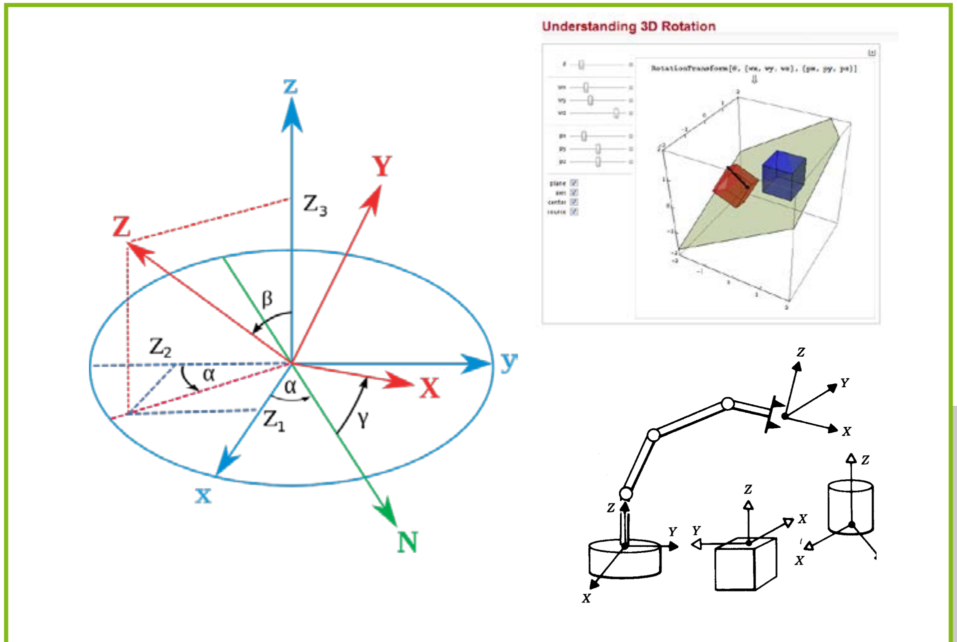
technische universität dortmund



Summer Term 2019

version 09.04.2019

Modeling and Control of Robotic Manipulators

Modellierung und Regelung von Robotern

Spatial Transformations

apl. Prof. Dr. rer. nat. Frank Hoffmann

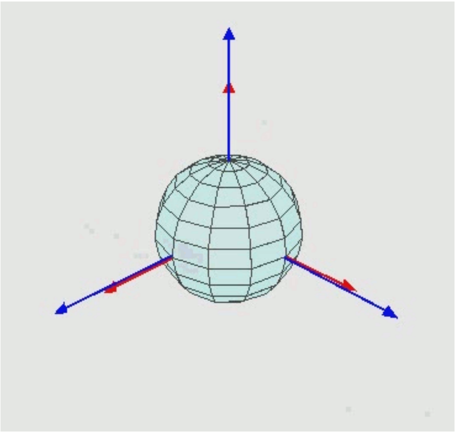
Lehrstuhl für Regelungssystemtechnik

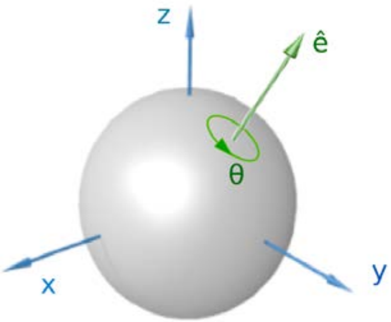
Spatial Transformations

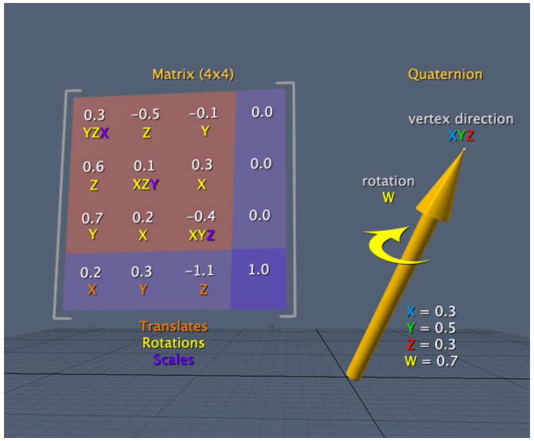
- rotation matrices
- Euler angles
- angle-axis
- unit quaternions
- homogeneous transformations

$$\mathbf{R} = \begin{bmatrix} \mathbf{x}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{x} & \mathbf{z}'^T\mathbf{x} \\ \mathbf{x}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{y} & \mathbf{z}'^T\mathbf{y} \\ \mathbf{x}'^T\mathbf{z} & \mathbf{y}'^T\mathbf{z} & \mathbf{z}'^T\mathbf{z} \end{bmatrix}$$

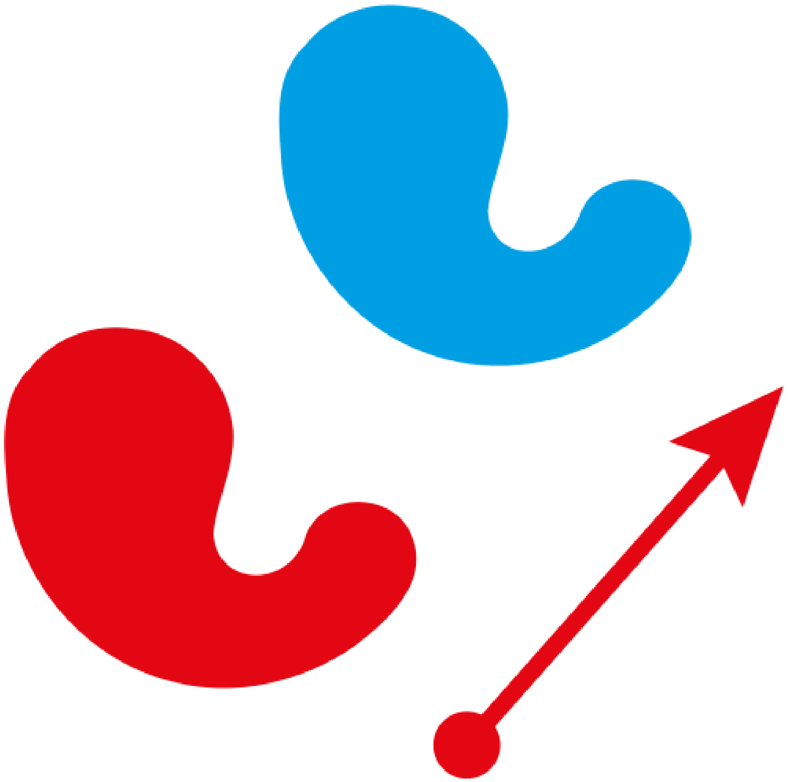
$$\mathbf{A}_1^0 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{o}_1^0 \\ \mathbf{0}^T & 1 \end{bmatrix}$$





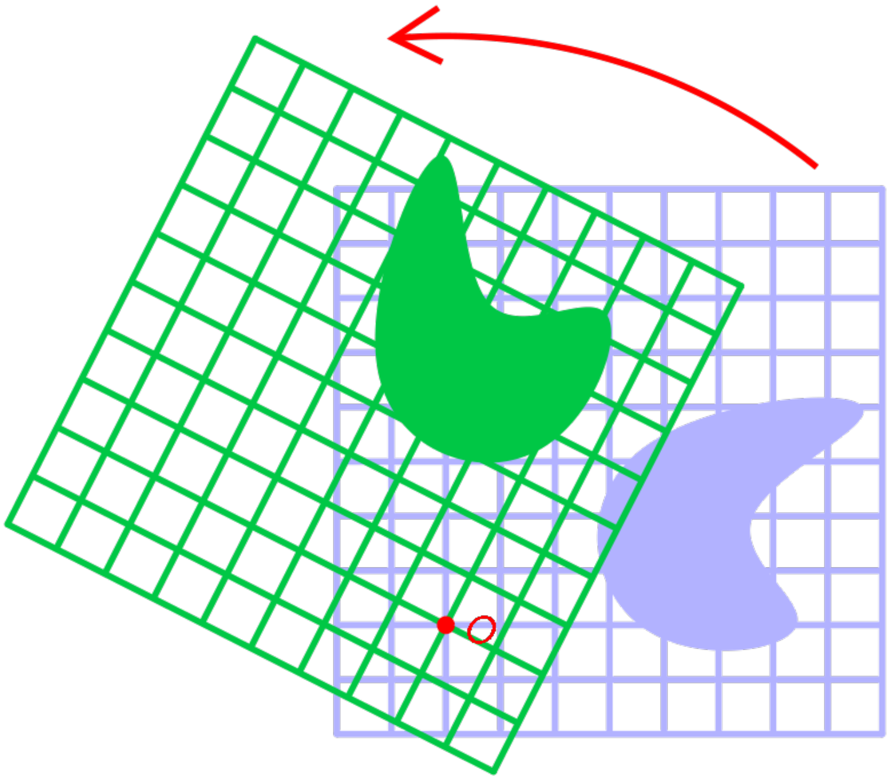


Translation



By Fred the Oyster, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=35017753

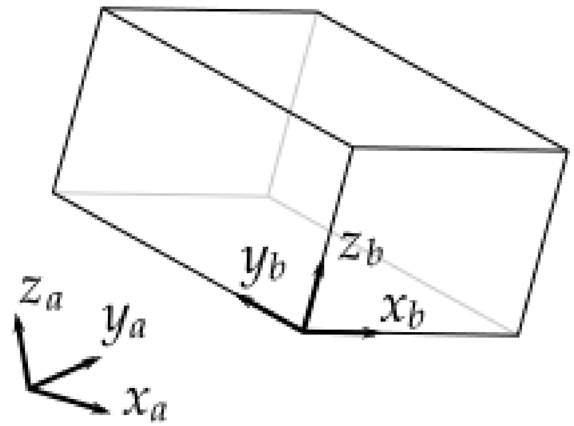
Rotation



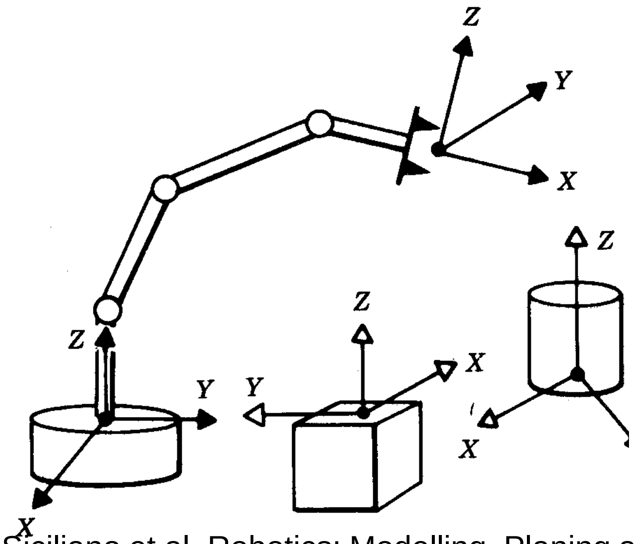
By Oleg Alexandrov - self-made, with MATLAB, then tweaked with en:Inkscape, Public Domain, https://commons.wikimedia.org/w/index.php?curid=2220861

Representation of Pose and Orientation of Rigid Body

- Transformation among two reference frames
  - translation (3 DOF)
  - rotation (3 DOF)
- translation
  - 3D-vector
- rotation
  - Rotation matrix
  - Euler angles
  - Angle-axis
  - Quaternions
- translation and rotation
  - homogeneous matrix
  - screws
  - 6D-vector

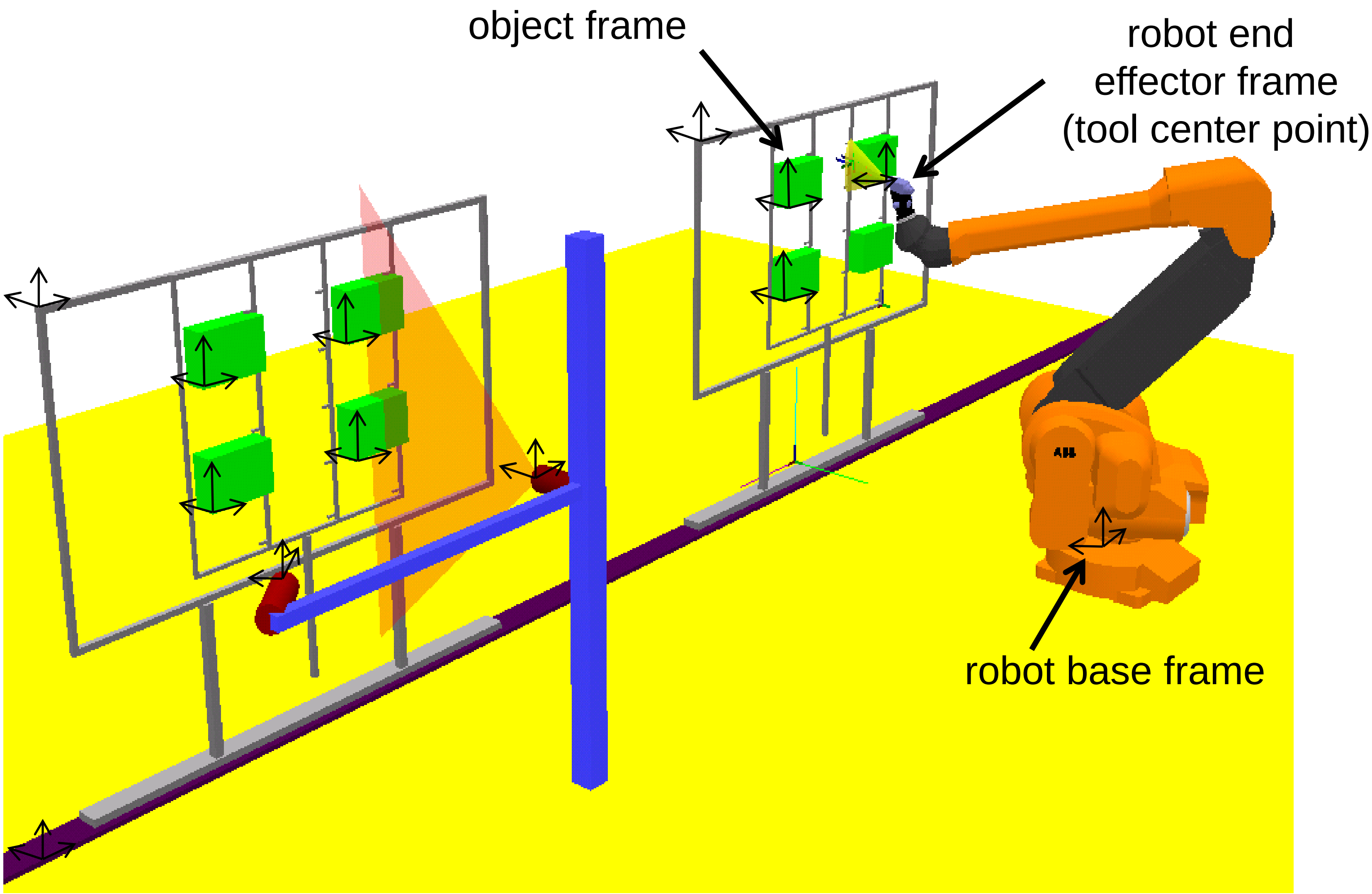


https://en.wikibooks.org/wiki/File:Rigid_body_attached_frame.svg



Siciliano et al, Robotics: Modelling, Planing and Control

Robotic Reference Frames

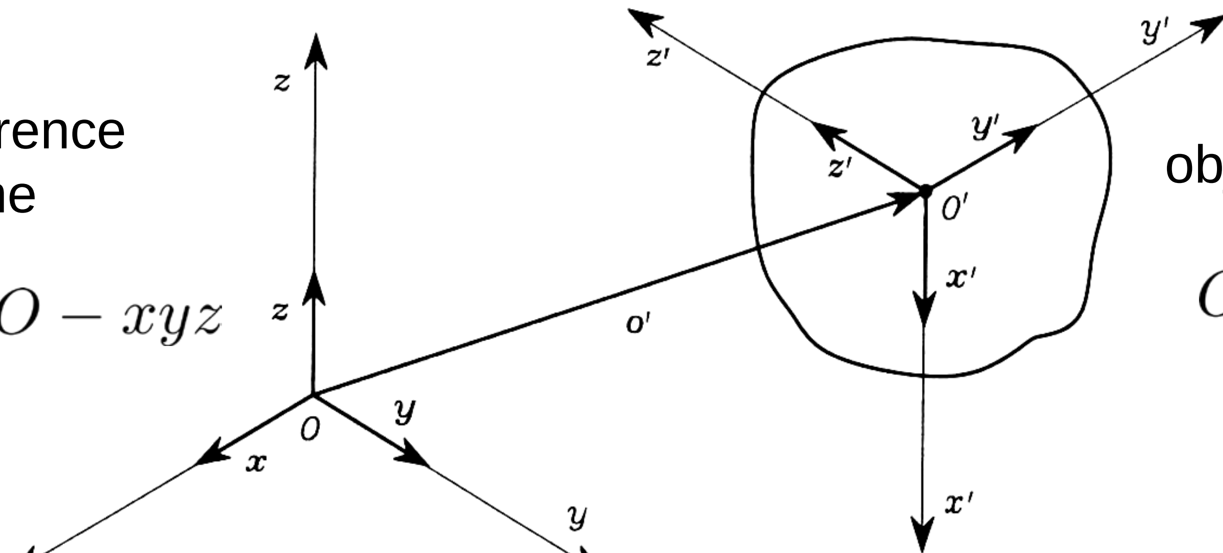


Position and Orientation of a Rigid Body

Pose : position + orientation

reference
frame

$O - xyz$



object frame

$O' - x'y'z'$

Siciliano et al, Robotics: Modelling, Planing and Control

$$\mathbf{o}' = \mathbf{o}'_x\mathbf{x} + \mathbf{o}'_y\mathbf{y} + \mathbf{o}'_z\mathbf{z}$$

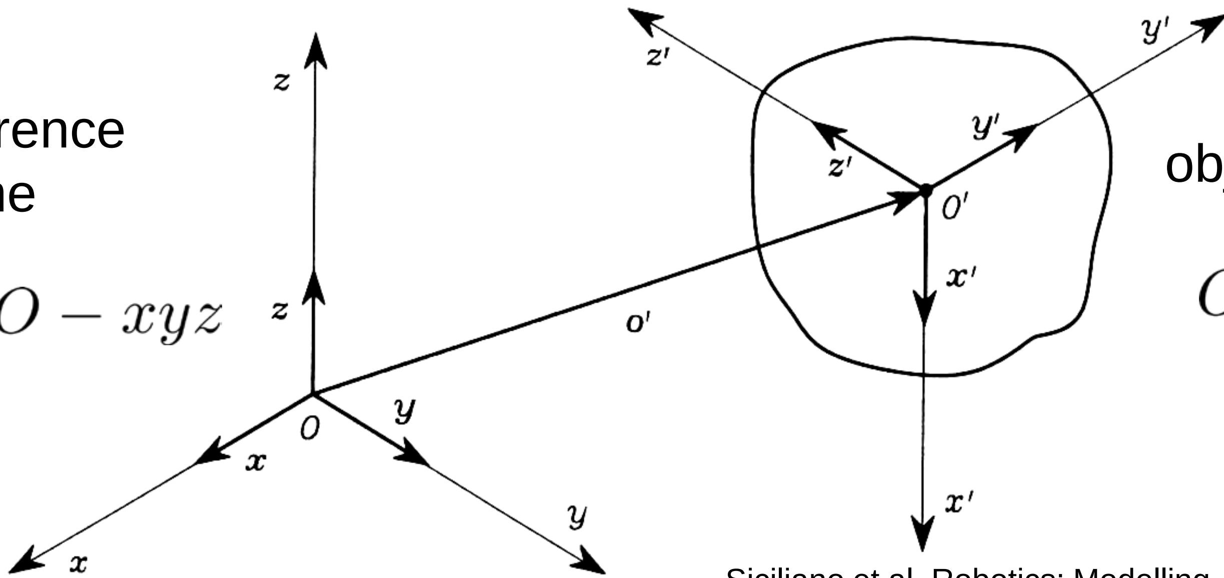
position: $$\mathbf{o}' = \begin{pmatrix} \mathbf{o}'_x \\ \mathbf{o}'_y \\ \mathbf{o}'_z \end{pmatrix}$$

orientation
$$\mathbf{x}' = \mathbf{x}'_x\mathbf{x} + \mathbf{x}'_y\mathbf{y} + \mathbf{x}'_z\mathbf{z}$$
$$\mathbf{y}' = \mathbf{y}'_x\mathbf{x} + \mathbf{y}'_y\mathbf{y} + \mathbf{y}'_z\mathbf{z}$$
$$\mathbf{z}' = \mathbf{z}'_x\mathbf{x} + \mathbf{z}'_y\mathbf{y} + \mathbf{z}'_z\mathbf{z}$$

Translation of a Rigid Body

reference frame

$O - xyz$



object frame

$O' - x'y'z'$

Siciliano et al, Robotics: Modelling, Planing and Control

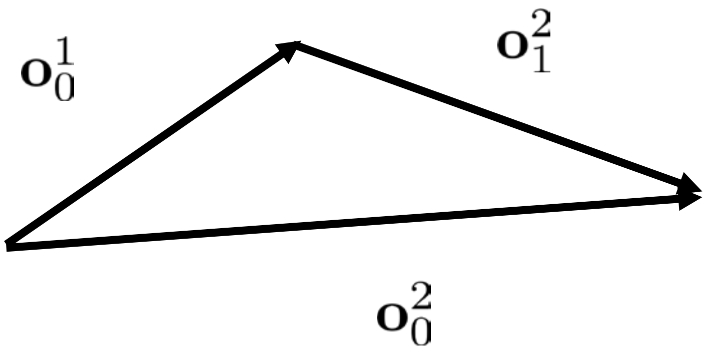
$$\mathbf{o}' = \mathbf{o}'_x \mathbf{x} + \mathbf{o}'_y \mathbf{y} + \mathbf{o}'_z \mathbf{z}$$

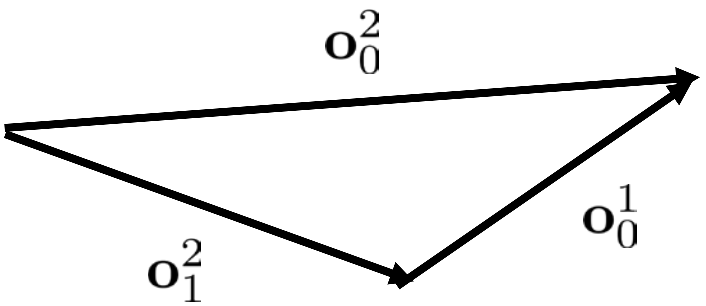
$$\mathbf{o}' = \begin{pmatrix} \mathbf{o}'_x \\ \mathbf{o}'_y \\ \mathbf{o}'_z \end{pmatrix}$$

Composition of Translations

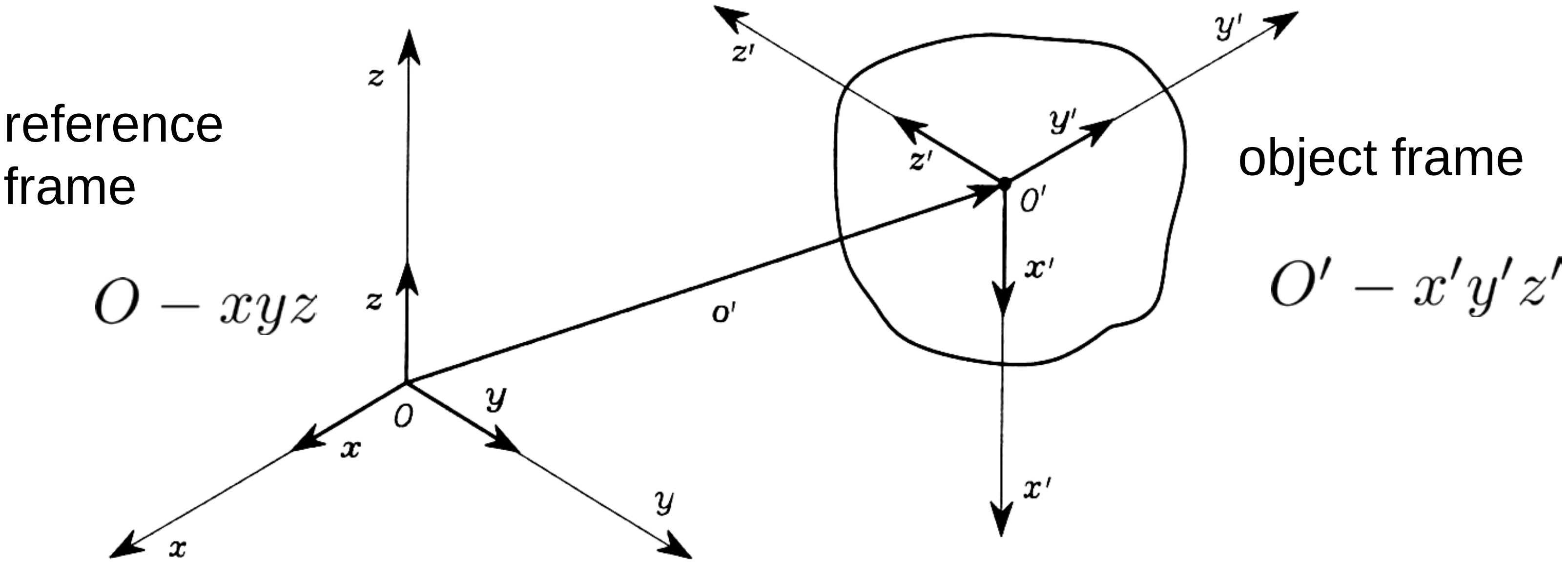
$$\mathbf{o}_0^2 = \mathbf{o}_0^1 + \mathbf{o}_1^2$$

$$\mathbf{o}_0^2 = \mathbf{o}_1^2 + \mathbf{o}_0^1$$





Orientation of a Rigid Body



Siciliano et al, Robotics: Modelling, Planing and Control

$$\mathbf{x}' = \mathbf{x}'_x \mathbf{x} + \mathbf{x}'_y \mathbf{y} + \mathbf{x}'_z \mathbf{z}$$
$$\mathbf{y}' = \mathbf{y}'_x \mathbf{x} + \mathbf{y}'_y \mathbf{y} + \mathbf{y}'_z \mathbf{z}$$
$$\mathbf{z}' = \mathbf{z}'_x \mathbf{x} + \mathbf{z}'_y \mathbf{y} + \mathbf{z}'_z \mathbf{z}$$

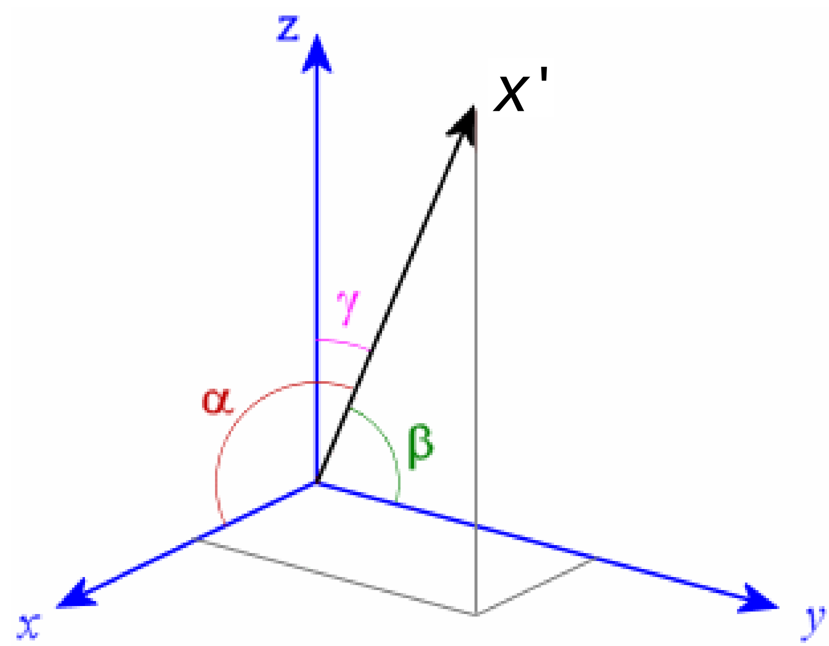
Rotation Matrix

$$\mathbf{R} = \begin{bmatrix}\mathbf{x}' & \mathbf{y}' & \mathbf{z}'\end{bmatrix} = \begin{bmatrix} \mathbf{x}'_x & \mathbf{y}'_x & \mathbf{z}'_x \\ \mathbf{x}'_y & \mathbf{y}'_y & \mathbf{z}'_y \\ \mathbf{x}'_z & \mathbf{y}'_z & \mathbf{z}'_z \end{bmatrix} = \begin{bmatrix} \mathbf{x}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{x} & \mathbf{z}'^T\mathbf{x} \\ \mathbf{x}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{y} & \mathbf{z}'^T\mathbf{y} \\ \mathbf{x}'^T\mathbf{z} & \mathbf{y}'^T\mathbf{z} & \mathbf{z}'^T\mathbf{z} \end{bmatrix}$$

Rotation matrix R captures the relative orientation among two frames

$x'_x, x'_y, \ldots, z'_z$ are the direction cosines of the axes of frame O´-x´y´z with respect to O-xyz

$$x'_x = \cos(\angle \mathbf{x}'\mathbf{x}) = \mathbf{x}' \circ \mathbf{x}$$
$$x'_y = \cos(\angle \mathbf{x}'\mathbf{y}) = \mathbf{x}' \circ \mathbf{y}$$
$$\ldots$$
$$z'_z = \cos(\angle \mathbf{z}'\mathbf{z}) = \mathbf{z}' \circ \mathbf{z}$$



http://intmstat.com/vectors/cosines.gif

Properties of a Rotation Matrix

$\mathbf{R}$ is an **orthogonal 3x3** matrix $\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$

$$\mathbf{x}'^T\mathbf{y}' = \mathbf{y}'^T\mathbf{z}' = \mathbf{z}'^T\mathbf{x}' = 0$$

$$\mathbf{x}'^T\mathbf{x}' = \mathbf{y}'^T\mathbf{y}' = \mathbf{z}'^T\mathbf{z}' = 1$$

Inverse of a rotation matrix : $\mathbf{R}^{-1} = \mathbf{R}^T$

A **square matrix** $\mathbf{R}$ is a rotation matrix if

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$$

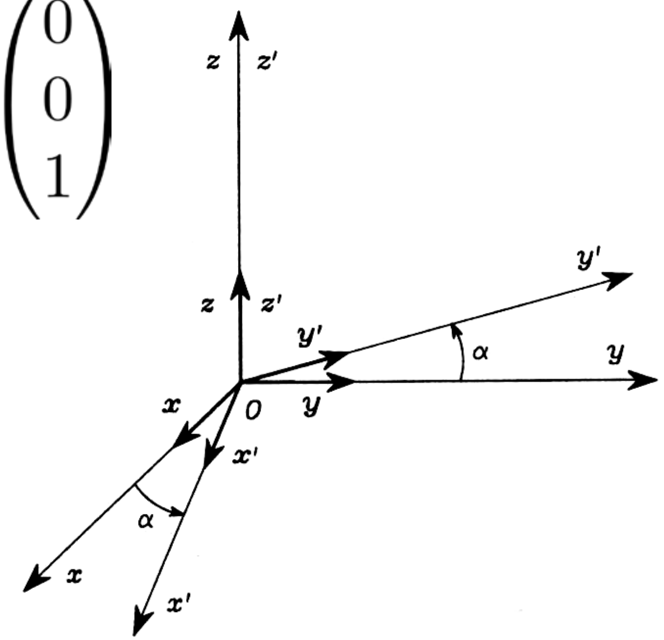
$$\det(\mathbf{R}) = 1$$

Elementary Rotations

- Rotation of frame O-xyz by angle α along z-axis

$$\mathbf{x}' = \begin{pmatrix} \cos\alpha \\ \sin\alpha \\ 0 \end{pmatrix} \quad \mathbf{y}' = \begin{pmatrix} -\sin\alpha \\ \cos\alpha \\ 0 \end{pmatrix} \quad \mathbf{z}' = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Siciliano et al, Robotics: Modelling, Planing and Control

Elementary Rotations

- Rotation by angle β along y-axis

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

- Rotation by angle γ along x-axis

$$\mathbf{R}_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

Rotation Matrices

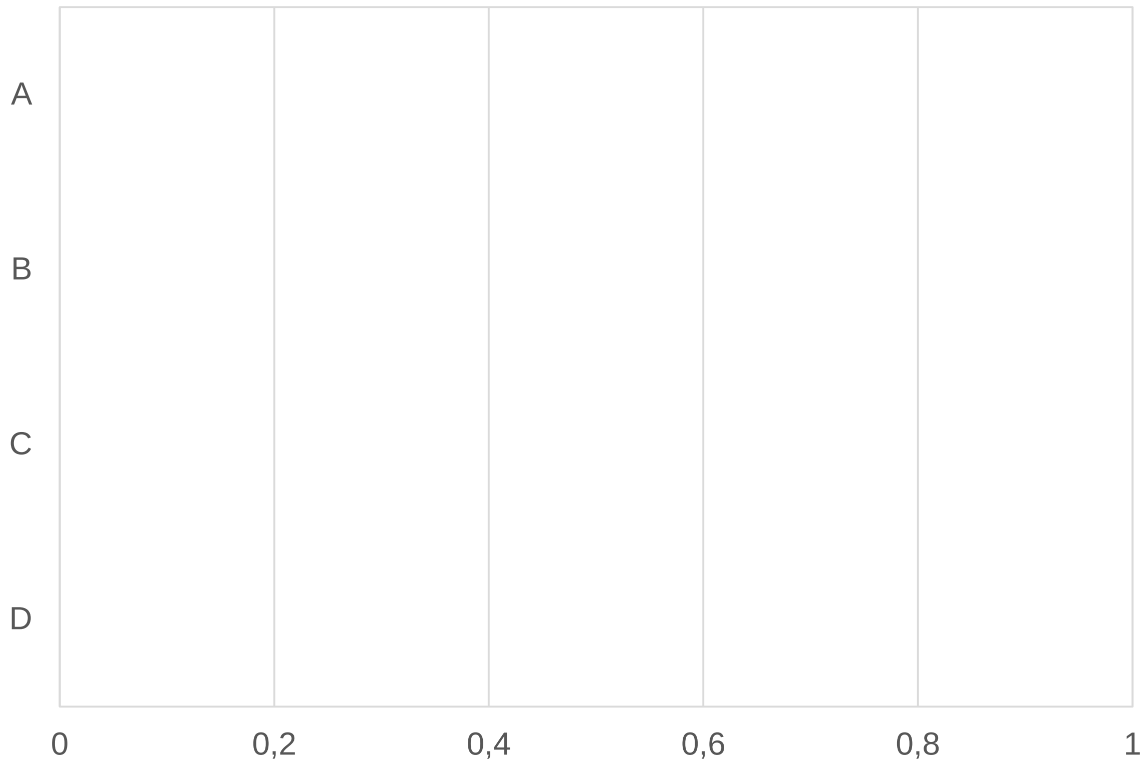
Which properties apply to rotation matrices?

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}$$

$$\mathbf{R}^T = -\mathbf{R}$$

$$\det(\mathbf{R}) = 1$$

$$||\mathbf{x}^T\mathbf{R}\mathbf{x}|| \geq 0$$



Umfrage starten

ID = frank.hoffmann@tu-dortmund.de
Umfrage noch nicht gestartet

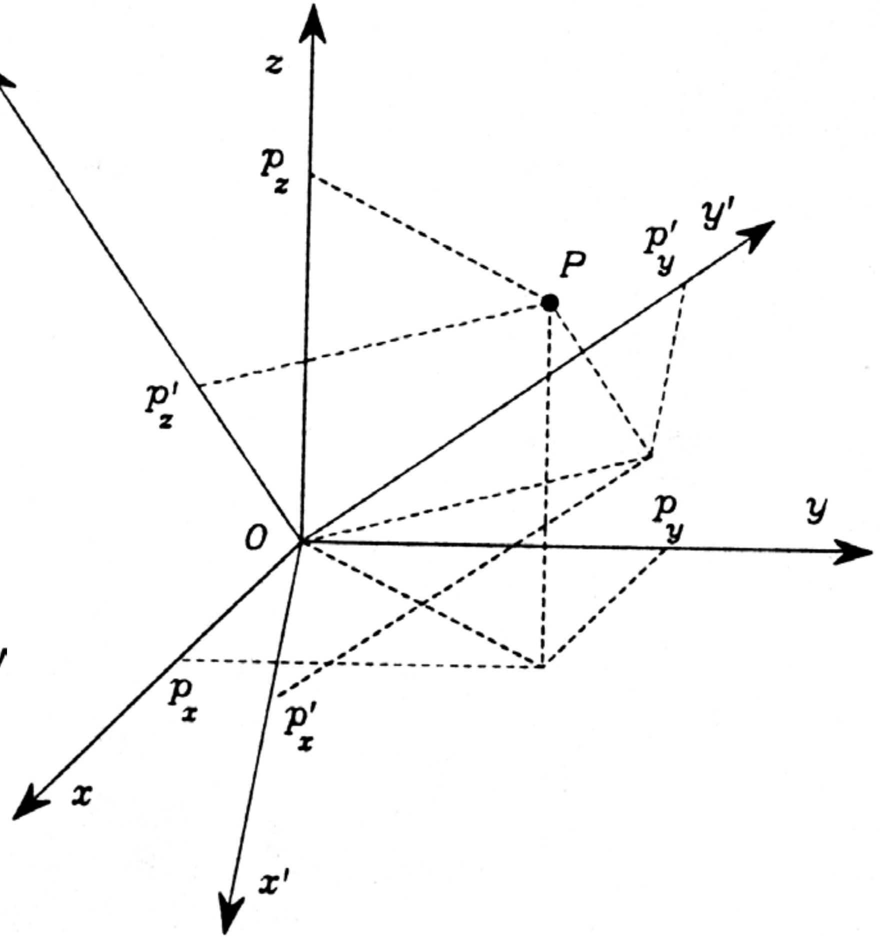
Representation of a Vector in Rotated Frames

A point P in space is represented w.r.t. *O-xyz* by coordinates

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = p_x\mathbf{x} + p_y\mathbf{y} + p_z\mathbf{z}$$

and w.r.t. frame *O-x'y'z'* by coordinates

$$\mathbf{p}' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = p'_x\mathbf{x}' + p'_y\mathbf{y}' + p'_z\mathbf{z}'$$



Siciliano et al, Robotics: Modelling, Planing and Control

Representation of a Vector in Rotated Frames

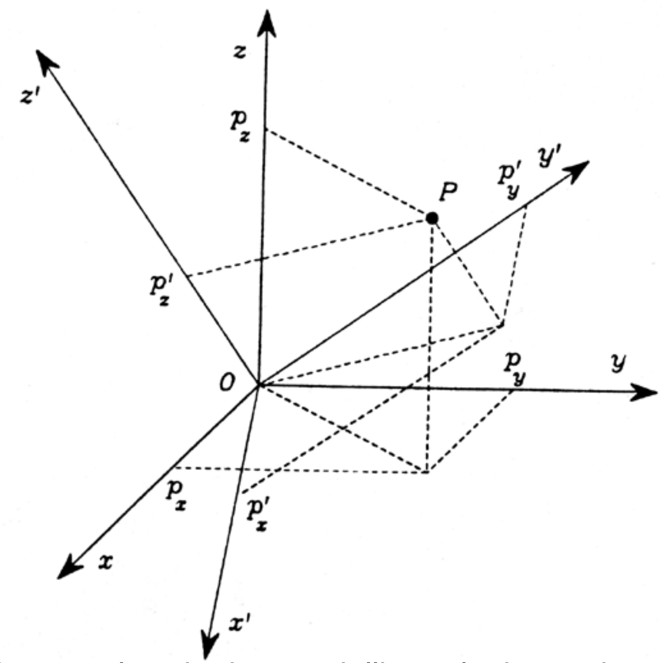
$$\mathbf{p} = p'_x \mathbf{x}' + p'_y \mathbf{y}' + p'_z \mathbf{z}' = \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \end{bmatrix} \mathbf{p}'$$

Rotation matrix **R** : **transformation matrix** to map coordinates of vector **p’** in frame *O-x’y’z’* into the coordinates of the same vector **p** in frame *O-xyz*.

$$\mathbf{p} = \mathbf{R}\mathbf{p}'$$

The inverse transformation is given by

$$\mathbf{p}' = \mathbf{R}^{\mathrm{T}}\mathbf{p}$$



Siciliano et al, Robotics: Modelling, Planing and Control

Representation of a Vector in Rotated Frames

- two frames with identical origin (pure rotation)
- rotation by α along Z-axis.
- $\mathbf{p}$: coordinates of P w.r.t. O-xyz
  $\mathbf{p}'$: coordinates of P w.r.t. O´-x´y´z´

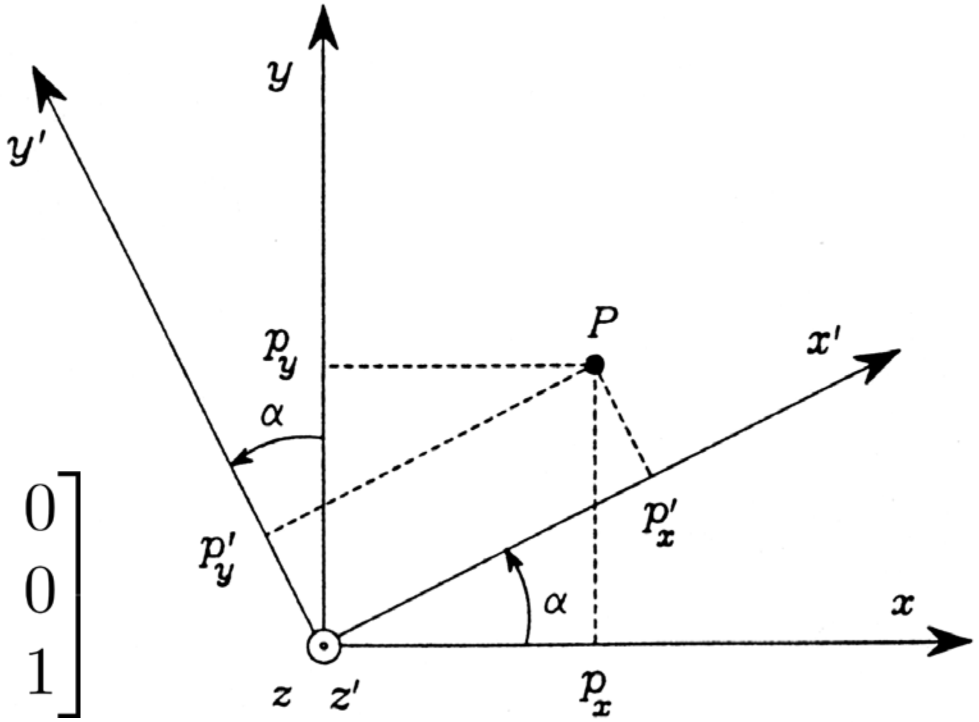
$$p_x = p'_x \cos\alpha - p'_y \sin\alpha$$

$$p_y = p'_x \sin\alpha + p'_y \cos\alpha$$

$$p_z = p'_z$$

$$\mathbf{p} = \mathbf{R}_z(\alpha)\mathbf{p}'$$

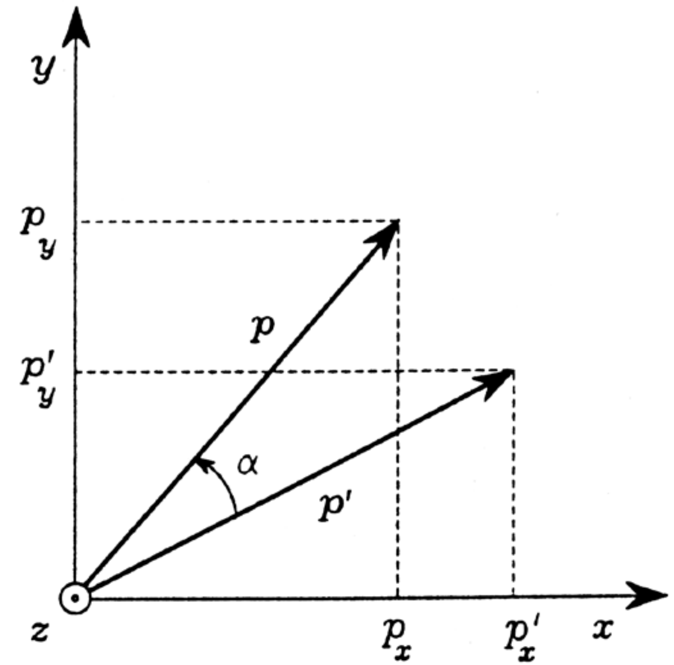
$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Rotation of a Vector

- **p´** vector in reference frame O-xyz.

- **Rp´** yields a vector **p**
  - same length as **p´**
  - rotated bei **R** w.r.t. **p´**



Siciliano et al, Robotics: Modelling, Planing and Contro

$$p_x = p'_x \cos\alpha - p'_y \sin\alpha$$
$$p_y = p'_x \sin\alpha + p'_y \cos\alpha$$
$$p_z = p'_z$$

$$\mathbf{p} = \mathbf{R}_z(\alpha)\mathbf{p}' \qquad \mathbf{R}_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Geometrical Interpretation of a Rotation Matrix

- mutual orientation between two frames

column vectors are the direction cosines of the rotated axis w.r.t. the original frame

$$\mathbf{R} = \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \end{bmatrix} = \begin{bmatrix} \mathbf{x}'_x & \mathbf{y}'_x & \mathbf{z}'_x \\ \mathbf{x}'_y & \mathbf{y}'_y & \mathbf{z}'_y \\ \mathbf{x}'_z & \mathbf{y}'_z & \mathbf{z}'_z \end{bmatrix} = \begin{bmatrix} \mathbf{x}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{x} & \mathbf{z}'^T\mathbf{x} \\ \mathbf{x}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{y} & \mathbf{z}'^T\mathbf{y} \\ \mathbf{x}'^T\mathbf{z} & \mathbf{y}'^T\mathbf{z} & \mathbf{z}'^T\mathbf{z} \end{bmatrix}$$

- coordinate transformation between the coordinates of a vector expressed in two different frames

$$\mathbf{p} = p'_x\mathbf{x}' + p'_y\mathbf{y}' + p'_z\mathbf{z}' = \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \end{bmatrix} \mathbf{p}'$$

- operator that generates a rotation of a vector in the same frame

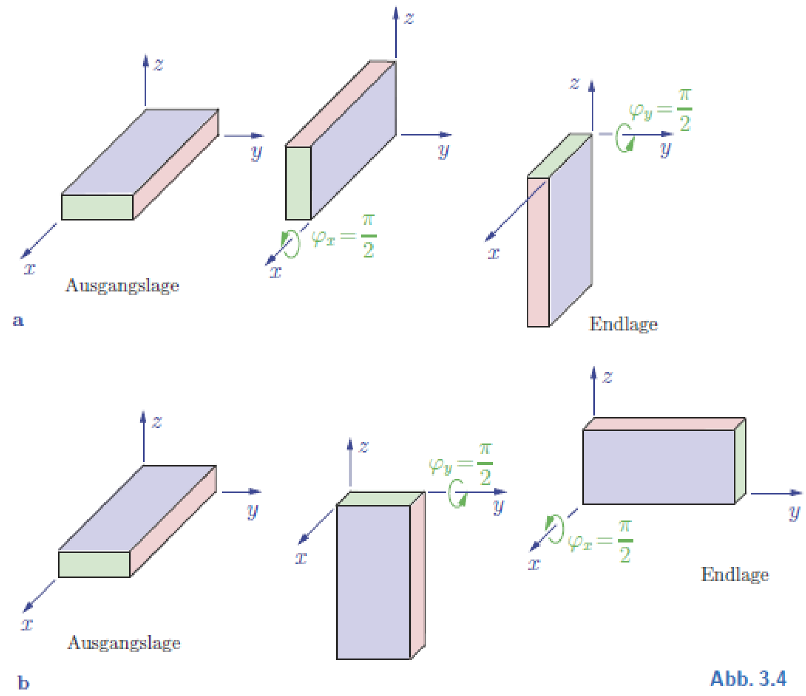
$$\mathbf{p} = \mathbf{R}_z(\alpha)\mathbf{p}'$$

Composition of Rotations

- Infinitesimal rotations and angular velocities are described by *vectors*
- Finite rotations are described by *matrices*
- Composition of rotations by matrix multiplication is <u>non-commutative</u>

first rotate along x-axis
then along y-axis

first rotate along y-axis
then along x-axis

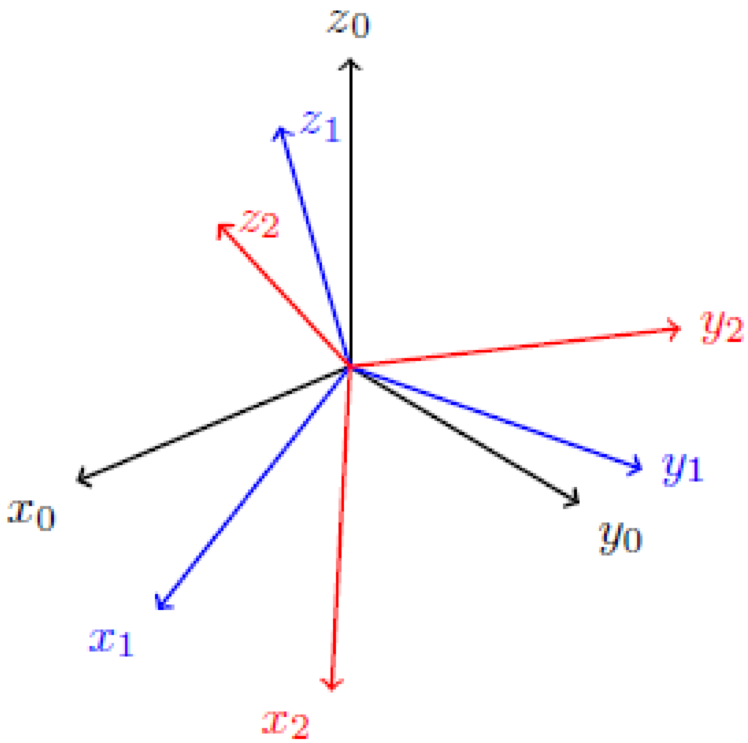


Gross, Hauger,Technische Mechanik

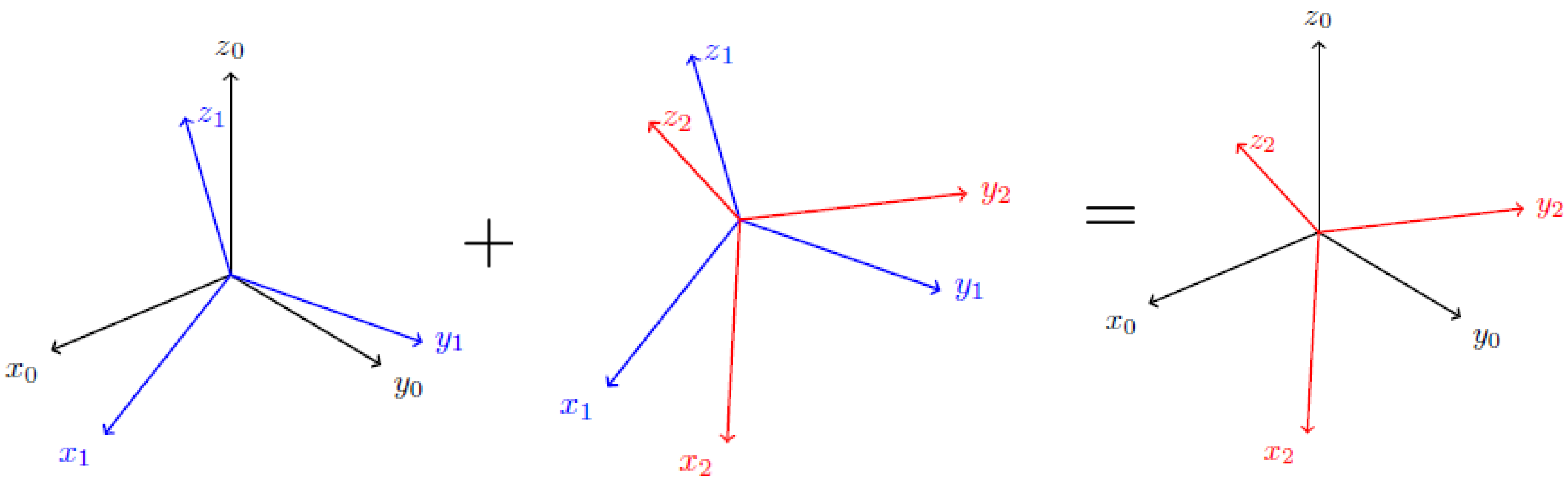
Composition of Rotation Matrices

Three frames with common origin

$$O - x_0 y_0 z_0 \qquad O - x_1 y_1 z_1 \qquad O - x_2 y_2 z_2$$



Composition of Rotation Matrices



$\mathbf{R}_1^0$ $\mathbf{R}_2^1$ $\mathbf{R}_2^0$

Composition of Rotation Matrices

vector $\mathbf{p}$ in the frames $O - x_i y_i z_i$

$$\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2$$

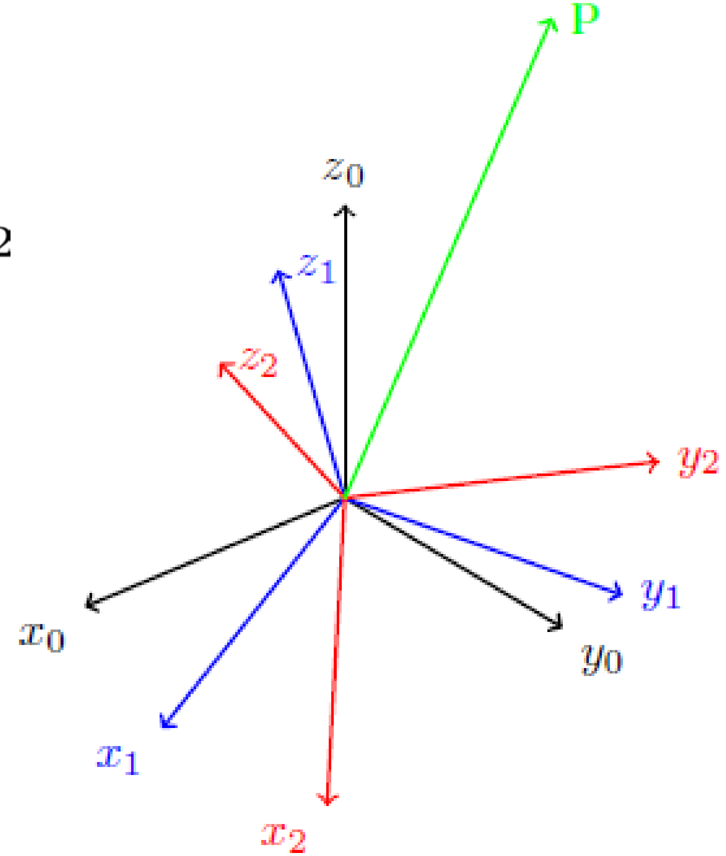
$$\mathbf{p}^1 = \mathbf{R}_2^1 \mathbf{p}^2$$

$$\mathbf{p}^0 = \mathbf{R}_1^0 \mathbf{p}^1$$

$$\mathbf{p}^0 = \mathbf{R}_2^0 \mathbf{p}^2$$

$$\mathbf{p}^0 = \mathbf{R}_1^0 \mathbf{R}_2^1 \mathbf{p}^2$$

$$\mathbf{R}_2^0 = \mathbf{R}_1^0 \mathbf{R}_2^1$$



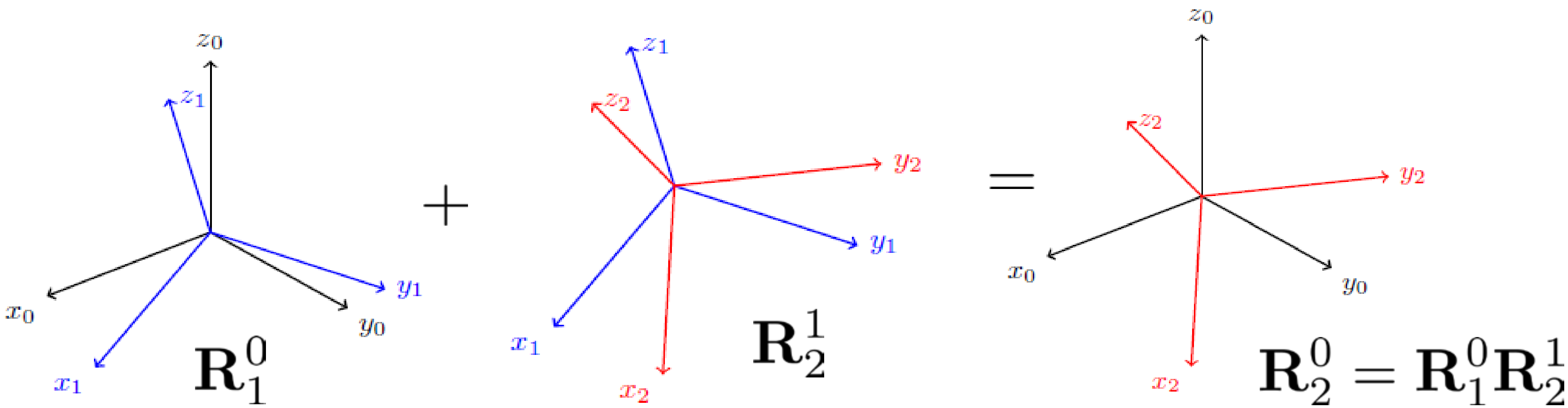
Composition of Rotations w.r.t. Current Frame

Composition of rotations by **post multiplication** of rotation matrices

$$\mathbf{R}_2^0 = \mathbf{R}_1^0 \mathbf{R}_2^1$$

rotation $\mathbf{R}_2^0$ obtained in two steps

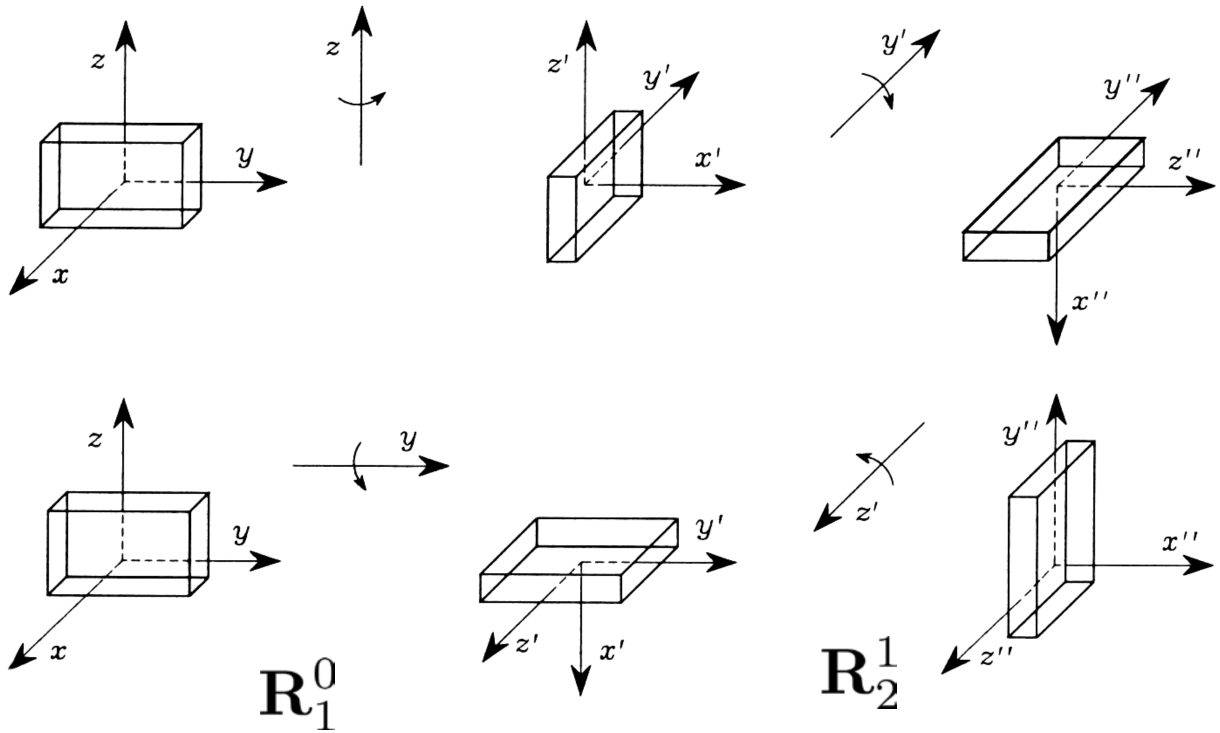
- first rotate frame $O - x_0 y_0 z_0$ by $\mathbf{R}_1^0$ to align it with $O - x_1 y_1 z_1$
- then rotate the frame by $\mathbf{R}_2^1$ to align it with $O - x_2 y_2 z_2$



rotation w.r.t. ***current frame*** $\longrightarrow$ postmultiplication of $\mathbf{R}_j^i$

Rotations w.r.t. to Axes of Current Frame

Siciliano et al, Robotics: Modelling, Planing and Control



$O - x_0 y_0 z_0$ $\quad$ $O - x_1 y_1 z_1$ $\quad$ $O - x_2 y_2 z_2$

$$\mathbf{R}_2^0 = \mathbf{R}_1^0 \mathbf{R}_2^1$$

Composition of Rotations w.r.t. Fixed Frame

$\mathbf{R}_1^0$ rotation matrix of frame $O - x_1y_1z_1$ w.r.t. fixed frame $O - x_0y_0z_0$
$\bar{\mathbf{R}}_2^0$ rotation matrix of frame $O - x_2y_2z_2$ w.r.t. fixed frame $O - x_0y_0z_0$

- Realign Frame 1 with Frame 0 by means of $\mathbf{R}_0^1$
- Make the rotation expressed by $\bar{\mathbf{R}}_2^1$ w.r.t. current frame
- Compensate for the initial realignment $\mathbf{R}_0^1$ by means of the inverse rotation $\mathbf{R}_1^0$
- Composition rule for rotations w.r.t. to current frame

$$\bar{\mathbf{R}}_2^0 = \mathbf{R}_1^0\mathbf{R}_0^1\bar{\mathbf{R}}_2^1\mathbf{R}_0^1$$

- Since $\mathbf{R}_1^0\mathbf{R}_0^1 = \mathbf{I}$

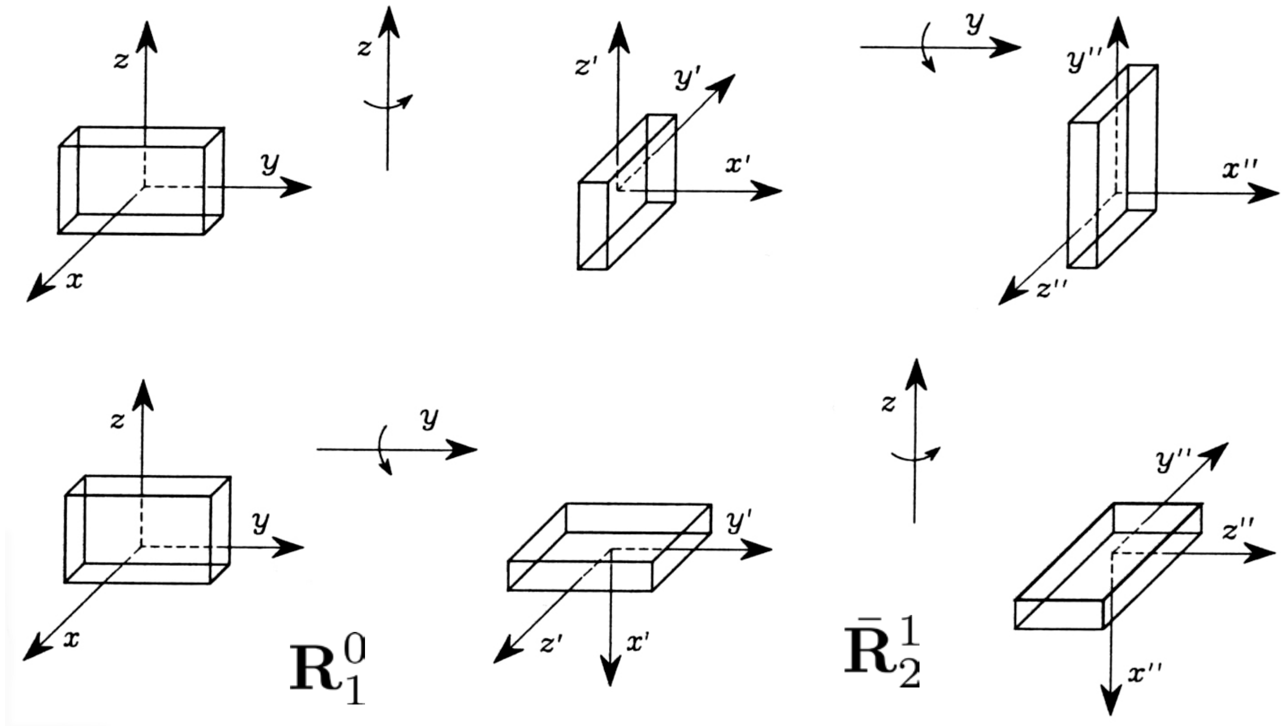
$$\bar{\mathbf{R}}_2^0 = \bar{\mathbf{R}}_2^1\mathbf{R}_0^1$$

- Premultiplication of rotation matrices

30

Rotations w.r.t to Axes of Fixed Frame

Siciliano et al, Robotics: Modelling, Planing and Control



$O - x_0y_0z_0$ $\qquad$ $O - x_1y_1z_1$ $\qquad$ $O - x_2y_2z_2$

$$\bar{\mathbf{R}}_2^0 = \bar{\mathbf{R}}_2^1\mathbf{R}_0^1$$

Rotation Matrices

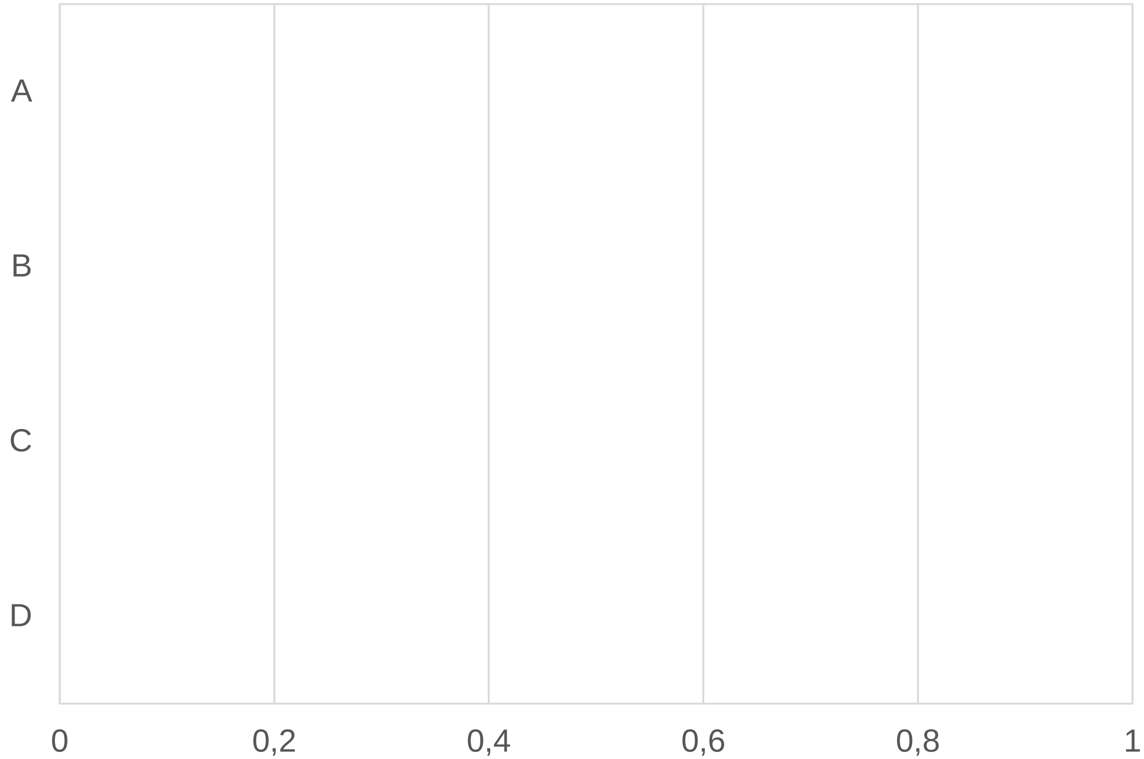
The composition of rotations R01 and R12 w.r.t. current frame is given by?

$$\mathbf{R}_2^0 = \mathbf{R}_2^1 \mathbf{R}_1^0$$

$$\mathbf{R}_2^0 = {\mathbf{R}_2^1}^T {\mathbf{R}_1^0}^T$$

$$\mathbf{R}_2^0 = \mathbf{R}_1^0 \mathbf{R}_2^1$$

$$\mathbf{R}_2^0 = \mathbf{R}_2^1 + \mathbf{R}_1^0$$



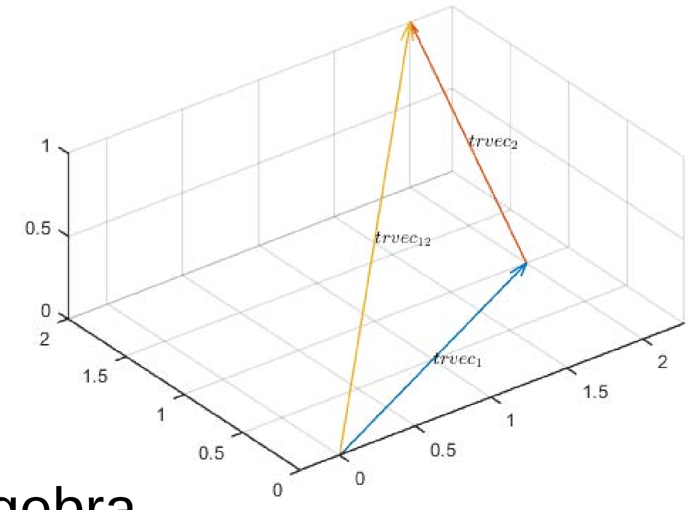
Umfrage starten

ID = frank.hoffmann@tu-dortmund.de
Umfrage noch nicht gestartet

Translations in Robotics System Toolbox

- Translations are ordinary 1-by-3 vectors

```
trvec = [3 0 2.5];
```



- Composition of translations by vector algebra

```
trvec1 = [2 1 0];
trvec2 = [0 1 1];
trvec12=trvec1+trvec2;
clf;
hold on;
quiver3(0,0,0,trvec1(1),trvec1(2),trvec1(3),0);
quiver3(trvec1(1),trvec1(2),trvec1(3), trvec2(1), trvec2(2), trvec2(3),0);
quiver3(0,0,0,trvec12(1),trvec12(2),trvec12(3),0);
```

Rotations in Robotics System Toolbox

- Define rotation matrix component wise

```
theta = pi/2;
% elemental rotation about the x-axis
rotmx = [1 0 0; 0 cos(theta) -sin(theta); 0 sin(theta) cos(theta)];
% elemental rotation about the y-axis
rotmy = [cos(theta) 0 sin(theta) 0; 0 1 0; -sin(theta) 0 cos(theta)];
% elemental rotation about the z-axis
rotmz = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];
```

- Define elemental rotations by angle axis notation

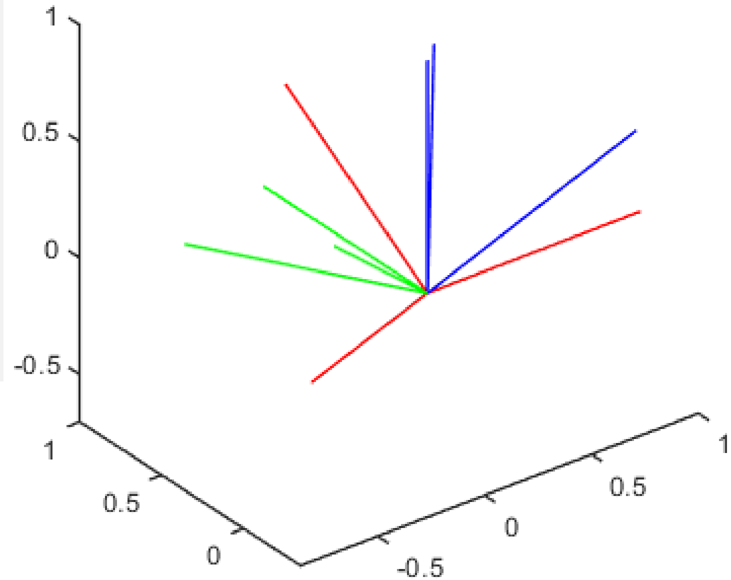
```
theta = pi/2;
% elemental rotation about the x-axis
rotmx = axang2rotm([1 0 0 theta]);
% elemental rotation about the y-axis
rotmy = axang2rotm([0 1 0 theta]);
% elemental rotation about the z-axis
rotmz = axang2rotm([0 0 1 theta]);
```

Sequence of Rotations in Robotics System Toolbox

- Composition w.r.t. current frame

$$\mathbf{R} = \mathbf{R}_z(\alpha)\,\mathbf{R}_y(\beta)\,\mathbf{R}_x(\gamma)$$

```
alpha = pi/2;
beta=pi/4;
gamma=pi/6;
rotmz = axang2rotm([0 0 1 alpha]);
rotmy = axang2rotm([0 1 0 beta]);
rotmx = axang2rotm([1 0 0 gamma]);
rotrpy= rotmz*rotmy*rotmx;
```



- Composition w.r.t. fixed frame

$$\mathbf{R} = \mathbf{R}_x(\gamma)\,\mathbf{R}_y(\beta)\,\mathbf{R}_z(\alpha)$$

```
roteul= rotmx*rotmy*rotmz;
```

```
plotTransforms(zeros(3,3),[1 0 0 0; rotm2quat(rotrpy); rotm2quat(roteul)]);
axis equal;
```

Apply Translation and Rotation to a 3DVector

- First apply rotation then translation $\mathbf{p}^0 = \mathbf{R}_1^0\mathbf{p}^1 + \mathbf{t}_1^0$

```
p1=[1 1 2];
trvec01=[1 0 -1];
rotmx01=axang2rotm([1 0 0 pi/2]);
p0=rotmx01*p1'+trvec01';
```

- First apply translation then rotation $\mathbf{p}^0 = \mathbf{R}_1^0(\mathbf{p}^1 + \mathbf{t}_1^0)$

```
p1=[1 1 2];
trvec01=[1 0 -1];
rotmx01=axang2rotm([1 0 0 pi/2]);
p0=rotmx01*(p1+trvec01)';
```

Matlab Grader Assignment

Your Script

```
%% 1.1 translations along x axis by a=1 and z-axis by d=2
a=1;
d=2;

```

Your Script

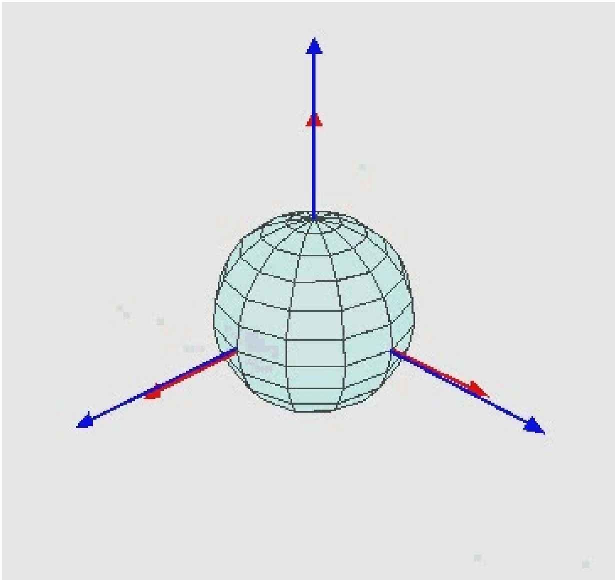
```
%% 1.2 rotations rotmx about x-axis by alpha = pi/2  and rotmz z-axis by gamma= pi/4
alpha=pi/2;
gamma=pi/4;
rotmx=
rotmz=

%% 1.3 rotations rotmc w.r.t. current and rotmf w.r.t. fixed frame
rotmc=
rotmf=

%% 1.4 apply rotations to 3D vector p
p=[1 2 0]';
pc=
pf=

```

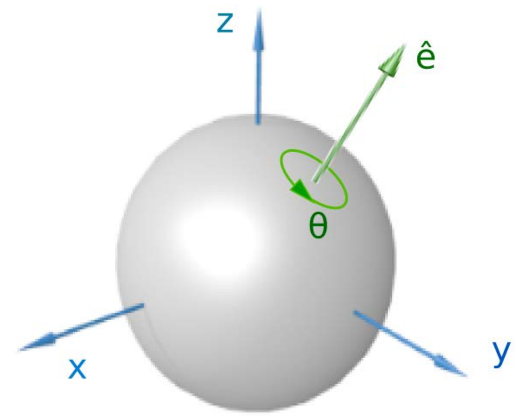
Minimal Representation

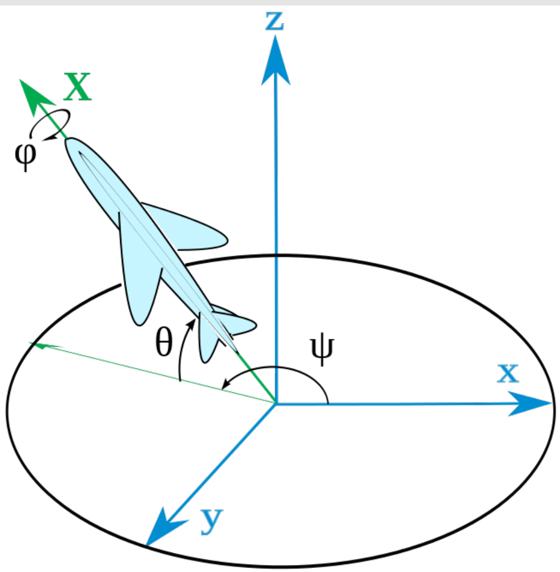
$$\mathbf{R} = \begin{bmatrix}\mathbf{x}' & \mathbf{y}' & \mathbf{z}'\end{bmatrix} = \begin{bmatrix} \mathbf{x}'_x & \mathbf{y}'_x & \mathbf{z}'_x \\ \mathbf{x}'_y & \mathbf{y}'_y & \mathbf{z}'_y \\ \mathbf{x}'_z & \mathbf{y}'_z & \mathbf{z}'_z \end{bmatrix} = \begin{bmatrix} \mathbf{x}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{x} & \mathbf{z}'^T\mathbf{x} \\ \mathbf{x}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{y} & \mathbf{z}'^T\mathbf{y} \\ \mathbf{x}'^T\mathbf{z} & \mathbf{y}'^T\mathbf{z} & \mathbf{z}'^T\mathbf{z} \end{bmatrix}$$

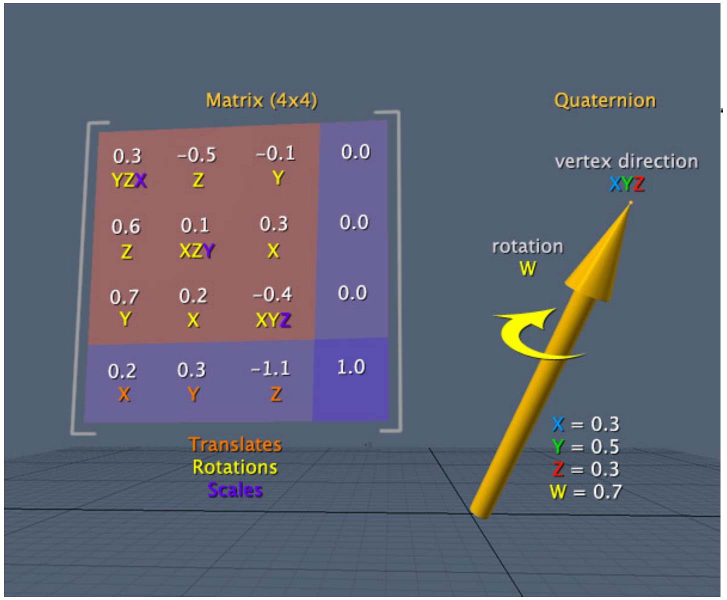
- Rotation matrix has ***redundant*** parameters
- $\mathbf{R}$ is a 3x3 matrix with *nine* components $\mathbf{R}_{ij}$
- $\mathbf{R}$ is constrained by *six* orthogonality constraints $\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$
- *Three independent* parameter are sufficient to determine orientation of a rigid body
- Euler angles provide such a *minimal representation*.

Rotation Representations

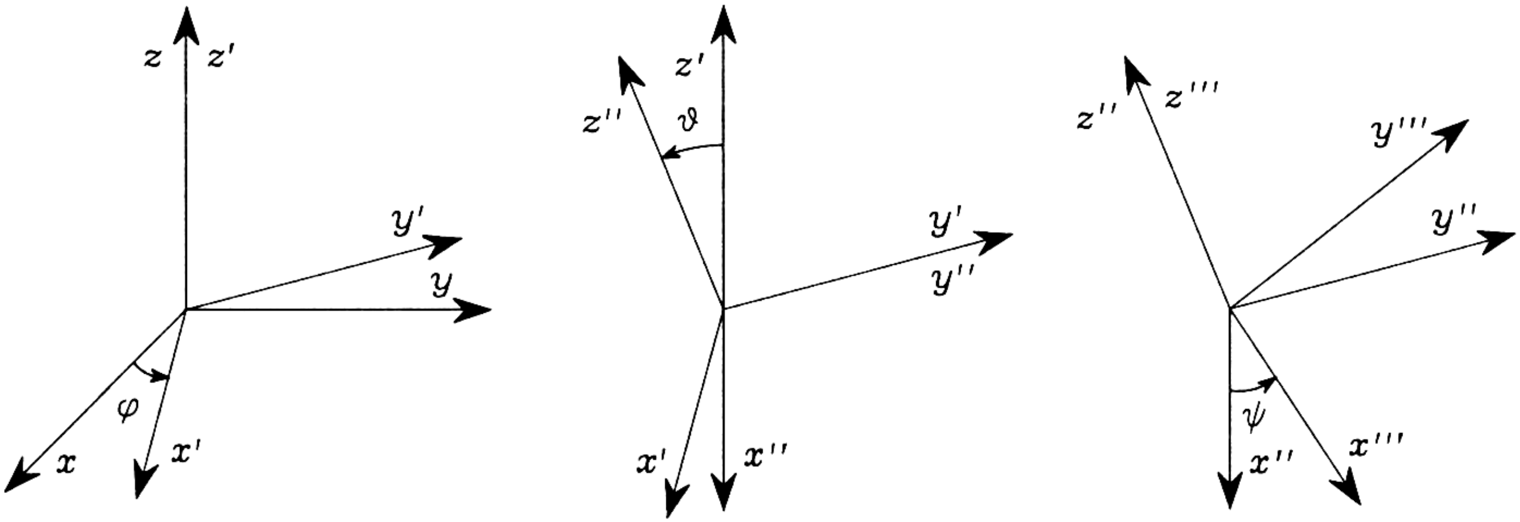








Euler Angles

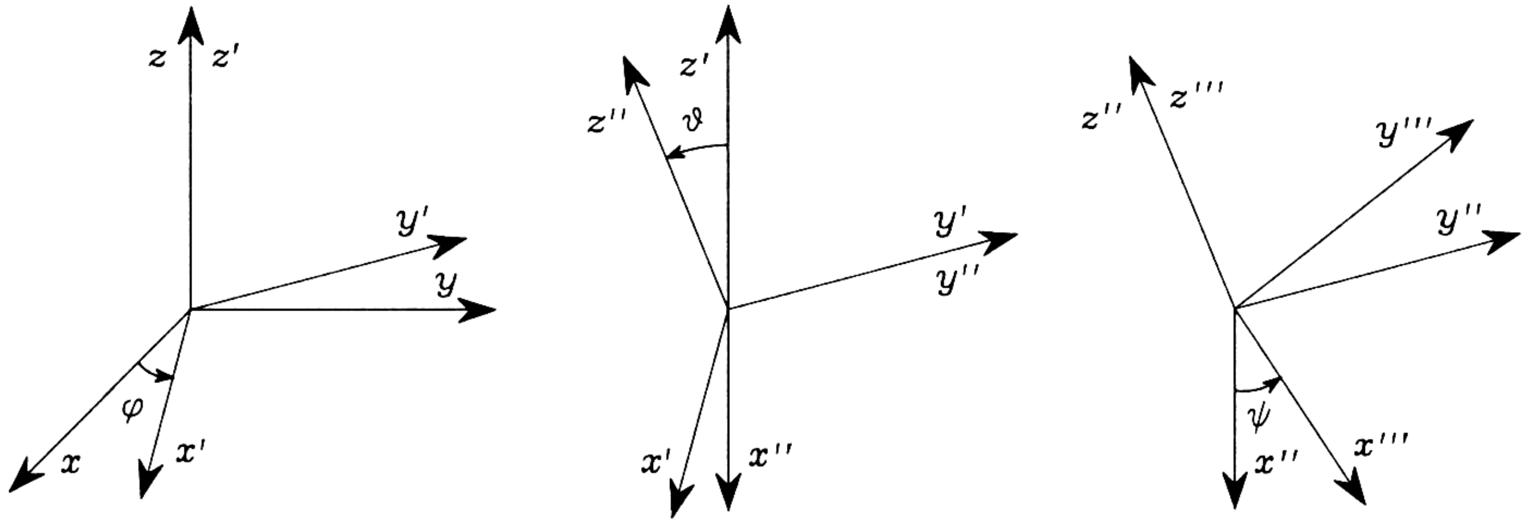


Siciliano et al, Robotics: Modelling, Planing and Control

Euler angles (*z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y*)

Tait–Bryan angles (*x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z*).

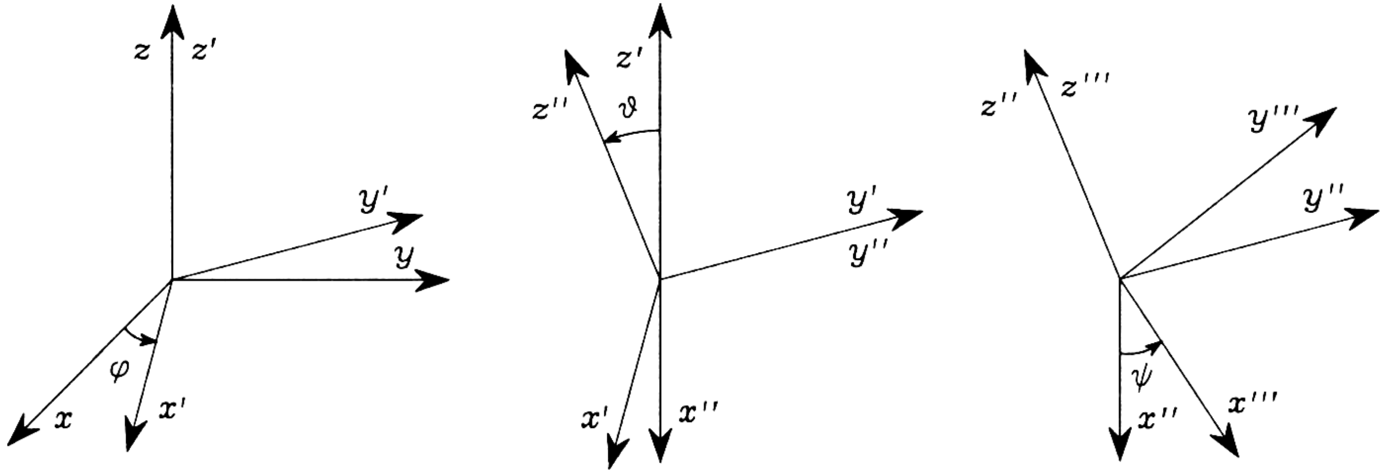
Euler Angles ZYZ



Siciliano et al, Robotics: Modelling, Planing and Control

- Rotate the reference frame by the angle φ about z-axis
- Rotate the current frame by the angle ϑ about y'-axis
- Rotate the current frame by the angle ψ about z''-axis

Euler Angles ZYZ



Siciliano et al, Robotics: Modelling, Planing and Control

$$\mathbf{R}_z(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_{y'}(\vartheta) = \begin{bmatrix} \cos\vartheta & 0 & -\sin\vartheta \\ 0 & 1 & 0 \\ \sin\vartheta & 0 & \cos\vartheta \end{bmatrix}$$

$$\mathbf{R}_{z''}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Euler Angles ZYZ

- composition of rotations w.r.t. current frame

$$\mathbf{R}(\phi) = \mathbf{R}_z(\varphi)\mathbf{R}_{y'}(\vartheta)\mathbf{R}_{z''}(\psi)$$

$$= \begin{bmatrix} c_\varphi c_\vartheta c_\psi - s_\varphi s_\psi & -c_\varphi c_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta \\ s_\varphi c_\vartheta c_\psi + c_\varphi s_\psi & -s_\varphi c_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta \\ -s_\vartheta c_\psi & s_\vartheta s_\psi & c_\vartheta \end{bmatrix}$$

Inverse Solution to Euler Angles

- Euler angles φ, γ, ψ that correspond to a rotation matrix **R**

$$\mathbf{R}(\phi) = \begin{bmatrix} c_\varphi c_\vartheta c_\psi - s_\varphi s_\psi & -c_\varphi c_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta \\ s_\varphi c_\vartheta c_\psi + c_\varphi s_\psi & -s_\varphi c_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta \\ -s_\vartheta c_\psi & s_\vartheta s_\psi & c_\vartheta \end{bmatrix}$$

$$\varphi = \mathrm{atan2}(r_{23}, r_{13})$$

$$\vartheta = \mathrm{atan2}\left(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}\right)$$

$$\psi = \mathrm{atan2}(r_{32}, -r_{31})$$

Inverse Solution to Euler Angles

$$\mathbf{R}(\phi) = \begin{bmatrix} c_\varphi c_\vartheta c_\psi - s_\varphi s_\psi & -c_\varphi c_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta \\ s_\varphi c_\vartheta c_\psi + c_\varphi s_\psi & -s_\varphi c_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta \\ -s_\vartheta c_\psi & s_\vartheta s_\psi & c_\vartheta \end{bmatrix}$$

$$\varphi = \operatorname{atan2}(r_{23}, r_{13})$$

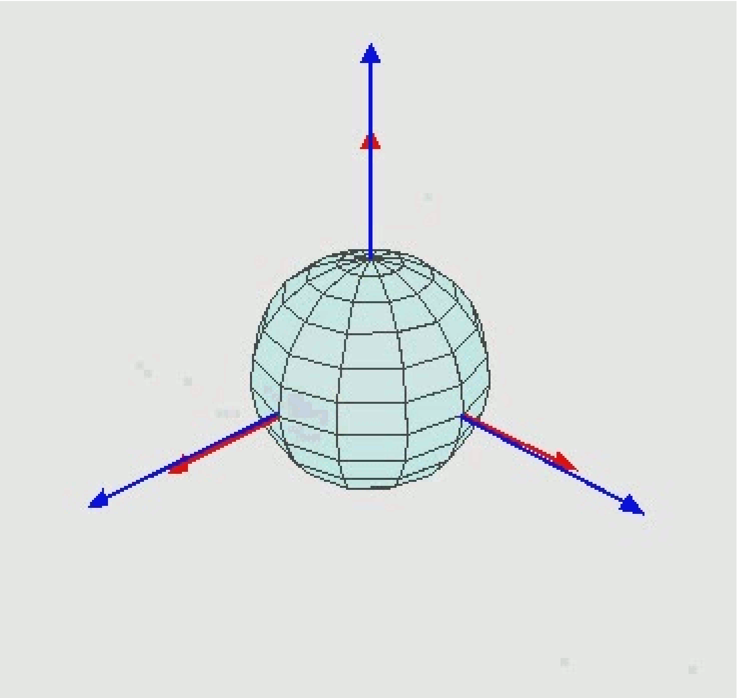
$$\vartheta = \operatorname{atan2}\left(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}\right)$$

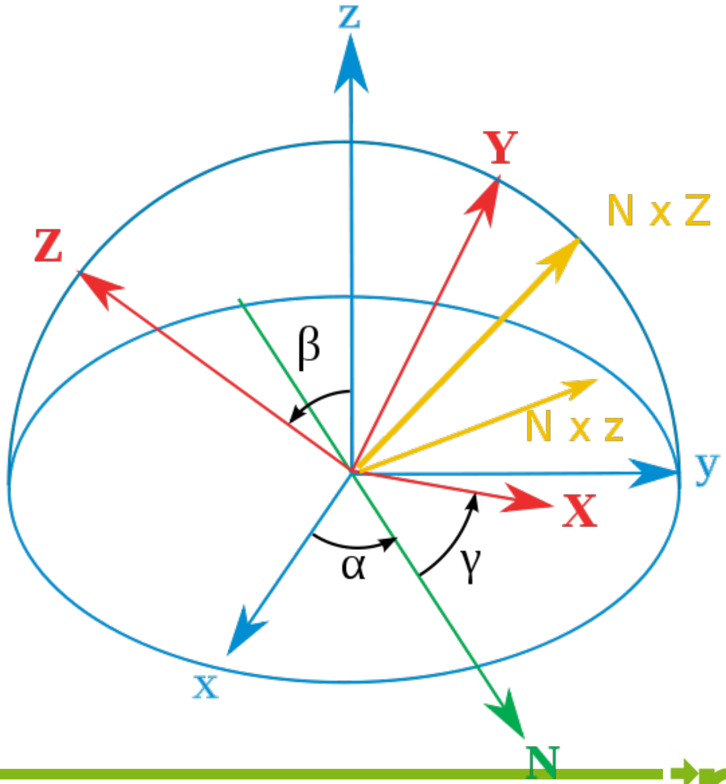
$$\psi = \operatorname{atan2}(r_{32}, -r_{31})$$

Singularity for $\sin\vartheta = 0$

Euler Angles ZXZ

- start with xyz-frame and XYZ-frame aligned
- rotate *XYZ*-frame along current *Z*-axis by α.
- rotate *XYZ*-frame w.r.t. to current *X*-axis by β.
- rotate *XYZ*-frame w.r.t. to current Z-axis by γ.

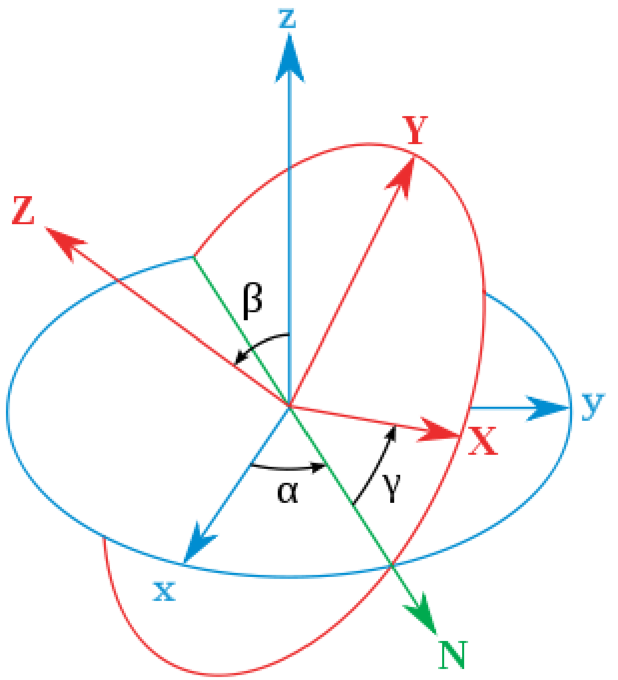




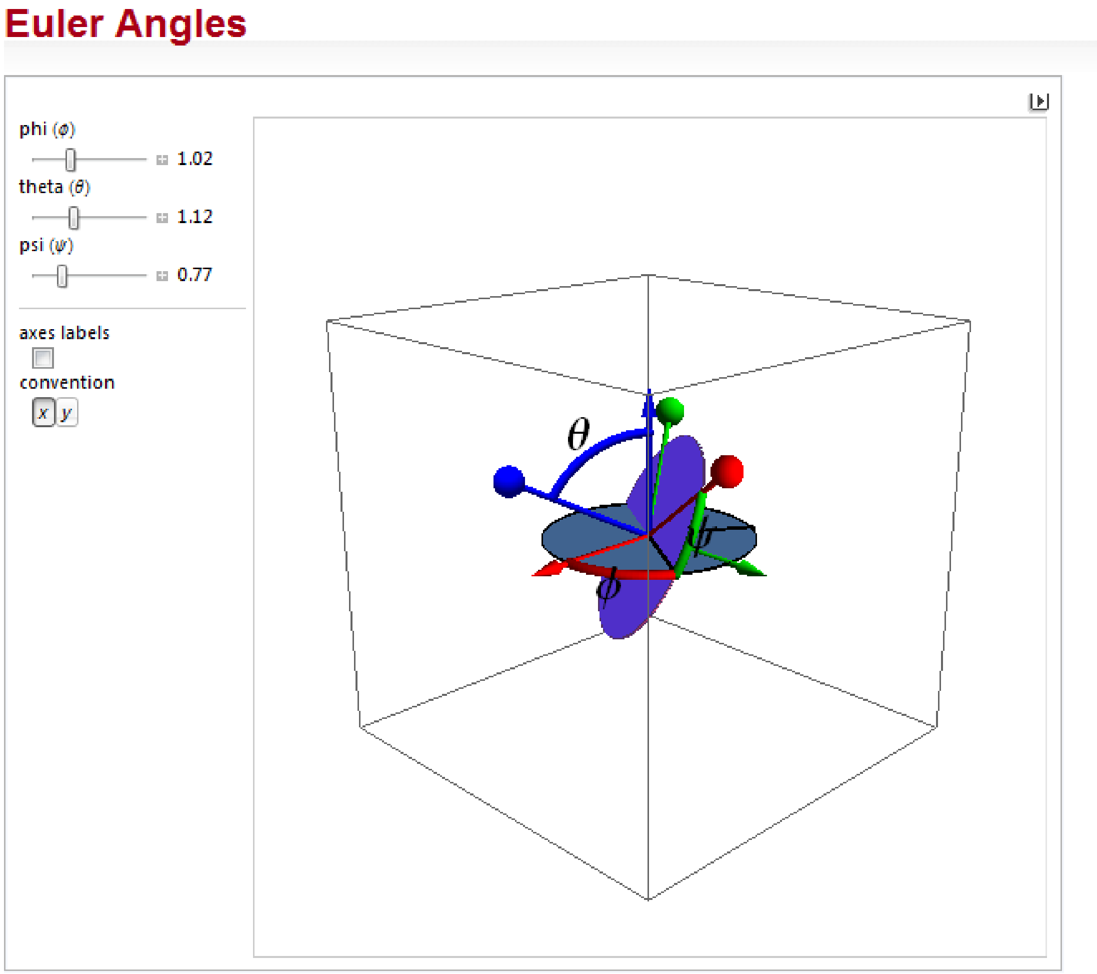
Euler Angles ZXZ

- intersection line N between xy and XY plane
- fixed frame xyz
- rotated frame XYZ

- α (φ) is the angle between *x*-axis and N
- β (θ) is the angle between *z*-axis and *Z*-axis
- γ (ψ) is the angle between N and *X*-axis.



Euler Angles Wolfram Demonstration



http://demonstrations.wolfram.com/EulerAngles/

Gimbal Lock Euler Angles

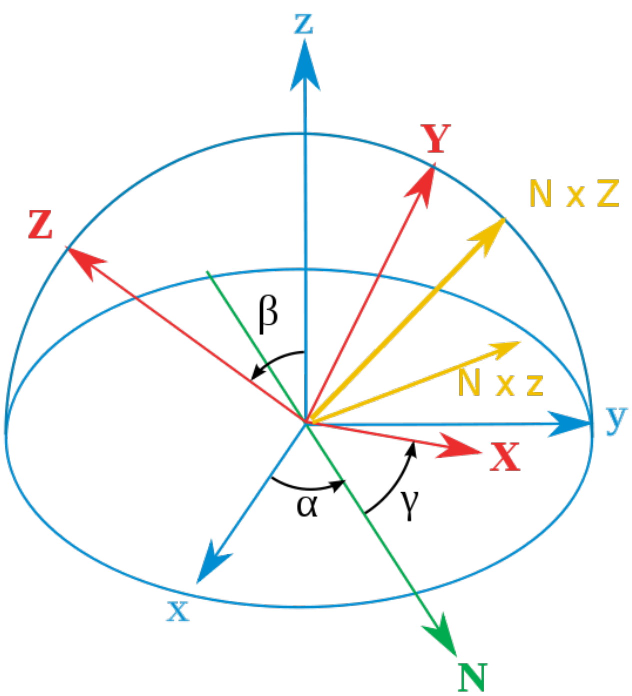
- singularity (gimbal lock)
- for $\vartheta,\beta=0$ the z- and Z-axis become parallel
- ambigious solutions for φ and ψ
- inverse solution degenerates for $\vartheta=0$ since $s_\vartheta=0$, $r_{13},r_{23},r_{32},r_{31}=0$

$$\boldsymbol{R}=\begin{bmatrix} c_\varphi c_\vartheta c_\psi - s_\varphi s_\psi & -c_\varphi c_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta \\ s_\varphi c_\vartheta c_\psi + c_\varphi s_\psi & -s_\varphi c_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta \\ -s_\vartheta c_\psi & s_\vartheta s_\psi & c_\vartheta \end{bmatrix}$$

$$\varphi = \mathrm{atan2}(r_{23}, r_{13})$$

$$\vartheta = \mathrm{atan2}\left(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}\right)$$

$$\psi = \mathrm{atan2}(r_{32}, -r_{31})$$



Euler Angles

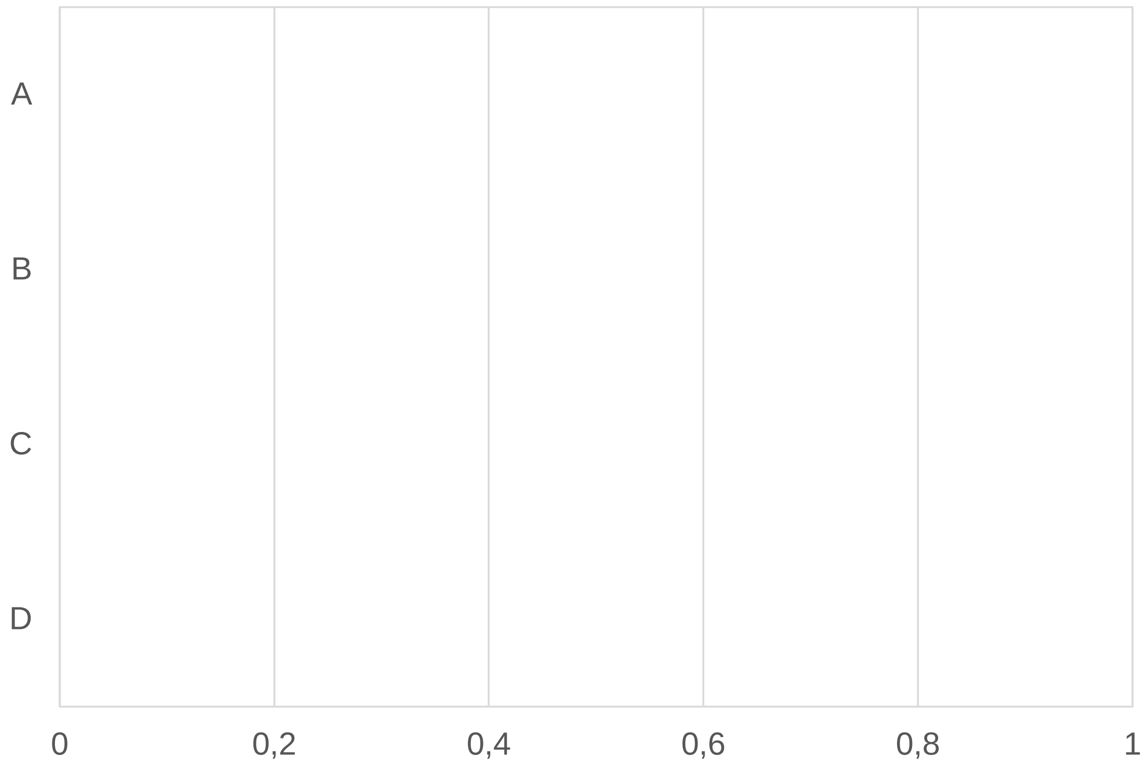
What is the most common order of axis in Euler angles?

XYZ

YZY

ZYX

ZYZ



Umfrage starten

ID = frank.hoffmann@tu-dortmund.de
Umfrage noch nicht gestartet

Representation of Rotations

Z-Y-X Euler angles (α, β, γ):

$$
{}^j\boldsymbol{R}_i = \begin{pmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{pmatrix}
$$

X-Y-Z fixed angles (ψ, θ, ϕ):

$$
{}^j\boldsymbol{R}_i = \begin{pmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{pmatrix}
$$

Angle-axis $\theta\hat{\boldsymbol{w}}$:

$$
{}^j\boldsymbol{R}_i = \begin{pmatrix} w_x^2 v_\theta + c_\theta & w_x w_y v_\theta - w_z s_\theta & w_x w_z v_\theta + w_y s_\theta \\ w_x w_y v_\theta + w_z s_\theta & w_y^2 v_\theta + c_\theta & w_y w_z v_\theta - w_x s_\theta \\ w_x w_z v_\theta - w_y s_\theta & w_y w_z v_\theta + w_x s_\theta & w_z^2 v_\theta + c_\theta \end{pmatrix}
$$

Unit quaternions $(\epsilon_0\ \epsilon_1\ \epsilon_2\ \epsilon_3)^{\mathrm{T}}$:

$$
{}^j\boldsymbol{R}_i = \begin{pmatrix} 1-2(\epsilon_2^2+\epsilon_3^2) & 2(\epsilon_1\epsilon_2 - \epsilon_0\epsilon_3) & 2(\epsilon_1\epsilon_3 + \epsilon_0\epsilon_2) \\ 2(\epsilon_1\epsilon_2 + \epsilon_0\epsilon_3) & 1-2(\epsilon_1^2+\epsilon_3^2) & 2(\epsilon_2\epsilon_3 - \epsilon_0\epsilon_1) \\ 2(\epsilon_1\epsilon_3 - \epsilon_0\epsilon_2) & 2(\epsilon_2\epsilon_3 + \epsilon_0\epsilon_1) & 1-2(\epsilon_1^2+\epsilon_2^2) \end{pmatrix}
$$

Rotation matrix:

$$
{}^j\boldsymbol{R}_i = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}
$$

Z-Y-X Euler angles (α, β, γ):

$$\beta = \mathrm{Atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)$$

$$\alpha = \mathrm{Atan2}\left(\frac{r_{21}}{\cos\beta}, \frac{r_{11}}{\cos\beta}\right)$$

$$\gamma = \mathrm{Atan2}\left(\frac{r_{32}}{\cos\beta}, \frac{r_{33}}{\cos\beta}\right)$$

X-Y-Z fixed angles (ψ, θ, ϕ):

$$\theta = \mathrm{Atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)$$

$$\psi = \mathrm{Atan2}\left(\frac{r_{21}}{\cos\theta}, \frac{r_{11}}{\cos\theta}\right)$$

$$\phi = \mathrm{Atan2}\left(\frac{r_{32}}{\cos\theta}, \frac{r_{33}}{\cos\theta}\right)$$

Angle axis $\theta\hat{\boldsymbol{w}}$:

$$\theta = \cos^{-1}\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right)$$

$$\hat{\boldsymbol{w}} = \frac{1}{2\sin\theta}\begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$$

Unit quaternions $(\epsilon_0\ \epsilon_1\ \epsilon_2\ \epsilon_3)^{\mathrm{T}}$:

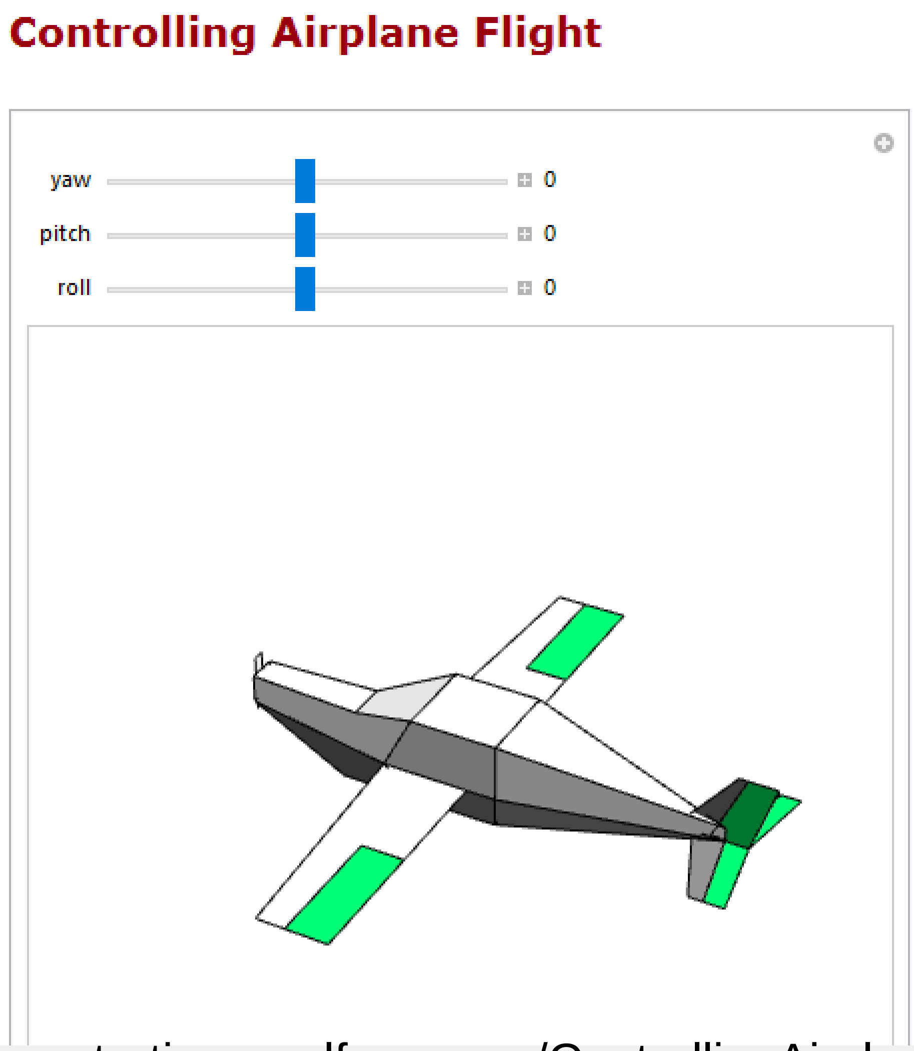
$$\epsilon_0 = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}}$$

$$\epsilon_1 = \frac{r_{32} - r_{23}}{4\epsilon_0}$$

$$\epsilon_2 = \frac{r_{13} - r_{31}}{4\epsilon_0}$$

$$\epsilon_3 = \frac{r_{21} - r_{12}}{4\epsilon_0}$$

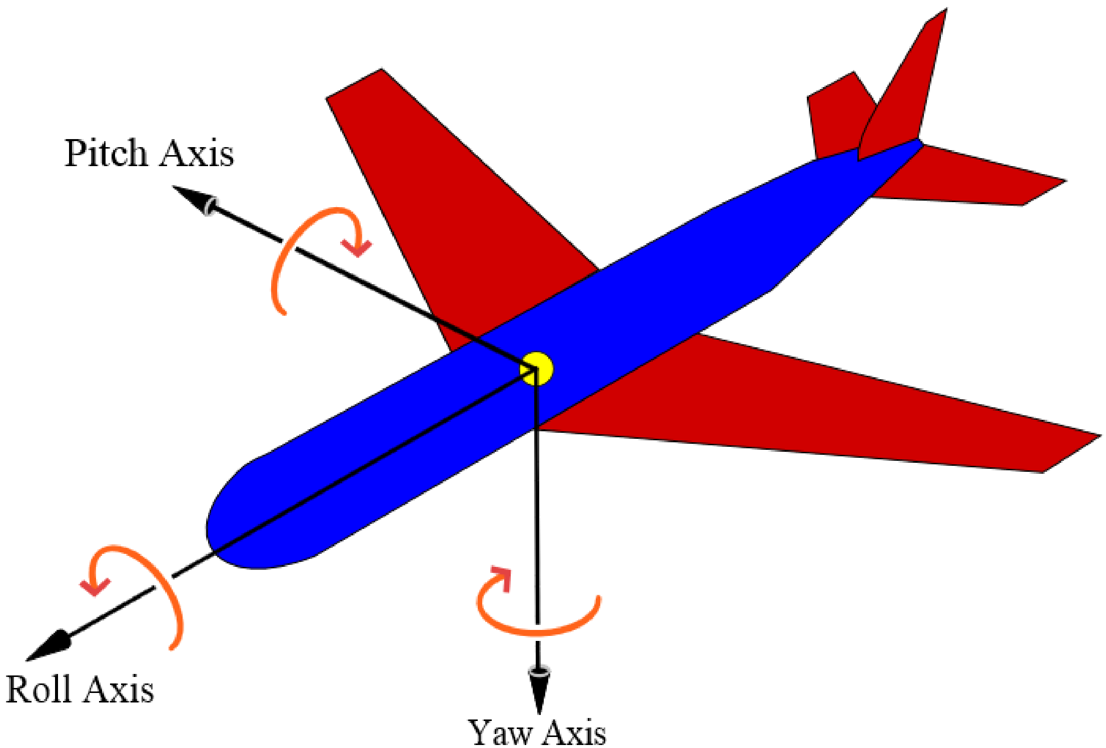
Roll Pitch Yaw



http://demonstrations.wolfram.com/ControllingAirplaneFlight/

Roll Pitch Yaw (Tait Bryan) Angles

- Aeronautic and automotive applications
- Example: attitiude of an aircraft

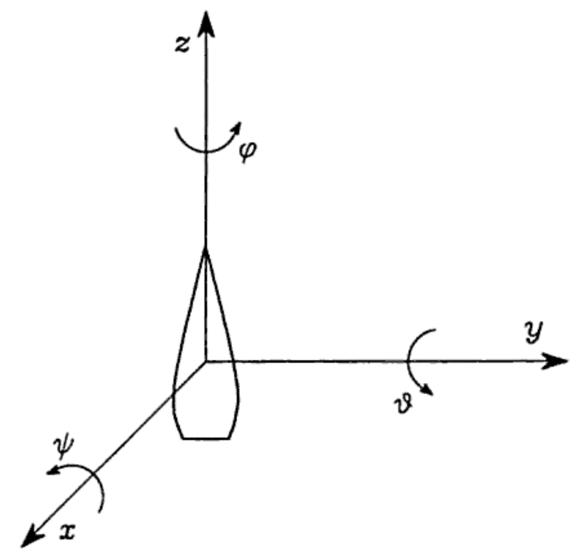


Roll Pitch Yaw Angles

- Rotate the reference frame by the angle ψ about x-axis (roll)
- Rotate the reference frame by the angle ϑ about y-axis (pitch)
- Rotate the reference frame by the angle φ about z'-axis (yaw)
- Rotations typically defined w.r.t fixed frame (extrinsic)

$$\phi = (\psi, \vartheta, \varphi) \qquad R(\phi) = R_z(\varphi) R_y(\vartheta) R_x(\psi)$$

$$\boldsymbol{R}(\phi) = \begin{bmatrix} c_\varphi c_\vartheta & -c_\varphi s_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta c_\psi + s_\varphi s_\psi \\ s_\varphi c_\vartheta & -s_\varphi s_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta c_\psi - c_\varphi s_\psi \\ -s_\vartheta & c_\vartheta s_\psi & c_\vartheta c_\psi \end{bmatrix}$$



Siciliano et al, Robotics: Modelling, Planing and Control

Roll Pitch Yaw Angles Inverse Mapping

$$\boldsymbol{R}(\phi) = \begin{bmatrix} c_\varphi c_\vartheta & -c_\varphi s_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta c_\psi + s_\varphi s_\psi \\ s_\varphi c_\vartheta & -s_\varphi s_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta c_\psi - c_\varphi s_\psi \\ -s_\vartheta & c_\vartheta s_\psi & c_\vartheta c_\psi \end{bmatrix}$$

$$\phi = (\psi, \vartheta, \varphi) \qquad R(\phi) = R_z(\varphi) R_y(\vartheta) R_x(\psi)$$

$$\varphi = \text{atan2}(r_{21}, r_{11})$$

$$\vartheta = \text{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)$$

$$\psi = \text{atan2}(r_{32}, r_{33})$$

RPY Angles

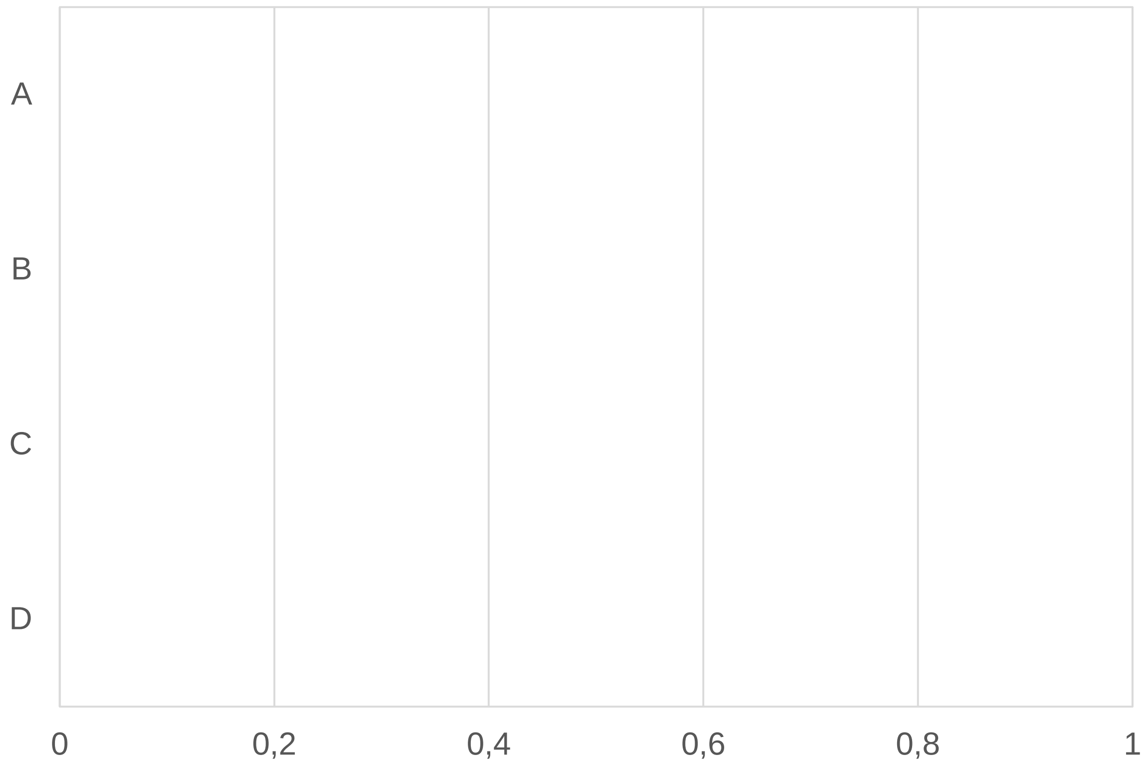
What is the most common order of axis in RPY angles?

XYZ

YZY

ZYX

ZYZ



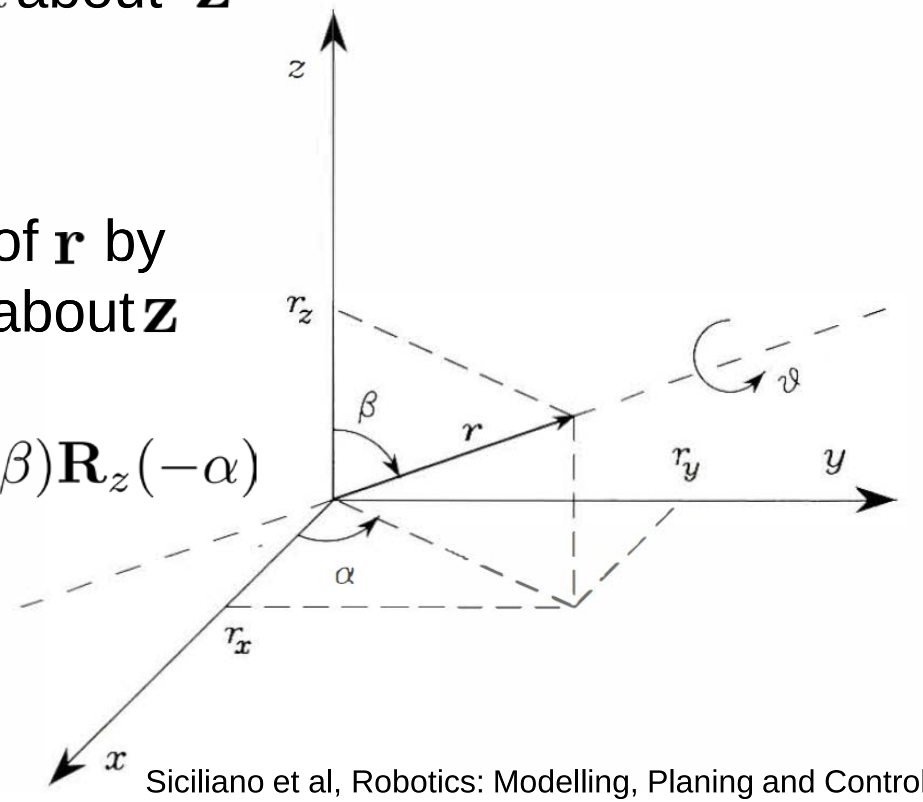
Umfrage starten

ID = frank.hoffmann@tu-dortmund.de
Umfrage noch nicht gestartet

Angle - Axis

- Unit vector of rotation axis $\mathbf{r} = [r_x \ r_y \ r_z]$ $\qquad r_x^2 + r_y^2 + r_z^2 = 1$
- rotation matrix $\mathbf{R}(\vartheta, \mathbf{r})$
- align $\mathbf{r}$ with $\mathbf{z}$ by rotating by $-\alpha$ about $\mathbf{z}$ and by $-\beta$ about $\mathbf{y}$.
- rotate by ϑ about $\mathbf{z}$
- realign with the initial direction of $\mathbf{r}$ by rotating by β about $\mathbf{y}$ and by α about $\mathbf{z}$

$$\mathbf{R}(\vartheta, \mathbf{r}) = \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma)\mathbf{R}_y(-\beta)\mathbf{R}_z(-\alpha)$$



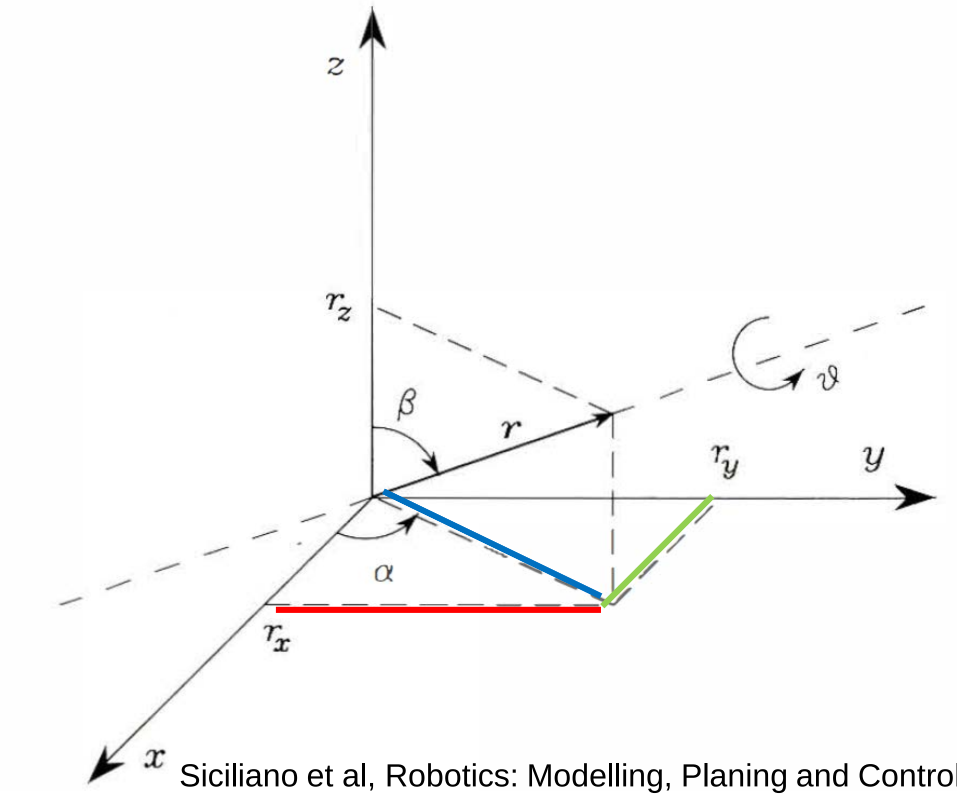
Siciliano et al, Robotics: Modelling, Planing and Control

Angle - Axis

$$\mathbf{R}(\vartheta, \mathbf{r}) = \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma)\mathbf{R}_y(-\beta)\mathbf{R}_z(-\alpha)$$

$$\sin\alpha = \frac{\mathbf{r}_x}{\sqrt{\mathbf{r}_x^2 + \mathbf{r}_y^2}}$$

$$\cos\alpha = \frac{\mathbf{r}_y}{\sqrt{\mathbf{r}_x^2 + \mathbf{r}_y^2}}$$



Siciliano et al, Robotics: Modelling, Planing and Control

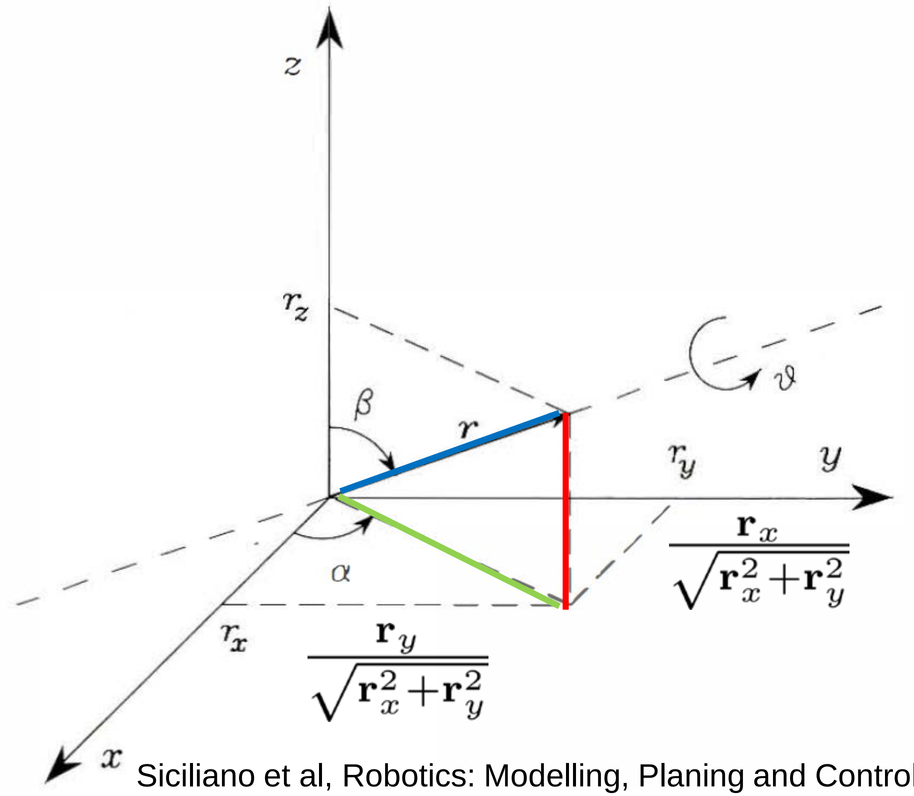
Angle - Axis

$$\mathbf{R}(\vartheta, \mathbf{r}) = \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma)\mathbf{R}_y(-\beta)\mathbf{R}_z(-\alpha)$$

$$\sin\beta = \sqrt{\mathbf{r}_x^2 + \mathbf{r}_y^2}$$

$$\cos\beta = \mathbf{r}_z$$

$$\sqrt{\mathbf{r}_x^2 + \mathbf{r}_y^2 + \mathbf{r}_z^2} = 1$$



Siciliano et al, Robotics: Modelling, Planing and Control

Angle - Axis

$$R(\vartheta,\mathbf{r}) = R_z(\alpha)R_y(\beta)R_z(\vartheta)R_y(-\beta)R_z(-\alpha)$$

$$\sin\alpha = \frac{r_y}{\sqrt{r_x^2 + r_y^2}} \qquad \cos\alpha = \frac{r_x}{\sqrt{r_x^2 + r_y^2}}$$

$$\sin\beta = \sqrt{r_x^2 + r_y^2} \qquad \cos\beta = r_z$$

$$\mathbf{R}(\vartheta,\mathbf{r}) = \begin{pmatrix} r_x^2(1-c_\vartheta)+c_\vartheta & r_x r_y(1-c_\vartheta)-r_z s_\vartheta & r_x r_z(1-c_\vartheta)+r_y s_\vartheta \\ r_x r_y(1-c_\vartheta)+r_z s_\vartheta & r_y^2(1-c_\vartheta)+c_\vartheta & r_x r_z(1-c_\vartheta)-r_x s_\vartheta \\ r_x r_z(1-c_\vartheta)-r_y s_\vartheta & r_y r_z(1-c_\vartheta)+r_x s_\vartheta & r_z^2(1-c_\vartheta)+c_\vartheta \end{pmatrix}$$

$$\mathbf{R}(\vartheta,\mathbf{r}) = \mathbf{R}(-\vartheta,-\mathbf{r})$$

inverse solution

$$\vartheta = \cos^{-1}\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right)$$

$$\mathbf{r} = \frac{1}{2\sin\vartheta}\begin{pmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{pmatrix} \qquad r_x^2 + r_y^2 + r_z^2 = 1$$

Rodrigues Rotation Formula (Euler Parameter)

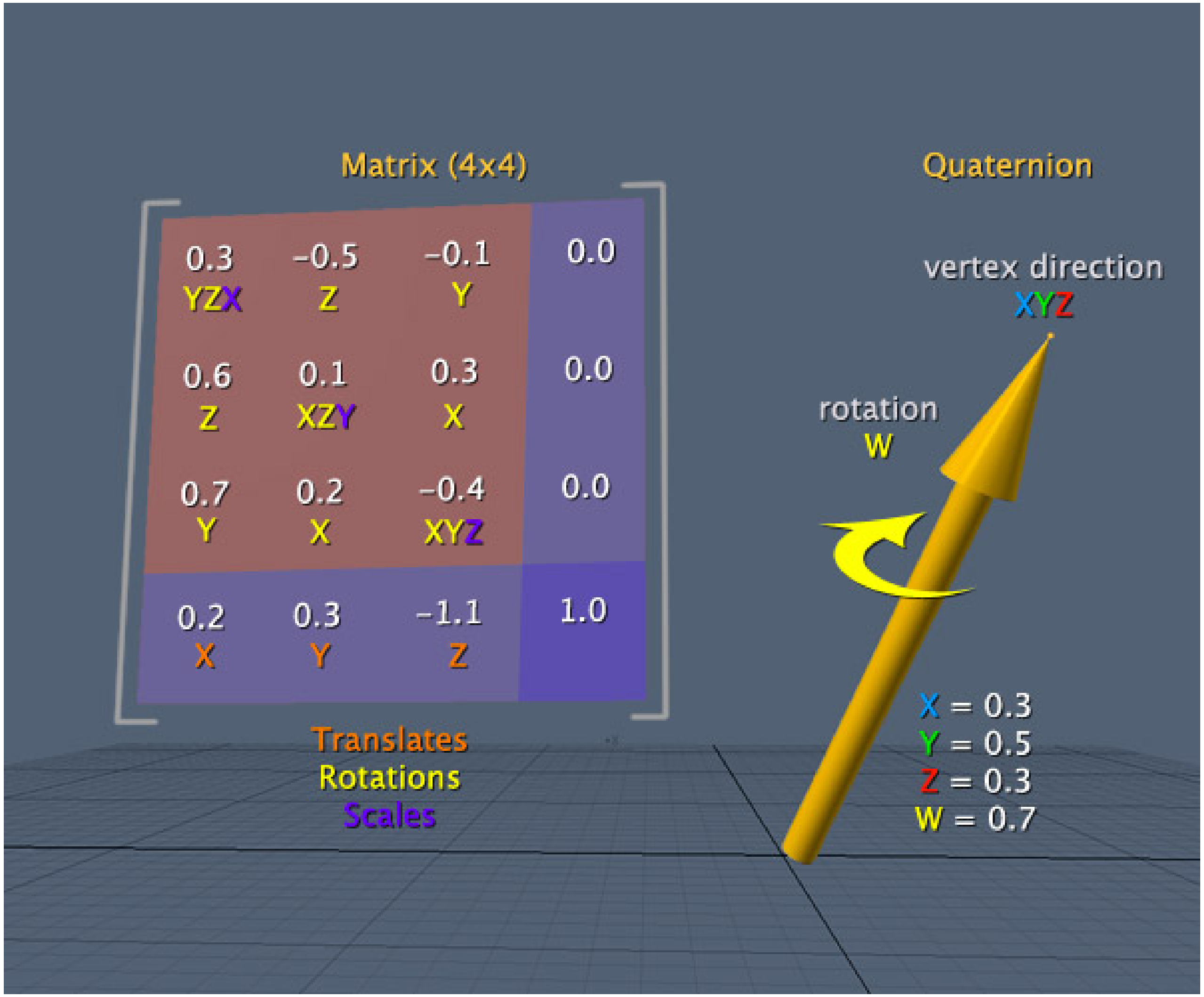
$$\mathbf{p}' = (\cos\vartheta)\mathbf{p} + \sin\vartheta(\mathbf{r}\times\mathbf{p}) + (1-\cos\vartheta)(\mathbf{r}\circ\mathbf{p})\mathbf{r}$$

$$\mathbf{p}' = \mathbf{R}\mathbf{p} \qquad\qquad \mathbf{R} = \mathbf{I}\cos\vartheta + \sin\vartheta[\mathbf{r}]_{\mathbf{x}} + (1-\cos\vartheta)\mathbf{r}\otimes\mathbf{r}$$

$$[\mathbf{r}]_{\times} = \mathbf{r}\times\ldots = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} \qquad \mathbf{r}\otimes\mathbf{r} = \begin{pmatrix} r_x{}^2(1-c_\vartheta) & r_x r_y(1-c_\vartheta) & r_x r_z(1-c_\vartheta) \\ r_x r_y(1-c_\vartheta) & r_y{}^2(1-c_\vartheta) & r_x r_z(1-c_\vartheta) \\ r_x r_z(1-c_\vartheta) & r_y r_z(1-c_\vartheta) & r_z{}^2(1-c_\vartheta) \end{pmatrix}$$

$$\mathbf{R}(\vartheta,\mathbf{r}) = \begin{pmatrix} r_x{}^2(1-c_\vartheta)+c_\vartheta & r_x r_y(1-c_\vartheta)-r_z s_\vartheta & r_x r_z(1-c_\vartheta)+r_y s_\vartheta \\ r_x r_y(1-c_\vartheta)+r_z s_\vartheta & r_y{}^2(1-c_\vartheta)+c_\vartheta & r_x r_z(1-c_\vartheta)-r_x s_\vartheta \\ r_x r_z(1-c_\vartheta)-r_y s_\vartheta & r_y r_z(1-c_\vartheta)+r_x s_\vartheta & r_z{}^2(1-c_\vartheta)+c_\vartheta \end{pmatrix}$$

Unit Quaternion



http://vignette4.wikia.nocookie.net/science/images/1/1e/Quaternion-03-goog.jpg/revision/latest?cb=20131024191311&path-prefix=el

Rotations and Quaternions

3D world
Euclidean space

4D space
(one real and
three imaginary
dimensions)

Rotation matrix to quaternion

3D rotation

$$\mathbf{p}' = \mathbf{R}\mathbf{p}$$

matrix
product

$$\mathbf{R}_2^0 = \mathbf{R}_1^0 \mathbf{R}_2^1$$

Quaternion
product

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$$

$$\mathbf{p} = [0 \quad \mathbf{p}_x \quad \mathbf{p}_y \quad \mathbf{p}_z]$$

Quaternion
product

$$\mathbf{q}_2^0 = \mathbf{q}_1^0 \mathbf{q}_2^1$$

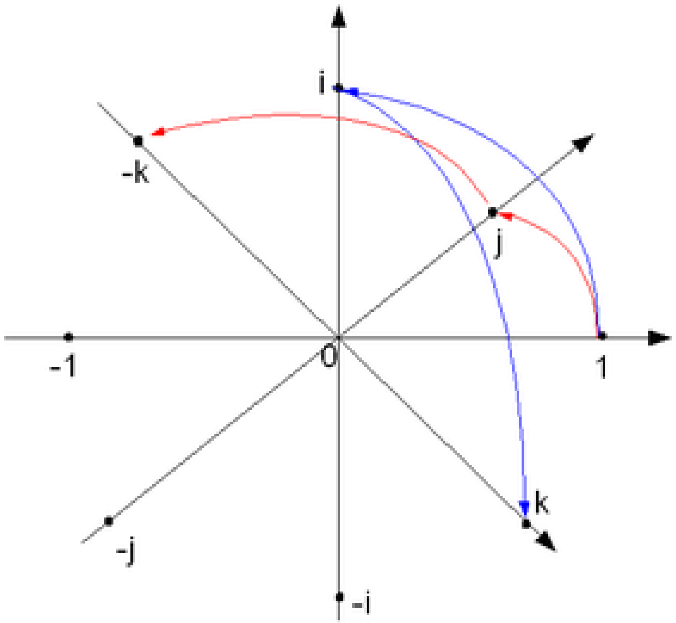
Quaternion to rotation matrix

Quaternion

$$\mathbf{q} = \{\eta, \boldsymbol{\varepsilon}\} = [\eta \quad \varepsilon_x \quad \varepsilon_y \quad \varepsilon_z]$$

$$= [\eta \quad \varepsilon_x \mathbf{i} \quad \varepsilon_y \mathbf{j} \quad \varepsilon_z \mathbf{k}]$$

×	1	*i*	*j*	*k*
1	1	*i*	*j*	*k*
i	*i*	−1	*k*	−*j*
j	*j*	−*k*	−1	*i*
k	*k*	*j*	−*i*	−1



Graphical representation of quaternion units product as 90°-rotation in 4D-space

ij = k
ji = -k
ij = -ji

- Unit quaternion

$$\eta^2 + \varepsilon_x^{\ 2} + \varepsilon_y^{\ 2} + \varepsilon_z^{\ 2} = 1$$

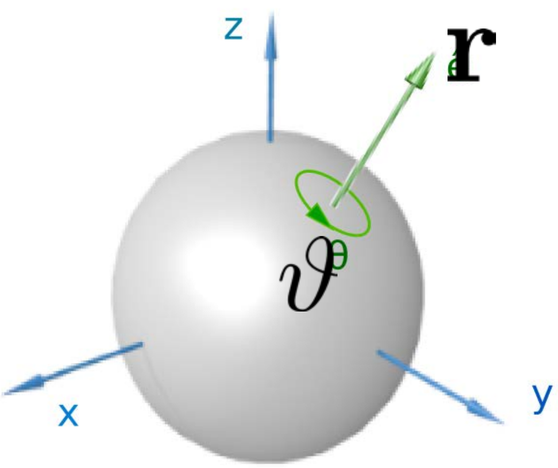
By ProkopHapala at English Wikipedia - Transferred from en.wikipedia to Commons by Ebe123., Public Domain, https://commons.wikimedia.org/w/index.php?curid=16097554

Eulers Formula

Euclidean 3D vector $\mathbf{r} = [r_x, r_y, r_z] = r_x\mathbf{i} + r_y\mathbf{j} + r_x\mathbf{k}$

$$\mathbf{q} = e^{\frac{\vartheta}{2}(r_x\mathbf{i}+r_y\mathbf{j}+r_z\mathbf{k})} = \cos\frac{\vartheta}{2} + (r_x\mathbf{i} + r_y\mathbf{j} + r_z\mathbf{k})\sin\frac{\vartheta}{2}$$

$$\mathbf{q}^{-1} = e^{-\frac{\vartheta}{2}(r_x\mathbf{i}+r_y\mathbf{j}+r_z\mathbf{k})} = \cos\frac{\vartheta}{2} - (r_x\mathbf{i} + r_y\mathbf{j} + r_z\mathbf{k})\sin\frac{\vartheta}{2}$$



By DF Malan - Own work, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=1354297

Unit Quaternion and Angle Axis

- Quaternion $$\mathbf{q} = \{\eta, \boldsymbol{\varepsilon}\} = [\eta \quad \boldsymbol{\varepsilon}_x \quad \boldsymbol{\varepsilon}_y \quad \boldsymbol{\varepsilon}_z]$$

- Relation to angle-axis $$\underbrace{\eta}_{\text{scalar part}} = \cos\frac{\vartheta}{2}, \quad \underbrace{\boldsymbol{\varepsilon}}_{\text{vector part}} = \begin{bmatrix} \boldsymbol{\varepsilon}_x & \boldsymbol{\varepsilon}_y & \boldsymbol{\varepsilon}_z \end{bmatrix} = \sin\frac{\vartheta}{2}\mathbf{r}$$

$$\mathbf{q} = \{\eta, \boldsymbol{\varepsilon}\} = [\cos\frac{\vartheta}{2} \quad \sin\frac{\vartheta}{2} r_x \quad \sin\frac{\vartheta}{2} r_y \quad \sin\frac{\vartheta}{2} r_z]$$

- Constraint $$\eta^2 + \varepsilon_x{}^2 + \varepsilon_y{}^2 + \varepsilon_z{}^2 = 1$$

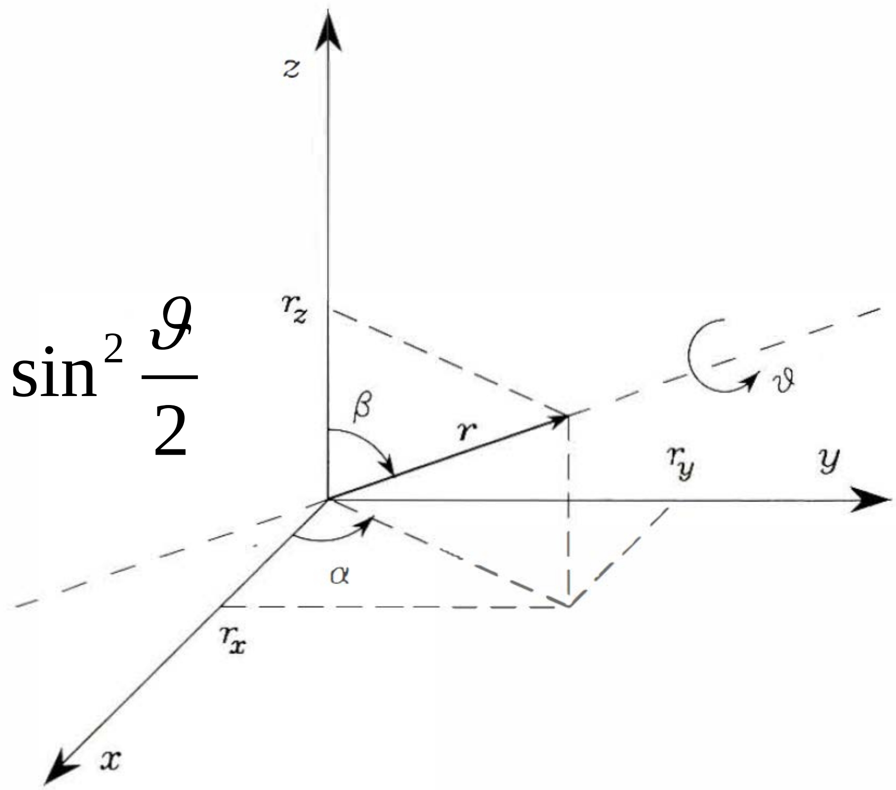
Unit Quaternion to Angle Axis

- Quaternion $\mathbf{q} = \{\eta, \boldsymbol{\varepsilon}\} = [\eta \quad \boldsymbol{\varepsilon}_x \quad \boldsymbol{\varepsilon}_y \quad \boldsymbol{\varepsilon}_z]$

- Rotation axis $\mathbf{r} = \dfrac{\boldsymbol{\varepsilon}}{\|\boldsymbol{\varepsilon}\|}$

- Rotation angle

$$\cos\vartheta = \eta^2 - \|\boldsymbol{\varepsilon}\|^2 = \cos^2\frac{\vartheta}{2} - \sin^2\frac{\vartheta}{2}$$



Siciliano et al, Robotics: Modelling, Planing and Control

Unit Quaternion to Rotation Matrix

- Quaternion $\mathbf{q} = \{\eta, \boldsymbol{\varepsilon}\}$

- Rotation matrix

$$\mathbf{R}(\eta,\boldsymbol{\varepsilon}) = \begin{pmatrix} 2(\eta^2+\varepsilon_x{}^2)-1 & 2(\varepsilon_x\varepsilon_y-\eta\varepsilon_z) & 2(\varepsilon_x\varepsilon_z-\eta\varepsilon_y) \\ 2(\varepsilon_x\varepsilon_y+\eta\varepsilon_z) & 2(\eta^2+\varepsilon_y{}^2)-1 & 2(\varepsilon_y\varepsilon_z-\eta\varepsilon_x) \\ 2(\varepsilon_x\varepsilon_z-\eta\varepsilon_y) & 2(\varepsilon_y\varepsilon_z+\eta\varepsilon_x) & 2(\eta^2+\varepsilon_z{}^2)-1 \end{pmatrix}$$

$$\mathbf{R}^{-1}(\eta,\boldsymbol{\varepsilon}) = \mathbf{R}^T(\eta,\boldsymbol{\varepsilon}) \longrightarrow \mathbf{q}^{-1} = \{\eta, -\boldsymbol{\varepsilon}\}$$

Rotation Matrix to Unit Quaternion

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

$$\eta = \frac{1}{2}\sqrt{r_{11}{}^2 + r_{22}{}^2 + r_{33}{}^2}$$

$$\boldsymbol{\varepsilon} = \frac{1}{2}\begin{bmatrix} \operatorname{sgn}(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ \operatorname{sgn}(r_{13} - r_{31})\sqrt{r_{22} - r_{22} - r_{11} + 1} \\ \operatorname{sgn}(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix}$$

Unit Quaternion

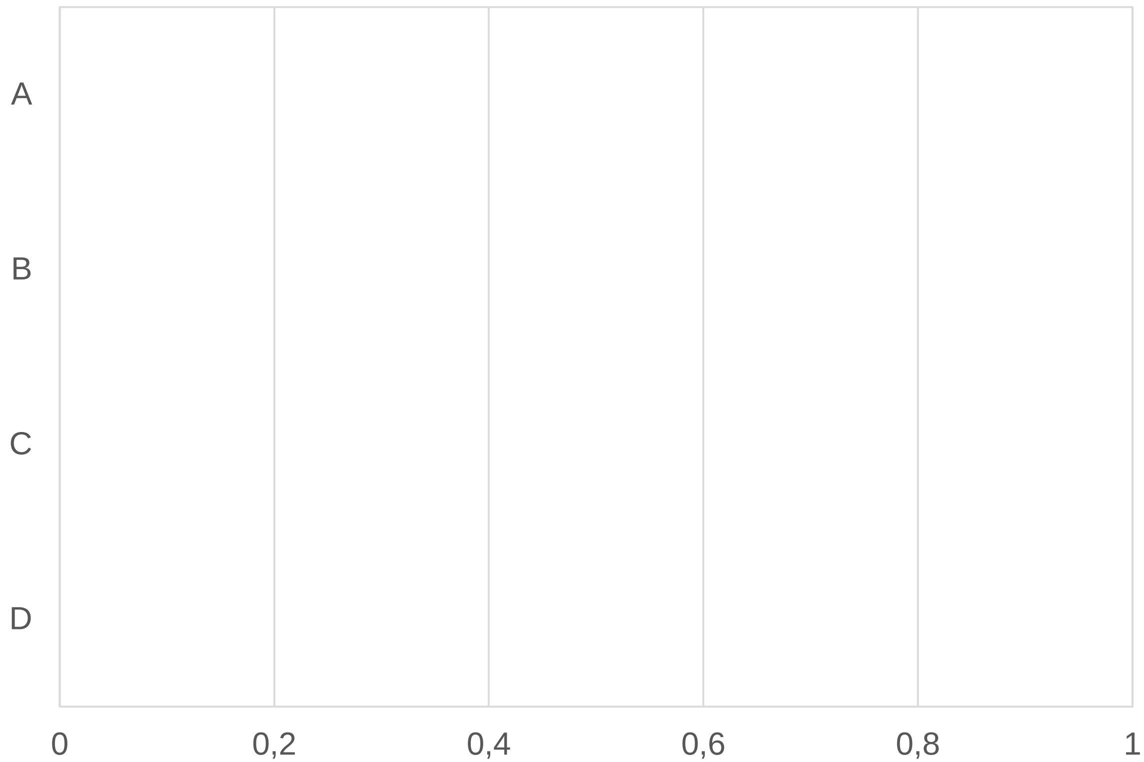
Which relations between unit quaternion and angle axis are correct?

$$\boldsymbol{\epsilon} = \sin(\vartheta)\mathbf{r}/2$$

$$\boldsymbol{\epsilon} = \sin(\vartheta/2)\mathbf{r}$$

$$\eta = \cos(\vartheta/2)$$

$$\eta = \sin(\vartheta)$$



Umfrage starten

ID = frank.hoffmann@tu-dortmund.de
Umfrage noch nicht gestartet

Product of Unit Quaternions

- Complex number $z = a + b\mathbf{i}$
- Product of complex numbers

$$(a+b\mathbf{i})(c+d\mathbf{i}) = (ac-bd)+(bc+ad)\mathbf{i}$$

- Quaternion: scalar + vector $\eta + \varepsilon_x\mathbf{i} + \varepsilon_y\mathbf{j} + \varepsilon_z\mathbf{k} = (\eta, \boldsymbol{\varepsilon})$
- Product of quaternions

$$(\eta^1 + \varepsilon_x{}^1\mathbf{i} + \varepsilon_y{}^1\mathbf{j} + \varepsilon_z{}^1\mathbf{k})(\eta^2 + \varepsilon_x{}^2\mathbf{i} + \varepsilon_y{}^2\mathbf{j} + \varepsilon_z{}^2\mathbf{k}) =$$

$$(\eta^1\eta^2 - \varepsilon_x{}^1\varepsilon_x{}^2 - \varepsilon_y{}^1\varepsilon_y{}^2 - \varepsilon_z{}^1\varepsilon_z{}^2) +$$

$$(\eta^1\varepsilon_x{}^2 + \varepsilon_x{}^1\eta^2 + \varepsilon_y{}^1\varepsilon_z{}^2 - \varepsilon_z{}^1\varepsilon_y{}^2)\mathbf{i} +$$

$$(\eta^1\varepsilon_y{}^2 - \varepsilon_x{}^1\varepsilon_z{}^2 + \varepsilon_y{}^1\eta^2 + \varepsilon_z{}^1\varepsilon_x{}^2)\mathbf{j} +$$

$$(\eta^1\varepsilon_z{}^2 + \varepsilon_x{}^1\varepsilon_y{}^2 - \varepsilon_y{}^1\varepsilon_x{}^2 + \varepsilon_z{}^1\eta^2)\mathbf{k}$$

$\mathbf{ij} = \mathbf{k}$ $\mathbf{ji} = -\mathbf{k}$

$\mathbf{jk} = \mathbf{i}$ $\mathbf{kj} = -\mathbf{i}$

$\mathbf{ki} = \mathbf{j}$ $\mathbf{ik} = -\mathbf{j}$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$$

Composition of Rotations with Unit Quaternions

- Quaternions $q_1 = \{\eta_1, \boldsymbol{\varepsilon}_1\}, q_2 = \{\eta_2, \boldsymbol{\varepsilon}_2\}$ corresponding to rotation matrices $\mathbf{R}_1, \mathbf{R}_2$

- Quaternion corresponding to product $\mathbf{R} = \mathbf{R}_1 \mathbf{R}_2$

$$q_1 * q_2 = \{\eta_1\eta_2 - \boldsymbol{\varepsilon}_1{}^T\boldsymbol{\varepsilon}_2, \eta_1\boldsymbol{\varepsilon}_2 + \eta_2\boldsymbol{\varepsilon}_1 + \boldsymbol{\varepsilon}_1 \times \boldsymbol{\varepsilon}_2\}$$

$$(\eta^1 + \varepsilon_x{}^1\mathbf{i} + \varepsilon_y{}^1\mathbf{j} + \varepsilon_z{}^1\mathbf{k})(\eta^2 + \varepsilon_x{}^2\mathbf{i} + \varepsilon_y{}^2\mathbf{j} + \varepsilon_z{}^2\mathbf{k}) =$$

$$(\eta^1\eta^2 - \varepsilon_x{}^1\varepsilon_x{}^2 - \varepsilon_y{}^1\varepsilon_y{}^2 - \varepsilon_z{}^1\varepsilon_z{}^2) +$$

$$(\eta^1\varepsilon_x{}^2 + \varepsilon_x{}^1\eta^2 + \varepsilon_y{}^1\varepsilon_z{}^2 - \varepsilon_z{}^1\varepsilon_y{}^2)\mathbf{i} +$$

$$(\eta^1\varepsilon_y{}^2 - \varepsilon_x{}^1\varepsilon_z{}^2 + \varepsilon_y{}^1\eta^2 + \varepsilon_z{}^1\varepsilon_x{}^2)\mathbf{j} +$$

$$(\eta^1\varepsilon_z{}^2 + \varepsilon_x{}^1\varepsilon_y{}^2 - \varepsilon_y{}^1\varepsilon_x{}^2 + \varepsilon_z{}^1\eta^2)\mathbf{k}$$

Rotation of 3D Vectors Using Quaternions

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$$

$$\mathbf{p} = [0 \quad \mathbf{p}_x \quad \mathbf{p}_y \quad \mathbf{p}_z]$$

$$\mathbf{q} = \{\eta, \boldsymbol{\varepsilon}\} = [\eta \quad \boldsymbol{\varepsilon}_x \quad \boldsymbol{\varepsilon}_y \quad \boldsymbol{\varepsilon}_z]$$

$$\mathbf{q}^{-1} = \{\eta, -\boldsymbol{\varepsilon}\} = [\eta \quad -\boldsymbol{\varepsilon}_x \quad -\boldsymbol{\varepsilon}_y \quad -\boldsymbol{\varepsilon}_z]$$

$$\mathbf{q} = e^{\frac{\vartheta}{2}(r_x\mathbf{i}+r_y\mathbf{j}+r_x\mathbf{k})} = \cos\frac{\vartheta}{2} + (r_x\mathbf{i} + r_y\mathbf{j} + r_x\mathbf{k})\sin\frac{\vartheta}{2}$$

$$\mathbf{q}^{-1} = e^{-\frac{\vartheta}{2}(r_x\mathbf{i}+r_y\mathbf{j}+r_x\mathbf{k})} = \cos\frac{\vartheta}{2} - (r_x\mathbf{i} + r_y\mathbf{j} + r_x\mathbf{k})\sin\frac{\vartheta}{2}$$

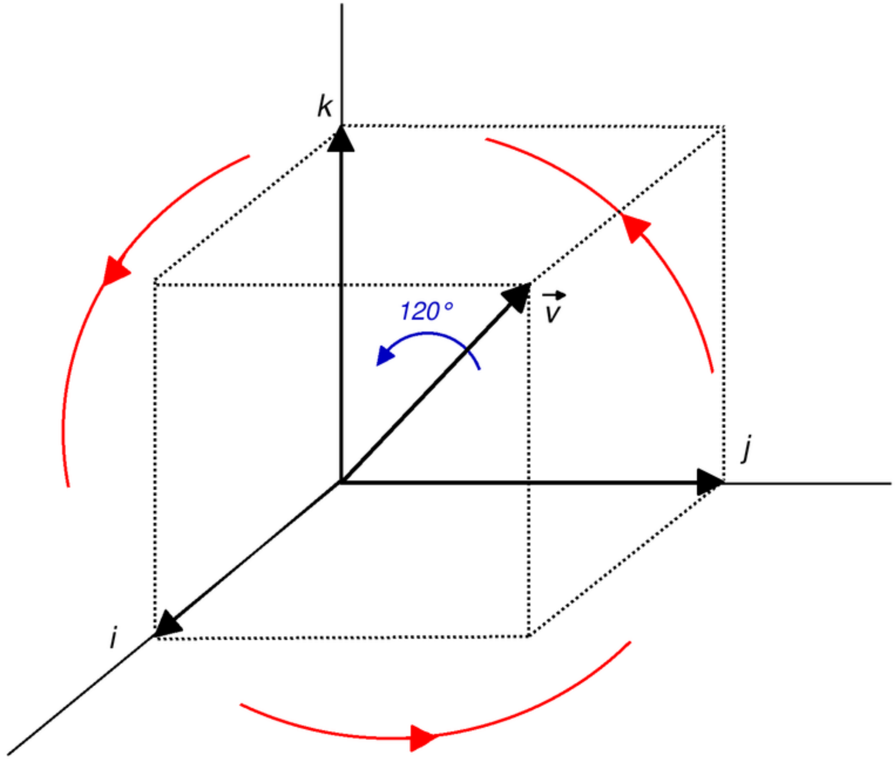
Rotation of 3D Vectors Using Quaternions

$$\vec{\mathbf{v}} = \begin{bmatrix}1 & 1 & 1\end{bmatrix} = \mathbf{i} + \mathbf{j} + \mathbf{k} \qquad \vartheta = \frac{2\pi}{3}$$

$$\mathbf{q} = \frac{1 + \mathbf{i} + \mathbf{j} + \mathbf{k}}{2}$$

$$\mathbf{q}^{-1} = \frac{1 - \mathbf{i} - \mathbf{j} - \mathbf{k}}{2}$$

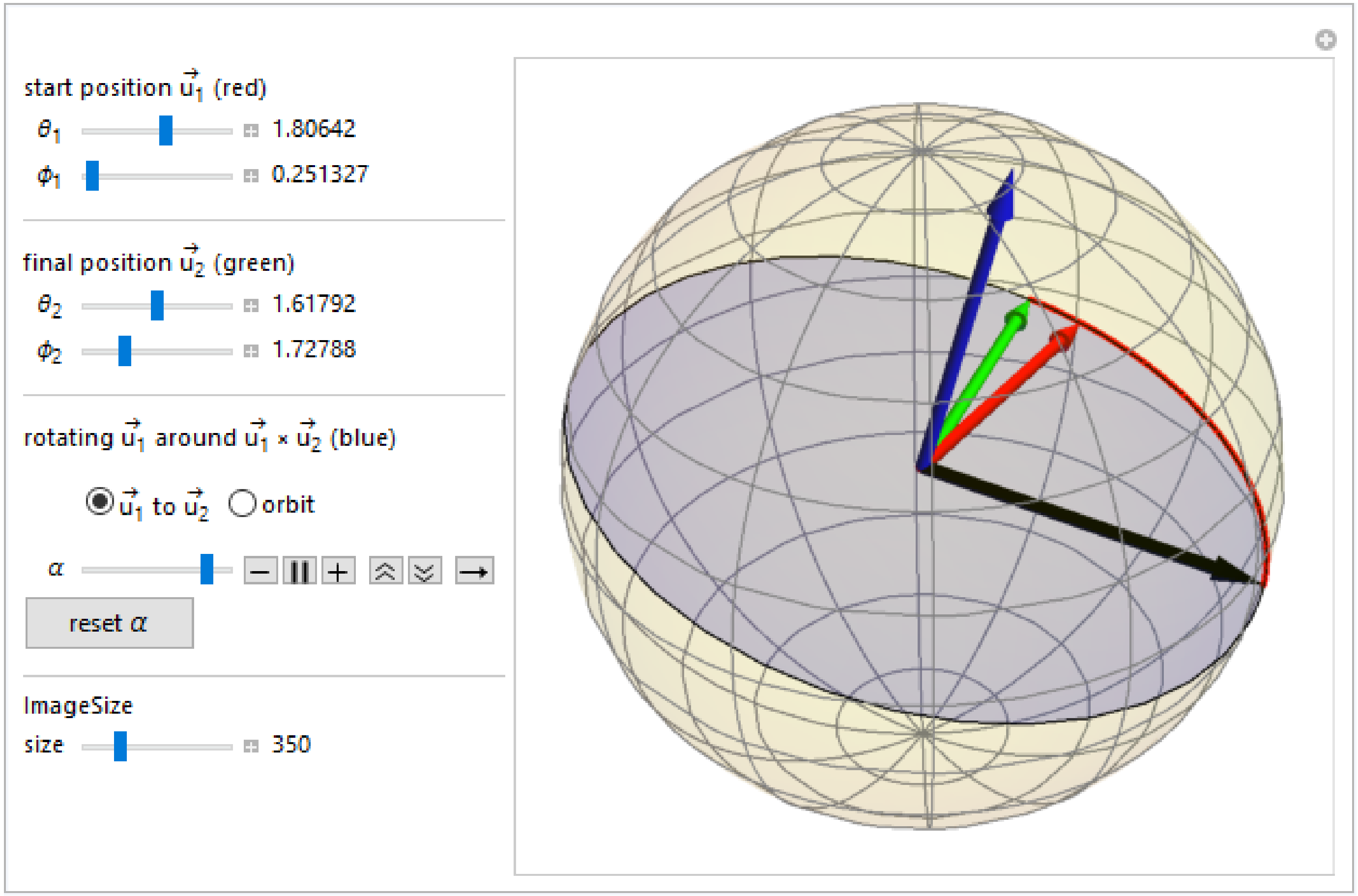
$$\mathbf{p} = \begin{bmatrix}p_x & p_x & p_x\end{bmatrix} = 0 + p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$$



$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1} = \frac{1 + \mathbf{i} + \mathbf{j} + \mathbf{k}}{2}(0 + p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k})\frac{1 - \mathbf{i} - \mathbf{j} - \mathbf{k}}{2}$$

$$= p_z\mathbf{i} + p_x\mathbf{j} + p_y\mathbf{k}$$

Rotating a Unit Vector in 3D Using Quaternions



Quaternion Arithmetic

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1} \qquad \mathbf{p} = [0 \quad \mathbf{p}_x \quad \mathbf{p}_y \quad \mathbf{p}_z]$$

$$\mathbf{q} = e^{\frac{\vartheta}{2}(r_x\mathbf{i}+r_y\mathbf{j}+r_x\mathbf{k})} = \cos\frac{\vartheta}{2} + (r_x\mathbf{i} + r_y\mathbf{j} + r_x\mathbf{k})\sin\frac{\vartheta}{2}$$

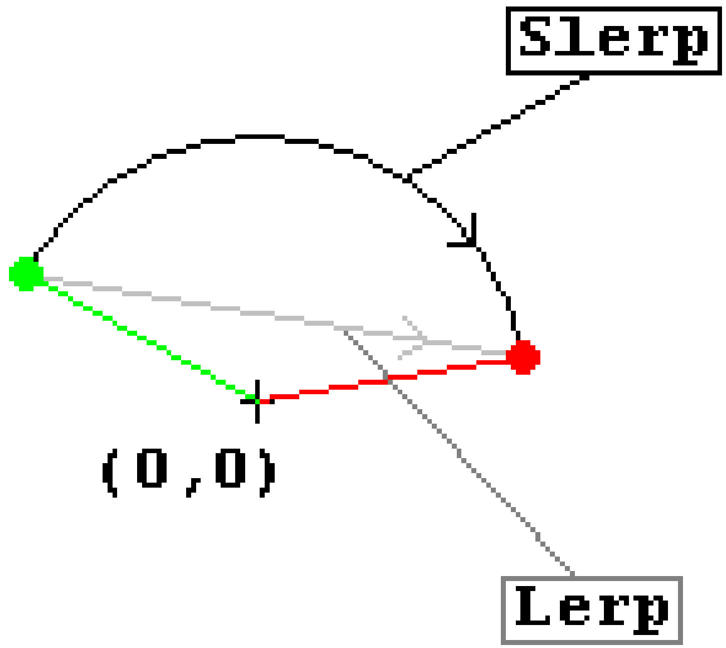
$$\mathbf{q}^{-1} = e^{-\frac{\vartheta}{2}(r_x\mathbf{i}+r_y\mathbf{j}+r_x\mathbf{k})} = \cos\frac{\vartheta}{2} - (r_x\mathbf{i} + r_y\mathbf{j} + r_x\mathbf{k})\sin\frac{\vartheta}{2}$$

$$\mathbf{R}(\vartheta, \mathbf{r}) = \begin{pmatrix} r_x^{\,2}(1-c_\vartheta)+c_\vartheta & r_x r_y(1-c_\vartheta)-r_z s_\vartheta & r_x r_z(1-c_\vartheta)+r_y s_\vartheta \\ r_x r_y(1-c_\vartheta)+r_z s_\vartheta & r_y^{\,2}(1-c_\vartheta)+c_\vartheta & r_x r_z(1-c_\vartheta)-r_x s_\vartheta \\ r_x r_z(1-c_\vartheta)-r_y s_\vartheta & r_y r_z(1-c_\vartheta)+r_x s_\vartheta & r_z^{\,2}(1-c_\vartheta)+c_\vartheta \end{pmatrix}$$

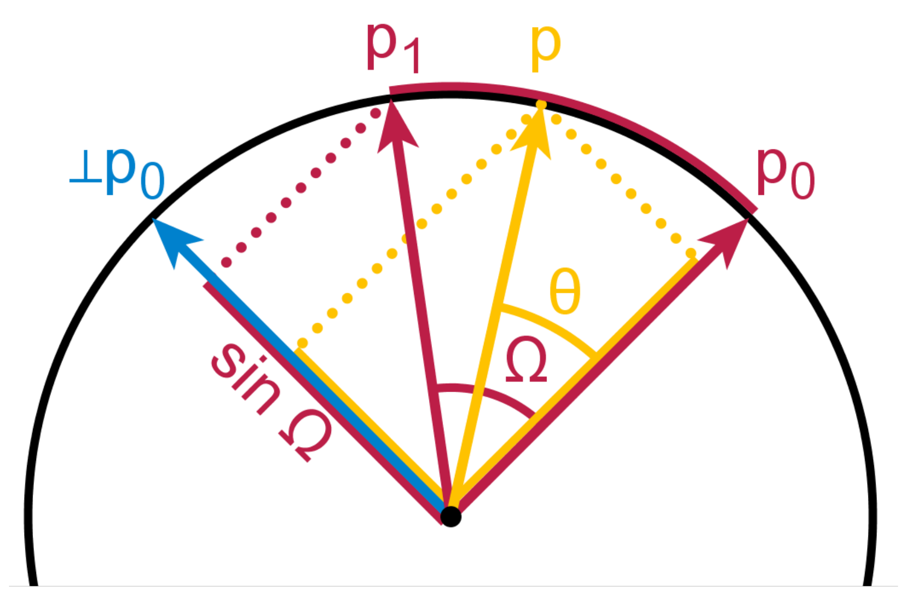
Spherical Linear Interpolation

$$\mathrm{lerp}(\mathbf{p}_0, \mathbf{p}_1, t) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1$$

$$\mathrm{slerp}(\mathbf{p}_0, \mathbf{p}_1, t) = \frac{sin((1-t)\Omega)}{\sin\Omega}\mathbf{p}_0 + \frac{sin(t\Omega)}{\sin\Omega}\mathbf{p}_1$$

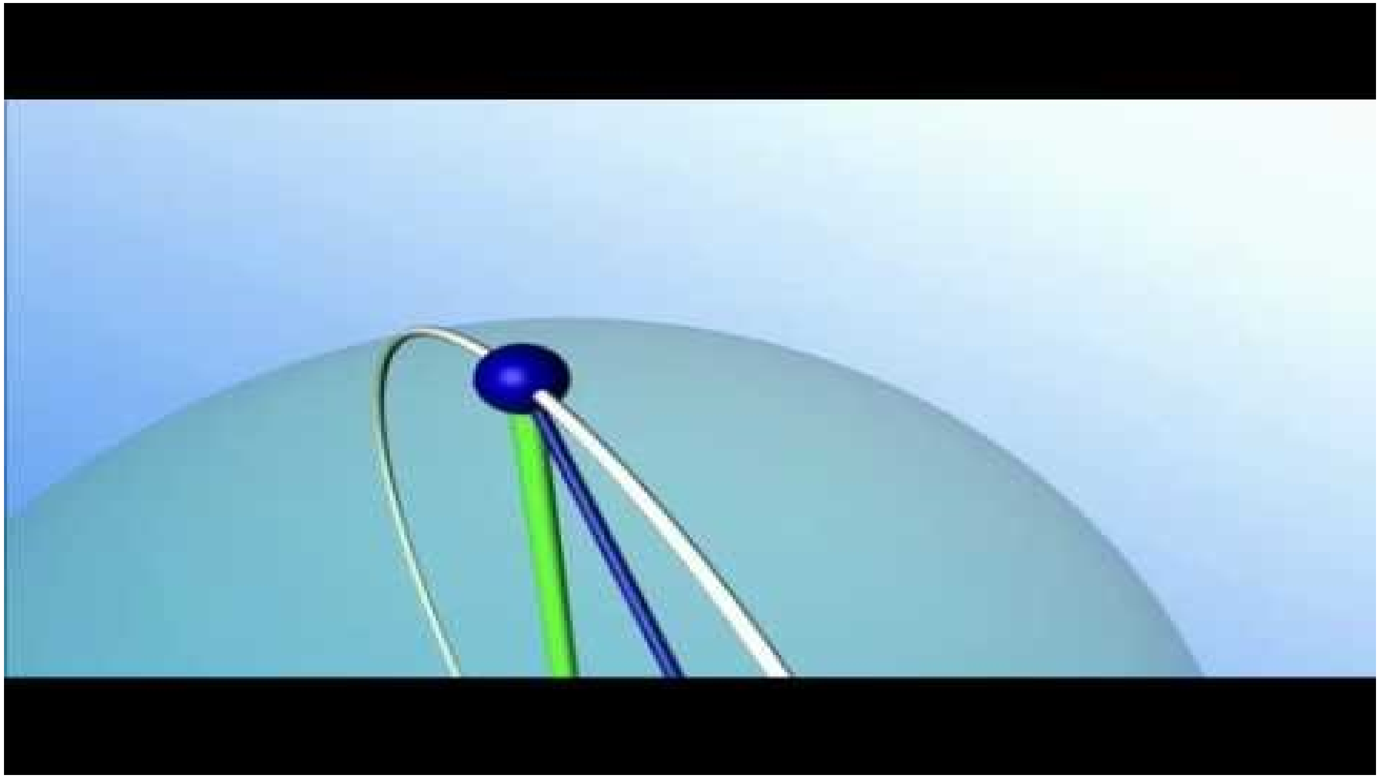


http://answers.unity3d.com/storage/temp/7325-slerp.png



https://commons.wikimedia.org/w/index.php?curid=267089

Slerp vs. Lerp

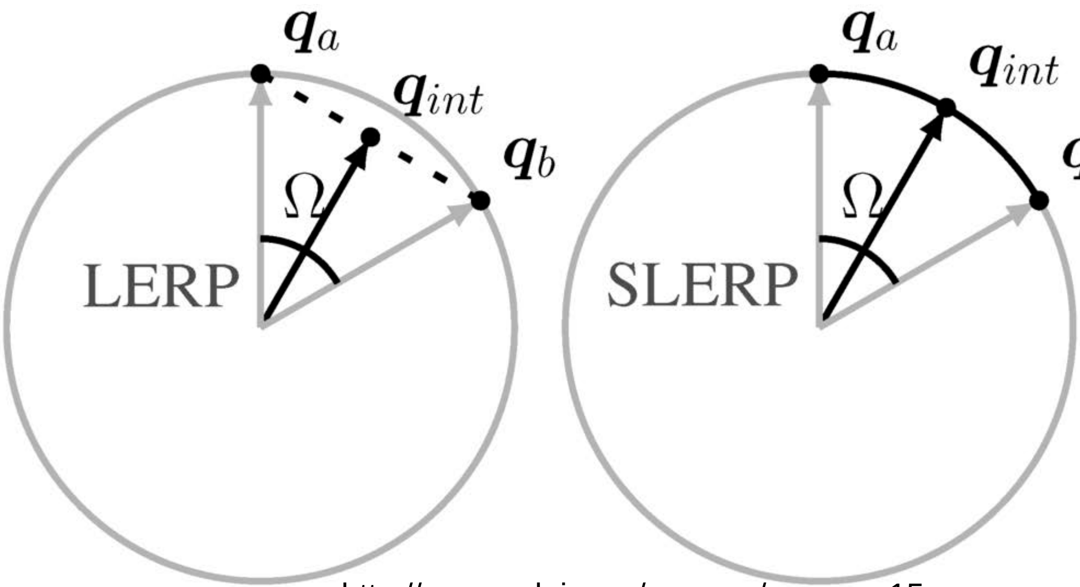


https://www.youtube.com/watch?v=uNHIPVOnt-Y

Quaternion Slerp

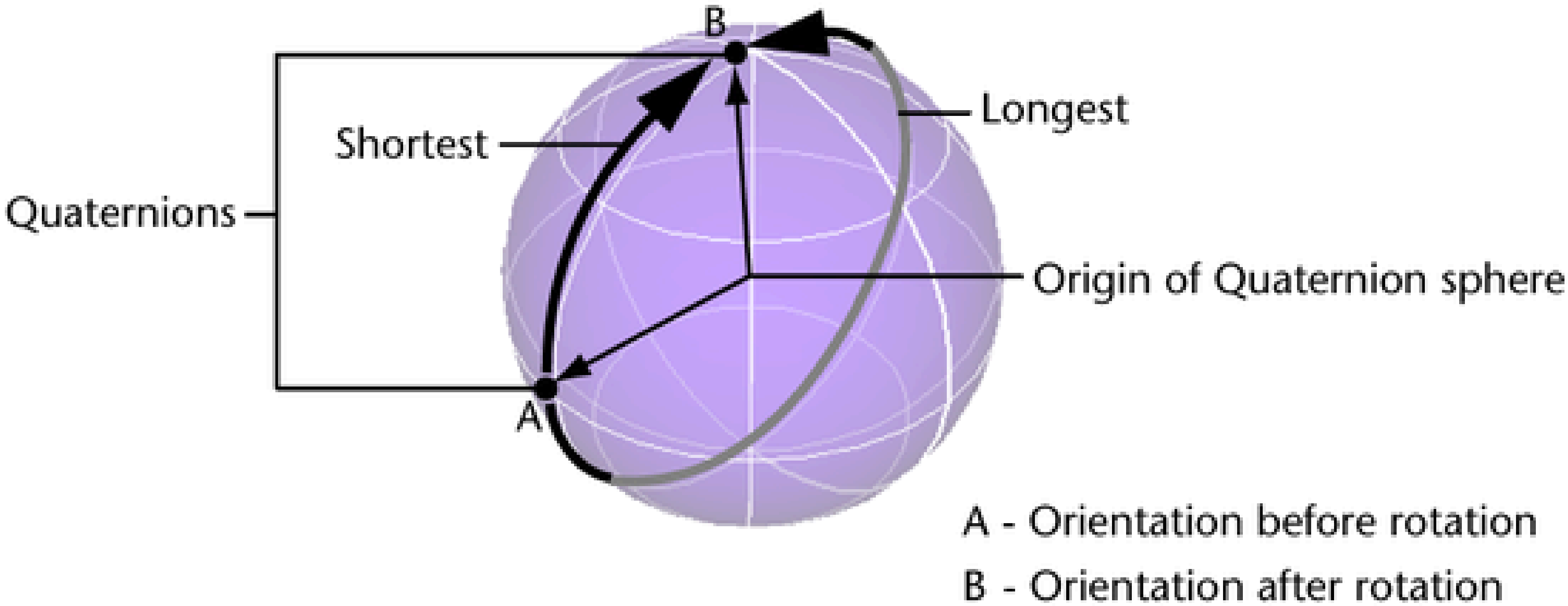
- Power series $$e^q = 1 + \frac{q^2}{2} + \frac{q^3}{6} + \ldots + \frac{q^n}{n!}$$
- Versor form : $$q^t = \cos\Omega t + \mathbf{v}\sin\Omega t \qquad e^{\mathbf{v}\Omega} = q$$

$$\mathbf{q} = \mathbf{q}_1\mathbf{q}_0^{-1} \qquad \mathrm{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}_0(\mathbf{q}_0^{-1}\mathbf{q}_1)^t$$



http://www.mdpi.com/sensors/sensors-15-19302/article_deploy/html/images/sensors-15-19302-g004-1024.png

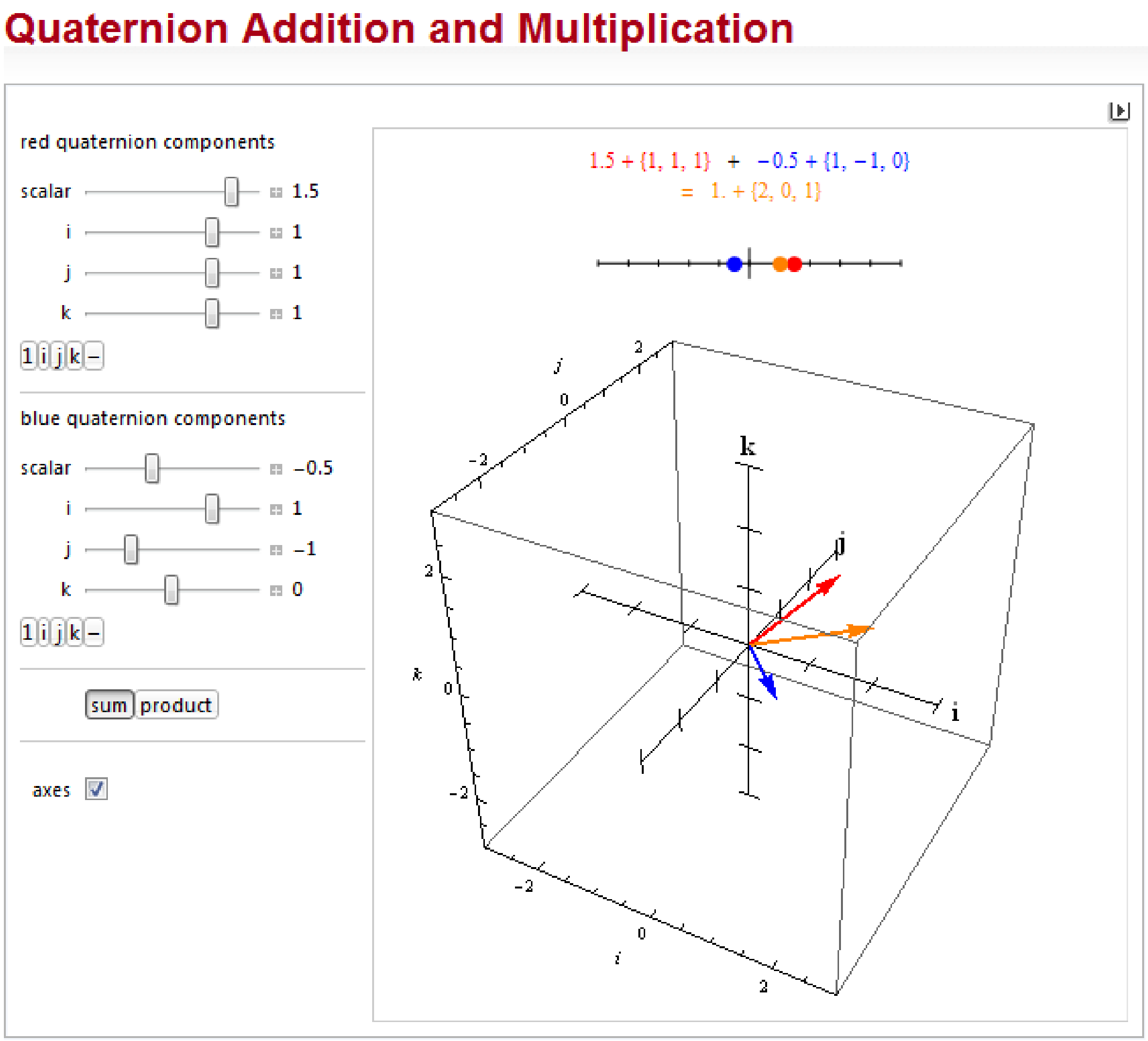
Quaternion Rotation Interpolation



Quaternion rotation interpolation

http://help.autodesk.com/cloudhelp/2015/ENU/Maya/images/GUI
D-D6410250-2B26-4B80-B810-C2D9AAE79F9E.png

Quaternion



Quaternion Addition and Multiplication
red quaternion components
scalar 1.5
i 1
j 1
k 1
1 i j k –
blue quaternion components
scalar −0.5
i 1
j −1
k 0
1 i j k –
sum product
axes
1.5 + {1, 1, 1} + −0.5 + {1, −1, 0}
= 1. + {2, 0, 1}
k
j
i

Spatial Transformations

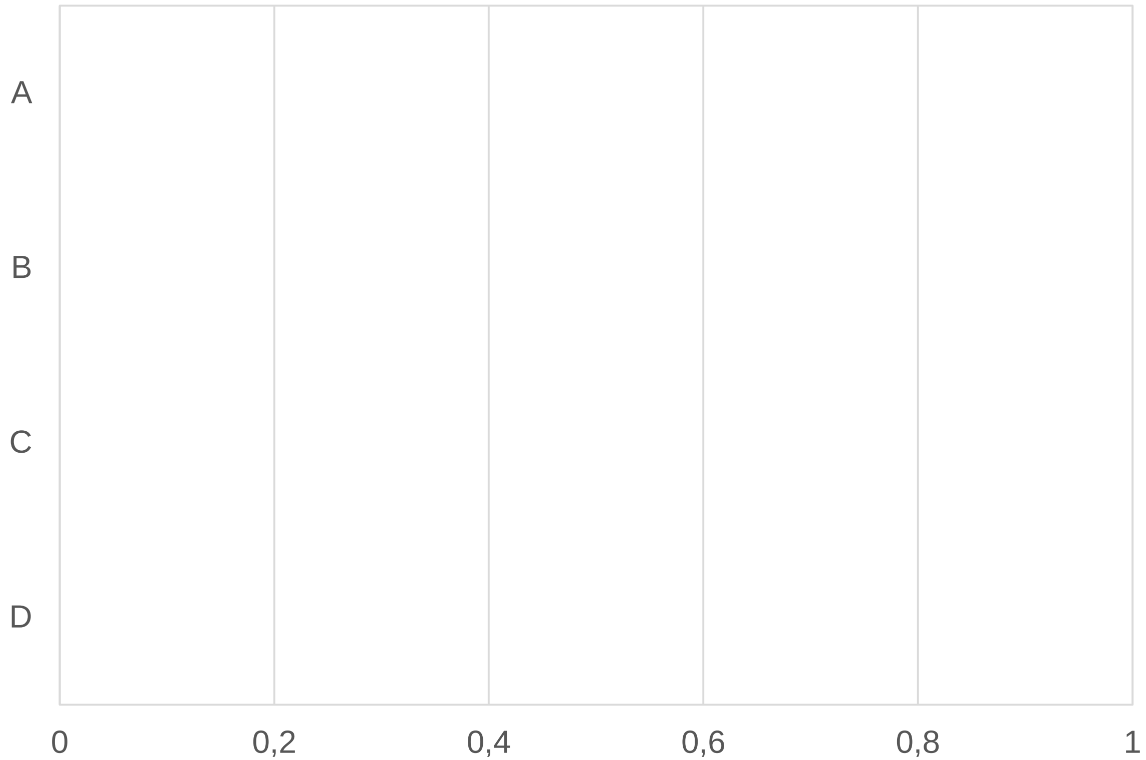
Which representations of rotations have redundant parameters?

Rotation matrix

Euler angles

Unit quaternion

Angle axis



Umfrage starten

ID = frank.hoffmann@tu-dortmund.de
Umfrage noch nicht gestartet

Homogeneous Transformations

Position of point P w.r.t. the reference frame

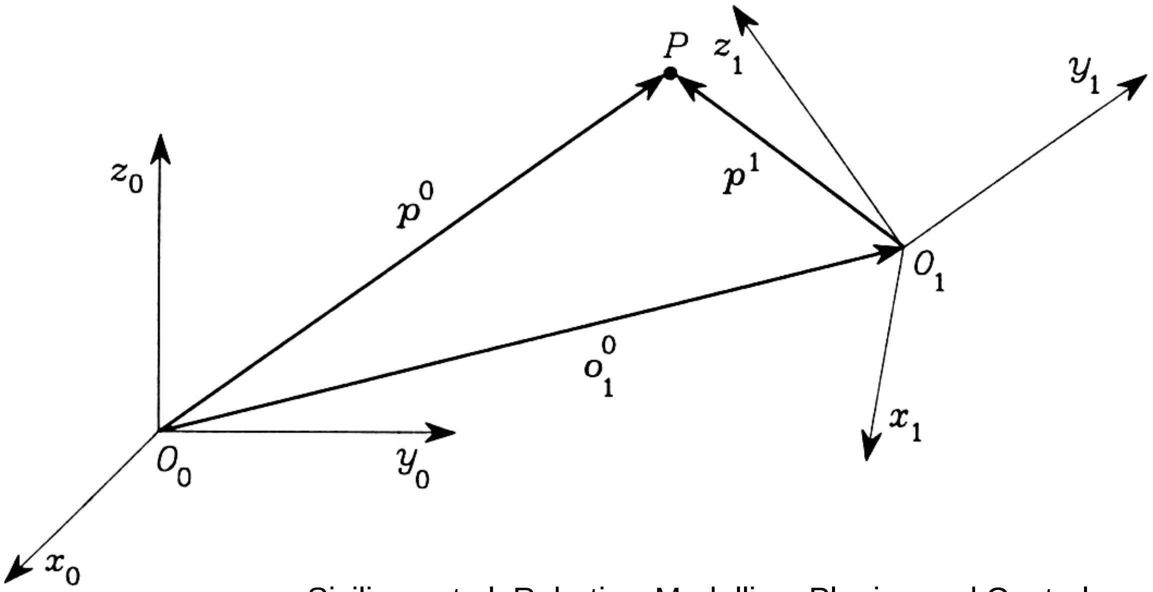
$$\boldsymbol{p}^0 = \boldsymbol{o}_1^0 + \boldsymbol{R}_1^0 \boldsymbol{p}^1.$$

coordinate transformation (*translation* + *rotation*) of a bound vector between two frames.

Inverse transformation

$$\boldsymbol{p}^1 = -\boldsymbol{R}_1^{0^T} \boldsymbol{o}_1^0 + \boldsymbol{R}_1^{0^T} \boldsymbol{p}^0$$

$$\boldsymbol{p}^1 = -\boldsymbol{R}_0^1 \boldsymbol{o}_1^0 + \boldsymbol{R}_0^1 \boldsymbol{p}^0$$



Siciliano et al, Robotics: Modelling, Planing and Control

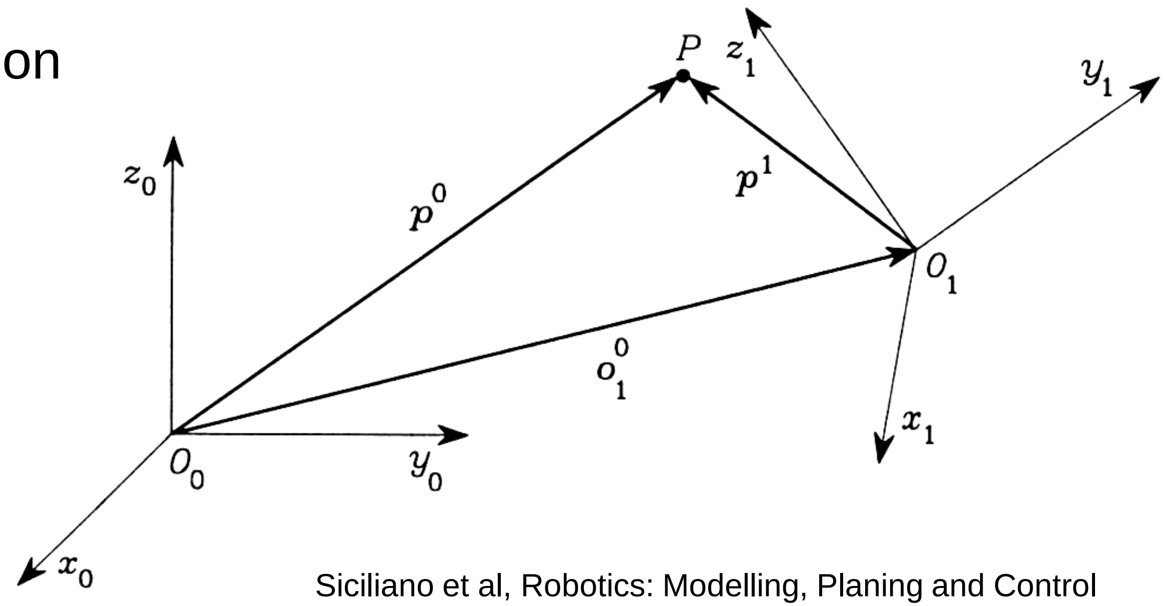
Homogeneous Transformations

Homogeneous representation of a vector

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix}$$

$$\tilde{\boldsymbol{p}}_0 = \begin{bmatrix} \boldsymbol{p}_0 \\ 1 \end{bmatrix} \qquad \tilde{\boldsymbol{p}}_1 = \begin{bmatrix} \boldsymbol{p}_1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{A}_1^0 = \begin{bmatrix} \boldsymbol{R}_1^0 & \boldsymbol{o}_1^0 \\ \boldsymbol{0}^T & 1 \end{bmatrix}.$$



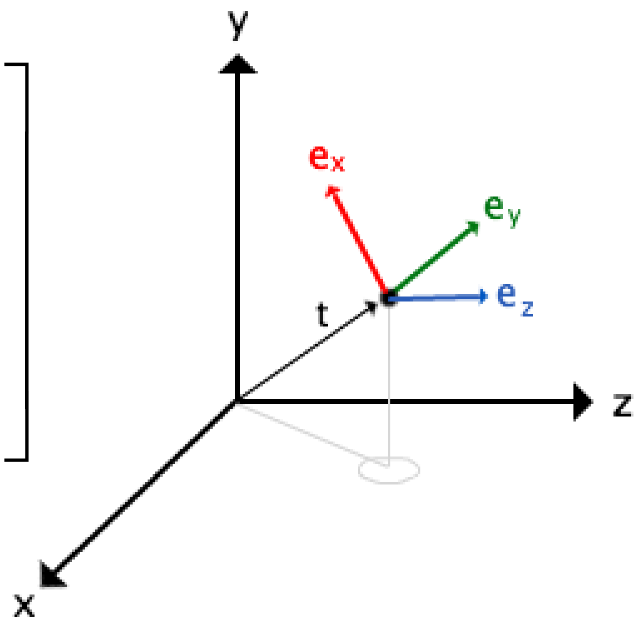
Siciliano et al, Robotics: Modelling, Planing and Control

$$\boldsymbol{p}^0 = \boldsymbol{o}_1^0 + \boldsymbol{R}_1^0 \boldsymbol{p}^1 \longrightarrow \tilde{\boldsymbol{p}}^0 = \boldsymbol{A}_1^0 \tilde{\boldsymbol{p}}^1$$

$$\mathbf{o}_1^0 \in \mathbb{R}^3, \mathbf{R}_1^0 \in SO(3)$$
$$\mathbf{A}_1^0 \in SE(3) = \mathbb{R}^3 \times SO(3)$$

Homogeneous Transformations

$$H = \begin{bmatrix} e_{xx} & e_{yx} & e_{zx} & t_x \\ e_{xy} & e_{yy} & e_{zy} & t_y \\ e_{xz} & e_{yz} & e_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



http://mabulous.com/wp-content/uploads/2013/10/TransformationMatrixBasisVectors.png

Inverse of Homogeneous Transformation

$$\tilde{\mathbf{p}}^1 = \mathbf{A}_0^1 \tilde{\mathbf{p}}^0 = (\mathbf{A}_1^0)^{-1} \tilde{\mathbf{p}}^0$$

$$\mathbf{A}_0^1 = \begin{bmatrix} \mathbf{R}_1^{0\,T} & -\mathbf{R}_1^{0\,T} \mathbf{o}_1^0 \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_0^1 & -\mathbf{R}_0^1 \mathbf{o}_1^0 \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

$$\mathbf{A}^{-1} \neq \mathbf{A}^T$$

Homogeneous transformation has 6 independent parameters

- 3 for rotation R (3x3-Matrix with 6 constraints)
- 3 for translation o

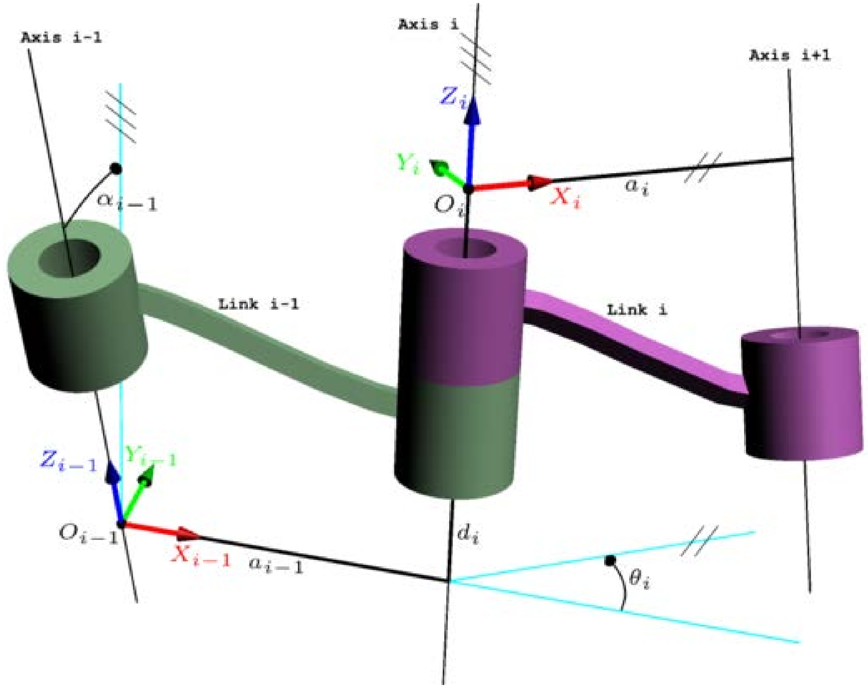
Composition of Homogeneous Transformation

$$\mathbf{A}_1^0 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{o}_1^0 \\ \mathbf{0}^T & 1 \end{bmatrix}. \quad \mathbf{A}_2^1 = \begin{bmatrix} \mathbf{R}_2^1 & \mathbf{o}_2^1 \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

$$\mathbf{A}_2^0 = \mathbf{A}_1^0 \mathbf{A}_2^1 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{o}_1^0 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_2^1 & \mathbf{o}_2^1 \\ \mathbf{0}^T & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{R}_1^0 \mathbf{R}_2^1 & \mathbf{R}_1^0 \mathbf{o}_2^1 + \mathbf{o}_1^0 \\ \mathbf{0}^T & 1 \end{bmatrix}$$

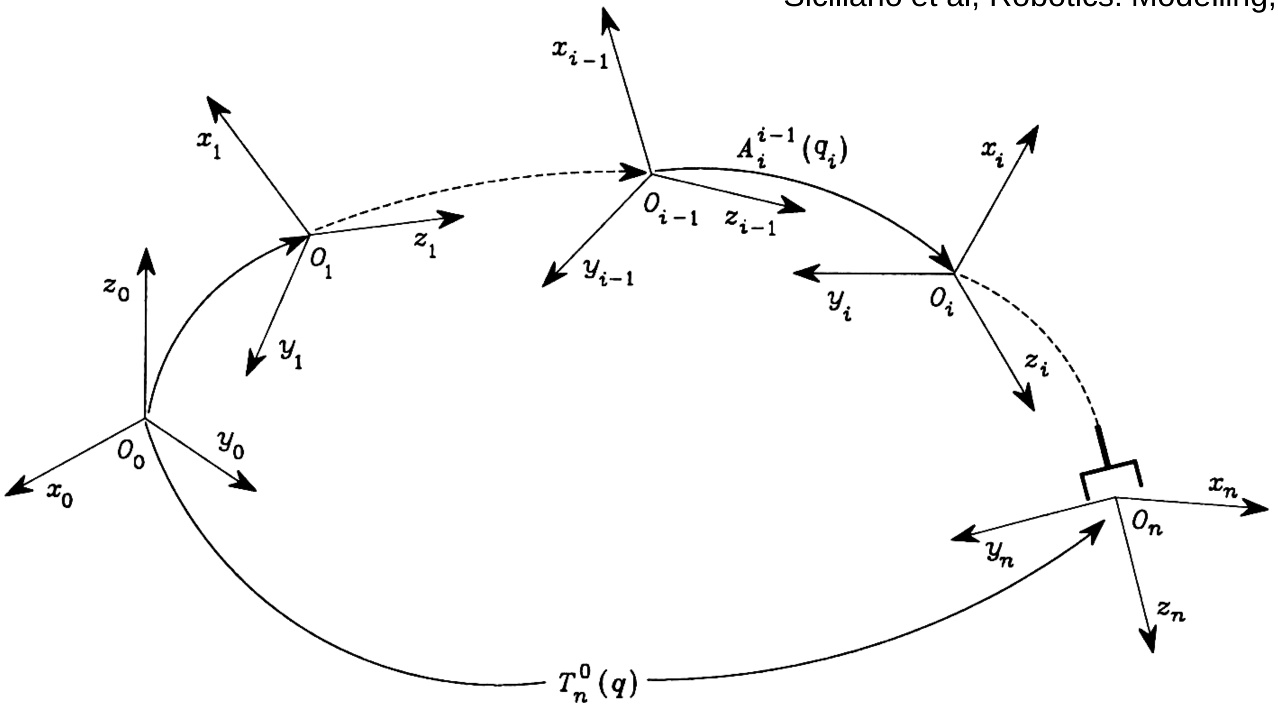
$$\tilde{\mathbf{p}}^0 = \mathbf{A}_1^0 \mathbf{A}_2^1 \tilde{\mathbf{p}}^2$$



https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/DHParameter.png/519px-DHParameter.png

Composition of Homogeneous Transformations

Siciliano et al, Robotics: Modelling, Planing and Control



Siciliano et al, Robotics: Modelling, Planing and Control

$$\tilde{\mathbf{p}}^0 = \mathbf{A}_1^0 \mathbf{A}_2^1 \ldots \mathbf{A}_n^{n-1} \tilde{\mathbf{p}}^n$$

Spatial Transformations Robotics System Toolbox

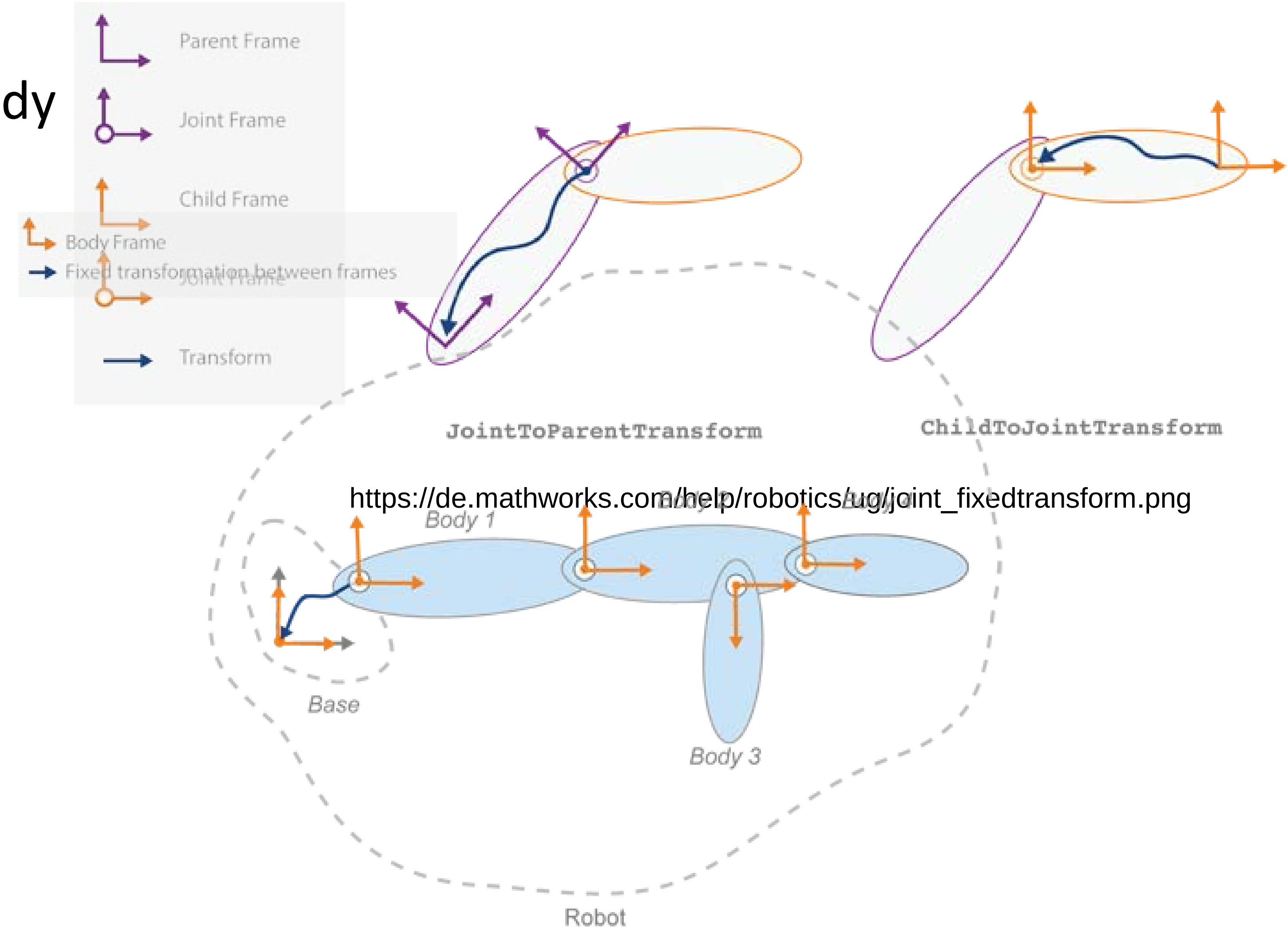
- axang
- eul
- quat
- rotm
- tform
- trvec

- axang2quat
- trvec2tform
- <a>2<b>

Converting To / Converting From	Axis-Angle (axang)	Euler Angles (eul)	Quaternion (quat)	Rotation Matrix (rotm)	Homogeneous Transformation (tform)	Translation Vector (trvec)
Axis-Angle (axang)						
Euler Angles (eul)						
Quaternion (quat)						
Rotation Matrix (rotm)						
Homogeneous Transformation (tform)						
Translation Vector (trvec)						

Rigid Body Tree

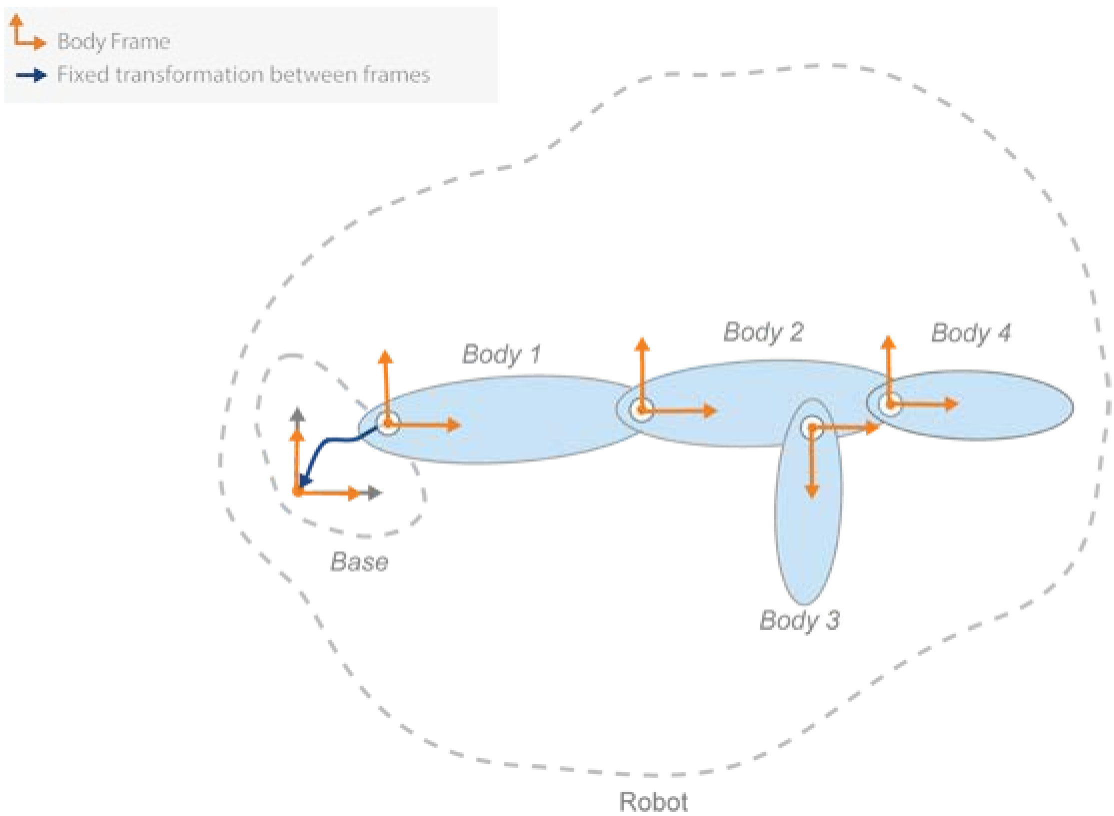
- Base
- Rigid Body
- Joint



https://de.mathworks.com/help/robotics/ug/joint_fixedtransform.png

Rigid Body Tree

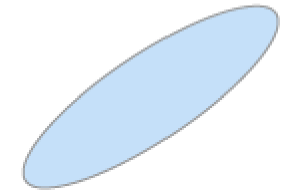
- Base
- Rigid Body
- Joint

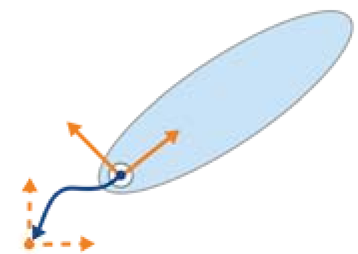


https://de.mathworks.com/help/robotics/ug/rigidbodytree_homeconfig.png

Rigid Body Tree

```
body1 = robotics.RigidBody('body1');
```

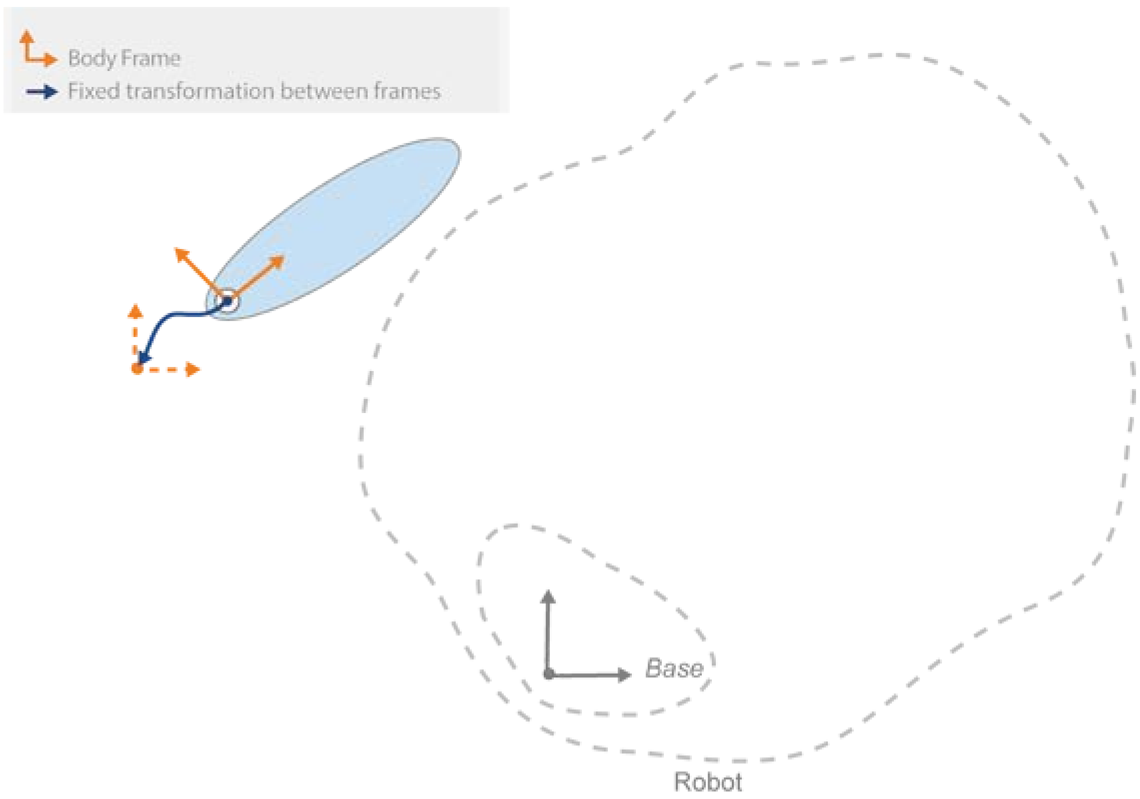




```
jnt1 = robotics.Joint('jnt1','revolute');
jnt1.HomePosition = pi/4;
tform = trvec2tform([0.25, 0.25, 0]); % User defined
setFixedTransform(jnt1,tform);
body1.Joint = jnt1;
```

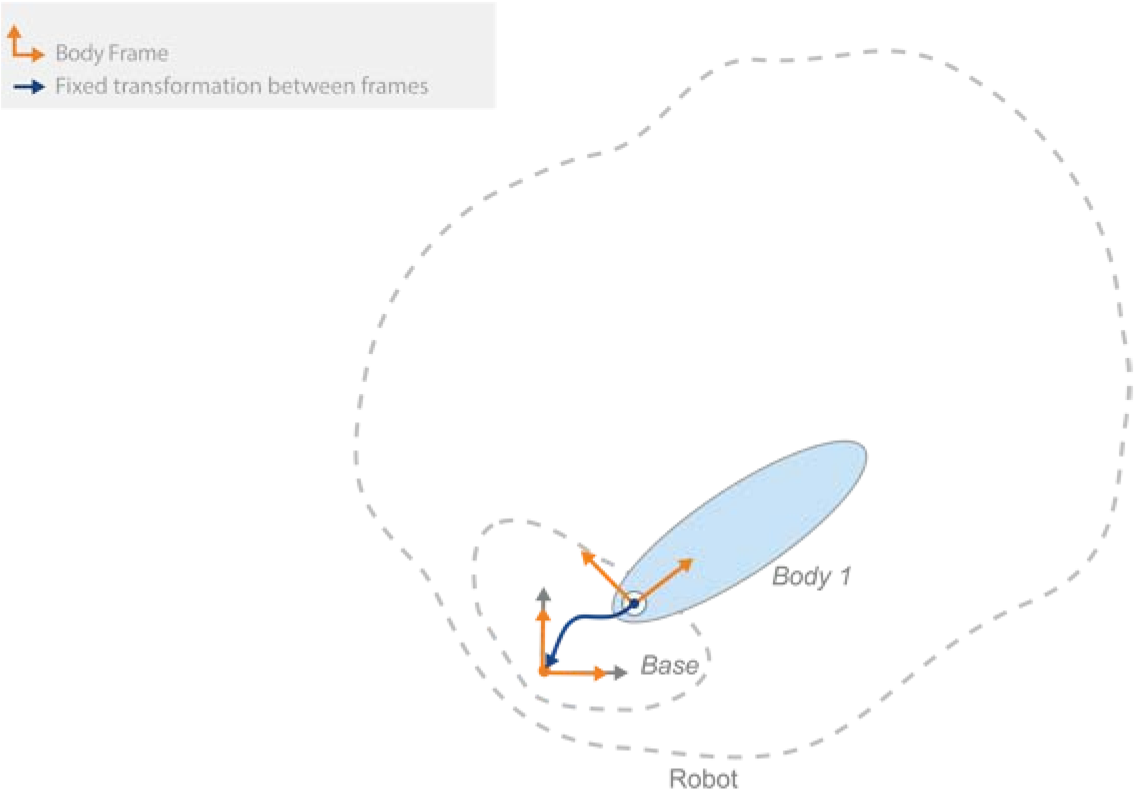
Rigid Body Tree

robot = robotics.RigidBodyTree;



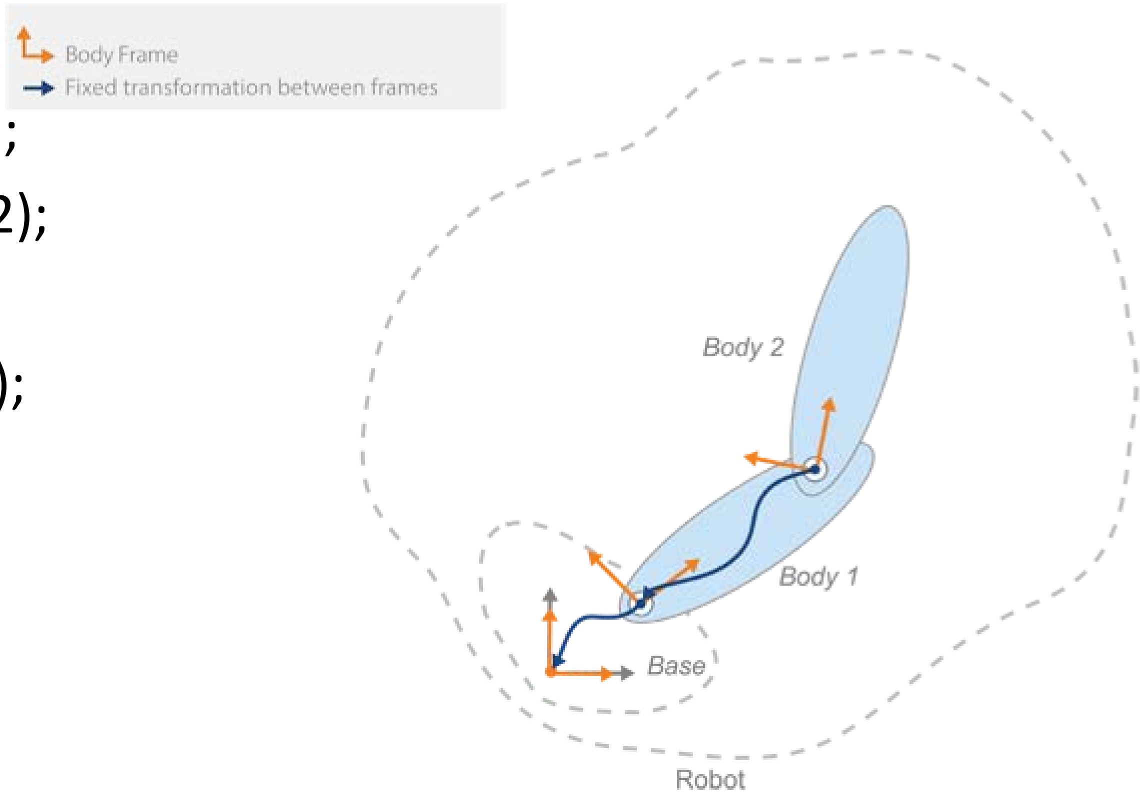
Rigid Body Tree

addBody(robot,body1,'base')



Rigid Body Tree

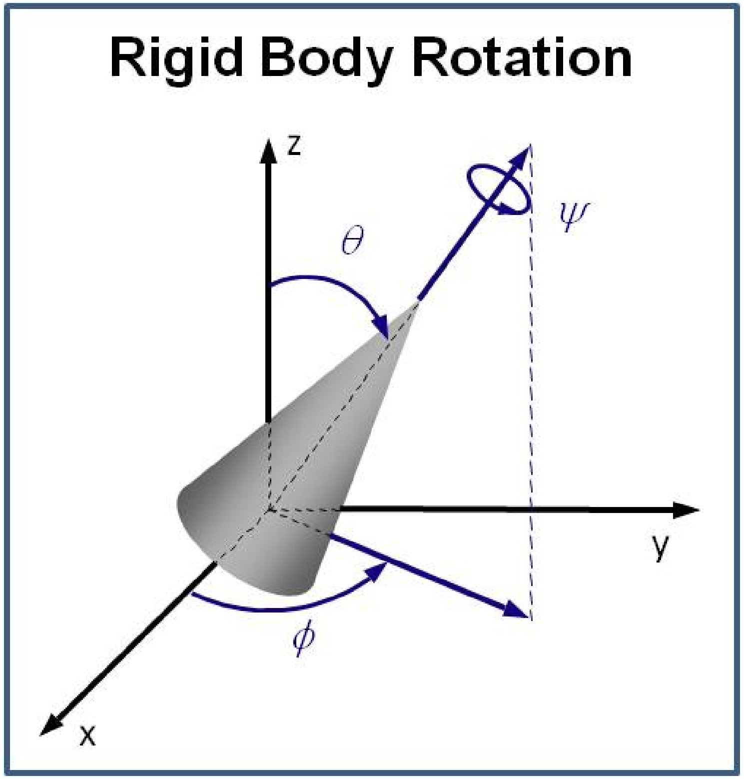
```
body2 = robotics.RigidBody('body2');
jnt2 = robotics.Joint('jnt2','revolute');
jnt2.HomePosition = pi/6;
tform2 = trvec2tform([1, 0, 0]);
setFixedTransform(jnt2,tform2);
body2.Joint = jnt2;
addBody(robot,body2,'body1');
```



Quaternion Class

- https://de.mathworks.com/matlabcentral/fileexchange/33341-quaternion-m

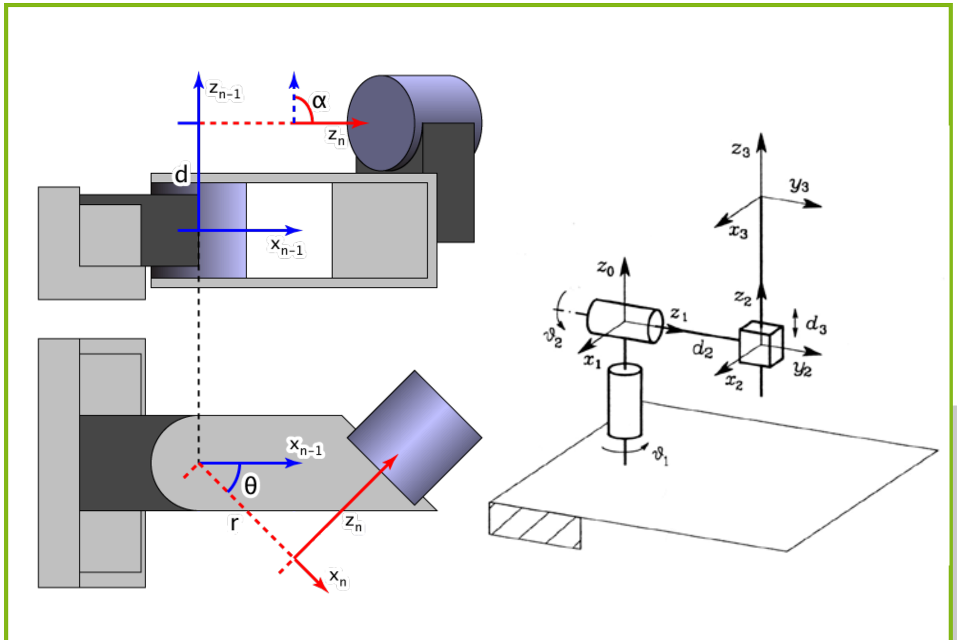
- vp = RotateVectorQ(q,v,dim);
- qc = conj(q);
- qe = exp(q);
- ql = log(q);
- q3 = mtimes(q1,q2);
- qp = power(q,p);
- qi = inverse(q) ;
- qi = interp1(t,q,ti,method);

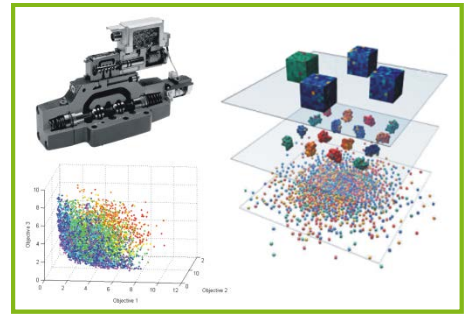


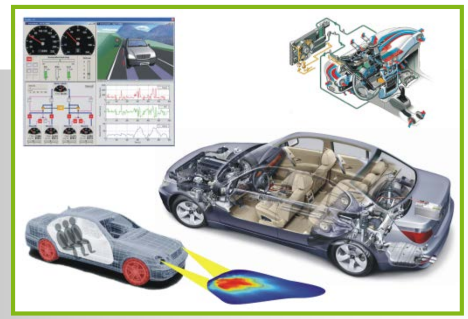
https://de.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/33341/versions/8/screenshot.jpg

Recommended Literature

- Robotics Modelling, Planning and Control, Chapter Kinematics, sections 2.1-2.7







What is next?
Kinematics

Univ.-Prof. Dr.-Ing. Prof. h.c. Torsten Bertram
Lehrstuhl für Regelungssystemtechnik