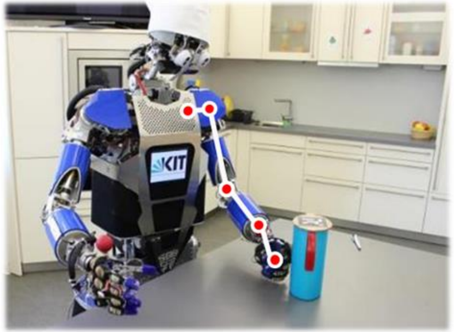
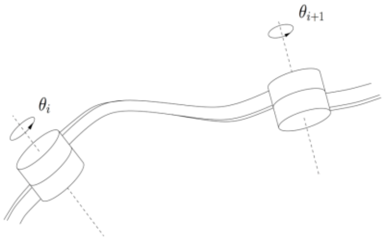
Robotics I: Introduction to Robotics

Chapter 1 – Mathematical Foundations and Concepts of Robotics

Tamim Asfour

https://www.humanoids.kit.edu





Mathematical Foundations of Robotics

Motivation

ARMAR!
Bring me the apple juice from the fridge

What basic mathematical means are needed?

We need to describe **positions of objects** in space:

- Where is the apple juice box? (at which coordinates?)
- Relative to which coordinate system?
  - Relative to camera coordinate system?
  - Relative to arm base (shoulder) coordinate system?
  - Relative to robot mobile base coordinate system?
  - Relative to world coordinate system?
    (in the left corner of the kitchen)

What basic mathematical means are needed?

We need to describe **positions of objects** in space:

We need to describe **orientations of objects** in space:

- Is the bottle located directly in front of the robot?
- Or to the left or to the right of the robot?

A framework to describe positions (translations) and orientations (rotations) is needed!

Kinematic Basis

- This chapter is an introduction to the mathematical foundations of robotics
- Mathematical methods for the description of rigid body transformations (based on linear algebra)
- Application of these methods to model robots

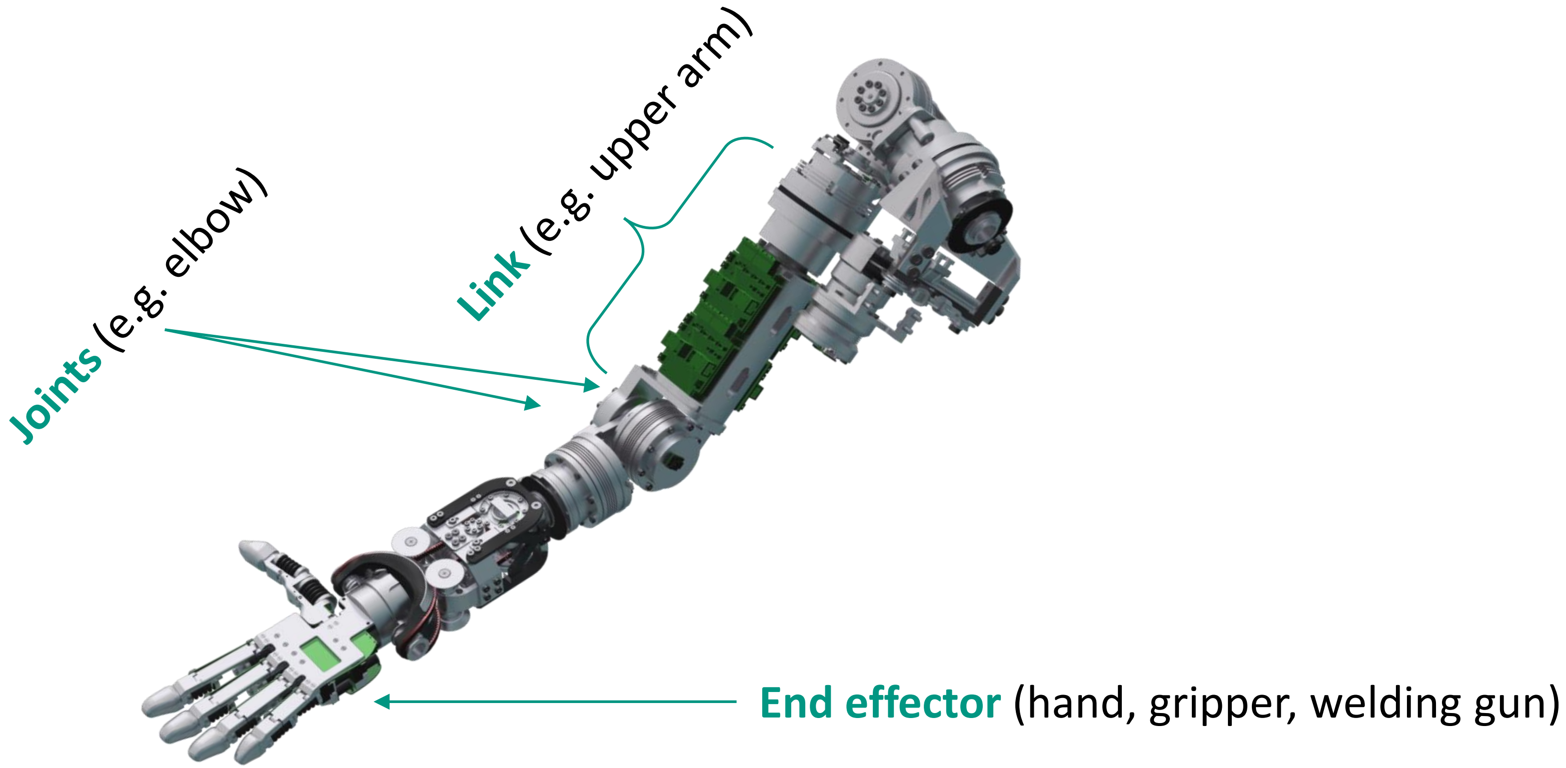
Definitions

- **Kinematics** is the study of motion of bodies and systems based **only on geometry**, i.e. without considering the physical properties and the forces acting on them. The essential concept is a **pose** (position and orientation).

- **Statics** studies forces and moments acting on an object **at rest**. The essential concept is a **stiffness**.

- **Dynamics** studies the relationship between the **forces and moments** acting on a robot and accelerations they produce,

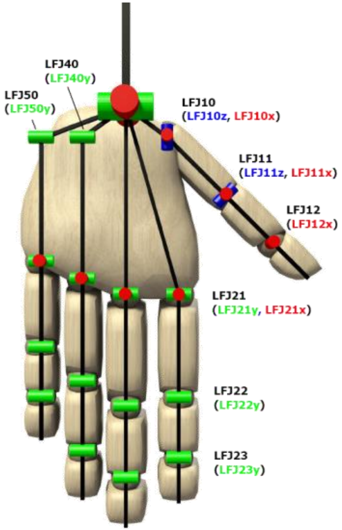
Kinematics – Terminology (I)

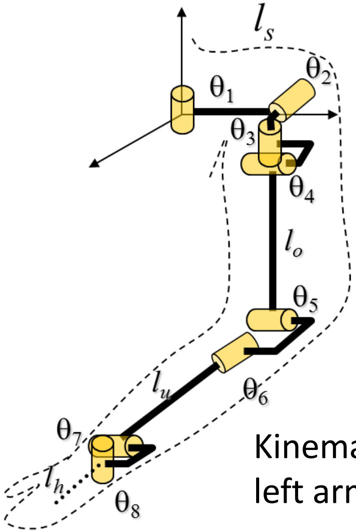


Kinematics – Terminology (II)

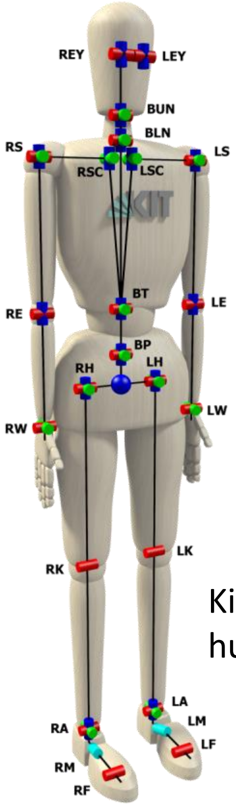
- **Kinematic chain** is a set of links connected by joints.
- Kinematic chain can be represented by a graph.
  The vertices represent joints and edges represent links.

Kinematics chain human hand





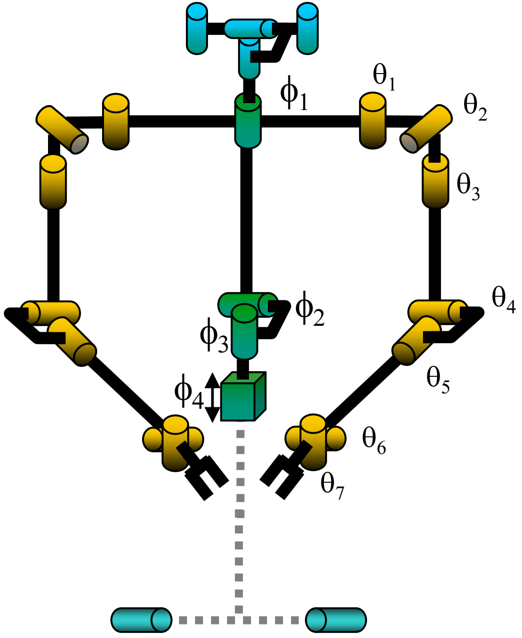
Kinematics chain left arm



Kinematics chain human body

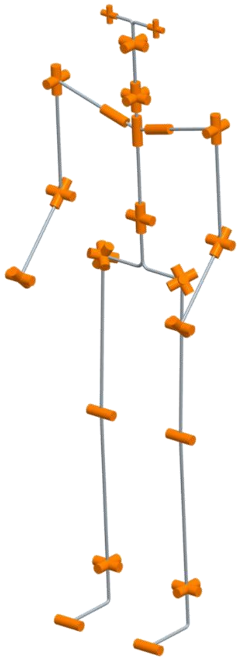
Kinematics – Terminology (II)

- Kinematic chains: examples



Kinematic chain ARMAR-I

ARMAR-IV



Kinematics chain ARMAR-IV

Kinematics – Degrees of Freedom (DoF)

Degrees of freedom (less formal definition) is the **number of independent parameters** needed to specify the position of an object completely.

Examples:

- A point on a plane has 2 DoF
- A point in 3D space has 3 DoF
- Rigid body in a 2D space (i.e. on a plane) has 3 DoF
- Rigid body in 3D space has 6 DoF

Conventions

In this lecture, we will use the following conventions for equation symbols:

- Scalars: lower-case Latin letters
  - Example: $s, t \in \mathbb{R}$
- Vectors: bold lower-case Latin letters
  - Example: $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3$
- Matrices: upper-case Latin letters
  - Example: $\mathbf{A} \in \mathbb{R}^{3\times 3}$
- Linear maps (linear transformations): upper-case Greek letters
  - Example: $\phi(\cdot): \mathbb{R}^3 \rightarrow \mathbb{R}^3$

Rigid Body Motion

A rigid body is a body that does not deform or change shape

Rigid body motion is characterized by **two properties:**

1. The distance between any two points remains invariant
   → The motion of the body is completely specified by the motion of any point in the body.
   → All points of the body have the same velocity and same acceleration.
2. The orientations are preserved.
   → A right-handed coordinate system remains right-handed

$\mathrm{SO}(3)$ and $\mathrm{SE}(3)$

Two groups which are of particular interest to us in robotics are

- **$\mathrm{SO}(3)$ – the special orthogonal group** that represents **rotations** and
- **$\mathrm{SE}(3)$ – the special Euclidean group** that represents rigid body **motions**

- Elements of $\mathrm{SO}(3)$ are represented as 3×3 real matrices and satisfy

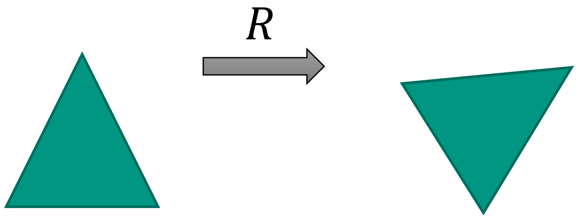
$$\mathbf{R}^T\mathbf{R} = \mathbf{I} \qquad \text{with } \det(\mathbf{R}) = 1$$

i.e., R is a special orthogonal matrix

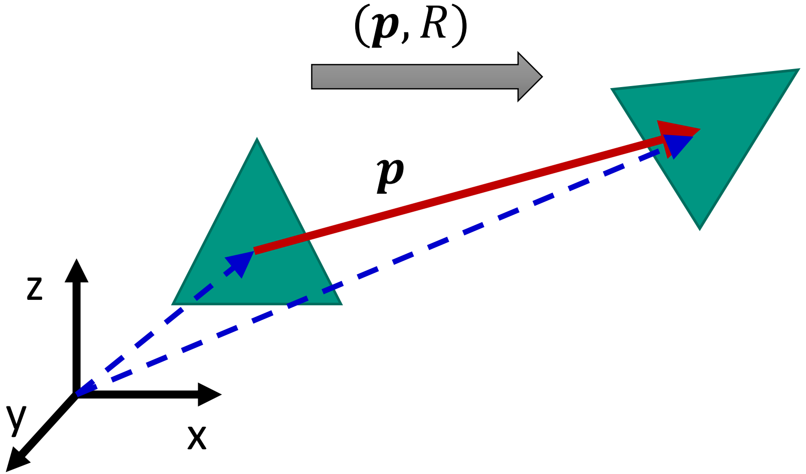
- Element $\mathrm{SE}(3)$ are of the form $(\mathbf{p}, \mathbf{R})$, where $\mathbf{p} \in \mathbb{R}^3$ and $\mathbf{R} \in \mathrm{SO}(3)$

$\mathbf{SO(3)}$ und $\mathbf{SE(3)}$

- SO(3)
  - Orientation
  - $R \in SO(3) \subset \mathbb{R}^{3\times3}$



- SE(3)
  - Position and orientation
  - $(\boldsymbol{p}, R) \in SE(3)$
    with $\boldsymbol{p} \in \mathbb{R}^3, R \in SO(3)$



Affine Geometry

- We use affine geometry to describe spatial transformations.
- These transformations are **concatenations of rotations and translations**

- Spatial transformations can be represented mathematically in several ways:
  - rotation matrices and translation vectors
  - homogeneous matrices
  - quaternions
  - dual quaternions

- This lecture will introduce the above representations.

Euclidean Space (I)

- Euclidean space is the **vector space $\mathbb{R}^3$** with the **standard scalar product** (also know as dot product or inner product).

- Example:

A point $\mathbf{c}$ located on a line between two points $\mathbf{a}$ and $\mathbf{b}$ can be represented as

$$\mathbf{c} = t \cdot \mathbf{a} + (1 - t) \cdot \mathbf{b}, \quad t \in (0,1) \subset \mathbb{R}, \quad \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3.$$

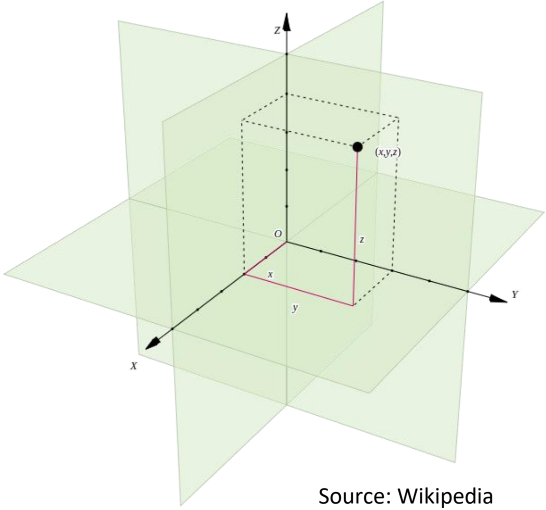
Euclidean Space (II)

- A **point** $\mathbf{a}$ in Euclidean space is represented by coordinates referring to a **coordinate system** $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$.

$$\mathbf{a} = a_x \cdot \mathbf{e}_x + a_y \cdot \mathbf{e}_y + a_z \cdot \mathbf{e}_z = (a_x, a_y, a_z)^T. \quad \mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z \in \mathbb{R}^3$$

- Conventions:
  - We use **orthonormal coordinate systems**, i.e. the base vectors $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ are unit vectors and perpendicular (orthogonal) to one another.
  - We use **right-hand coordinate systems**.

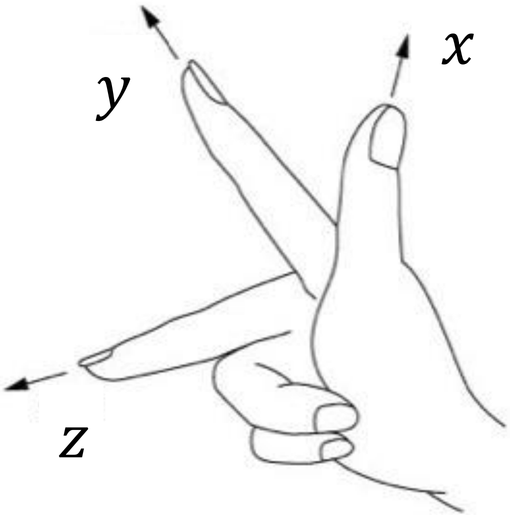
    Right hand rule: If the thumb points in the direction of the x-axis and the index finger points in the direction of the y-axis then the middle finger indicates the direction of the z-axis.

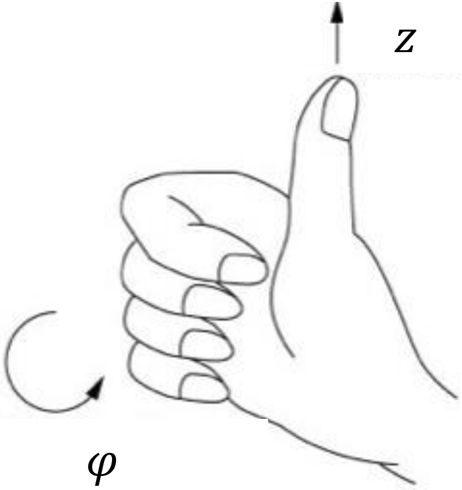


Source: Wikipedia

Coordinate Systems (I)

Right-hand rule for right-handed coordinate systems



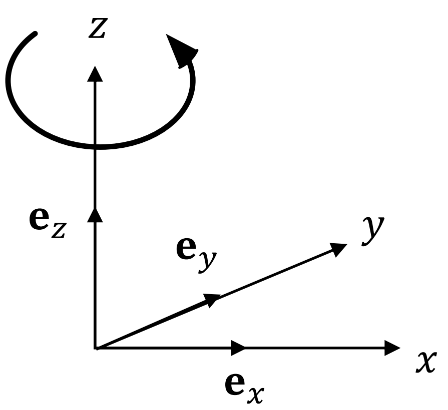


Coordinate Systems (II)

Right-handed coordinate system

$$\mathbf{e}_x \times \mathbf{e}_y = \mathbf{e}_z$$

$$\mathbf{x} \times \mathbf{y} = \mathbf{z}$$

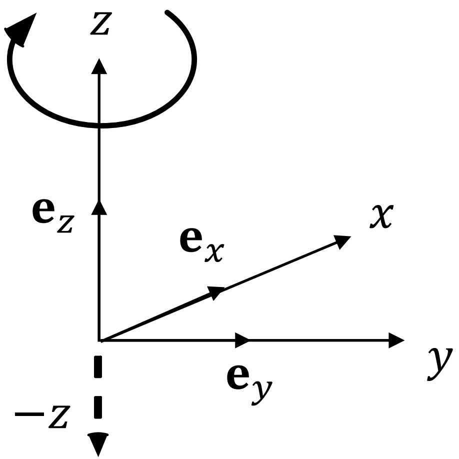


Left-handed coordinate system

$$\mathbf{e}_x \times \mathbf{e}_y = -\mathbf{e}_z$$

$$\mathbf{x} \times \mathbf{y} = -\mathbf{z}$$

× : cross product



Linear Maps, Endomorphism

- **Linear maps (transformations)** which map Euclidean space onto itself are called **endomorphisms**:

$$\phi(\cdot): \mathbb{R}^3 \rightarrow \mathbb{R}^3$$

- Endomorphisms can be represented by **square matrices**:

$$\phi(\mathbf{a}) = \mathbf{A} \cdot \mathbf{a}, \quad A \in \mathbb{R}^{3\times 3}$$

- A describes a **change of basis** resulting from the original basis vectors $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ and the new basis vectors $\mathbf{e}'_x, \mathbf{e}'_y, \mathbf{e}'_z$

$$A = \begin{pmatrix} \mathbf{e}'_x & \mathbf{e}'_y & \mathbf{e}'_z \end{pmatrix} \cdot \begin{pmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \end{pmatrix}^{-1}$$

Isomorphismus

- **Bijective** (reversible) endomorphisms are called **isomorphisms**.

- Isomorphisms may have special, interesting properties:
  - They may preserve angles. (Examples: scaling and rotation)
  - They may preserve lengths. (Example: rotation)
  - They may preserve handedness.
    (Example: rotation. Right-hand coordinate frame is preserved, etc.)

- A special set of isomorphisms which fulfills all of the above criteria is the **rotation group** (or special orthogonal group) $\mathrm{SO}(3)$.

The Rotation Group $\mathrm{SO}(3)$

- $\mathrm{SO}(3)$ contains **all possible rotations** around arbitrary axes through the origin
- $\mathrm{SO}(3)$ is non-abelian **(not commutative)**, i.e.

$$\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{x} \neq \mathbf{B} \cdot \mathbf{A} \cdot \mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^3, \quad \mathbf{A}, \mathbf{B} \in SO_3.$$

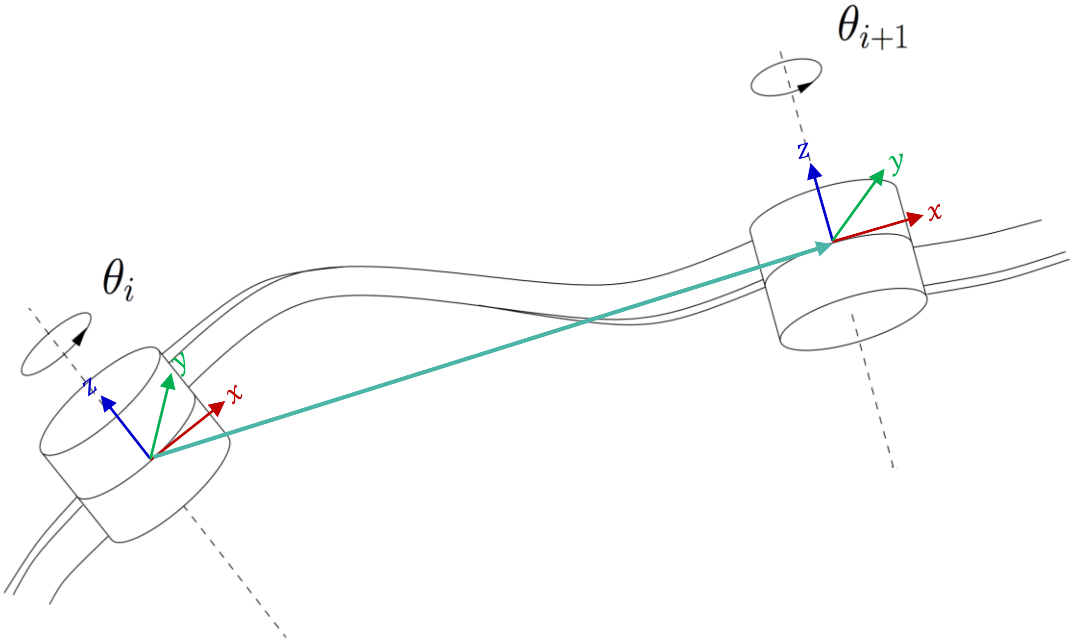
Why are $\mathrm{SO}(3)$ and $SE(3)$ interesting for robotics?

- Using $\mathrm{SO}(3)$ and $SE(3)$, an **object‘s pose** (i.e. position and orientation) in space as well as transformations between two robot joint axes can be represented as a **combination** of a **translation** and a **rotation**:

$$\phi(\cdot): \mathbb{R}^3 \rightarrow \mathbb{R}^3, \quad \phi(\mathbf{x}) = \mathbf{t} + \mathbf{R} \cdot \mathbf{x}, \quad \mathbf{x}, \mathbf{t} \in \mathbb{R}^3, \quad \boldsymbol{R} \in SO_3.$$

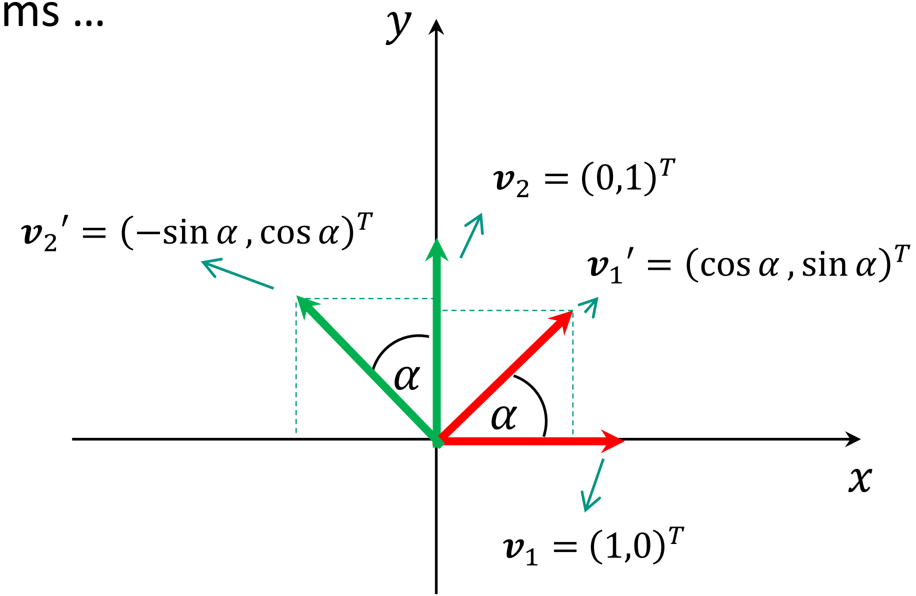
- The map $\phi(\cdot)$ is not linear! It is called **affine**.

Transformation between two Robot Joints



Rotations in 2D (1)

- Rotation in the xy-plane around $(0,0)$ is a linear transformation.
- Rotation of angle θ around $(0,0)$ transforms …
  - Vector $(1,0)^T$ to $(\cos\alpha, \sin\alpha)^T$
  - Vector $(0,1)^T$ to $(-\sin\alpha, \cos\alpha)^T$



- Rotation matrix

$$\mathbf{R}_\theta(\mathbf{x}) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \mathbf{x}$$

with $\mathbf{R}\mathbf{R}^\mathrm{T} = \mathbf{R}^\mathrm{T}\mathbf{R} = \mathbf{I}, \;\; \det(\mathbf{R}) = 1$

Rotations in 2D (2)

- Rotation around a point $\mathbf{c} \neq (0, 0)$ is not a linear transformation. It transforms $(0, 0)$ to a point other than $(0,0)$.

- Rotation around an arbitrary rotation center c:
  - We shift the plane by $-\mathbf{c}$ such that the rotation center will be $(0, 0)$.
  - Then we perform a **rotation** around $(0, 0)$.
  - Then we shift back the plane by $+\mathbf{c}$.

$$\mathbf{R}_{c,\theta}(\mathbf{x}) = \mathbf{R}_\theta(\mathbf{x} - c) + \mathbf{c} = \mathbf{R}_\theta(\mathbf{x}) + (-\mathbf{R}_\theta(\mathbf{c}) + \mathbf{c})$$

Affine Transformation

$$\mathbf{R}_{c,\theta}(\mathbf{x}) = \mathbf{R}_{\theta}(\mathbf{x} - c) + \mathbf{c} = \mathbf{R}_{\theta}(\mathbf{x}) + (-\mathbf{R}_{\theta}(\mathbf{c}) + \mathbf{c})$$

- $\mathbf{R}_{c,\theta}$ is a non-linear transformation. It differs from R_{θ} only in the addition of a constant.
- Transformations (like $\mathbf{R}_{c,\theta}$) of the form

$$\mathbf{T}(\boldsymbol{x}) = \mathbf{A}(\boldsymbol{x}) + \mathbf{b}$$

are called **affine transformations**.

Rotations in 3D

- 2D rotation in xy-plane is a rotation in 3D around the z-axis.
- Rotation of points around z does not depend on their z values and points on the z-axis are not affected by this rotation.
- The rotation matrix around the z-axis takes a simple form:
  - The submatrix corresponding to xy is identical to the 2D case,
  - the value multiplying the z-value is 1,
  - The entries corresponding to the influence of z (of the rotated vector) on its x and y and vice versa are zero

$$\mathbf{R}_{z,\theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotations in 3D

$$\mathbf{R}_{z,\theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_{x,\theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

$$\mathbf{R}_{y,\theta} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

Inverse of a Rotation Matrix

The **inverse** of a rotation matrix is **its transpose**:

$$\mathbf{R}_{x,\theta}^{-1} = \mathbf{R}_{x,-\theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(-\theta) & -\sin(-\theta) \\ 0 & \sin(-\theta) & \cos(-\theta) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} = \mathbf{R}_{x,\theta}^{T}$$

$$\mathbf{R}_{x,\theta}^{-1} = \mathbf{R}_{x,\theta}^{T}$$

Note:

This is the defining property for **all orthogonal** matrices.

(Rotation matrices $\mathbf{R}$ additionally have $\det(\mathbf{R}) = 1$.)

Concatenation of Rotations

Karlsruhe Institute of Technology

- The concatenation of rotations

$$\phi_{z,\theta_3}(\phi_{y,\theta_2}(\phi_{x,\theta_1}(\mathbf{a}))), \quad \mathbf{a} \in \mathbb{R}^3$$

- Important: there are two ways to interpret the above concatenation
  - **Left to right:** With each rotation, the unit vectors change; rotations are performed around **local axes**.

$$\left(\left(R_{z,\theta_3} \cdot R_{y',\theta_2}\right) \cdot R_{x'',\theta_1}\right) \cdot \mathbf{a} = R_{z,\theta_3} \cdot R_{y',\theta_2} \cdot R_{x'',\theta_1} \cdot \mathbf{a}$$

  - **Right to left:** Rotations are performed around **global axes** (which do not change).

$$R_{z,\theta_3} \cdot \left(R_{y,\theta_2} \cdot \left(R_{x,\theta_1} \cdot \mathbf{a}\right)\right) = R_{z,\theta_3} \cdot R_{y,\theta_2} \cdot R_{x,\theta_1} \cdot \mathbf{a}$$

High Performance Humanoid Technologies

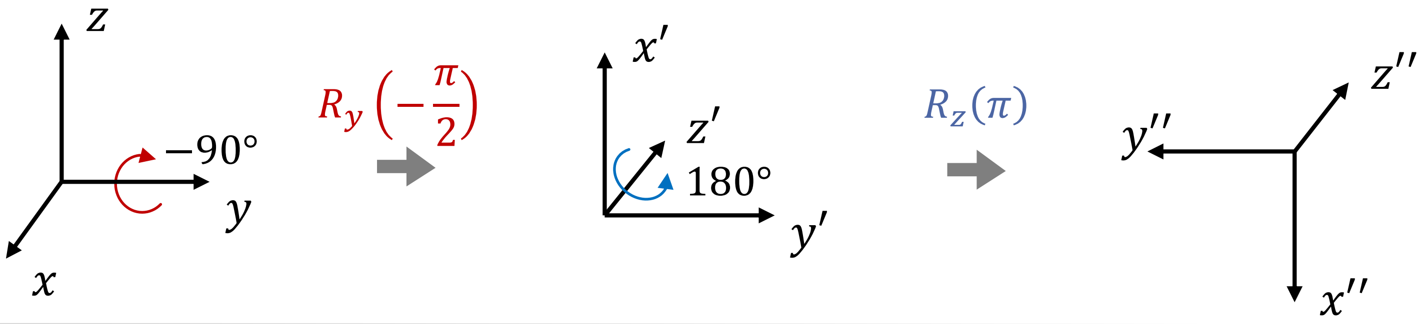
Example: Concatenation of Rotations (1)

- Concatenation of the following rotations:
  - Rotation around y-axis: $-90°\ \left(-\frac{\pi}{2}\right)$

$$R_y\left(-\frac{\pi}{2}\right) = \begin{pmatrix} \cos\left(-\frac{\pi}{2}\right) & 0 & \sin\left(-\frac{\pi}{2}\right) \\ 0 & 1 & 0 \\ -\sin\left(-\frac{\pi}{2}\right) & 0 & \cos\left(-\frac{\pi}{2}\right) \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

  - Rotation around z-axis: $180°\ (\pi)$

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



Example: Concatenation of Rotations (2)

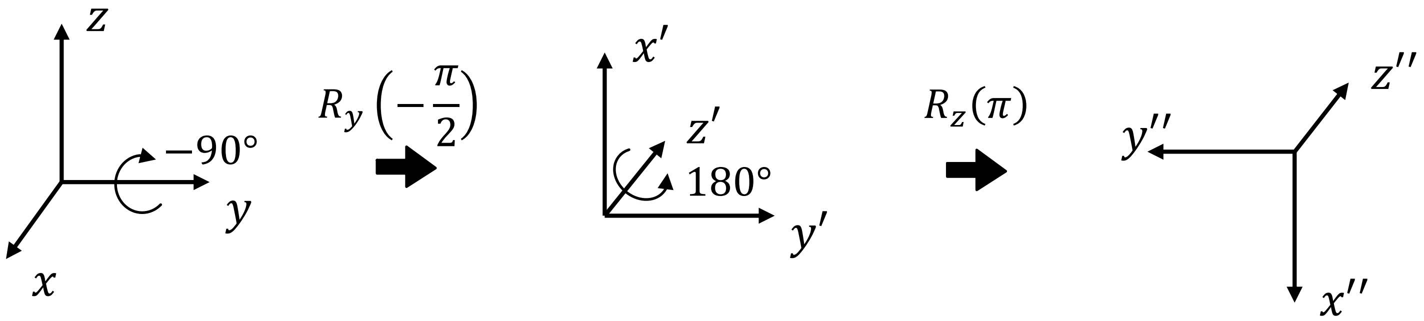
- Calculation of the rotation matrix

$$R = R_y\left(-\frac{\pi}{2}\right) \cdot R_z(\pi) = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

- Transformation of a vector

$$\mathbf{p}'' = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \cdot \mathbf{p} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} -p_3 \\ -p_2 \\ -p_1 \end{pmatrix}$$

From left to right:
The unit vectors change with each rotation. Rotations around local axes.



Problems with Rotation Matrices

- Rotation matrices have a number of **drawbacks**:
  - **Redundancy:** nine values for one rotation matrix
  - **In machine learning**: If the entries of a rotation matrix are predicted independently, it is likely that the resulting matrix is not a valid rotation matrix! (more on that later…)

- How to deal with these problems?
  - Use other representation for rotations, e.g. Euler angles.
  - Orthonormalize the matrix.

Euler Angles

- It is possible to represent every thinkable rotation by **three rotations around three coordinate axes**.
- The axes can be chosen arbitrarily, but due to historic reasons, a very common convention is the so-called **Euler $z\,x'z''$ convention**.
- The angles α, β and γ are the Euler angles. They describe the rotation matrix

$$R_{z,\alpha}\, R_{x',\beta}\, R_{z'',\gamma} =$$

$$\begin{pmatrix} \cos\gamma \cdot \cos\alpha - \sin\gamma \cdot \cos\beta \cdot \sin\alpha & -\sin\gamma \cdot \cos\alpha - \cos\gamma \cdot \cos\beta \cdot \sin\alpha & \sin\beta \cdot \sin\alpha \\ \cos\gamma \cdot \sin\alpha + \sin\gamma \cdot \cos\beta \cdot \cos\alpha & -\sin\gamma \cdot \sin\alpha + \cos\gamma \cdot \cos\beta \cdot \cos\alpha & -\sin\beta \cdot \cos\alpha \\ \sin\gamma \cdot \sin\beta & \cos\gamma \cdot \sin\beta & \cos\beta \end{pmatrix}$$

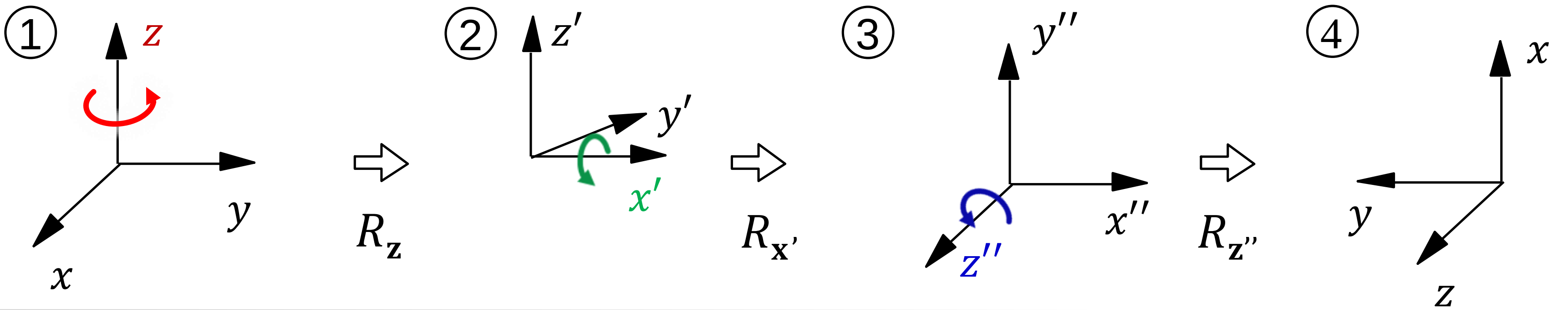
Euler Angles $z\ x'\ z''$

Sequence of rotations:

1. Rotation by α around the z-axis $\mathbf{z}$ — $R_{\mathbf{z}}$
2. Rotation by β around the x-axis $\mathbf{x}'$ — $R_{\mathbf{x}'}$
3. Rotation by γ around the z-axis $\mathbf{z}''$ — $R_{\mathbf{z}''}$

$$R_s = R_{\mathbf{z}} R_{\mathbf{x}'} R_{\mathbf{z}''}$$

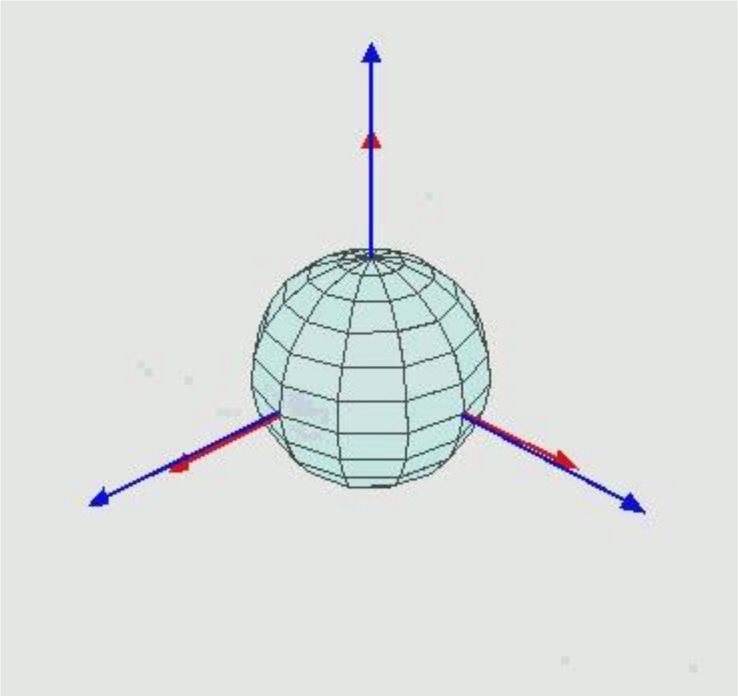
Important: Rotation around different axes!



Euler Angles

- **12 possible sequences** of rotation axis
  - $zxz, xyx, yzy, zyz, xzx, yxy$
  - $xyz, yzx, zxy, xzy, zyx, yxz$

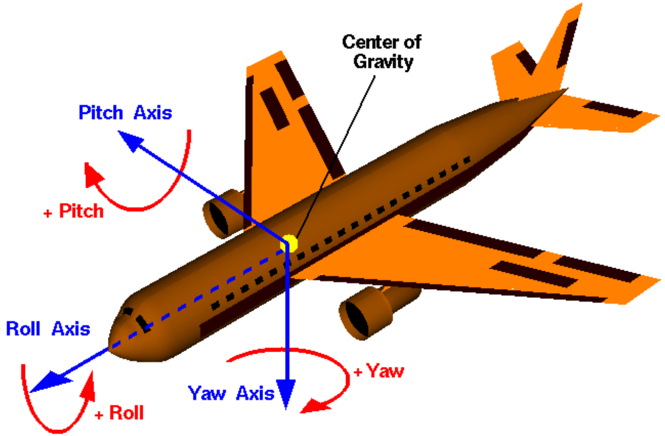
- Rotations around **local** or **fixed axis**
  $\Rightarrow$ in total **24 possible rotation**



Source: Wikipedia

Roll, Pitch und Yaw

- Another common convention is **Euler convention x, y, z**
- These special Euler angles are called **Roll, Pitch, Yaw**
- **Order of rotations:**
  1. Global x-axis around α (Roll)
  2. Global y-axis around β (Pitch)
  3. Global z-axis around γ (Yaw)



by NASA [Public domain], via wikimedia Commons

Euler Angles (III)

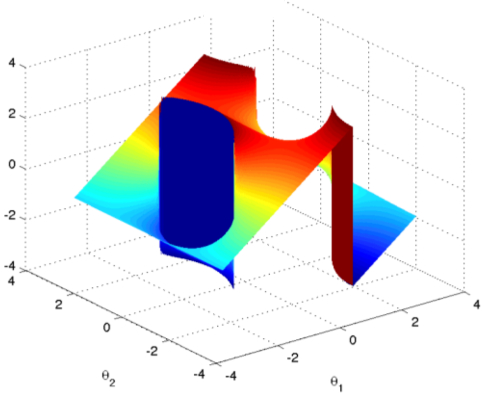
- Advantages of Euler angles:
  - More **compact** than rotation matrices
  - More **descriptive** than rotation matrices
- Disadvantages of Euler angles:
  - **Not unique:**
    - Example: in Euler z, x', z'' convention, Euler angles $(45°, 30°, -45°)$ and $(0°, 30°, -0°)$ result in the same rotation! This is called **Gimbal Lock**.
  - **Not continuous:**
    - Euler angles of a continuous rotation are not continuous.
    - Small changes in the orientation may lead to large changes in the Euler angles (next slide).
    - Consequence: smooth interpolation between two Euler angles is not possible

Euler Angles: Interpolation Problem

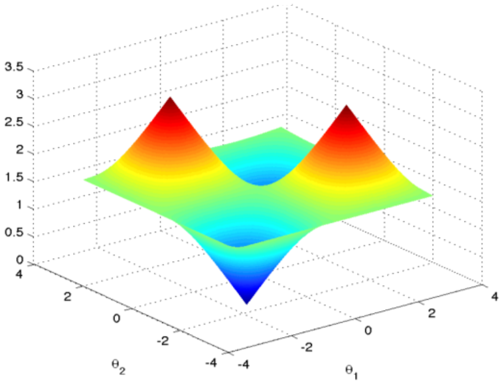
Not continuous:

- Euler angles of a continuous rotation are not continuous.
- Small changes in the orientation may lead to huge changes in the Euler angles
- Consequence: smooth interpolation between two Euler angles is not possible

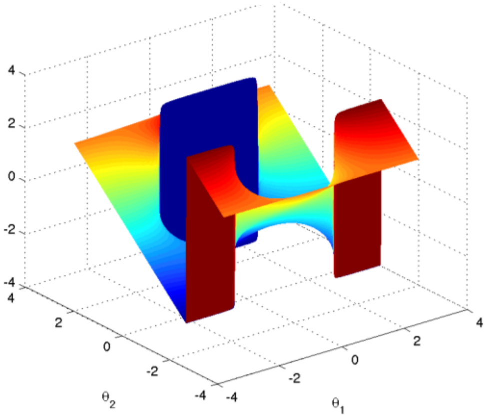
α



β



γ



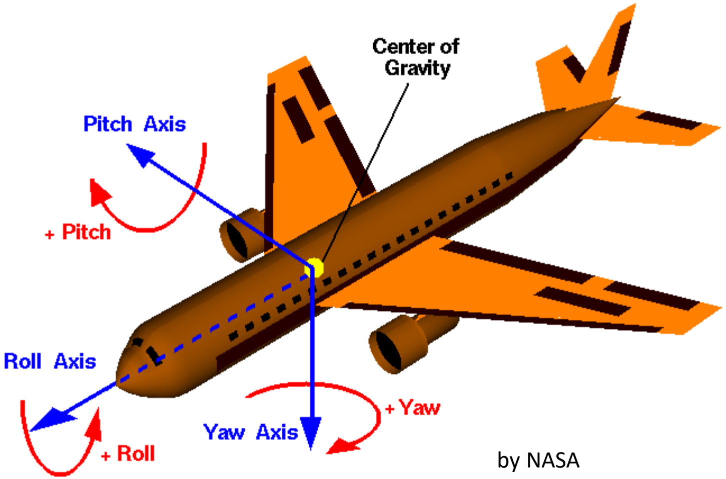
Euler Angles – Gimbal Lock (1)

- **12 different sequences** are possible for the rotation matrices:
  - $zxz \quad xyx \quad yzy \quad zyz \quad xzx \quad yxy$
  - $xyz \quad yzx \quad zxy \quad xzy \quad zyx \quad yxz$
- Rotation sequence xyz (Roll-Pitch-Yaw):

$$R_{\mathbf{z},\gamma} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{\mathbf{y},\beta} = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}$$

$$R_{\mathbf{x},\alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$



by NASA
[Public domain], via
Wikimedia Commons

Euler Angles – Gimbal Lock (2)

- Assumption: $\beta = -\frac{\pi}{2}$

$$\sin\left(-\frac{\pi}{2}\right) = -1, \quad \cos\left(-\frac{\pi}{2}\right) = 0 \quad \Rightarrow \quad R_{\mathbf{y},\,\beta=-\frac{\pi}{2}} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

- Multiplication of the matrices :

$$R = R_{z,\gamma} \cdot R_{y,\,\beta=-\frac{\pi}{2}} \cdot R_{x,\alpha} = \begin{pmatrix} 0 & -\sin\gamma & -\cos\gamma \\ 0 & \cos\gamma & -\sin\gamma \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

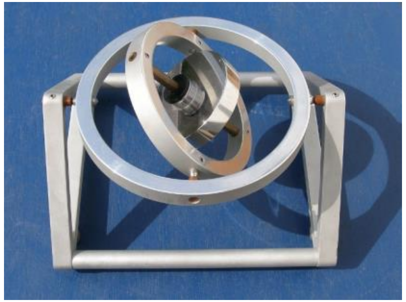
$$= \begin{pmatrix} 0 & -\sin\gamma\cos\alpha - \cos\gamma\sin\alpha & \sin\gamma\sin\alpha - \cos\gamma\cos\alpha \\ 0 & \cos\gamma\ \cos\alpha - \sin\gamma\sin\alpha & -\cos\gamma\sin\alpha - \sin\gamma\cos\alpha \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\sin(\alpha+\gamma) & -\cos(\alpha+\gamma) \\ 0 & \cos(\alpha+\gamma) & -\sin(\alpha+\gamma) \\ 1 & 0 & 0 \end{pmatrix}$$

⇒ Common rotation axis for rotation around α and γ → 1 DoF is lost

Changes to α and γ currently have the same effect

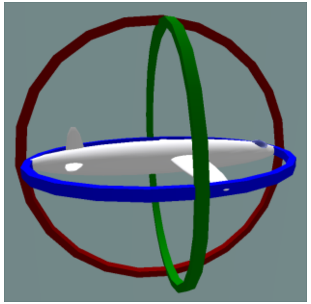
Euler Angles – Gimbal Lock (3)

- Gimbal (cardanic bearing) allows rotation around a predetermined axis
  - Combination of 3 elements to allow free movement
  - Measuring instruments such as gyroscope, compass

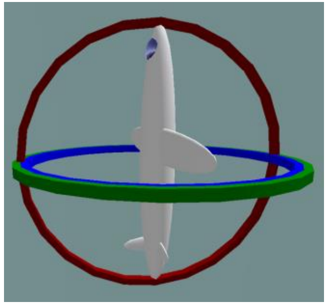


by Bautsch [Public domain], via Wikimedia Commons

- **Gimbal Lock**
  - At certain angles, two axes become dependent on each other
  - One degree of freedom is lost (→ no instantaneous speed possible in this degree of freedom)



3 DoF



2 DoF

by MathsPoetry [CC BY-SA 3.0], via Wikimedia Commons

Rotation Matrices vs. Euler Angles

Karlsruhe Institute of Technology

Rotation matrices	Euler angles
▪ “Natural” representation from the perspective of linear algebra ▪ Unambiguous, continuous ▪ Redundancy through 9 values	▪ More compact ▪ More meaningful ▪ Not unambiguous ▪ Gimbal Lock ▪ Not continuous

High Performance Humanoid Technologies

Euler Angles vs. Roll-Pitch-Yaw

Euler angles (z, x', z'')	**Roll-Pitch-Yaw (x, y, z)**
▪ Multiplication from left to right $\mathrm{R_s} = R_{\mathbf{z},\alpha}\, R_{\mathbf{x}',\beta}\, R_{\mathbf{z}'',\gamma}$	▪ Multiplication from right to left $\mathrm{R_s} = R_{\mathbf{z},\gamma}\, R_{\mathbf{y},\beta}\, R_{\mathbf{x},\alpha}$
▪ Each rotation is local (refers to the new coordinate system)	▪ Each rotation is global (refers to the global coordinate system)
▪ Rotation around **different** axes	▪ Rotation around **fixed** axes

Representation of orientation with 3×3 matrices

Assessment:

- **Advantage:** Vector and rotation matrix are descriptive and therefore a common way to represent poses (e.g. object and end effector pose)
- **Disadvantage:** Vector and matrix operations must be performed separately :

$$(\mathbf{p}, R) \text{ with } \mathbf{p} \in \mathbb{R}^3 \text{ and } R \in SO(3) \subset \mathbb{R}^{3\times 3}$$

Goal: **Closed representation** of rotation and translation in a matrix

$\rightarrow$ Use of affine transformations (projective geometry)

Affine Transformations (I)

- An affine space is an extension of the Euclidean space.
- It contains points and vectors expressed in **extended** (or **homogeneous**) coordinates:

$$\mathbf{a} = (a_x, a_y, a_z, h)^T, \quad \mathbf{a} \in \mathbb{R}^4, \quad h \in \{0,1\}$$

$h = 1$ for positions
$h = 0$ for directions

Affine Transformations (I)

- Affine transformations can be defined such that linear transformations in the Euclidean space (e.g., rotation, scaling and shear around the origin) can be combined with translations and be **expressed in homogeneous coordinates**:

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{t}$$

$$\mathbf{b} = \begin{pmatrix} \mathbf{b} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{o} \\ \mathbf{o}^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} + \begin{pmatrix} \mathbf{t} \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{o}^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

$$\boldsymbol{b}, \boldsymbol{x}, \boldsymbol{t}, \mathbf{o} \in \mathbb{R}^3 \quad \mathbf{A} \in \mathbb{R}^{3\times 3} \quad \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{o}^T & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4}$$

$\mathbf{o}$ represents the null vector

Affine Transformations: Advantages

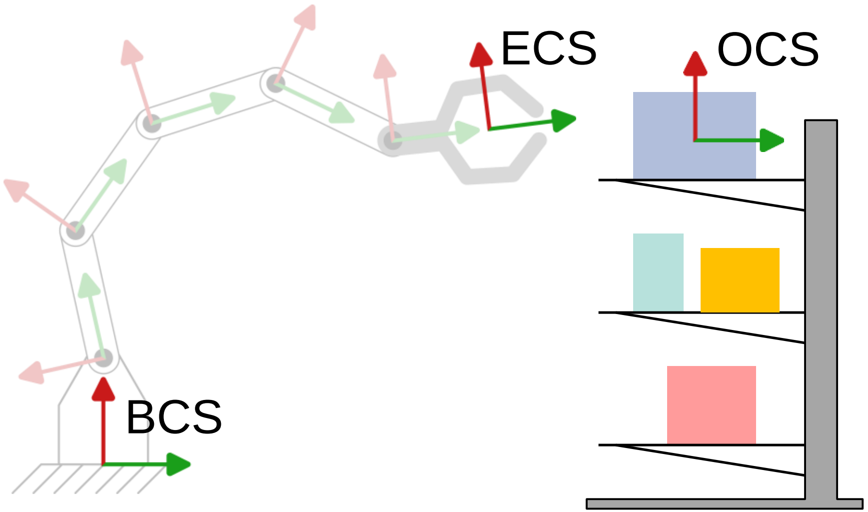
- It is possible to formulate **rotations around arbitrary axes** in affine space.

- **Rotations and translations** can be combine in a **single homogeneous 4×4 matrix**.

  This means that rotations and translations can be handled uniformly.

Coordinate Systems (Frames)

- Coordinate systems, also called frames:
  Can be defined at various locations
  - Basis coordinate system (**BCS**):
    Reference system, e.g.,
    in the **robot's base** or as a
    "world" coordinate system
  - End effector coordinate system (**ECS**):
    Attached to an **end effector**
  - Object coordinate system (**OCS**):
    Attached to an **object**



Homogeneous 4×4 –Matrix (1)

- Homogeneous 4×4 Matrix

$$T = \begin{pmatrix} A & \mathbf{t} \\ \mathbf{o}^T & 1 \end{pmatrix} \quad T \in SE(3) \quad \text{with} \quad \boldsymbol{t} \in \mathbb{R}^3 \text{ and } A \in SO(3)$$

- **Translation matrix:** Translation of object coordinate systems (OCS) to $(t_x, t_y, t_z)^T$ in the basis coordinate system (BCS)

$$T_{trans} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Homogeneous 4×4 –Matrix (2)

- Basic rotation matrices :

$$T_{x,\alpha} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{y,\beta} = \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{z,\gamma} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Example: Homogeneous Matrices

- Two points a and b should be translated by $+5$ units in x and by -3 units in z

$$\mathbf{a} = (4, 3, 2, 1)^\top \qquad \mathbf{b} = (6, 2, 4, 1)^\top$$

$$\mathbf{a}' = A \cdot \mathbf{a} = \begin{pmatrix} 1 & 0 & 0 & +5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 9 \\ 3 \\ -1 \\ 1 \end{pmatrix}$$

$$\mathbf{b}' = A \cdot \mathbf{b} = \begin{pmatrix} 1 & 0 & 0 & +5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 6 \\ 2 \\ 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 11 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

Homogeneous 4×4 Matrices: Inversion

$$\mathbf{b} = R \cdot \mathbf{x} + \mathbf{t} \quad \Leftrightarrow \quad \begin{pmatrix} \mathbf{b} \\ 1 \end{pmatrix} = T \cdot \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

1. Rotate $\mathbf{x}$ by R
2. Shift the result by $\mathbf{t}$ (in the *rotated* coordinate system)

- We are looking for the homogeneous matrix T^{-1}, which maps $\mathbf{b}$ back to $\mathbf{x}$:

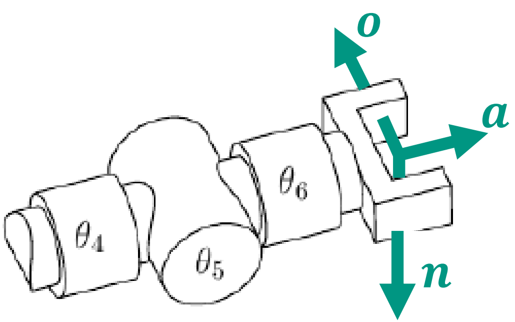
$$R \cdot \mathbf{x} + \mathbf{t} = \mathbf{b}$$
$$R \cdot \mathbf{x} = \mathbf{b} - \mathbf{t}$$
$$\mathbf{x} = R^{-1} \cdot (\mathbf{b} - \mathbf{t})$$
$$\mathbf{x} = R^{-1} \cdot \mathbf{b} - R^{-1} \cdot \mathbf{t}$$
$$\mathbf{x} = (R^{-1}) \cdot \mathbf{b} + (-R^{-1} \cdot \mathbf{t})$$
$$\mathbf{x} = (R^\top) \cdot \mathbf{b} + (-R^\top \cdot \mathbf{t})$$

$$\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = T^{-1} \cdot \begin{pmatrix} \mathbf{b} \\ 1 \end{pmatrix}$$

$$T^{-1} = \begin{pmatrix} R^\top & -R^\top \cdot \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}$$

Homogeneous 4×4 –Matrices

- Transformation of vector p_{OKS} (in OCS) into BCS:



$$p_{BCS} = T \cdot p_{OCS}$$

$$\text{mit:} \quad T = \begin{pmatrix} n_x & o_x & a_x & u_x \\ n_y & o_y & a_y & u_y \\ n_z & o_z & a_z & u_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{n} & \boldsymbol{o} & \boldsymbol{a} & \boldsymbol{u} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

u: Origin of OCS
n, o, a: Unit vectors of OCS in relation to BCS

$\boldsymbol{n}$ *normal*
$\boldsymbol{a}$ *approach*
$\boldsymbol{o}$ *orientation*

Homogeneous 4×4 –Matrices

- Inversion:

$$T = \begin{pmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{u} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} n_x & o_x & a_x & u_x \\ n_y & o_y & a_y & u_y \\ n_z & o_z & a_z & u_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T^{-1} = \begin{pmatrix} & R^\top & & -R^\top \mathbf{u} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} n_x & n_y & n_z & -\mathbf{n}^\top \mathbf{u} \\ o_x & o_y & o_z & -\mathbf{o}^\top \mathbf{u} \\ a_x & a_y & a_z & -\mathbf{a}^\top \mathbf{u} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Homogeneous 4×4 –Matrices

- A homogeneous 4×4 matrix contains 12 $(\mathbf{n}, \boldsymbol{o}, \mathbf{a}, \mathbf{u})$ n on-trivial variables as opposed to 6 $(x, y, z, \alpha, \beta, \gamma)$ necessary
- Redundancy, but with additional boundary conditions that guarantee orthogonality $(R \cdot R^\top = I)$
- Axes of rotation and rotation sequence are implicitly included

Comparison: Cartesian and Homogeneous Representation

- In Cartesian coordinates:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

- In homogeneous coordinates:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} n_x & o_x & a_x & t_x \\ n_y & o_y & a_y & t_y \\ n_z & o_z & a_z & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Interpretation of Homogeneous 4×4 Matrices

- **Pose description** of a coordinate system:

$^{A}P_{B}$ describes the position (pose) of the coordinate system B relative to the coordinate system A

- **Transformation mapping** (between coordinate systems):

$$^{A}T_{B}: \quad {}^{B}P \rightarrow {}^{A}P, \qquad\qquad {}^{A}P = {}^{A}T_{B} \cdot {}^{B}P$$

- **Transformation operator** (within a coordinate system):

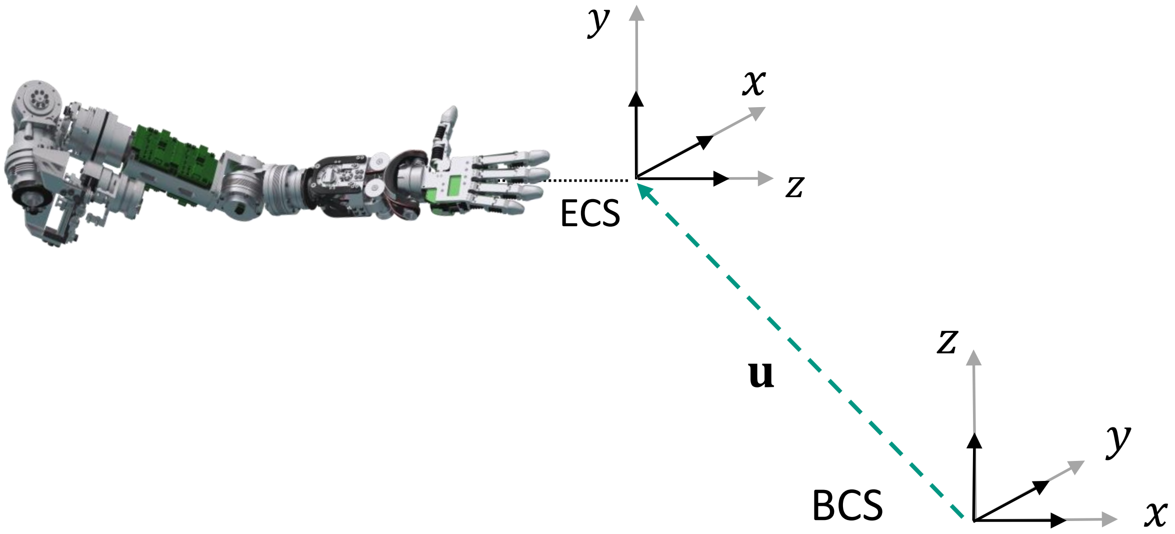
$$T: \quad {}^{A}P_{1} \rightarrow {}^{A}P_{2}, \qquad\qquad {}^{A}P_{2} = T \cdot {}^{A}P_{1}$$

Example: Coordinate System Transformation (1)

- Given: Point in the end effector coordinate system (ECS)

$$^{\mathrm{ECS}}\mathbf{p} \;=\; (0, -3, 5)^\mathsf{T}$$

- Requested: Point in the base coordinate system (BCS) $^{\mathrm{BCS}}\mathbf{p}$



$$\mathbf{u} = \begin{pmatrix} -7 \\ 0 \\ 8 \end{pmatrix}$$

$$R = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Example: Coordinate System Transformation (2)

- Given: Point in the end effector coordinate system (ECS)

$$^{\mathrm{ECS}}\mathbf{p} = (0, -3, 5)^{\mathsf{T}}$$

- Requested: Point in the base coordinate system (BCS) $^{\mathrm{BCS}}\mathbf{p}$

$$\mathbf{u} = \begin{pmatrix} -7 \\ 0 \\ 8 \end{pmatrix} \qquad R = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$^{\mathrm{BCS}}\mathbf{p} = \begin{pmatrix} 0 & 0 & 1 & -7 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -3 \\ 5 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \\ 5 \\ 1 \end{pmatrix}$$

Composition of Transformations (1)

Given

${}^{\mathrm{BCS}}T_A$	pose of object A in BCS
${}^{A}T_B$	pose of object B relative to OCS of A
${}^{\mathrm{BCS}}T_B$	pose of object B relative to BCS

$$\rightarrow \quad {}^{\mathrm{BCS}}T_B = {}^{\mathrm{BCS}}T_A \cdot {}^{A}T_B$$

More compact notation compared to Cartesian representation:

$$R_{Bneu} + \mathbf{t}_{Bneu} = R_A \cdot (R_B + \mathbf{t}_B) + \mathbf{t}_A = R_A \cdot R_B + (R_A \cdot \mathbf{t}_B + \mathbf{t}_A)$$

Composition of Transformations (1)

- Pose of object 1 in BCS: ${}^{\mathrm{BCS}}T_{O1}$
- Pose of object 2 relative to object 1 : ${}^{O_1}T_{O_2}$
- Pose of object 3 relative to object 2 : ${}^{O_2}T_{O_3}$
- Pose of object 3 relative to BCS ${}^{\mathrm{BCS}}T_{O_3}$

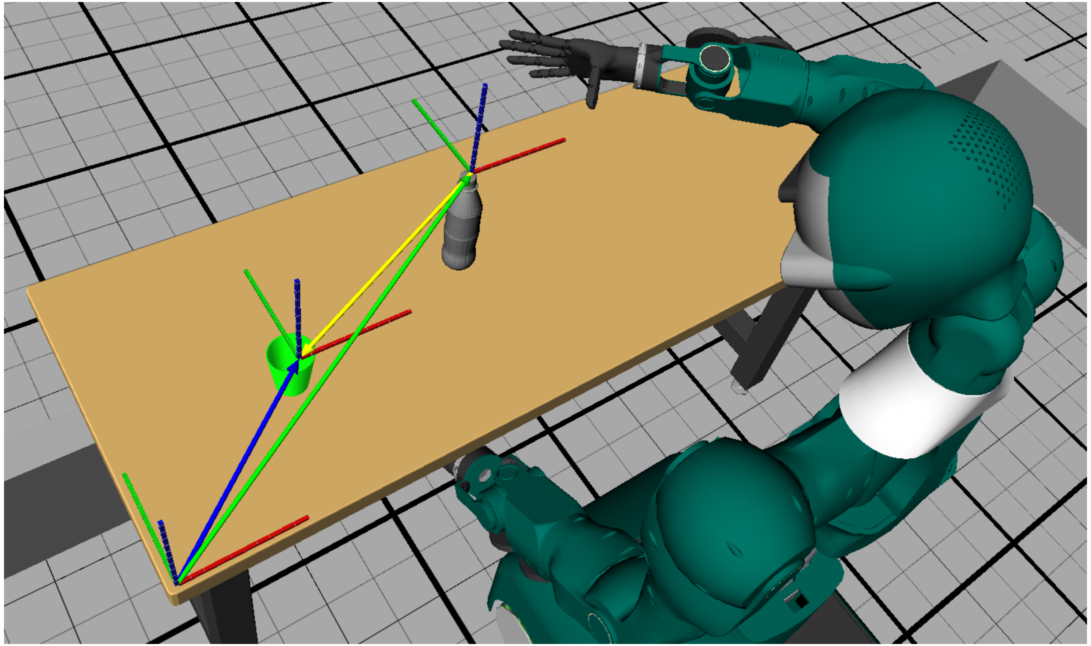
$$ {}^{\mathrm{BCS}}T_{O_3} = {}^{\mathrm{BCS}}T_{O_1} \cdot {}^{O_1}T_{O_2} \cdot {}^{O_2}T_{O_3} $$

In representations using product of matrices, each matrix must refer to the position defined by the matrix on the left:

$$ {}^{A_0}T_{A_n} = \prod_{i=1}^{n} {}^{A_{i-1}}T_{A_i} \qquad \text{with } A_0 = \mathrm{BCS} $$

Example

$$ {}^{BCS}H_{\text{cup}} = {}^{BCS}H_{\text{bottle}} \cdot {}^{bottle}H_{\text{cup}} $$



Problems with Rotation Matrices and Euler Angles ?

- Problems with rotation matrices
  - Highly redundant
  - Computationally intensive (matrix multiplication)
  - Interpolation difficult

- Problems with Euler angles:
  - Singularities (discontinuous)

- **Are there other representations for rotations which avoid these problems?**

Quaternions to Represent Orientations

- Are there other representations for rotations which avoid these problems?

- **Answer: Yes, Quaternions!**
  - Quaternions are a extension of complex numbers (”hypercomplex numbers“)
  - Introduced 1843 by William Rowan Hamilton
  - Used in robotics and computer graphics
  - See Horn 1987 for an overview

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

Quaternions

- Broome Bridge in Dublin

Quaternions: Definition

- The set of **quaternions** $\mathbb{H}$ is defined by

$$\mathbb{H} = \mathbb{C} + \mathbb{C}\, j \quad \text{with} \quad j^2 = -1 \quad \text{and} \quad i \cdot j = -j \cdot i = k$$

- An element $\mathbf{q} \in \mathbb{H}$ has the following form

$$\mathbf{q} = (a, \mathbf{u})^\top = a + u_1 i + u_2 j + u_3 k \quad \text{with } a \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^3 \text{ and } k = i \cdot j$$

  - a is referred to as the **real part**
  - $\mathbf{u} = (u_1, u_2, u_3)^\top$ is referred to as the **imaginary part**

- In code, common notations are (w, x, y, z) or (x, y, z, w) with $w = a$ and $(x, y, z) = \mathbf{u}$

Formula for Quaternions (1)

$$\boldsymbol{q} = (a, \boldsymbol{u})^\top = a + u_1 i + u_2 j + u_3 k$$

$$\begin{aligned} i^2 &= j^2 = k^2 = i \cdot j \cdot k = -1 \\ i \cdot j &= -j \cdot i = k \\ k \cdot i &= -i \cdot k = j \end{aligned}$$

(not commutative!)

$\cdot$	$\mathbf{1}$	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
$\mathbf{1}$	1	i	j	k
$\boldsymbol{i}$	i	-1	k	$-j$
$\boldsymbol{j}$	j	$-k$	-1	i
$\boldsymbol{k}$	k	j	$-i$	-1

Formula for Quaternions (2)

- Given two quaternions $\mathbf{q}, \mathbf{r}$:

$$\mathbf{q} = (a, \mathbf{u})^\top, \qquad \mathbf{r} = (b, \mathbf{v})^\top$$

- **Addition**:

$$\mathbf{q} + \mathbf{r} = (a + b, \mathbf{u} + \mathbf{v})^\top$$

- **Scalar product**:

$$\langle \mathbf{q} | \mathbf{r} \rangle = a \cdot b + \langle \mathbf{v} | \mathbf{u} \rangle = a \cdot b + v_1 \cdot u_1 + v_2 \cdot u_2 + v_3 \cdot u_3$$

- **Multiplication**:

$$\mathbf{q} \cdot \mathbf{r} = (a + u_1 i + u_2 j + u_3 k) \cdot (b + v_1 i + v_2 j + v_3 k)$$

Formula for Quaternions (3)

- Quaternion:

$$\mathbf{q} = (a, \mathbf{u})^\top$$

- **Conjugated** quaternion:

$$\mathbf{q}^* = (a, -\mathbf{u})^\top$$

- **Norm** of a quaternion:

$$\|\mathbf{q}\| = \sqrt{\mathbf{q} \cdot \mathbf{q}^*} = \sqrt{\mathbf{q}^* \cdot \mathbf{q}} = \sqrt{a^2 + u_1^2 + u_2^2 + u_3^2}$$

- **Inverse** of a quaternion:

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2}$$

Quaternions: Rotations (1)

Unit quaternions $\mathbb{S}^3 = \{\mathbf{q} \in \mathbb{H} \mid ||\mathbf{q}||^2 = 1\}$

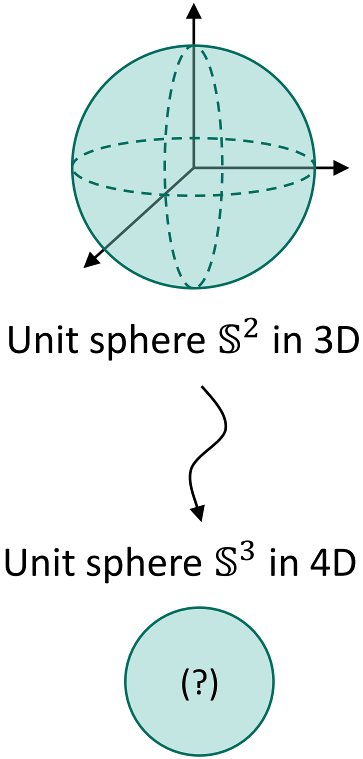
- Exist on the unit sphere $\mathbb{S}^3$ in 4D
  - Norm = 1

  $\Rightarrow$ 1 of 4 „degrees of freedom“ defined

  $\Rightarrow$ 3 „ degrees of freedom“ remaining

- Form a group
  - Group properties (reminder):
    - Associative law
    - Existence of an inverse element for each group element
    - Existence of an identity

- **Define rotations** There is an embedding from $\mathrm{SO}(3) \subset \mathbb{R}^3$ to $\mathbb{H}$



Quaternions: Rotations (2)

Question: How do you represent a rotation of, e.g., 46° around the axis $(0,1,0)^\top$ as a quaternion?

- **vector** $\mathbf{p} \in \mathbb{R}^3$ as a quaternion $\mathbf{q}$:

$$\mathbf{p} = (x, y, z)^\top \quad \Longrightarrow \quad \mathbf{q} = (0, \mathbf{p})^\top$$

- **scalars** $s \in \mathbb{R}$ as a quaternion $\mathbf{q}$:

$$\mathbf{q} = (s, \mathbf{0})^\top$$

Quaternions: Rotations (3)

- A rotation described by a **rotation axis** $\mathbf{a}$ with unit length and an **angle** ϕ can be represented by a quaternion:

$$\mathbf{q} = \left(\cos\frac{\phi}{2}, \mathbf{a} \cdot \sin\frac{\phi}{2}\right)$$

- Applying the rotation $\mathbf{q}$ to a point $\mathbf{p}$:

$$\mathbf{v}' = \mathbf{q} \cdot \mathbf{v} \cdot \mathbf{q}^{-1} \qquad \text{with } \mathbf{v} = (0, \mathbf{p})^{\mathrm{T}}$$

- As $\mathbf{q}$ is a unit quaternion, we have $\mathbf{q}^{-1} = \mathbf{q}^*$, and therefore:

$$\mathbf{v}' = \mathbf{q} \cdot \mathbf{v} \cdot \mathbf{q}^*$$

Quaternions: Rotations (4)

- Concatenation of rotations of a vector $\mathbf{v}$ with two quaternions $\mathbf{q}$ and $\mathbf{r}$:

$$\mathbf{q} = \left(\cos\frac{\phi_q}{2}, \mathbf{u_q} \cdot \sin\frac{\phi_q}{2}\right), \qquad \mathbf{r} = \left(\cos\frac{\phi_r}{2}, \mathbf{u_r} \cdot \sin\frac{\phi_r}{2}\right)$$

- Rotation with one quaternion:

$$\mathrm{f}(\mathbf{v}) = \mathbf{q} \cdot \mathbf{v} \cdot \mathbf{q}^*, \qquad \mathrm{h}(\mathbf{v}) = \mathbf{r} \cdot \mathbf{v} \cdot \mathbf{r}^*$$

- Then $\mathrm{f} \circ \mathrm{h}$ describes the rotation by the quaternion $\mathbf{p} = \mathbf{q} \cdot \mathbf{r}$

$$(\mathrm{f} \circ \mathrm{h})(\mathrm{v}) = \mathrm{f}(\mathrm{h}(\mathrm{v})) = \mathbf{q} \cdot (\mathbf{r} \cdot \mathbf{v} \cdot \mathbf{r}^*) \cdot \mathbf{q}^*$$

- $f \circ h$ corresponds to the rotation with the quaternion $\mathbf{s} = \mathbf{q} \cdot \mathbf{r}$
  $\Rightarrow$ concatenation $\widehat{=}$ multiplication

Quaternions: Example

- Rotation of the point $\mathbf{p} = (1, 0, 9)^\top$
  about the axis of rotation $\mathbf{a} = (1, 0, 0)^\top$
  with angles $\theta = 90°$

Quaternions: Example

- Example: Rotation of the point about the axis of rotation with angles

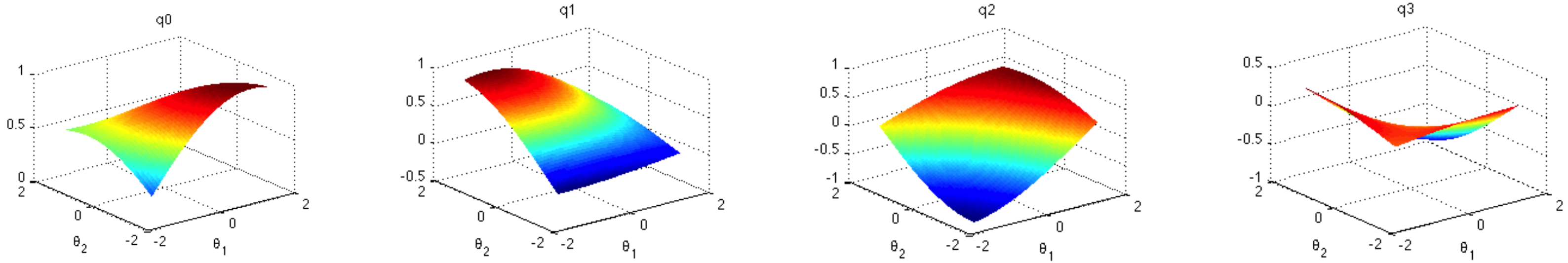
$$\mathbf{p} = (1, 0, 9)^\mathsf{T}$$
$$\mathbf{a} = (1, 0, 0)^\mathsf{T}$$
$$\theta = 90°$$

1. Representation of $\mathbf{p}$ as quaternion $\mathbf{v}$ $\quad \mathbf{v} = 0 + 1i + 0j + 9k$
2. Rotation quaternion $\mathbf{q}$ $\quad \mathbf{q} = \cos\frac{\theta}{2} + 1i \cdot \sin\frac{\theta}{2} + 0j + 0k$
3. Conjugated Quaternion $\mathbf{q}^*$ $\quad \mathbf{q}^* = \cos\frac{\theta}{2} - 1i \cdot \sin\frac{\theta}{2} - 0j - 0k$
4. Rotation of $\mathbf{v}$ around $\mathbf{q}$ $\quad \mathbf{v}_\mathrm{r} = \mathbf{q}\,\mathbf{v}\,\mathbf{q}^* \quad \rightarrow \mathbf{v}_\mathrm{r} = 0 + 1i - 9j + 0k$
5. Representation as point $\mathbf{p}_\mathrm{r}$ $\quad \mathbf{p}_\mathrm{r} = (1, -9, 0)^\mathsf{T}$

Note: The multiplication of quaternions is not commutative.

Representing Rotations with Quaternions

- Advantages:
  - Compact: 4 Values instead of 9 (rotation matrix)
  - Illustrative (related to the axis/angle representation)
  - Can be concatenated similar to rotation matrices
  - Can be used for the calculation of the inverse kinematics (later)
  - Unambiguous (no Gimbal lock)
  - The representation is continuous (no jumps, see figures below)
- Drawback:
  - Only for rotations, not for translations



Quaternions: Interpolation

- **Goal:** Continuous rotation between two orientations
- **Problems:**
  - Euler angles are not continuous
  - Rotation matrices have many degrees of freedom
- Interpolation of quaternions using **SLERP** (Spherical Linear Interpolation)
- Similar to linear interpolation: $a \cdot (1 - t) + b \cdot t$

Quaternions: SLERP

- SLERP interpolation from $\mathbf{q}_1$ to $\mathbf{q}_2$ with the parameter $t \in [0, 1]$:

$$\mathrm{Slerp}(\mathbf{q}_1, \mathbf{q}_2, t) = \mathbf{q_1} \cdot (\mathbf{q}_1^{-1} \cdot \mathbf{q}_2)^t$$

  (Powers of quaternions are not covered in the lecture)

- Direct formulation of the SLERP interpolation:

$$\mathrm{Slerp}(\mathbf{q}_1, \mathbf{q}_2, t) = \frac{\sin\big((1-t)\cdot\theta\big)}{\sin\theta} \cdot \mathbf{q}_1 + \frac{\sin(t\cdot\theta)}{\sin\theta} \cdot \mathbf{q}_2 \quad \text{with} \quad \langle \mathbf{q_1} | \mathbf{q_2} \rangle = \cos\theta$$

- Result: Rotation with constant angular velocity

Quaternions: Interpolation Problems

- **Problem:** Orientations in $\mathrm{SO}(3)$ are covered twice by unit quaternions because the unit quaternions $\mathbf{q}$ and $-\mathbf{q}$ correspond to the same rotation.

  Proof:
  - Rotation of $\mathbf{v}$ around $\mathbf{q}$ correspond to rotation of $\mathbf{v}$ around $-\mathbf{q}$.
  - $\mathbf{v}_\mathrm{r} = \mathbf{q}\,\mathbf{v}\,\mathbf{q}^* = (-\mathbf{q})\,\mathbf{v}\,(-\mathbf{q})^*$
  - The negative signs cancel each other out.

- SLERP therefore does not always calculate the shortest rotation
  $\Rightarrow$ It must be checked whether the rotation from $\mathbf{q}_1$ to $\mathbf{q}_2$ or $-\mathbf{q}_1$ to $\mathbf{q}_2$ is shorter

Dual Quaternions (1)

Problem:

- Real quaternions (as before) are suitable for describing the orientation, …
- but **not to describe the position** of an object (translation is missing).

Idea:

- Replace the 4 real values of a quaternion with **dual numbers**
- Obtain additional translational components to express the position of an object

→ **Dual Quaternions**

Duals Quaternions (2): Dual Numbers

- Dual numbers are of the form

$$\boldsymbol{d} = \boldsymbol{p} + \boldsymbol{\varepsilon} \cdot \boldsymbol{s}, \ \textbf{with } \boldsymbol{\varepsilon}^2 = \boldsymbol{0}$$

- Primary part p, secondary part s
- Similar to complex numbers, the usual operations can be derived
- If $d_1 = p_1 + \varepsilon \cdot s_1$ and $d_2 = p_2 + \varepsilon \cdot s_2$ are dual numbers, then the following applies:
  - Addition: $d_1 + d_2 = p_1 + p_2 + \varepsilon \cdot (s_1 + s_2)$
  - Multiplication: $d_1 \cdot d_2 = p_1 \cdot p_2 + \varepsilon \cdot (p_1 \cdot s_2 + p_2 \cdot s_1)$

Duale Quaternions (3)

Description

$$DQ = (d_1, d_2, d_3, d_4), \qquad d_i = dp_i + \varepsilon \cdot ds_i$$

- Primary part dp_i contains the **angle value** $\theta/2$
- Secondary part ds_i contains the **translation value** $d/2$

Dual Quaternions (4)

Multiplication table for dual unit quaternions

$\cdot$	1	i	j	k	ε	εi	εj	εk
1	1	i	j	k	ε	εi	εj	εk
i	i	-1	k	$-j$	εi	$-\varepsilon$	εk	$-\varepsilon j$
j	j	$-k$	-1	i	εj	$-\varepsilon k$	$-\varepsilon$	εi
k	k	j	$-i$	-1	εk	εj	$-\varepsilon i$	$-\varepsilon$
ε	ε	εi	εj	εk	0	0	0	0
εi	εi	$-\varepsilon$	εk	$-\varepsilon j$	0	0	0	0
εj	εj	$-\varepsilon k$	$-\varepsilon$	εi	0	0	0	0
εk	εk	εj	$-\varepsilon i$	$-\varepsilon$	0	0	0	0

Dual Quaternions (5)

- Rotation around an axis **a** with the θ:

$$\mathbf{q}_r = \left(\cos\left(\frac{\theta}{2}\right), \mathbf{a}\cdot\sin\left(\frac{\theta}{2}\right)\right) + \varepsilon\cdot(0,0,0,0)$$

- Translation with the vector $\mathbf{t} = (t_x, t_y, t_z)$

$$\mathbf{q}_t = (1,0,0,0) + \varepsilon\cdot\left(0,\frac{t_x}{2},\frac{t_y}{2},\frac{t_z}{2}\right)$$

- Combination for a transformation T:

$$\boldsymbol{q}_T = \boldsymbol{q}_t\,\boldsymbol{q}_r$$

Duale Quaternions (6)

- A transformation $\boldsymbol{T}$ with the rotational part $\boldsymbol{r}$ and the translational part $\boldsymbol{t}$, can be described as a dual quaternion:

$$\mathbf{q}_T = \mathbf{q}_t\, \mathbf{q}_r$$

- A transformation $\mathbf{q}_T$ is applied to a point $\mathbf{p}$ (as a dual quaternion) as follows:

$$\mathbf{p}' = \mathbf{q}_T\, \mathbf{p}\, \mathbf{q}_T{}^*, \text{ with } \mathbf{q}_T{}^* = (\mathbf{q}_t\, \mathbf{q}_r)^* = \mathbf{q}_r{}^* \mathbf{q}_t{}^*$$

- Conjugate (complex and dual) from $\mathbf{q} = \mathbf{p} + \varepsilon \cdot \mathbf{s}$:

$$\mathbf{q}^* = \mathbf{p}^* - \varepsilon \cdot \mathbf{s}^*$$

Duale Quaternions: Example (1)

- Example: Rotation of point around rotation axis and translation with
  $\mathbf{p} = (3,4,5)^\top$
  $\mathbf{a} = (1,0,0)^\top$ mit $\theta = 180°$
  $\mathbf{p}_t = (4,2,6)^\top$

- $\mathbf{p}$ as a dual quaternion $\mathbf{v}_d$ — $\mathbf{v}_d = 1 + 3\varepsilon i + 4\varepsilon j + 5\varepsilon k$
- Rotation as dual quaternion $\mathbf{q}_r$ — $\mathbf{q}_r = \cos\frac{\theta}{2} + 1i \cdot \sin\frac{\theta}{2} + 0j + 0k = i$
- Translation as a dual quaternion $\mathbf{q}_t$ — $\mathbf{q}_t = 1 + 2\varepsilon i + 1\varepsilon j + 3\varepsilon k$

- Combination as dual quaternion $\mathbf{q}_T$

$$\mathbf{q}_T = \mathbf{q}_t \cdot \mathbf{q}_r = (1 + 2i\varepsilon + 1j\varepsilon + 3k\varepsilon) \cdot i = i - 2\varepsilon - 1\varepsilon k + 3\varepsilon j$$

Duale Quaternions: Example (2)

- Example: Rotation of point $\mathbf{p} = (3,4,5)^\top$
  around rotation axis $\mathbf{a} = (1,0,0)^\top$ with $\theta = 180°$
  and translation with $\mathbf{p}_t = (4,2,6)^\top$

$$\mathbf{q}_T = (0 + i) + \varepsilon(-2 - 1k + 3j) = i - 2\varepsilon - 1\varepsilon k + 3\varepsilon j$$
$$\mathbf{q}_T^* = (0 - i) - \varepsilon(-2 + 1k - 3j) = -i + 2\varepsilon + 3\varepsilon j - 1\varepsilon k$$

- Transformation:

$$\mathbf{v}_T = \mathbf{q}_T\, \mathbf{v}_d\, \mathbf{q}_T^* = (i - 2\varepsilon - 1\varepsilon k + 3\varepsilon j)(1 + 3\varepsilon i + 4\varepsilon j + 5\varepsilon k)\, \mathbf{q}_T^*$$

$$= (i - 5\varepsilon - 2\varepsilon j + 3\varepsilon k)(-i + 2\varepsilon + 3\varepsilon j - 1\varepsilon k)$$

$$= 1 + 7\varepsilon i - 2\varepsilon j + 1\varepsilon k$$

- Result: $\mathbf{p}_T = (7,\ -2,\ 1)^\top$

Duale Quaternions: Example (3)

- Example: Rotation of point $\mathbf{p} = (3, 4, 5)^\top$
  around rotation axis $\mathbf{a} = (1, 0, 0)^\top$ with $\theta = 180°$
  and translation with $\mathbf{p}_t = (4, 2, 6)^\top$
- Result: $\mathbf{p}_T = (7,\ -2,\ 1)^\top$

- Test:
  - Rotation around the x axis with $\phi = 180°$

$$\mathbf{p}_r = (3, -4, -5)^\top$$

  - Translation with $\mathbf{p}_t = (4,2,6)^\top$:

$$\mathbf{p}_T = \mathbf{p}_r + \mathbf{p}_t = (3, -4, -5)^\top + (4, 2, 6)^\top = (7, -2, 1)^\top$$

Dual Quaternions: Evaluation

Advantages:

- Dual quaternions are suitable for describing the pose of an object
- Operations on dual quaternions also allow all required transformations
- Low redundancy, as only 8 values compared to 12 values of the homogeneous matrix representation
- Generally low number of individual operations per arithmetic operation

Disadvantages:

- Difficulty for the user to describe a pose by specifying a dual quaternion
- Complex processing instructions (e.g. for multiplication)

Summary

- Different forms of representation for rotations and translations in Euclidean space
  - Rotation matrix and translation vector
  - Euler angles
  - Homogeneous $4x4$ matrix
  - Quaternions
  - Dual quaternions

- Each representation has specific advantages and disadvantages
- Concrete application determines the choice of method