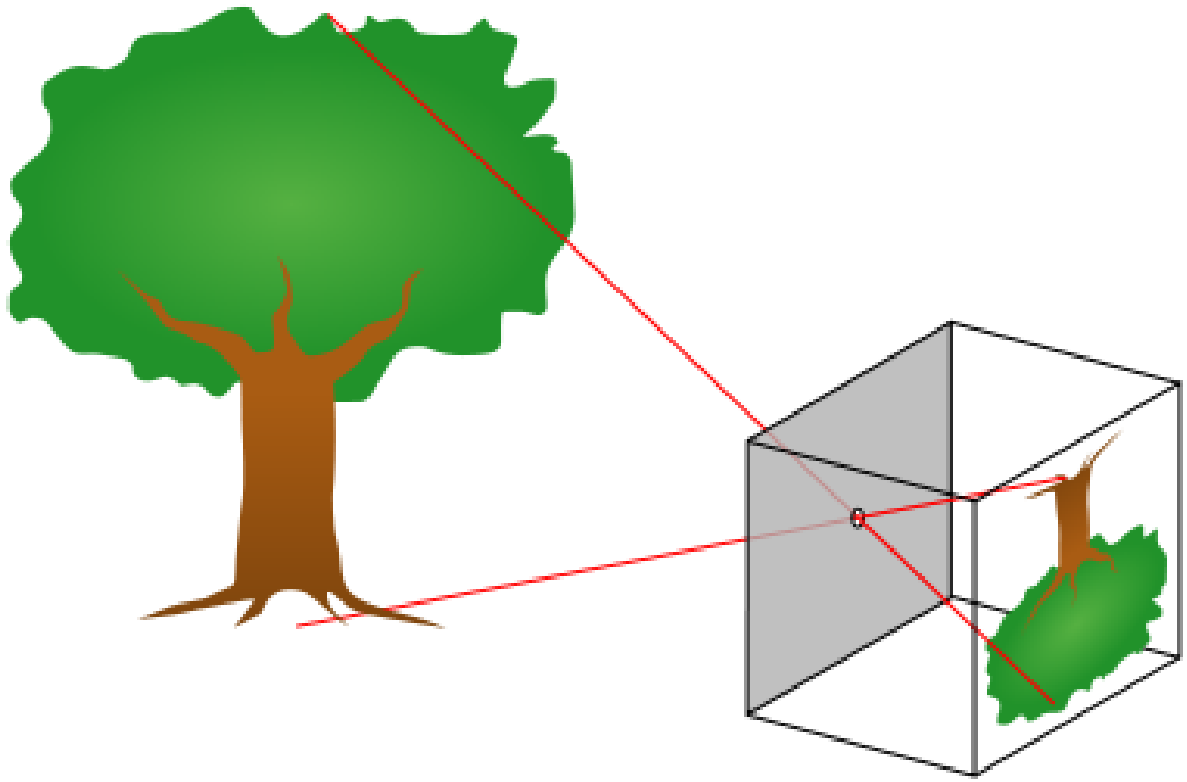
- These transformations are used in various applications:
  - Automatic assembly
  - Robot calibration
  - Tracking
  - Trajectory analysis
  - 3D metrology
  - ...

Camera calibration (cont.)

Several calibration techniques have been established

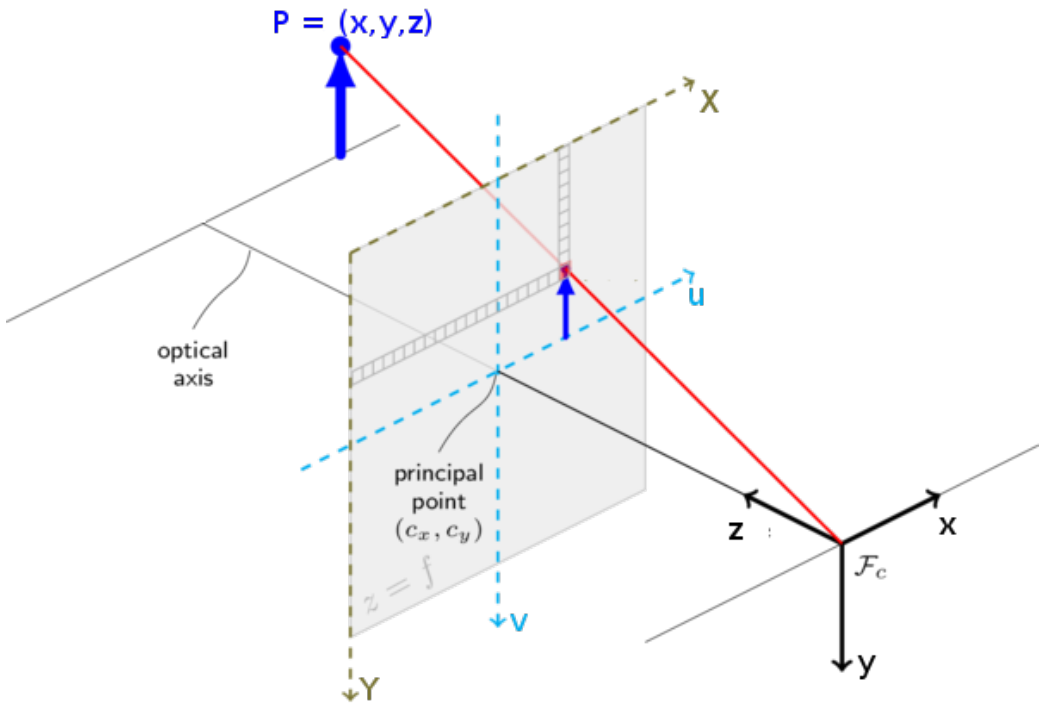
- Camera calibration can be done *on-line* or *off-line*
- Using a calibration object:
  - Identification of the camera parameters
  - Determination of coordinate transformation between camera coordinates and world coordinates
- Using self-calibration approaches
- Using machine learning methods

Pinhole camera model



Pinhole camera without lens distortion

Pinhole camera model (cont.)



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
- (x, y, z): 3D coordinate system of the camera with the origin O (optical center)
- (X, Y): 2D image coordinate system with the origin O_1
- f: Focal length of the camera

Pinhole camera model (cont.)

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

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:

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

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

$$X = s_u u \qquad Y = s_v v$$

- The scaling factors s_u and s_v are used to convert the image coordinates from meters to pixels
- s_u, s_v and f are the intrinsic camera parameters

Pinhole camera model (cont.)

Transformation of world coordinates into image coordinates

- Since only two independent intrinsic parameters exist, one defines:

$$f_x \equiv f s_u \quad \text{and} \quad f_y \equiv f s_v$$

- Previous equations yield the *distortion-free camera model*:

$$X_f = f_x \frac{r_1 x_w + r_2 y_w + r_3 z_w + t_x}{r_7 x_w + r_8 y_w + r_9 z_w + t_z}$$

$$Y_f = f_y \frac{r_4 x_w + r_5 y_w + r_6 z_w + t_y}{r_7 x_w + r_8 y_w + r_9 z_w + t_z}$$

Pinhole camera model (cont.)

Pixel coordinates

- The coordinates (C_x, C_y) of the image center need to be subtracted from the image coordinates (X_f, Y_f) determined during perspective projection
- Due to the above, one has:

$$X = X_f - C_x$$
$$Y = Y_f - C_y$$

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, C_x and C_y

Calibration

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

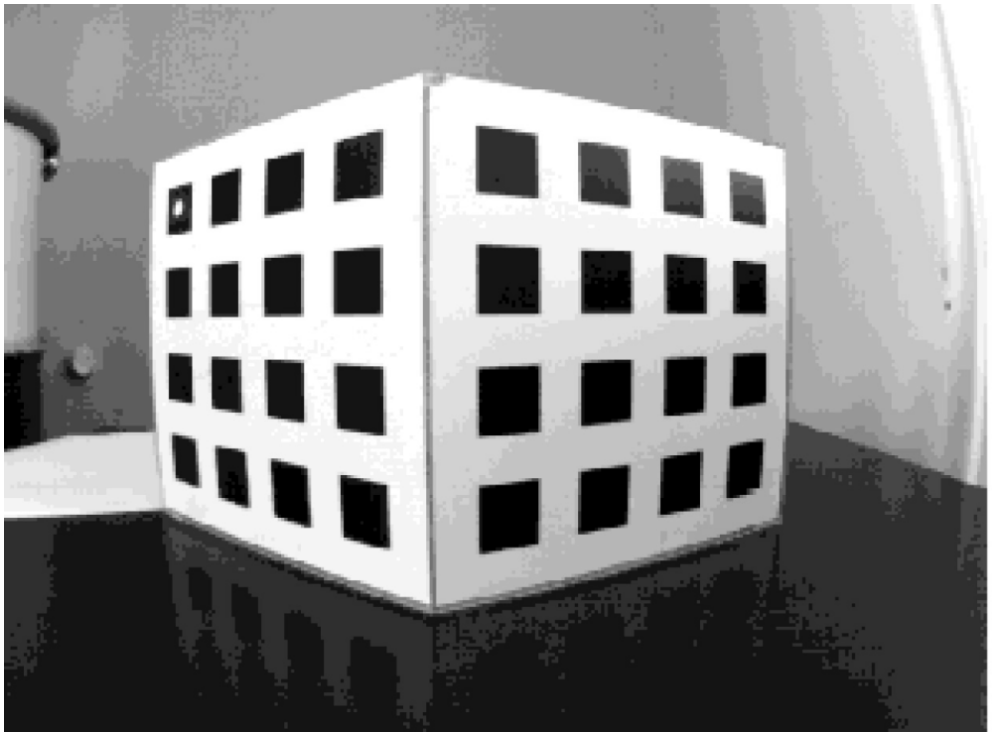
- The computation of these parameters for the distortion-free camera model yields the position of the camera in world coordinates

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

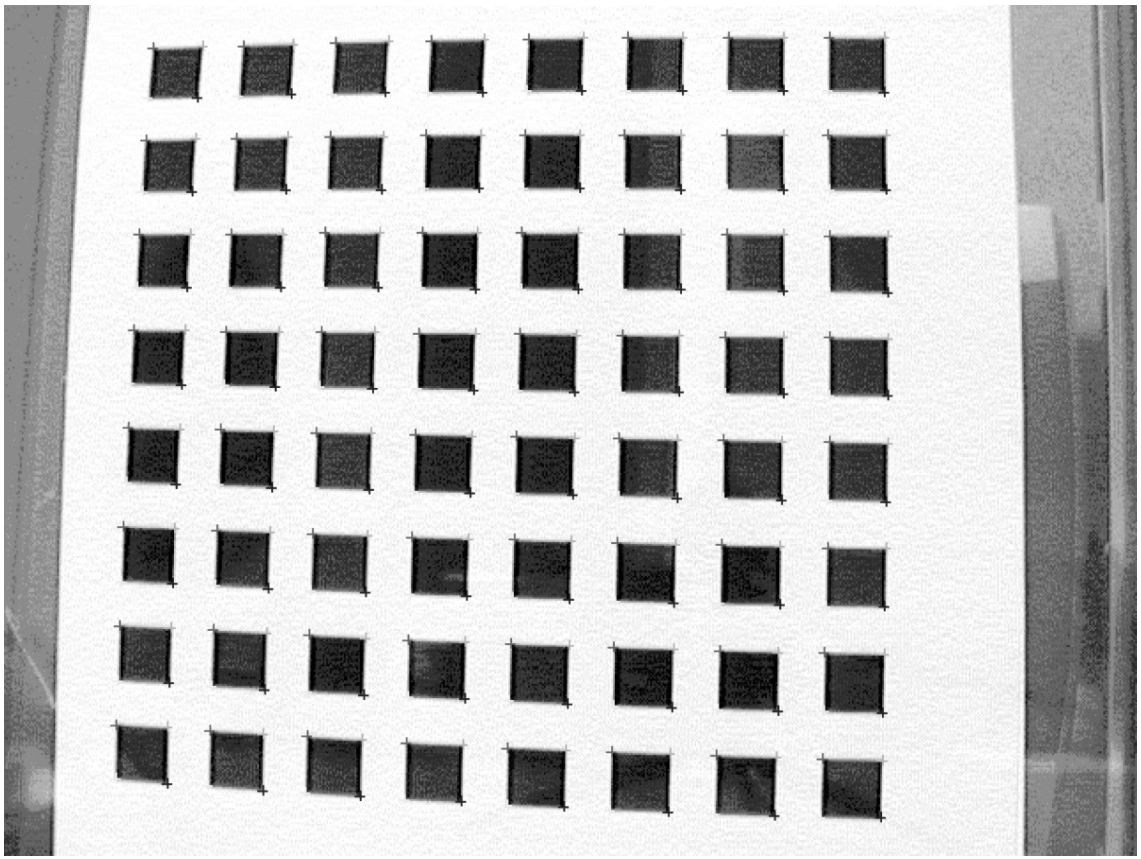
These *calibration points* are detected in the camera image with their respective image coordinates $\{X_i, Y_i\}$

Calibration objects



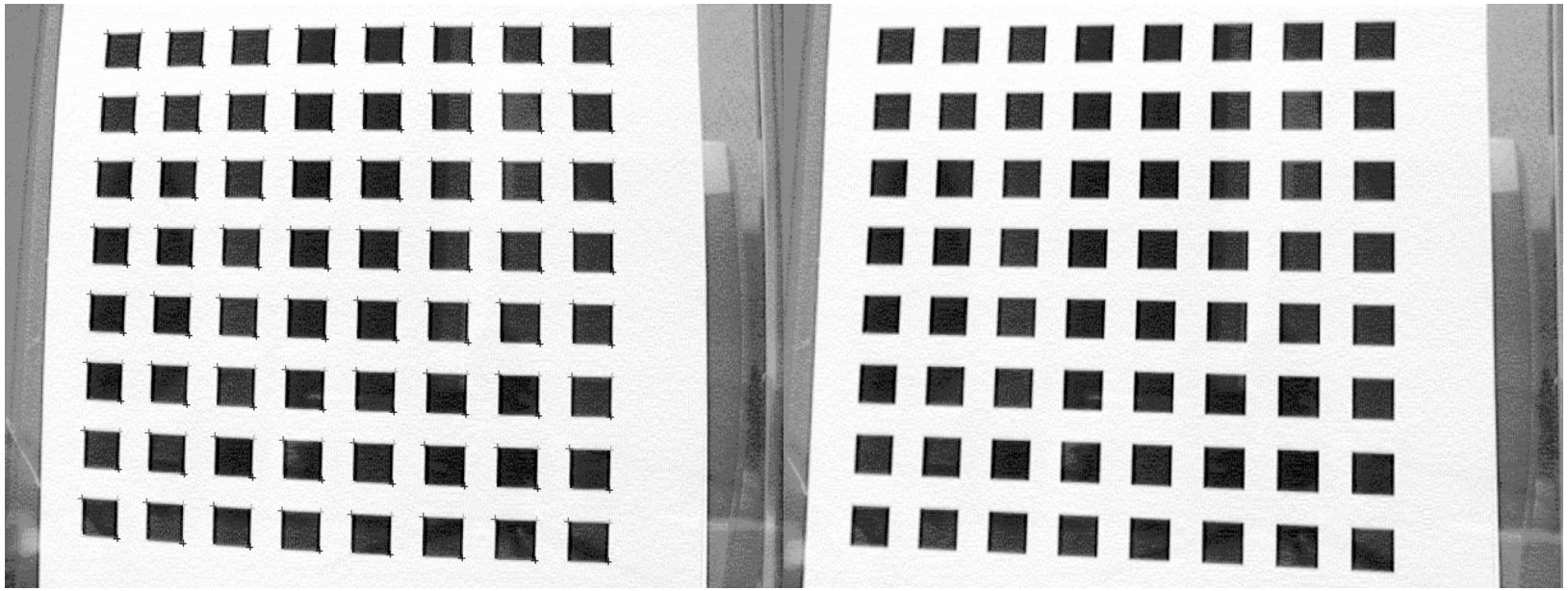
A typical 3D calibration object

Calibration objects (cont.)



Typical calibration pattern

Use of calibration parameters



Using the determined camera parameters, the image can be undistorted

Lens distortion

Real cameras and lenses 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 (e.g. center of the image sensor does not lie on the optical axis, sensor is not parallel to the lens, etc.)

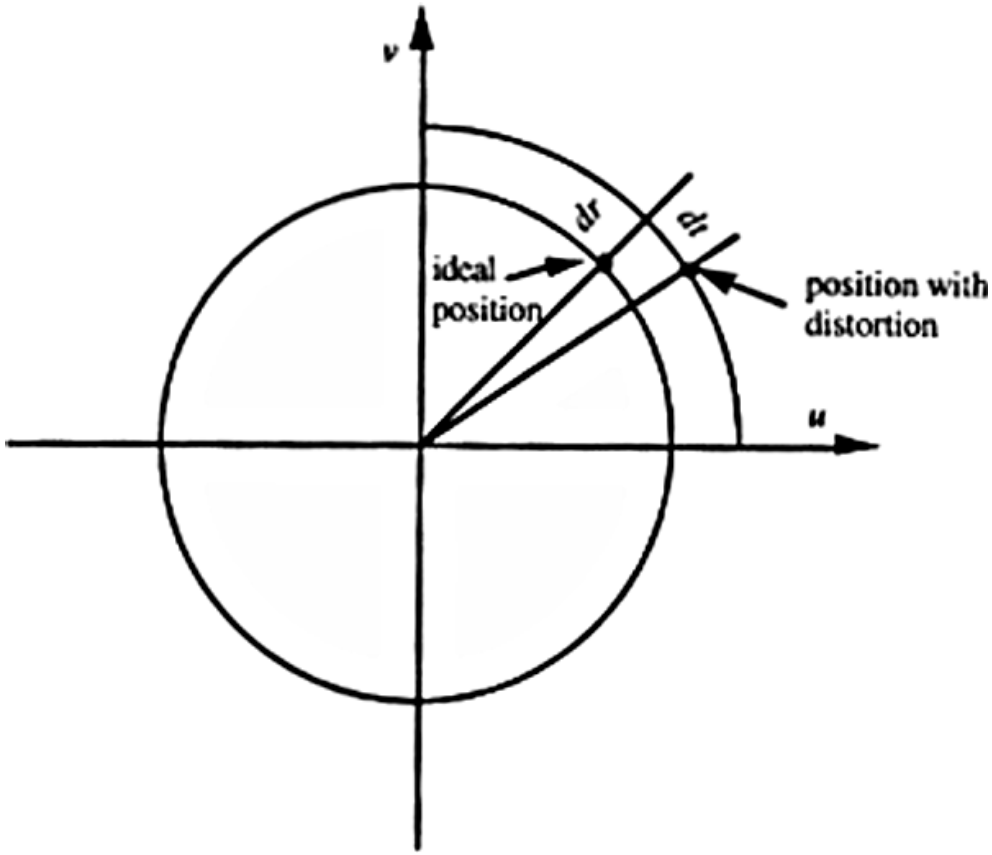
Lens distortion (cont.)

- 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.)



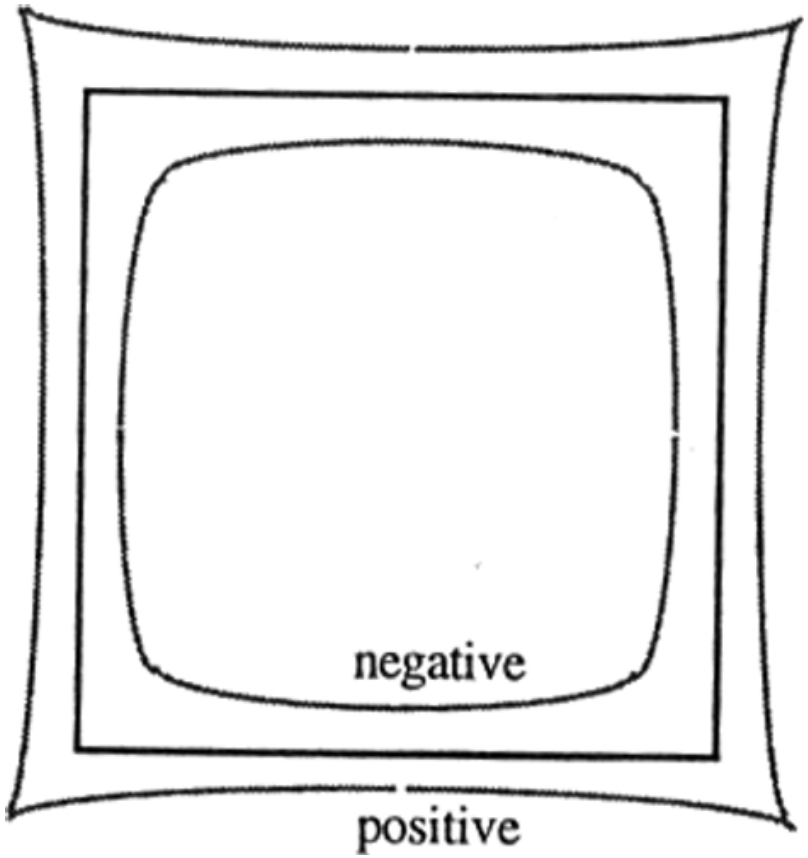
Lens distortion (cont.)

Types of distortion

- There are **two** primary types of distortions:
  - *Radial*
  - *Tangential*

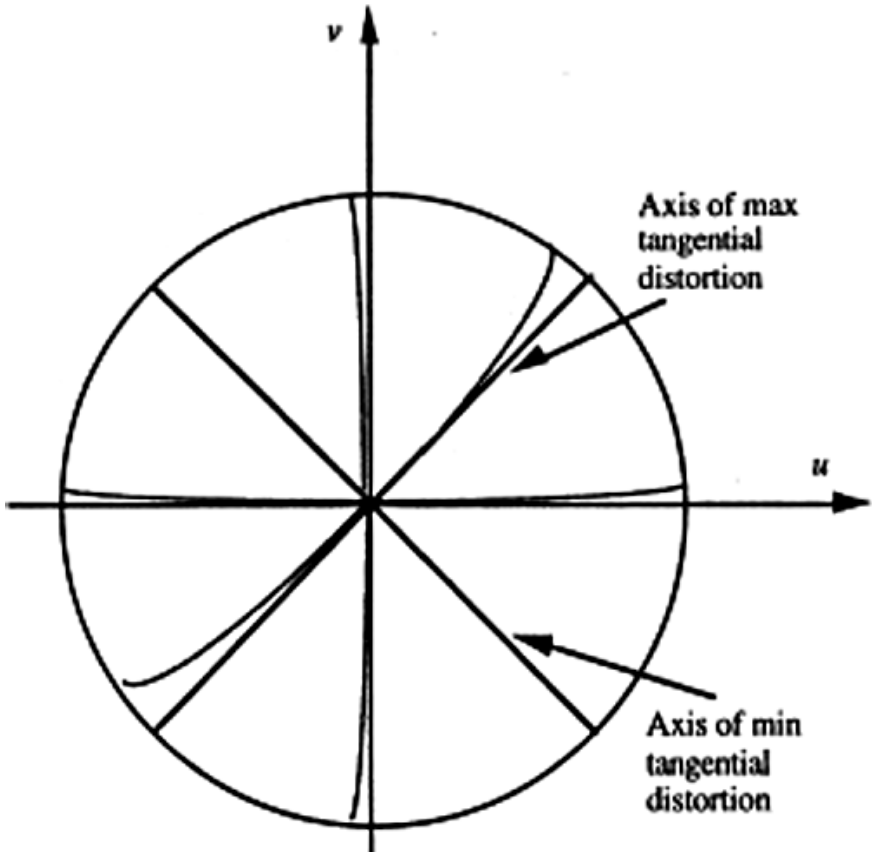
- Radial distortion causes an offset of the ideal position inwards (barrel distortion) or outwards (pincushion distortion)
  Possible cause: Flawed radial bend of the lens
- Tangential distortion shifts the ideal position along a tangential curve
  Possible cause: Non-parallel sensor/lens

Lens distortion



Radial distortion: Straight lines $\rightarrow$ no distortion

Lens distortion (cont.)



Tangential distortion: Straight lines $\rightarrow$ no distortion

Lens distortion (cont.)

Modeling of the lens distortion

- Various types of distortion are distinguished in literature:
  1. Radial distortion
  2. Decentering distortion
  3. Thin prism distortion
- Decentering distortion and thin prism distortion are both radial and tangential
- In the case of decentering distortion, optical centers of the lenses are not colinear

Lens distortion (cont.)

Radial lens distortion

$$D_{ur} = ku(u^2 + v^2)$$
$$D_{vr} = kv(u^2 + v^2)$$

with k being the first radial distortion coefficient and $(u^2 + v^2)$ being the squared radius

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:

Simplified camera model with distortion:

$$u' = u(1 + k' r'^2)$$
$$v' = v(1 + k' r'^2)$$

with $r'^2 = u^2 + v^2$

Camera model (cont.)

Radial distortion coefficient:

Since u and v are unknown, they are replaced by the measurable image coordinates X and Y and one has

$$r'^2 = (X/s_u)^2 + (Y/s_v)^2$$

and with

$$\mu \equiv \frac{f_y}{f_x} = \frac{s_v}{s_u}$$

one gets

$$r^2 \equiv \mu^2 X^2 + Y^2$$

Camera model (cont.)

Model for small radial distortions:

With the previously mentioned modifications, one gets the following camera model for small radial distortions

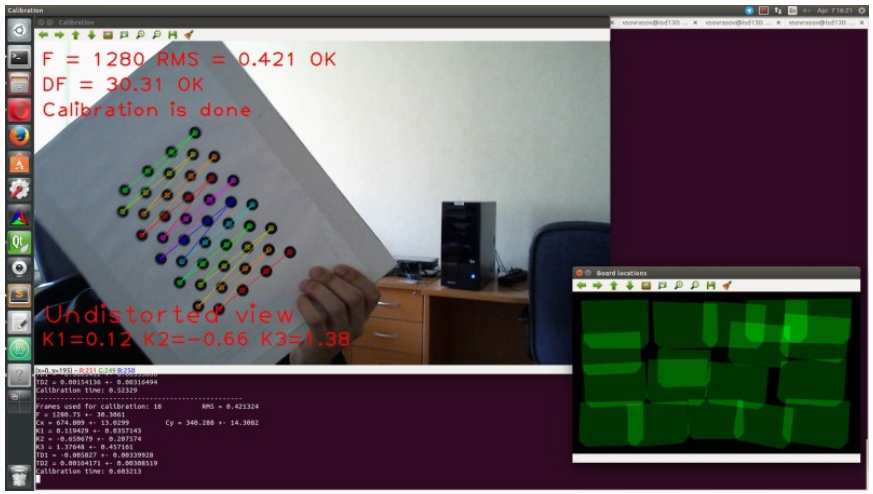
$$X(1 + kr^2) \cong f_x \frac{r_1 x_w + r_2 y_w + r_3 z_w + t_x}{r_7 x_w + r_8 y_w + r_9 z_w + t_z},$$

$$Y(1 + kr^2) \cong f_y \frac{r_4 x_w + r_5 y_w + r_6 z_w + t_y}{r_7 x_w + r_8 y_w + r_9 z_w + t_z}$$

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)



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

Outline

2. Vision systems
- Introduction
- Camera calibration
- Lens distortion
- Further Reading

Further Reading

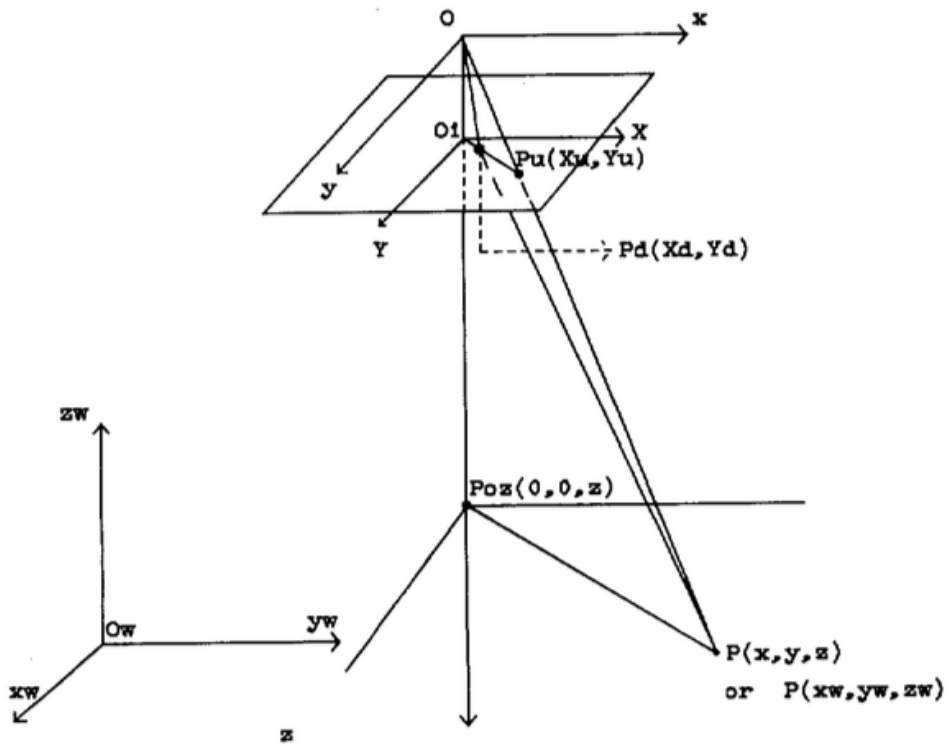
— ADDITIONAL SLIDES FOR FURTHER READING —

Calibration procedure (R. Y. Tsai)

Tsai's calibration procedure represents one of the most established techniques for determination of extrinsic and intrinsic camera parameters

- Determines the camera parameters in two stages
- Exploits a geometric constraint in order to simplify the parameter search problem
- Requires at least 5 *coplanar* calibration points
- In case of *noncoplanar* points at least 7 are required
- Both calibration stages can be executed in real time

Radial alignment constraint



Radial alignment constraint: Direction of vectors $O_i\vec{P}_d$ and $O_z\vec{P}_w$ is a function of only the relative rotation and translation

Radial alignment constraint (cont.)

The radial alignment constraint (RAC) is a function of *only* the relative rotation and translation (without z component) between camera and calibration points

- Radial distortion does not affect the direction of the vector from origin to (effective) image point
- The focal length scales X_d and Y_d with the same rate $\rightarrow$ Does not affect the direction either

With the given observations the *radial alignment constraint* is defined as:

$$\frac{X_d}{Y_d} = \frac{r_1 x_w + r_2 y_w + t_x}{r_4 x_w + r_5 y_w + t_y}$$

RAC-based calibration

The procedure assumes that C_x, C_y and $\mu = \frac{f_y}{f_x}$ are known parameters (obtained from manufacturer)

- Extrinsic parameters R and t and the intrinsic parameters f_x, f_y and k are to be determined
- For calibration, a reasonable number of **coplanar/noncoplanar** calibration points must be acquired
- Calibration is performed in two stages
  1. Determine the rotation matrix R and the components t_x and t_y of the translation vector
  2. Compute effective focal length, distortion coefficient, and the z component of the translation

RAC-based calibration (cont.)

Stage 1

1. Compute the distorted image coordinates $(X_{d,i}, Y_{d,i})$

With N as the number of calibration points, for $i = 1, 2, \ldots, N$ one has

$$X_{d,i} = X_{f,i} - C_x$$
$$Y_{d,i} = Y_{f,i} - C_y$$

where $X_{f,i}$ and $Y_{f,i}$ are the pixel values in the computer

RAC-based calibration (cont.)

Stage 1

2. Compute five unknowns for each calibration point

- With RAC independent from k and $f \rightarrow R$, t_x and t_y can be calculated
- Following from the RAC equation, five unknowns are established

$$\{v_1, v_2, v_3, v_4, v_5\} \equiv \{t_y^{-1} r_1, t_y^{-1} r_2, t_y^{-1} t_x, t_y^{-1} r_4, t_y^{-1} r_5\}$$

RAC-based calibration (cont.)

Stage 1

- Rearrangement of the RAC equation as a function of t_y leads to the following linear equation

$$\begin{bmatrix} Y_{d,i}x_{w,i} & Y_{d,i}y_{w,i} & Y_{d,i} & -X_{d,i}x_{w,i} & -X_{d,i}y_{w,i} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} = X_{d,i}$$

where $x_{w,i}$ and $y_{w,i}$ are the x- and y-coordinates of the i-th calibration point

RAC-based calibration (cont.)

Stage 1

- The minimum number of necessary calibration points is $N = 5$
- With the minimum number of points provided a system of linear equations can be established and solved for the unknowns
- In practice, an appropriate number of calibration points would be $N > 5$

- **Note:** If $t_y = 0$, the above equation can also be formulated as a function of t_x
- If one determines $t_x = t_y = 0$, the chosen calibration setup needs to be adjusted

RAC-based calibration (cont.)

Stage 1

3. Compute R, t_x and t_y

- Define $C \equiv \begin{bmatrix} v_1 & v_2 \\ v_4 & v_5 \end{bmatrix}$ a submatrix of R
- If no line or column equals zero, t_y^2 is determined as follows:

$$t_y^2 = \frac{S_r - \sqrt{S_r^2 - 4(v_1 v_5 - v_4 v_2)^2}}{2(v_1 v_5 - v_4 v_2)^2} \quad \text{with} \quad S_r \equiv v_1^2 + v_2^2 + v_4^2 + v_5^2$$

- Otherwise one has:

$$t_y^2 = (v_i^2 + v_j^2)^{-1}$$

where v_i and v_j are the elements from C, which are non-zero

RAC-based calibration (cont.)

Stage 1

- Physically, the algebraic signs of $x_{w,i}$ and $X_{d,i}$ as well as $y_{w,i}$ and $Y_{d,i}$ should be equal
- Assuming $t_y > 0$ following components can be calculated

$$r_1 = v_1 t_y$$
$$r_2 = v_2 t_y$$
$$r_4 = v_4 t_y$$
$$r_5 = v_5 t_y$$
$$t_x = v_3 t_y$$

RAC-based calibration (cont.)

Stage 1

- Using an arbitrary calibration point, the following coordinates can be determined:

$$X_d = r_1 x_w + r_2 y_w + t_x$$
$$Y_d = r_4 x_w + r_5 y_w + t_y$$

- If the signs of $x_{w,i}$ and $X_{d,i}$ as well as the signs of $y_{w,i}$ and $Y_{d,i}$ re equal, then the assumption $t_y > 0$ is true and we keep r_1, r_2, r_4, r_5 and t_x
- Otherwise, $t_y < 0$ and we change the signs of r_1, r_2, r_4, r_5 and t_y accordingly

RAC-based calibration (cont.)

Stage 1

- There are two possible solutions for the rotation matrix R, if a 2×2 submatrix C is known
- R can be calculated based on:

$$r_3 = \pm(1 - r_1^2 - r_2^2)^{1/2}$$

$$r_6 = \pm sign(r_1 r_4 + r_2 r_5)(1 - r_4^2 - r_5^2)^{1/2}$$

$$[r_7\ r_8\ r_9]^T = [r_1\ r_2\ r_3]^T \times [r_4\ r_5\ r_6]^T$$

- One of the two solutions leads to a positive sign of the focal length in *Stage 2* of the calibration procedure

RAC-based calibration (cont.)

Stage 2

Determination of the parameters t_z, k, f_x and f_y

- If R, t_x and t_y are known, the remaining parameters for the i-th calibration point can be determined using the following linear equation:

$$\begin{bmatrix} -Y_i & y_i & -y_i r_i^2 \end{bmatrix} \begin{bmatrix} t_z \\ f_x \\ k f_x \end{bmatrix} = Y_i w_i$$

where

$$y_i \equiv r_4 x_{w,i} + r_5 y_{w,i} + t_y$$
$$w_i \equiv r_7 x_{w,i} + r_8 y_{w,i}$$

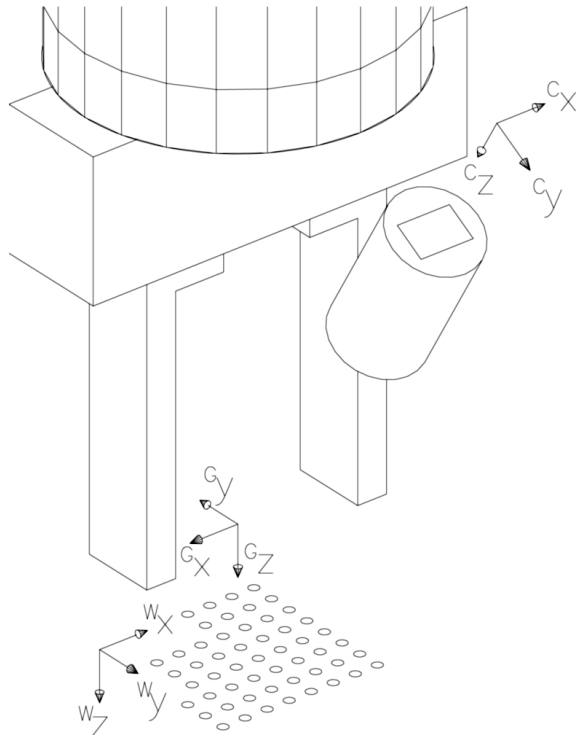
RAC-based calibration (cont.)

Stage 2

- With the minimum amount of 5 calibration points, we already have an over-determined system of linear equations
- The solution of this system of linear equations yields the parameters k, t_z and f_x
- Using f_x the other parameters can be computed:

$$f_y = f_x \mu$$
$$k = (k f_x) f_x^{-1}$$

Hand camera calibration



Camera, gripper and world coordinate frames

Hand camera calibration (cont.)

Task:

Determination of the fixed spatial relation between camera- (C) and gripper-coordinate system (G) represented by the homogeneous transformation ${}^{C}H_{G}$

Idea:

Direct computation of ${}^{C}H_{G}$ through model based localization of visible gripper features

Hand camera calibration (cont.)

Solution:

- Positioning of the gripper on a planar calibration object with several calibration points
- Result: Gripper and calibration coordinate frames coincide
- Plane coincidence allows composition of the problem

$$^{C}H_{G} = {}^{C}H_{W}\,{}^{W}H_{G}$$

Hand camera calibration (cont.)

Approach:

1. Determination of intrinsic and extrinsic camera parameters using the calibration object $\Rightarrow \quad {}^{C}H_{W}$
2. Determination of the parameters of a 2D-transformation ${}^{W}H_{G}$ using visible gripper features

Hand camera calibration (cont.)

Advantages of "self-visibility":

- "Self-visibility" allows calibration of the configuration without test movements of the manipulator as opposed to classical procedures
- Two dot-shaped gripper features are sufficient for the determination of ${}^{C}H_{G}$ through solution in closed form
- Online-surveillance of the relative position between gripper and object is possible
- Higher accuracy of offline calibration due to exclusion of kinematic errors
- Higher level of robustness due to the possibility of online re-calibration

Visually controlled grasping

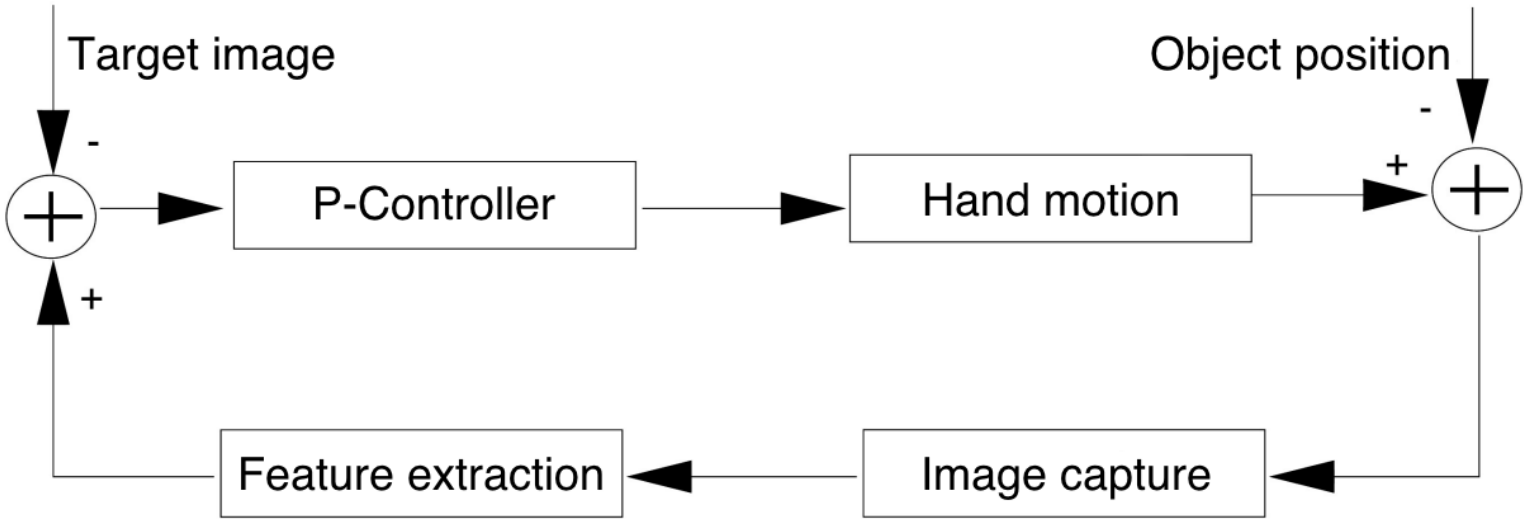
Task:

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

Procedure:

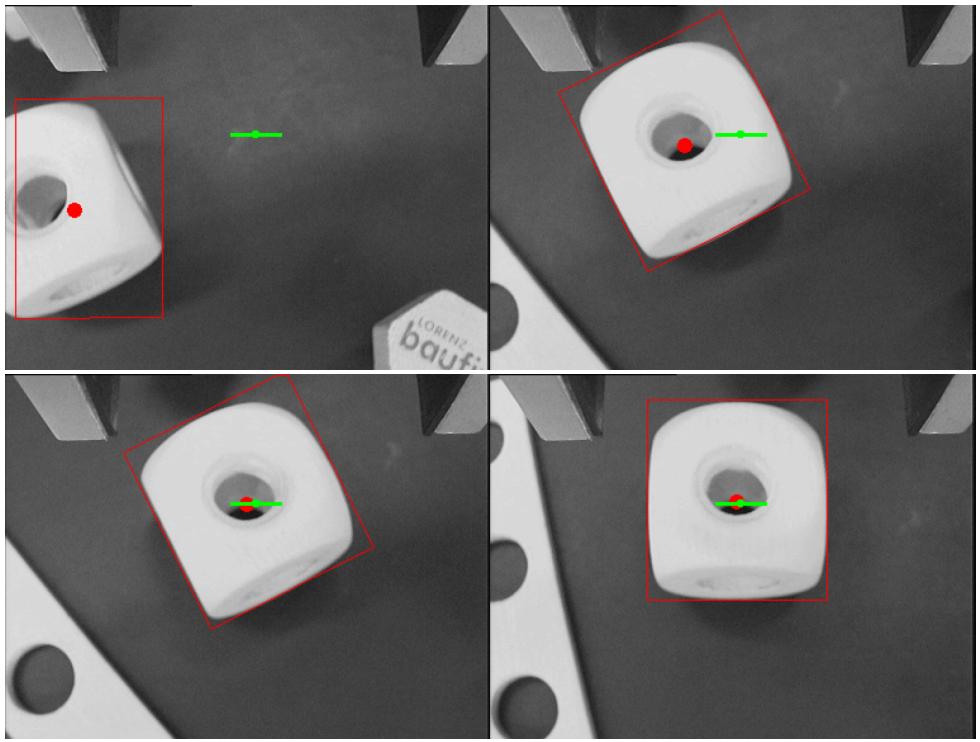
1. Offline-specification of the target position (e.g. object features from stereo image processing)
2. Online transformation of the current difference in relation to the target position (e.g. with a hand camera)

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

Literature list

[1] Richard Hartley and Andrew Zisserman.
Multiple view geometry in computer vision.
Cambridge Univ. Press, 2. edition, 2003.

[2] Roger Y. Tsai.
A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses.
IEEE Journal of Robotics and Automation, 3(4):323–344, August 1987.