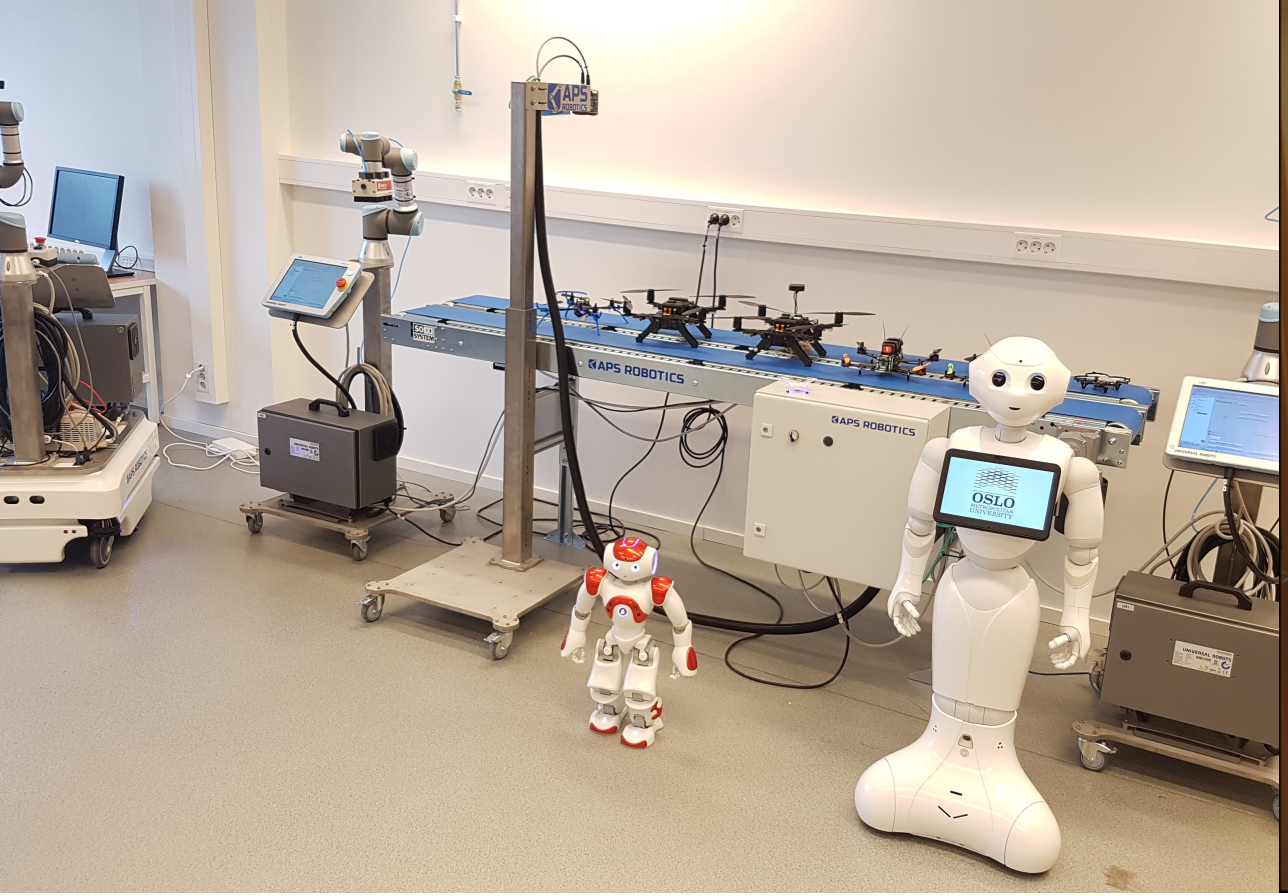
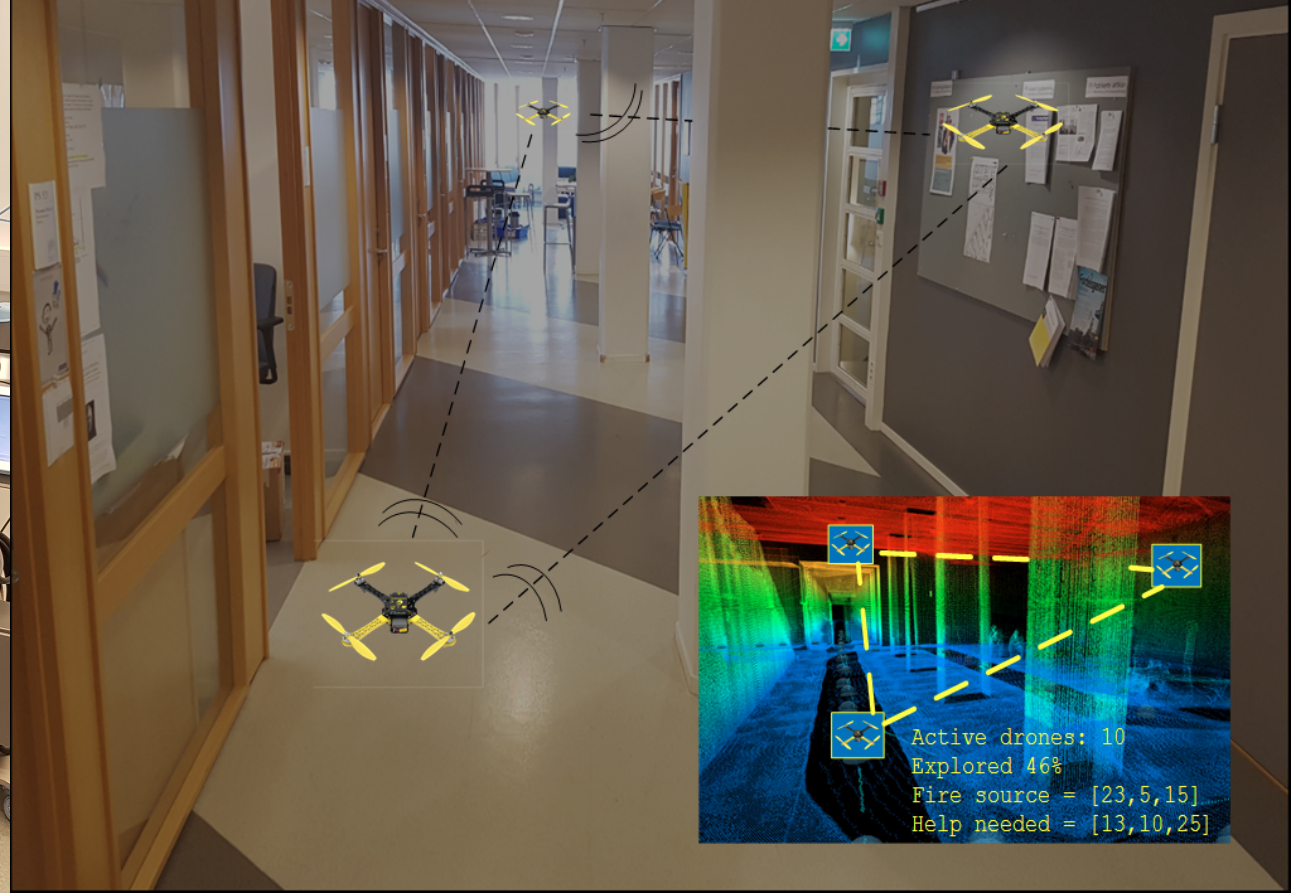




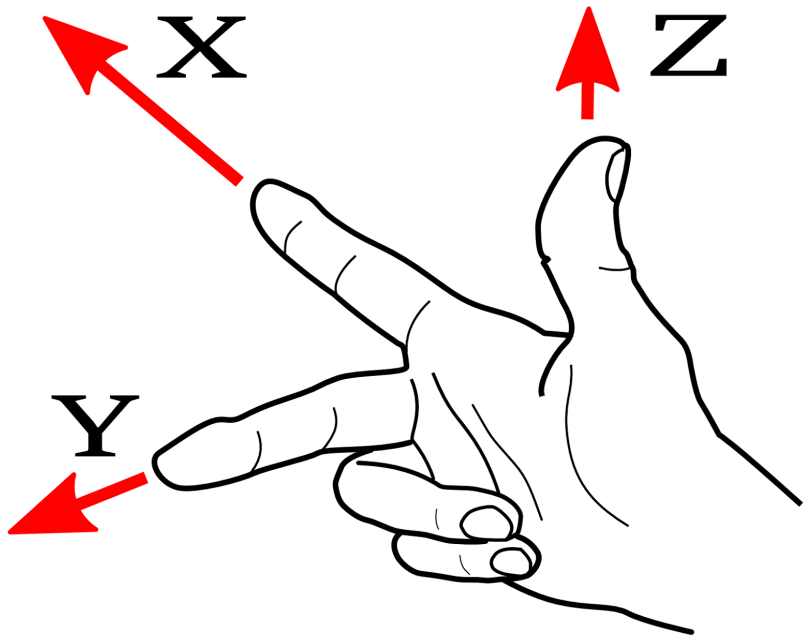
Introduction to Robotics
Representing pose in 3D

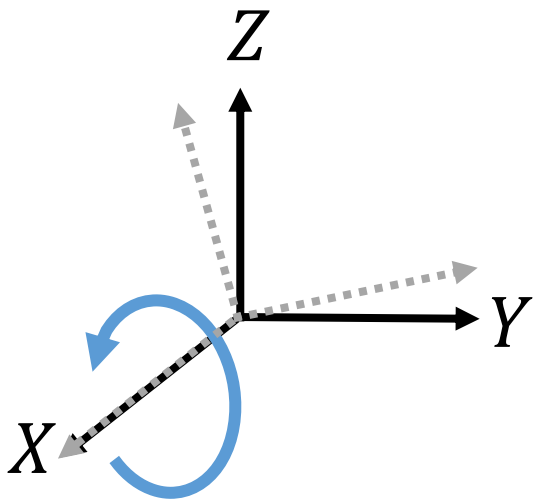
OsloMet ELVE3610 Robotteknikk

Outline

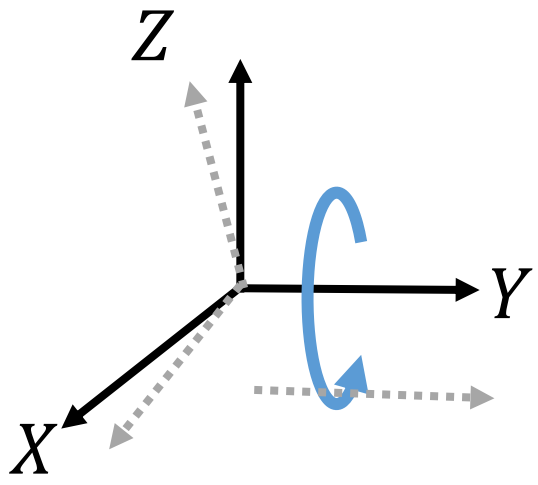
- Representing rotation in 3D
- Rotation matrices
- Euler angles
- Angle-axis representations
- Unit quaternions
- 3D Pose and homogeneous transformations

Right-handed coordinate frame

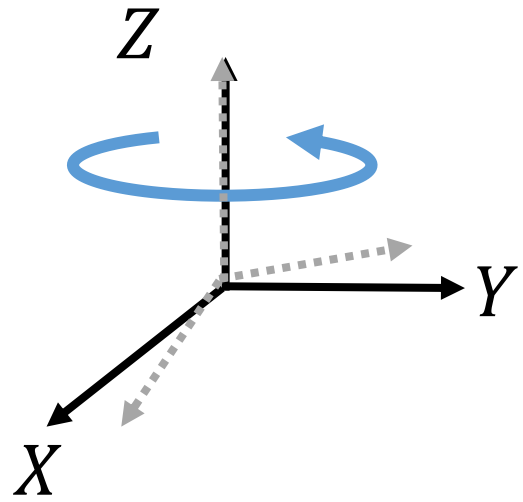




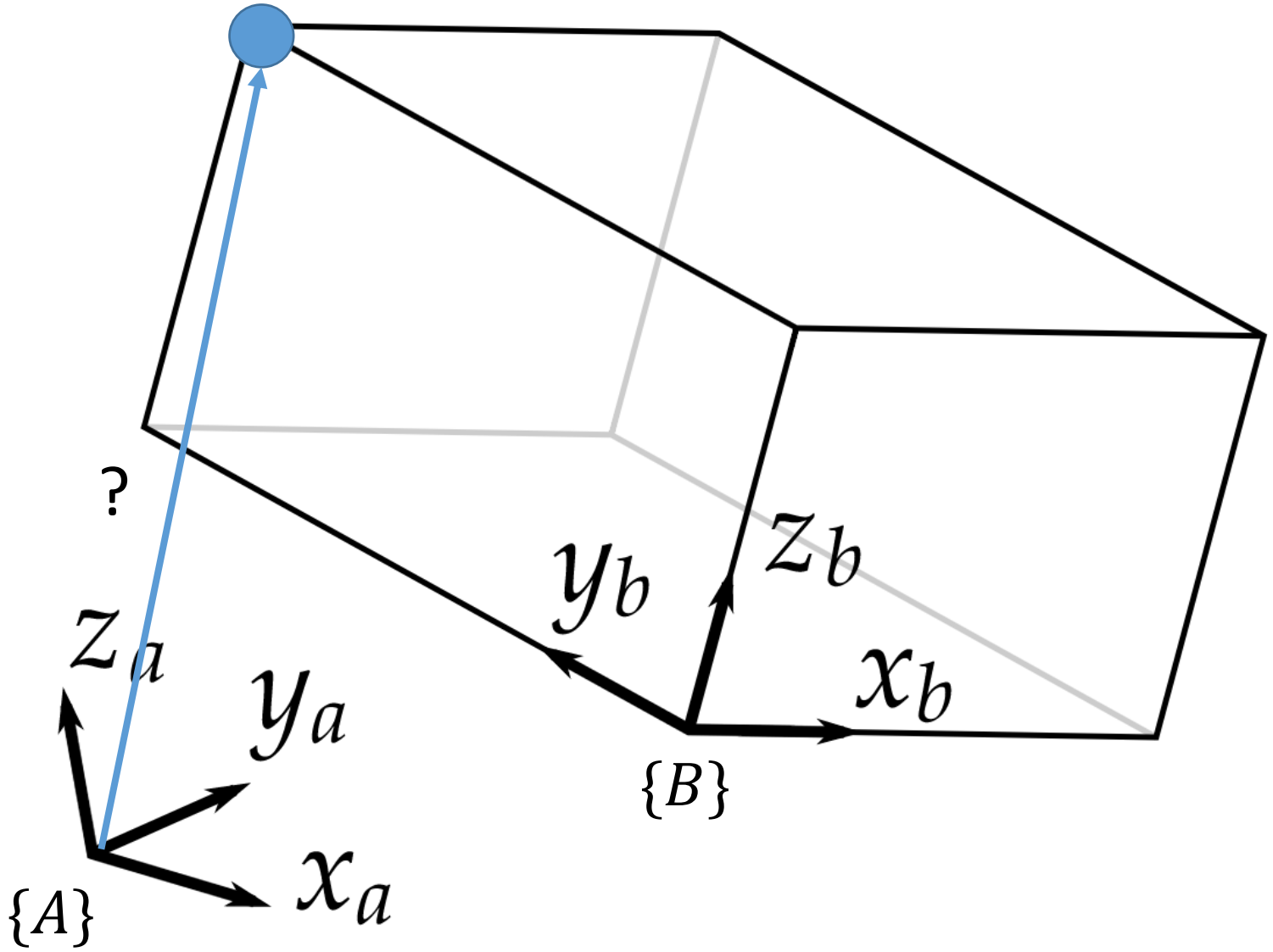
Positive rotation about X-axis

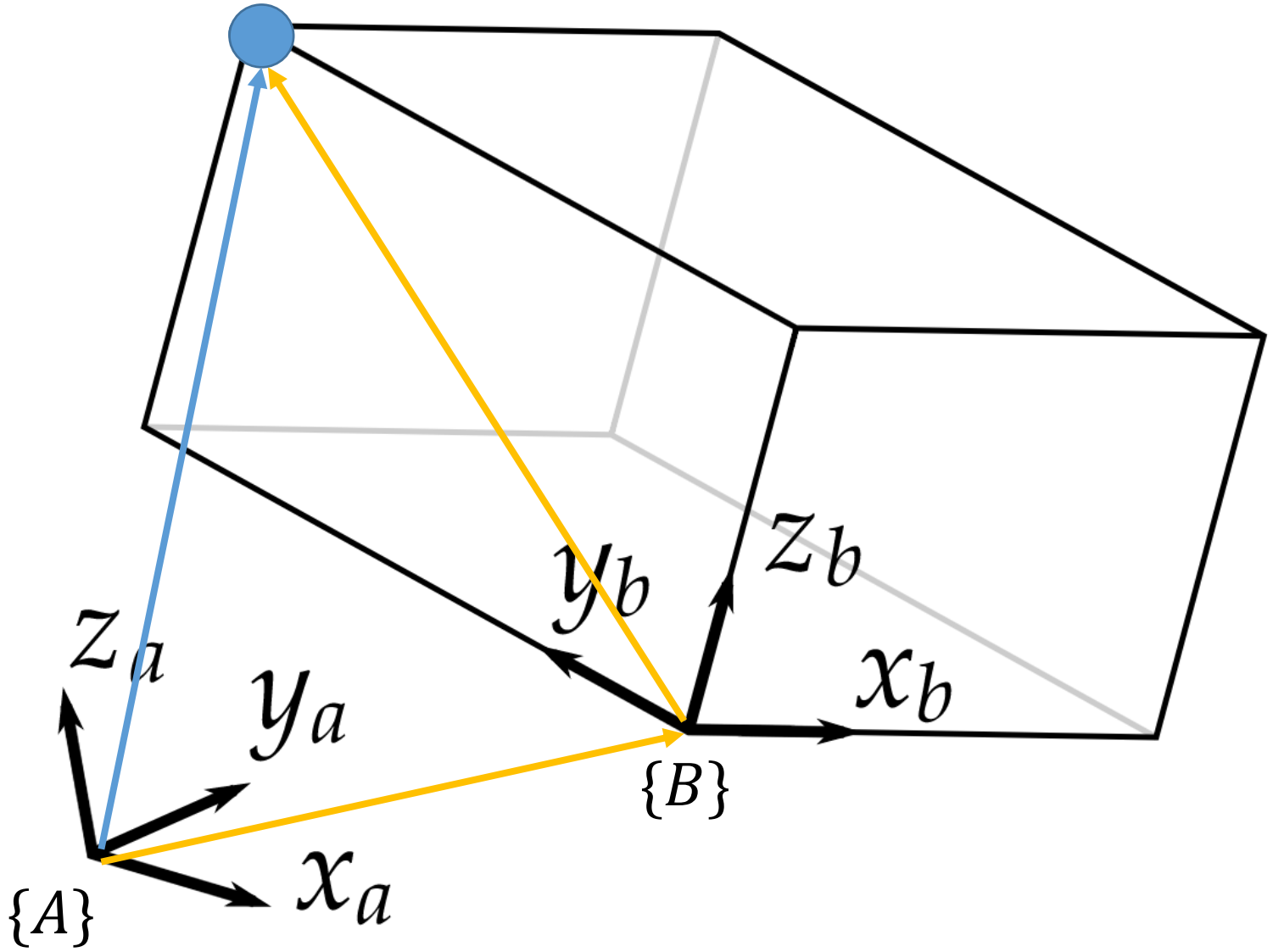


Positive rotation about Y-axis



Positive rotation about Z-axis


?
z_a
y_a
x_a
$\{A\}$
y_b
z_b
x_b
$\{B\}$


z_a
y_a
x_a
$\{A\}$
y_b
z_b
x_b
$\{B\}$

3D Transformation

Reference frame

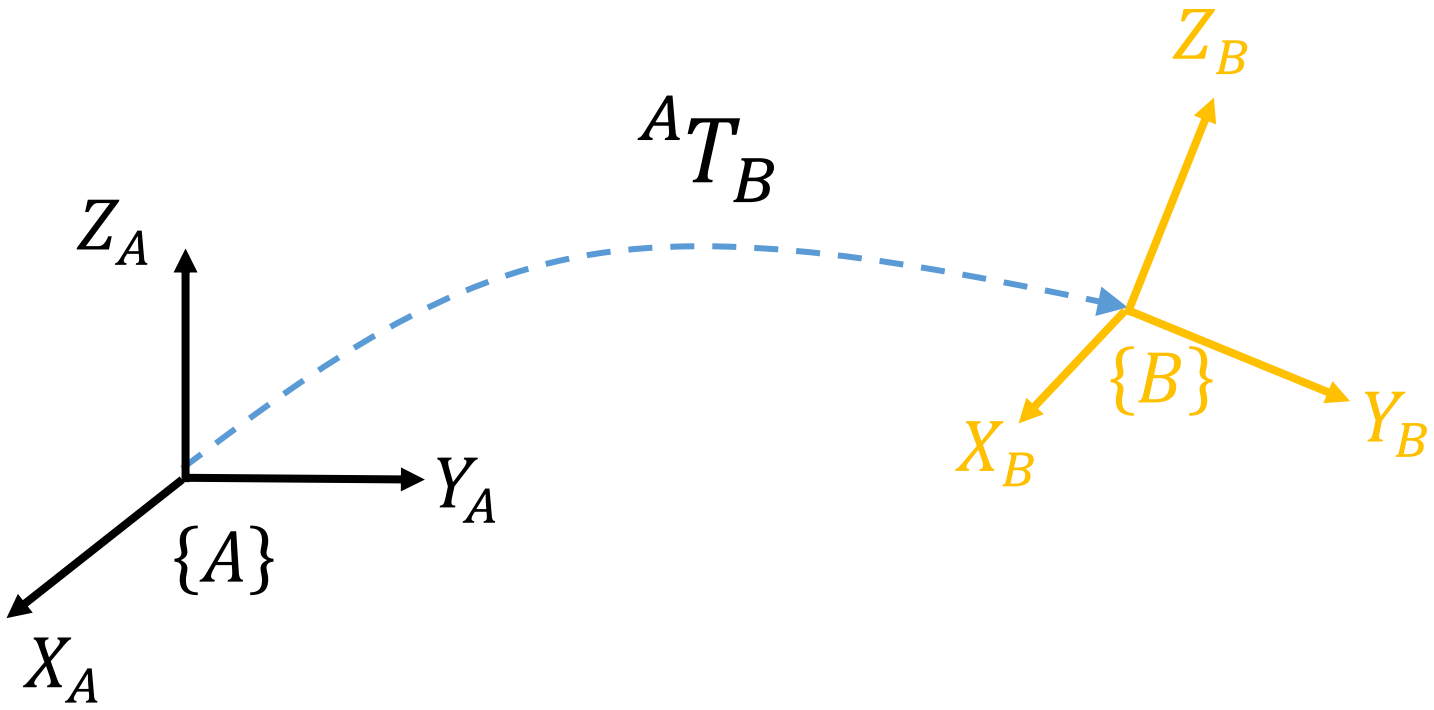
$${}^{A}T_{B}$$

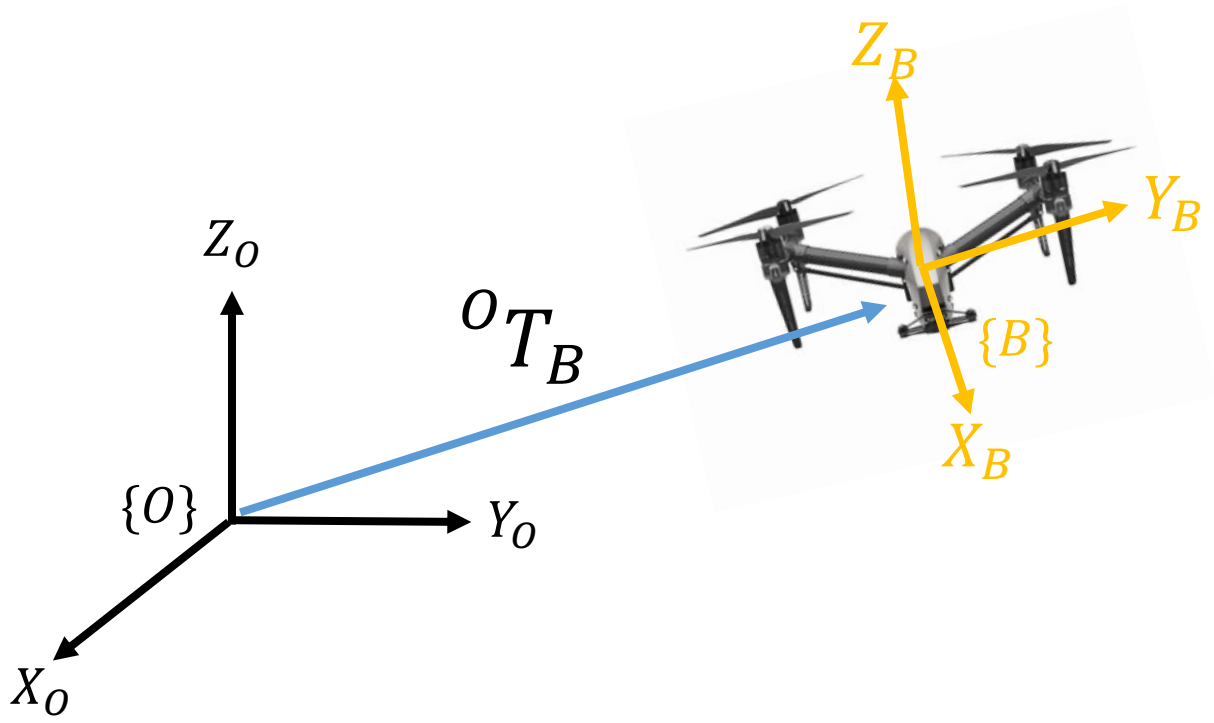
Frame being described

$\Longleftrightarrow$

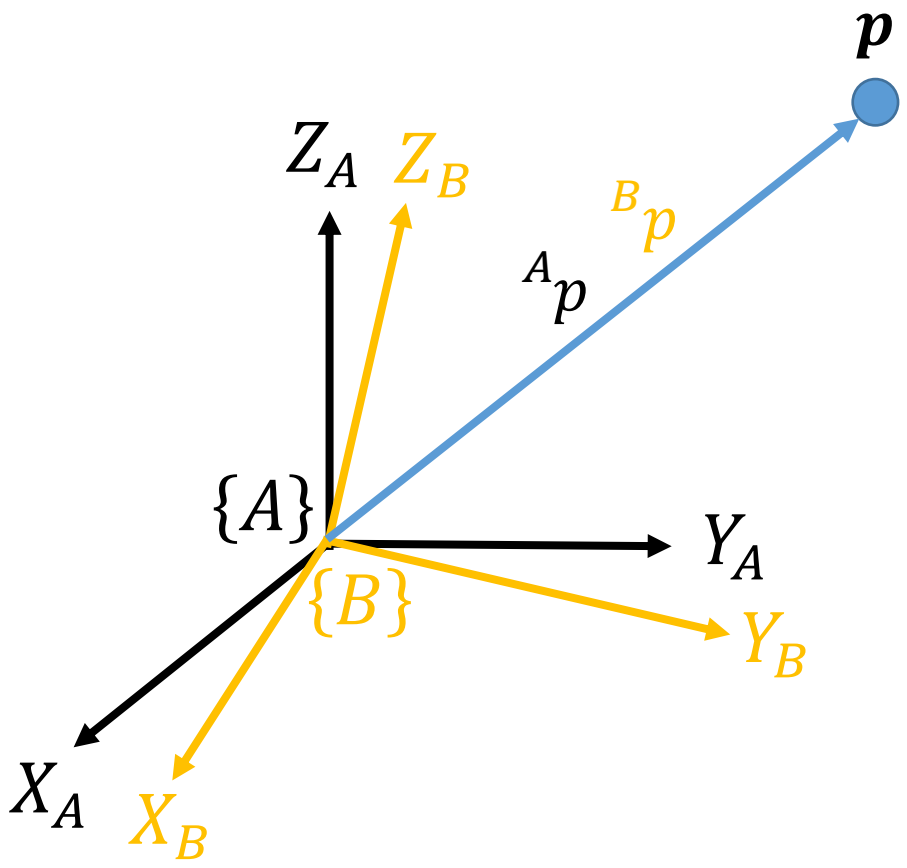
Pose of $\{B\}$ with respect to $\{A\}$

Pose transform from $\{A\}$ to $\{B\}$




Z_B
Y_B
$\{B\}$
X_B
Z_O
${}^{O}T_B$
$\{O\}$
Y_O
X_O

3D rotation matrices



$$ {}^B p = {}^B R_A \, {}^A p $$

$$ \begin{bmatrix} {}^B p_x \\ {}^B p_y \\ {}^B p_z \end{bmatrix} = {}^B R_A \begin{bmatrix} {}^A p_x \\ {}^A p_y \\ {}^A p_z \end{bmatrix} $$

3D rotation matrices

$$SO(3) = \{R \in \mathbb{R}^{3\times 3} : R^T R = I_3 \; ; \; \det(R) = +1\}$$

$$R = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} = [\boldsymbol{a} \quad \boldsymbol{b} \quad \boldsymbol{c}] \in \mathbb{R}^{3\times 3}$$

$\boldsymbol{a}^T\boldsymbol{a} = \boldsymbol{b}^T\boldsymbol{b} = \boldsymbol{c}^T\boldsymbol{c} = 1$ Columns (and rows) have unit norm

$\boldsymbol{a}^T\boldsymbol{b} = \boldsymbol{b}^T\boldsymbol{c} = \boldsymbol{c}^T\boldsymbol{a} = 0$ Columns (and rows) Orthongonal

$R^{-1} = R^T$ $\det(R) = +1$

Exercise: Verify that columns/rows have unit norm and orthogonal

$$R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Exercise: Make a skecth in paper to represent the rotation matrix. What is the new frame with respect to the original frame?

Exercise: Verify that columns/rows have unit norm and orthogonal

$$R_1 = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.707 & -0.707 & 0 \\ 0.707 & 0.707 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise: Make a skecth in paper to represent the rotation matrix. What is the new frame with respect to the original frame?

Exercise: are the following rotation matrices?

$$R^T R = I$$

$$\det(R) = +1$$

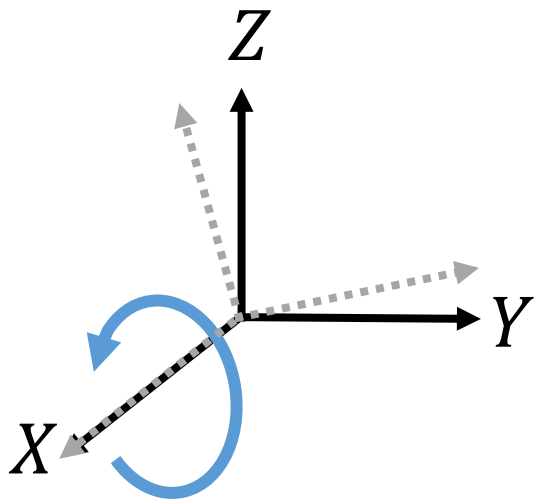
$$R_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad R_2 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$R_3 = \begin{bmatrix} -0.866 & -0.5 & 0 \\ 0.5 & -0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_4 = \begin{bmatrix} 1.692 & -0.2094 & 0.6908 \\ 0.5431 & 0.7815 & -0.3072 \\ -0.4755 & 0.5878 & 0.6545 \end{bmatrix}$$

$$R_5 = \begin{bmatrix} 0.692 & -0.2094 & 0.6908 \\ 0.5431 & 0.7815 & -0.3072 \\ -0.4755 & 0 & 0.6545 \end{bmatrix}$$

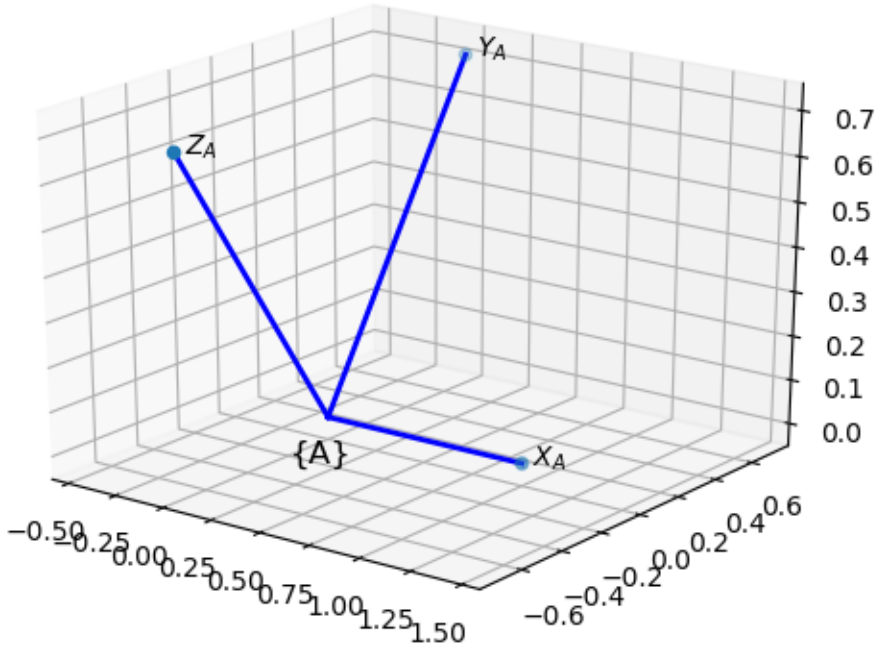
```
# python
R = np.array([[-1,0,0],
              [0,0,1],
              [0,1,0]])
# check determinant is +1
np.linalg.det(R)
# check R^T*R = I
R.T.dot(R)
```

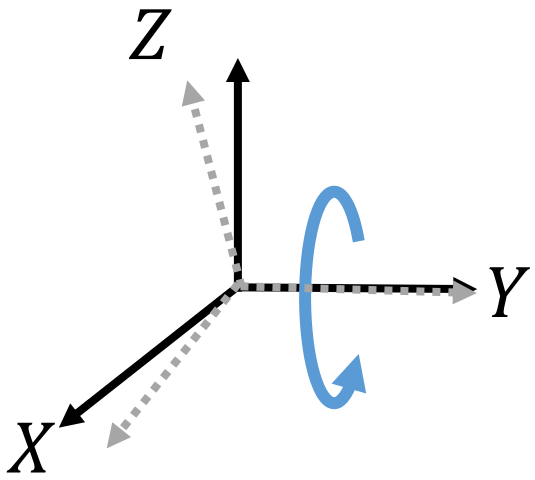


Positive rotation about X-axis

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

```
import numpy as np
import matplotlib.pyplot as plt
import robotteknikk as rob

theta = np.pi/4
R = np.array([[1, 0, 0],
              [0,np.cos(theta),-np.sin(theta)],
              [0,np.sin(theta),np.cos(theta)]])

T = rob.trotmat(R)
T = rob.trotx(theta) #same result

fig = plt.figure()
ax = plt.axes(projection='3d')
rob.trplot3(ax,T,name="A",color="b")
ax.axis('equal')
```

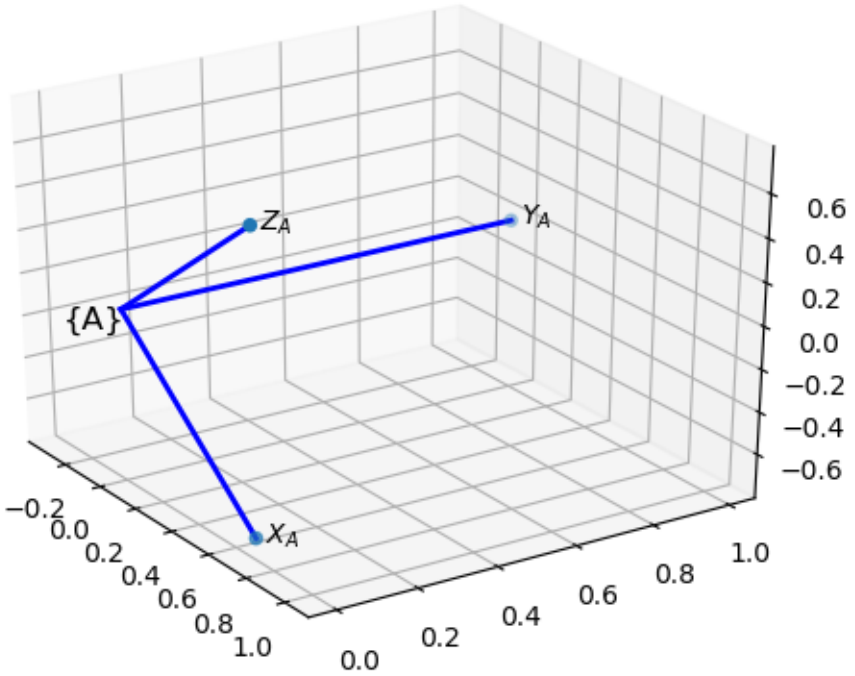


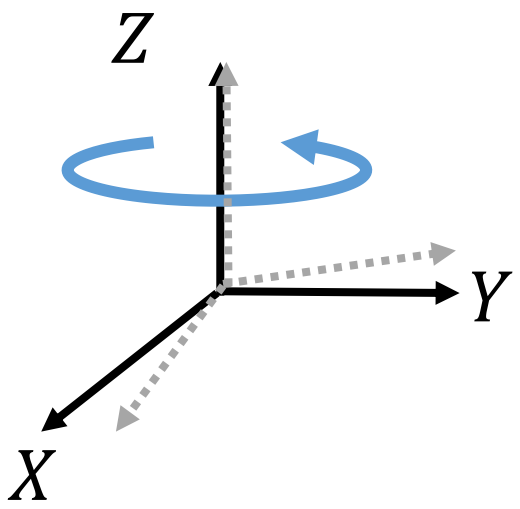


Positive rotation about Y-axis

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

```
theta = np.pi/4
T = rob.troty(theta)
fig = plt.figure(2)
ax = plt.axes(projection='3d')
rob.trplot3(ax,T,name="A",color="b")
ax.axis('equal')
```

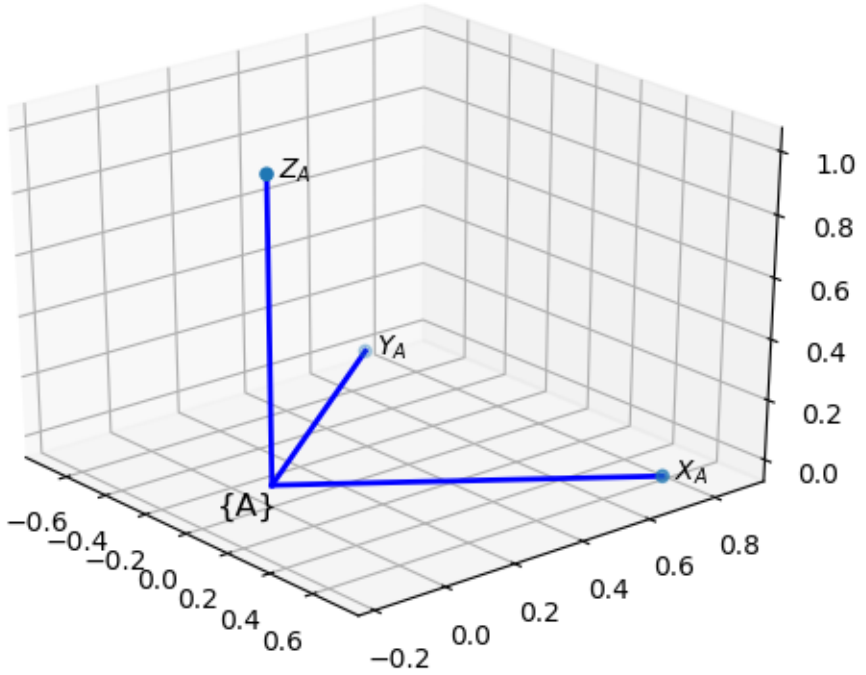




Positive rotation about Z-axis

```
theta = np.pi/4
T = rob.trotz(theta)
fig = plt.figure(2)
ax = plt.axes(projection='3d')
rob.trplot3(ax,T,name="A",color="b")
ax.axis('equal')
```

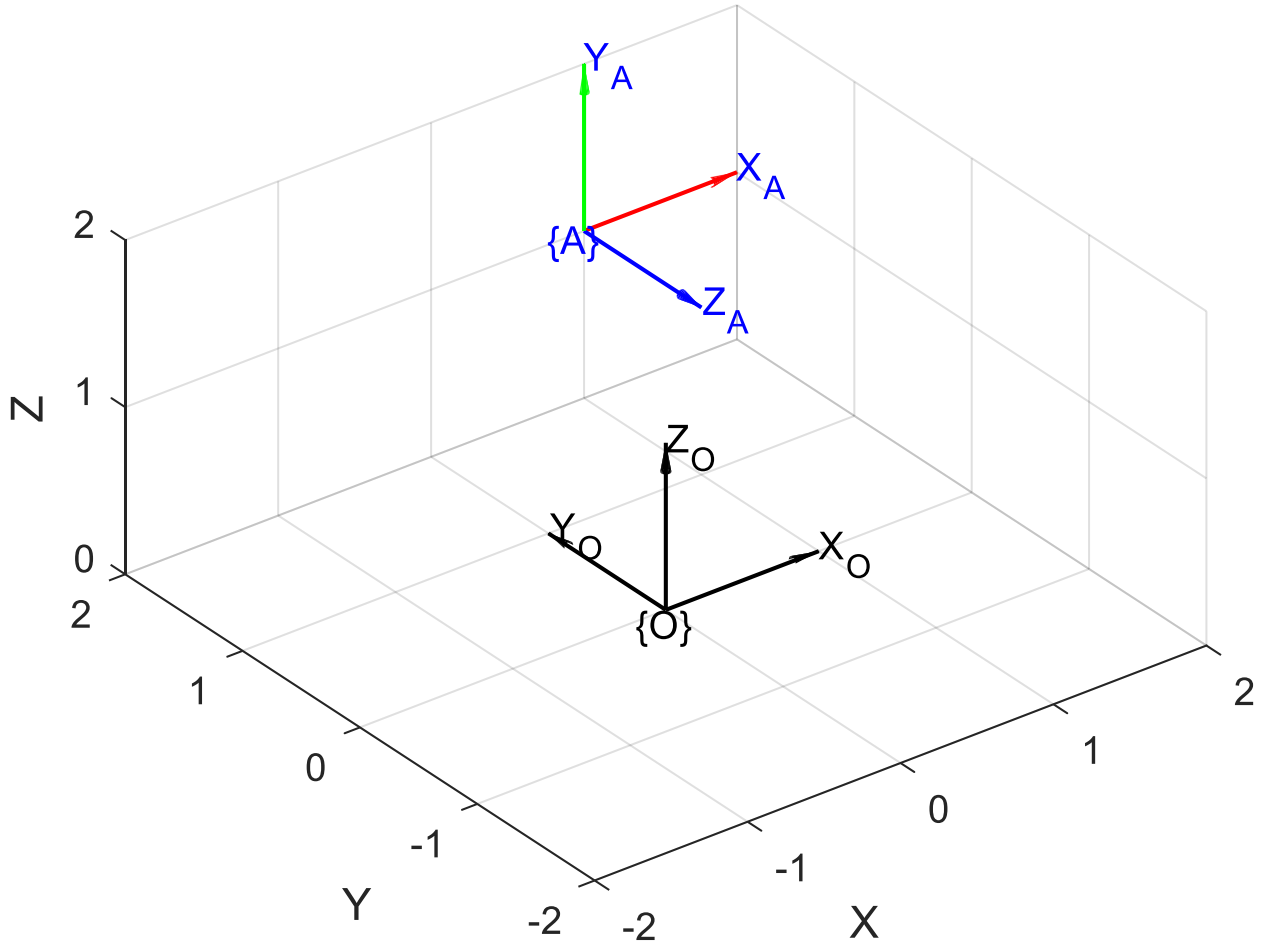
$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



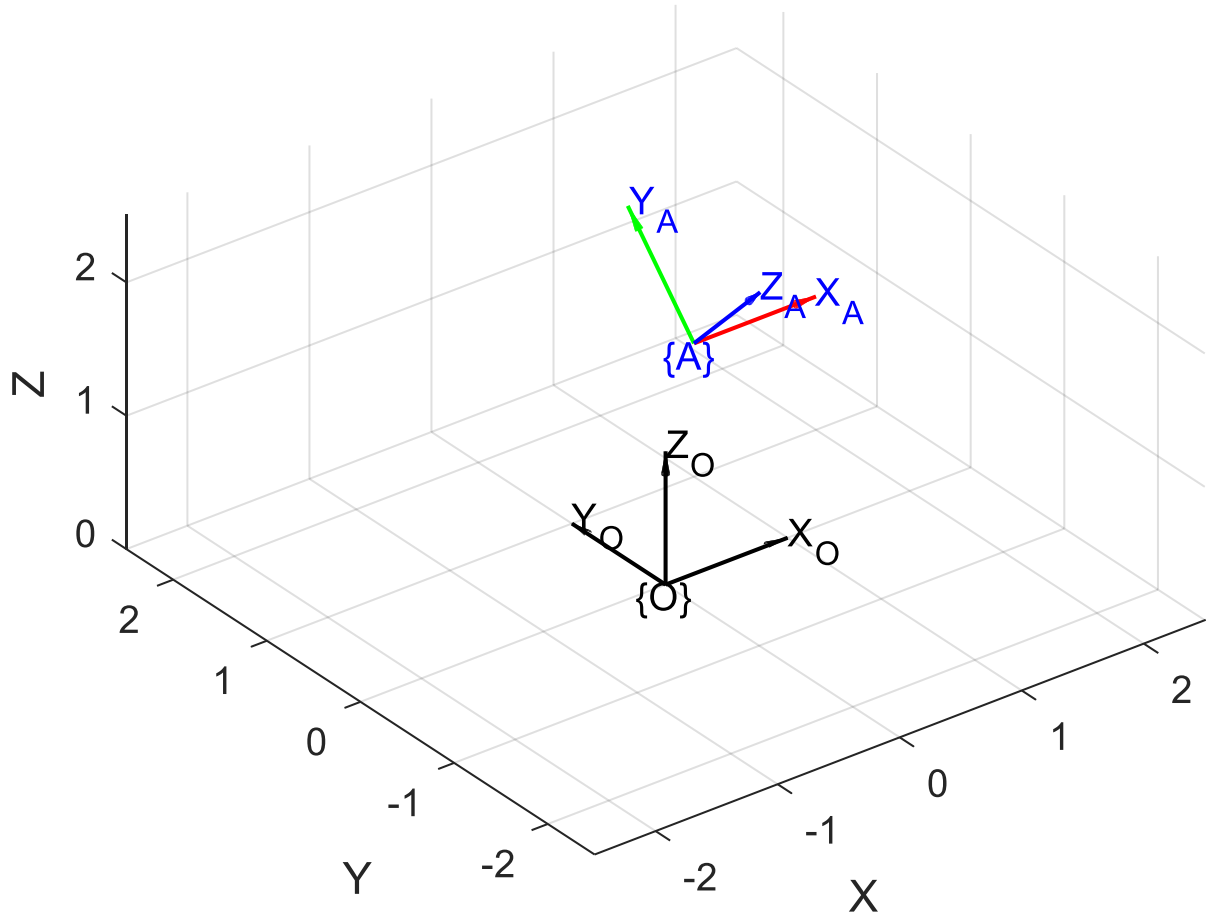
Exercise: are the following rotation matrices the same?

```
a = np.pi/2
T1 = rob.trotx(a).dot(rob.troty(a))
T2 = rob.troty(a).dot(rob.trotx(a))

fig = plt.figure()
ax = plt.axes(projection='3d')
rob.trplot3(ax,T1,name="1",color="b")
ax.axis('equal')

fig = plt.figure()
ax = plt.axes(projection='3d')
rob.trplot3(ax,T2,name="2",color="g")
ax.axis('equal')
```

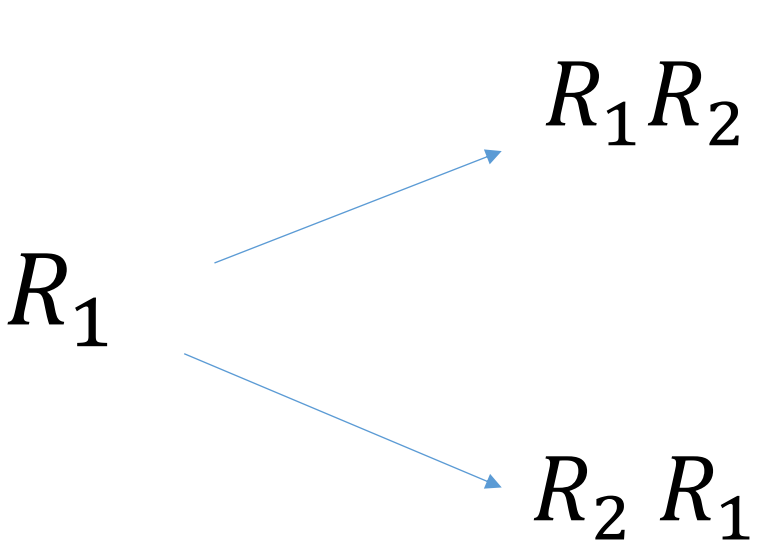
Exercise: What is the rotation from O to A?



Exercise: What is the rotation from O to A?



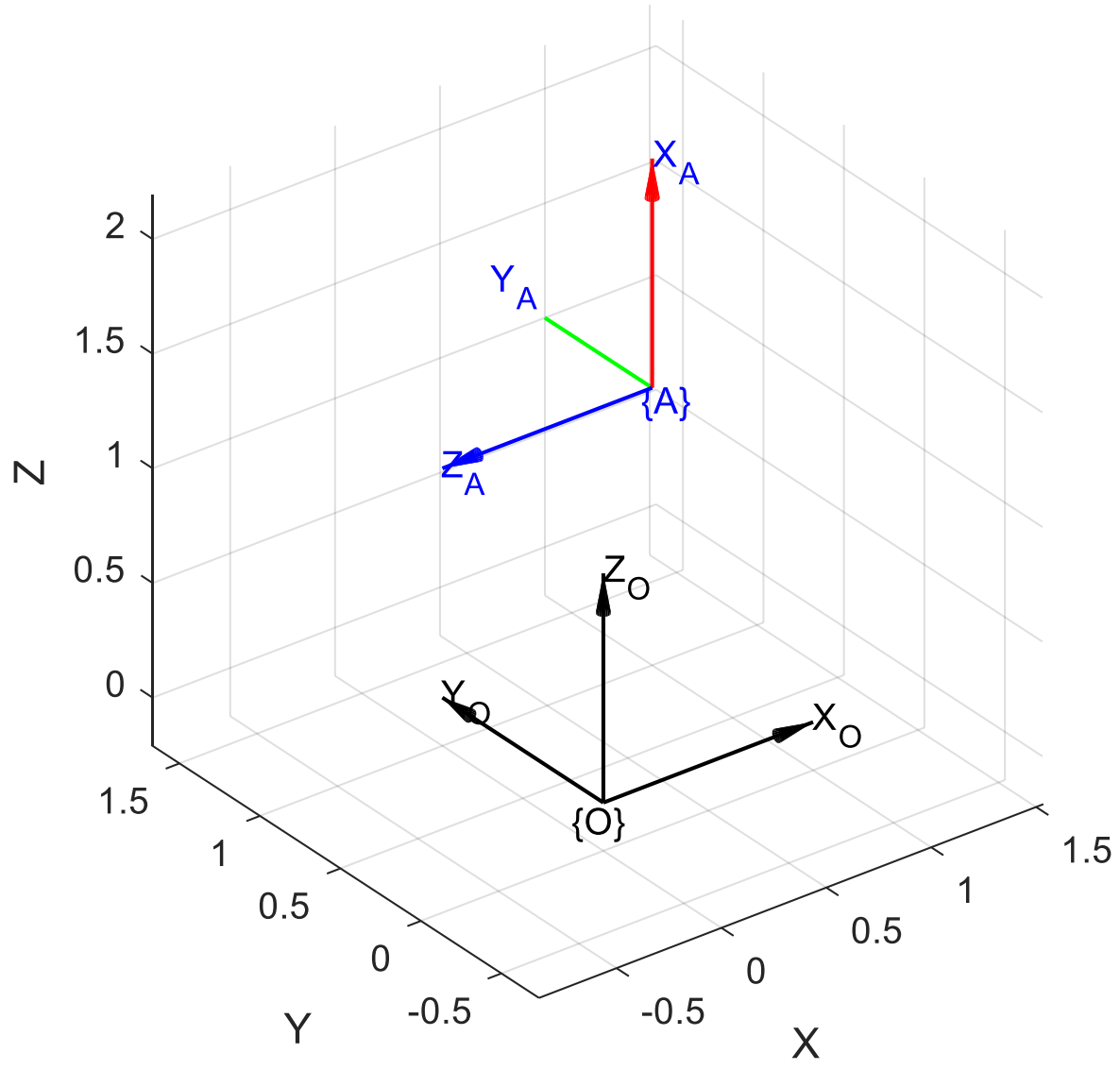
Composition of rotations



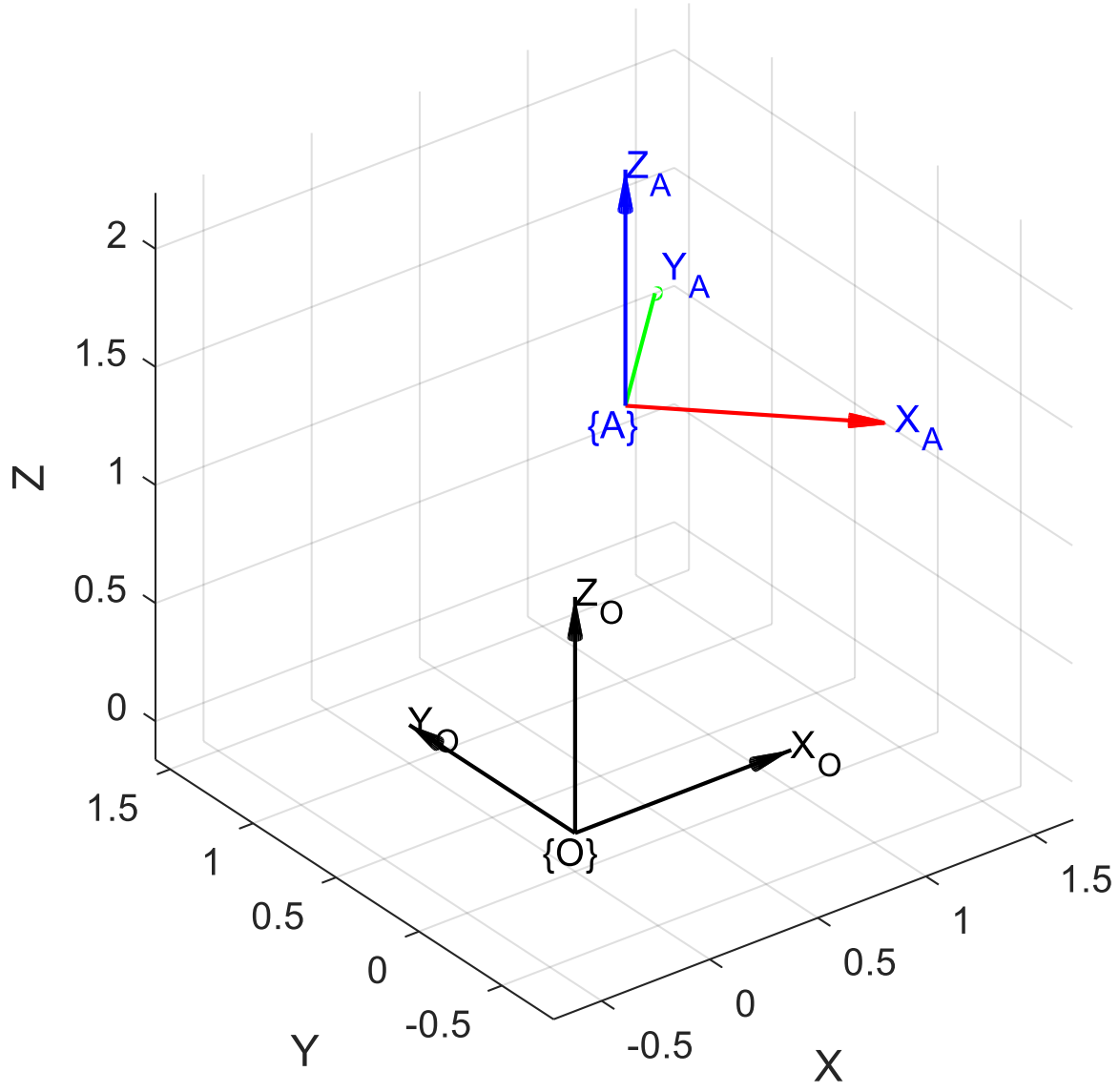
Rotation 1 followed by Rotation 2 along **new** axis

Rotation 1 followed by Rotation 2 along **old** axis

Exercise: What is the rotation from O to A?



Exercise: What is the rotation from O to A?



$$R = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} = [\boldsymbol{a} \quad \boldsymbol{b} \quad \boldsymbol{c}] \in \mathbb{R}^{3\times 3}$$ → 9 elements

$$\boldsymbol{a}^T\boldsymbol{a} = 1$$

$$\boldsymbol{b}^T\boldsymbol{b} = 1$$

$$\boldsymbol{c}^T\boldsymbol{c} = 1$$

Unit norm:
3 constraints

$$\boldsymbol{a}^T\boldsymbol{b} = 0$$

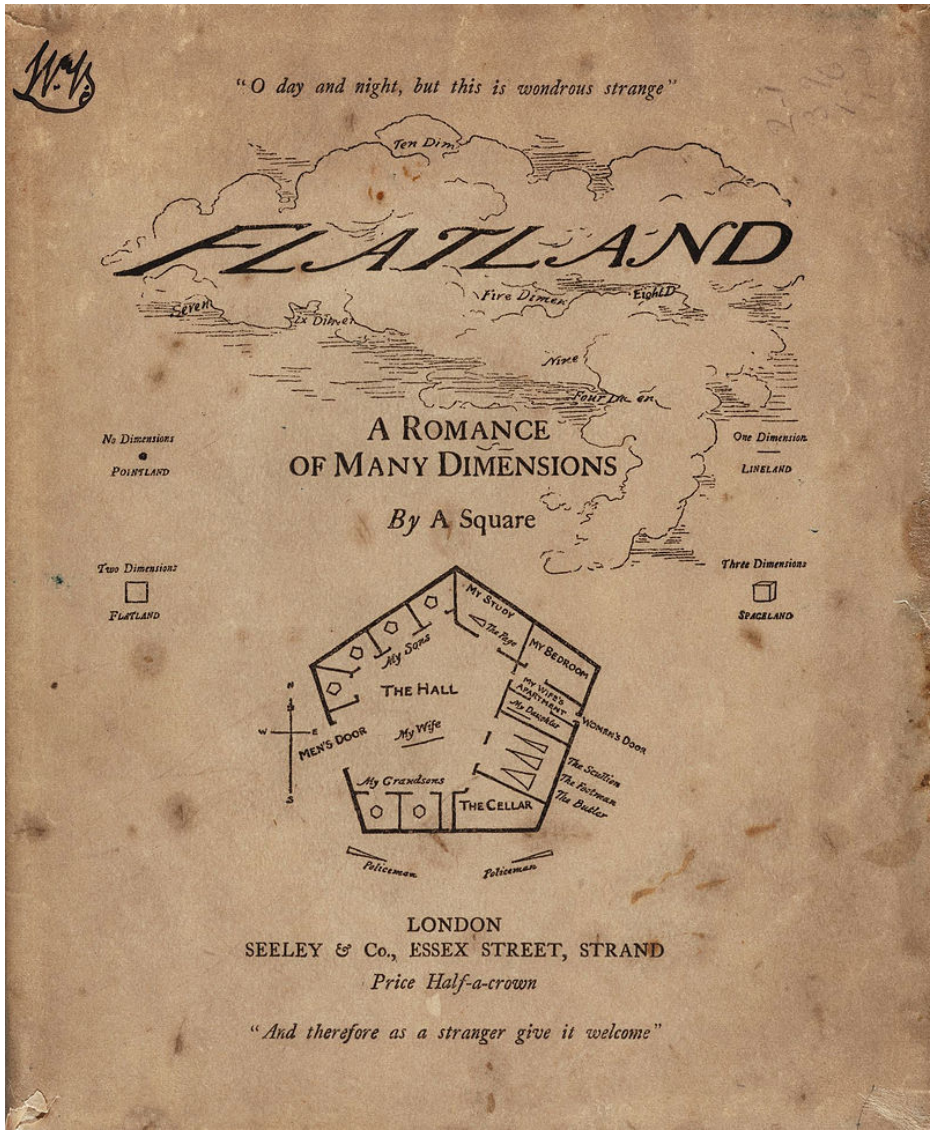
$$\boldsymbol{b}^T\boldsymbol{c} = 0$$

$$\boldsymbol{c}^T\boldsymbol{a} = 0$$

Orthogonality:
3 constraints

9 -3 -3 = 3 → 3 independent variables

9 elements and 6 constraints



Flatland: A Romance of Many Dimensions
Edwin Abbott Abbott, 1884

Three angle representations

Leonhard Euler
1707 – 1783

Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than ***three****) about coordinate axes, where no two successive rotations may be about the same axis.*
Euler's rotation theorem (Kuipers 1999).

Euler angles

XYX, XZX, YXY, YZY, ZYZ

Most common Euler angle representation

XYZ, XZY, YXZ, YZX, ZXY, ZYX

Cardan angles

Tait-Bryan angles

Roll-Pitch-Yaw

ZYZ Euler Angles

The ZYZ sequence

$$\boldsymbol{R} = \boldsymbol{R}_z(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)$$

Euler angles $\Gamma = (\phi, \theta, \psi)$

Exercise:

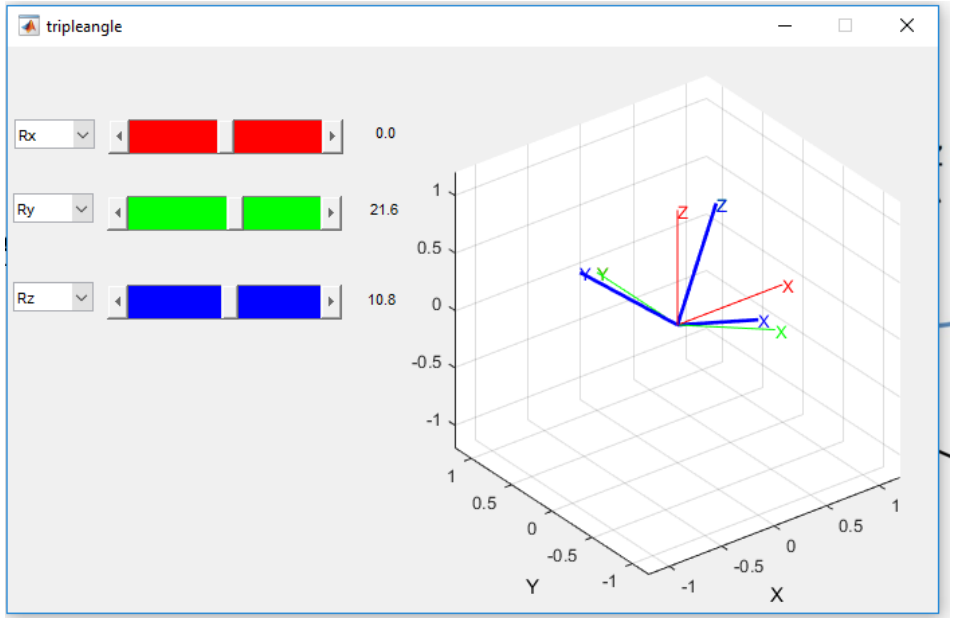
Can different rotations have the same Euler angles?

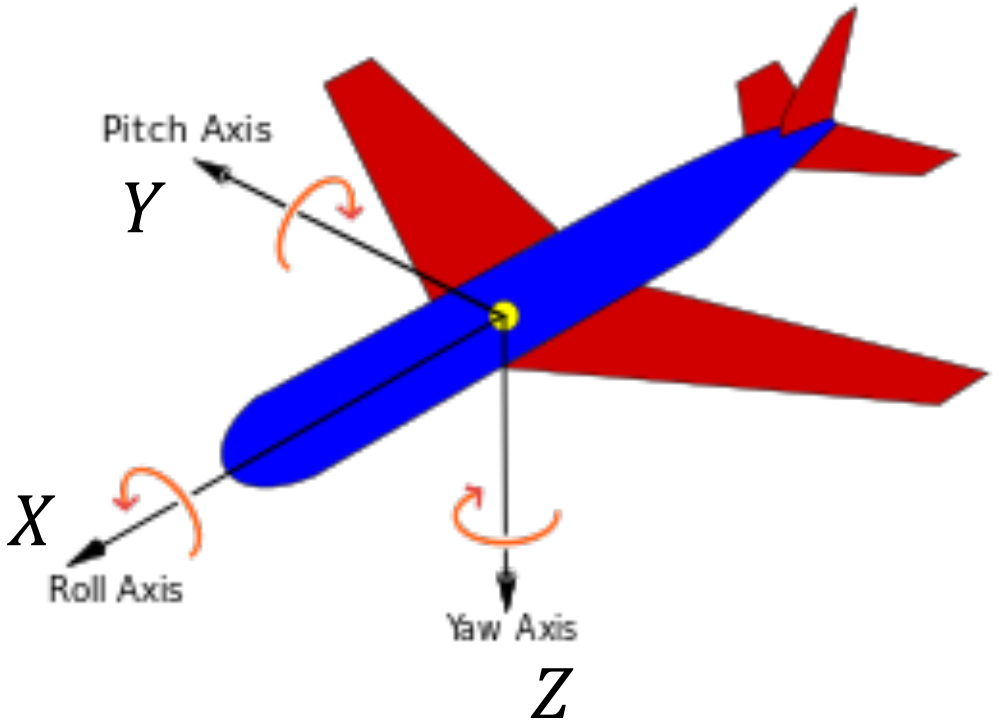
Roll-Pitch-Yaw Angles

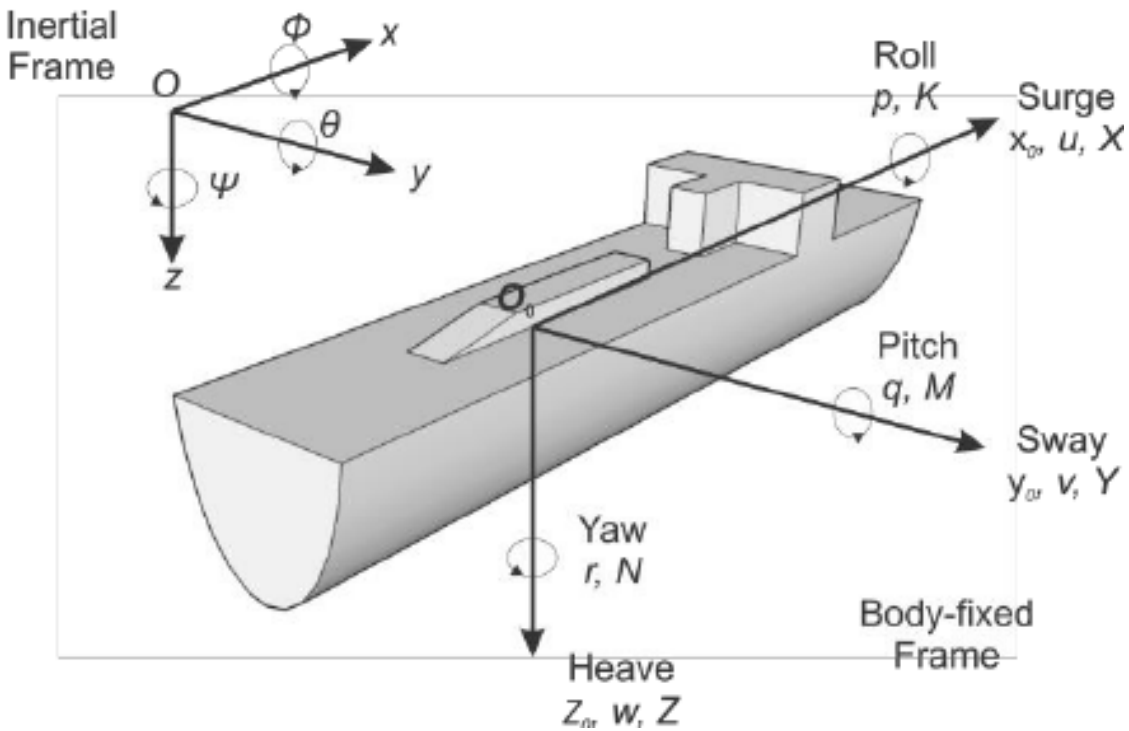
$$(\theta_r, \theta_p, \theta_y)$$

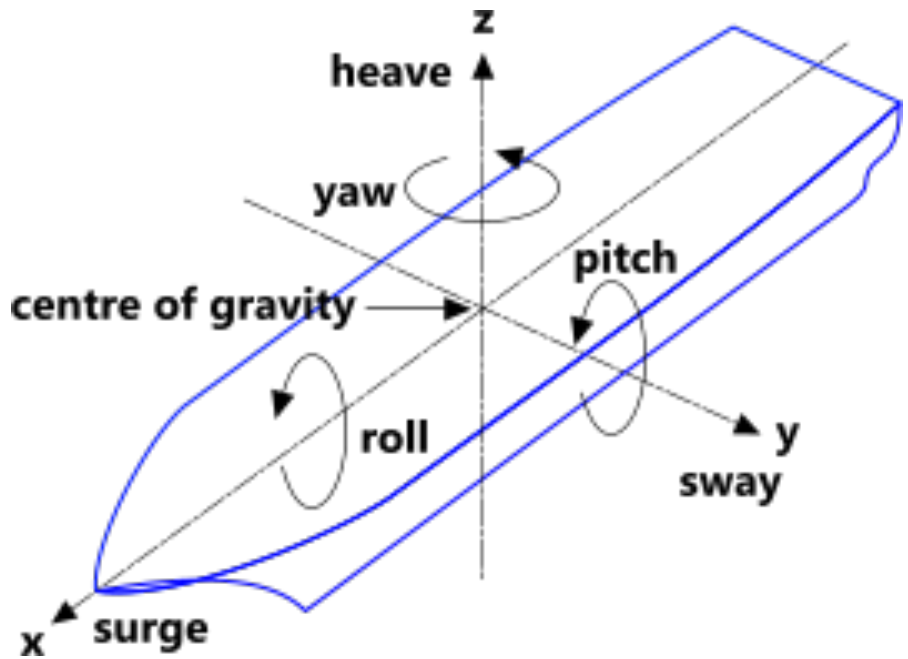
$$R = R_x(\theta_r)R_y(\theta_p)\,R_z(\theta_y)$$

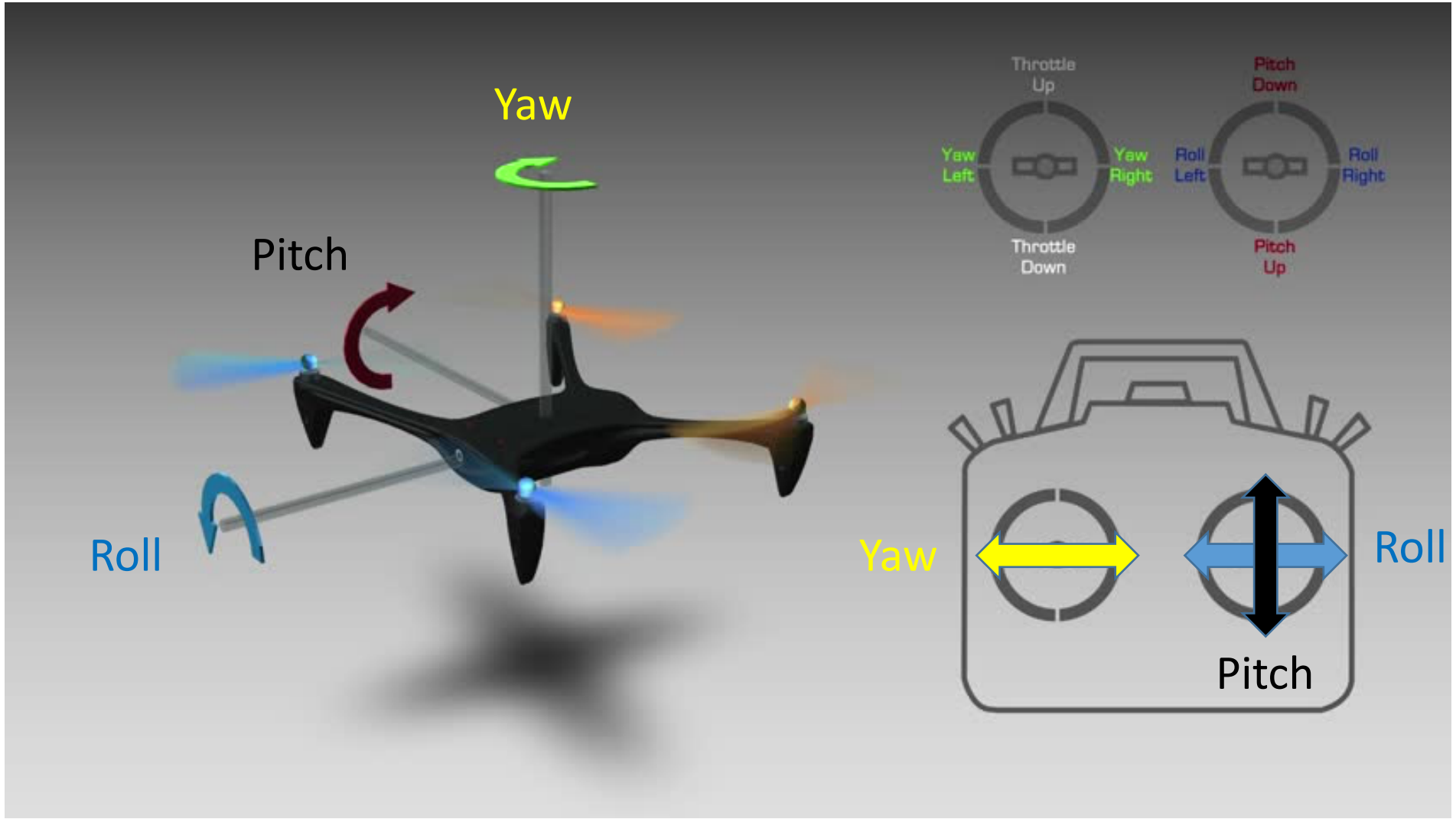
Write «rotation» in MATLAB command window






Inertial
Frame
O
Φ
x
θ
y
ψ
z
Roll
p, K
Surge
x0, u, X
Pitch
q, M
Sway
y0, v, Y
Yaw
r, N
Heave
z0, w, Z
Body-fixed
Frame


z
heave
yaw
pitch
centre of gravity
roll
y
sway
x
surge

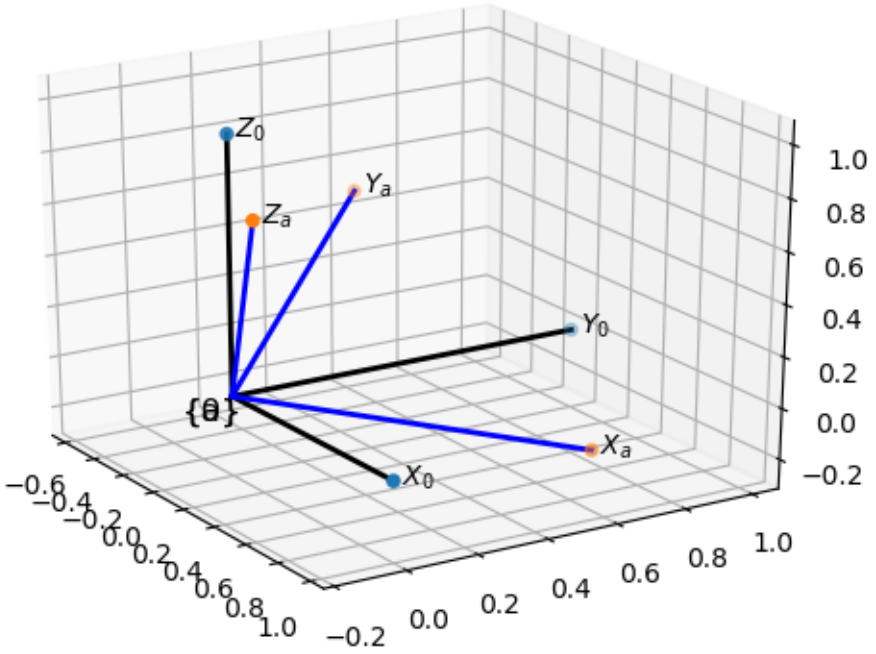

Yaw
Pitch
Roll
Throttle Up
Yaw Left
Yaw Right
Throttle Down
Pitch Down
Roll Left
Roll Right
Pitch Up
Yaw
Roll
Pitch

Roll-Pitch-Yaw Angles

```
roll = 10*np.pi/180
pitch = 30*np.pi/180
yaw = 45*np.pi/180

Ra = rob.rpy2rotmat(roll,pitch,yaw)
# homogeneous transform matrix
Ta = rob.trotmat(Ra)
fig = plt.figure()
ax = plt.axes(projection='3d')
rob.trplot3(ax,T0,name="0",color="k")
rob.trplot3(ax,Ta,name="a",color="b")
ax.axis('equal')
```

Python robotteknikk functions

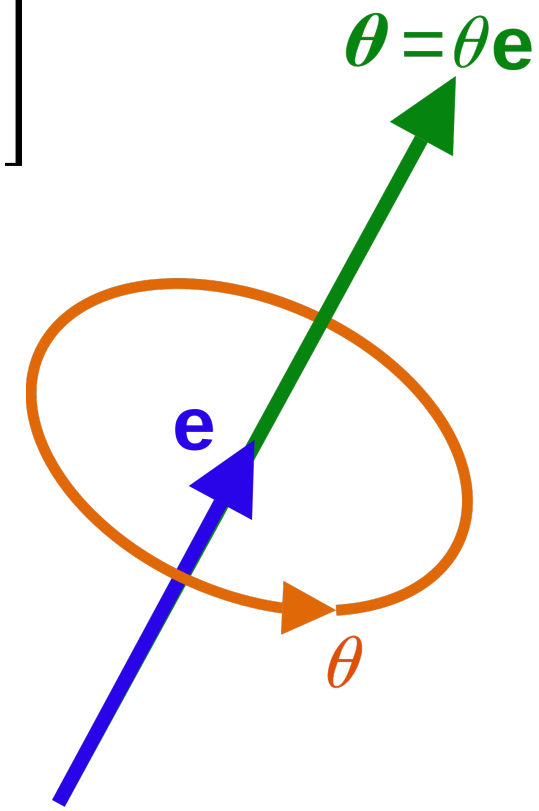
```
Ra = rob.rpy2rotmat(roll, pitch, yaw)
Ta = rob.trotmat(Ra)
```



Angle-Vector representations

Angle, vector

$$\theta, \boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$



Python robotteknikk functions

```
R = rob.angvec2rotmat(theta, v)
```

```
a = np.random.randn(3)
theta = np.random.randn()
R = angvec2rotmat(theta,v)
print(R.dot(R.T))
print(np.linalg.det(R))
```

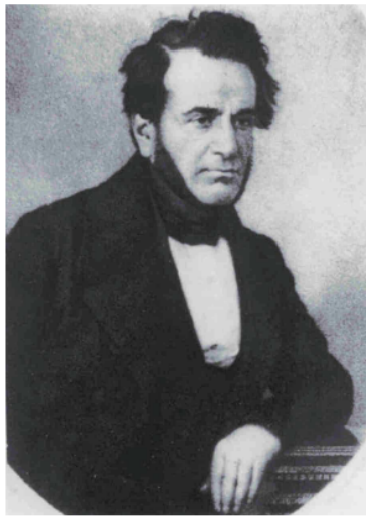
Rodrigues formula

Vector , Angle

$$\boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \theta$$

Rotation matrix

$$R$$



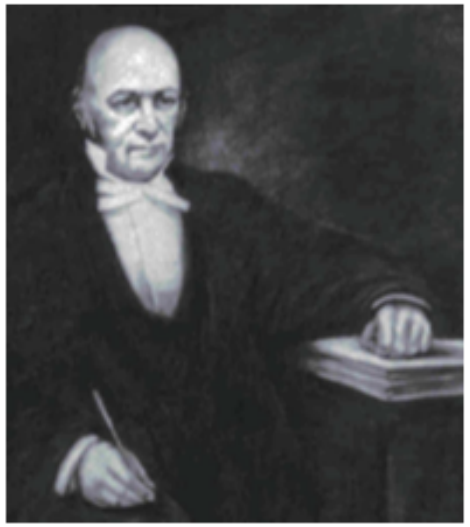
Olinde Rodrigues
1795-1851

$$R = I_{3\times3} + \sin\theta\, S(\boldsymbol{v}) + (1 - \cos\theta)(\boldsymbol{v}\boldsymbol{v}^T - I_{3\times3})$$

$$S(\boldsymbol{v}) = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$

```
def angvec2rotmat(theta,v):
    """ Computes rotation matrix from angle vector
    theta is angle in radians
    vector is 3 array vector
    Uses Rogrigues formula
    """
    s = np.sqrt(v.dot(v)) # compute length of v
    v = v/s # renormalize v. Now v has unit lenght
    I = np.eye(3) # identity matrix
    S = skew(v) # skew symmetrix matrix
    R = I + np.sin(theta)*S + (1-np.cos(theta))*(np.outer(v,v)-I)
    return R
```

Unit quaternions



Sir William Rowan Hamilton (1805–1865) was an Irish mathematician, physicist, and astronomer. He was a child prodigy with a gift for languages and by age thirteen knew classical and modern European languages as well as Persian, Arabic, Hindustani, Sanskrit, and Malay. Hamilton taught himself mathematics at age 17, and discovered an error in Laplace's Celestial Mechanics. He spent his life at Trinity College, Dublin, and was appointed Professor of Astronomy and Royal Astronomer of Ireland while still an undergraduate. In addition to quaternions he contributed to the development of optics, dynamics, and algebra. He also wrote poetry and corresponded with Wordsworth who advised him to devote his energy to mathematics.

According to legend the key quaternion equation, Eq. 2.17, occured to Hamilton in 1843 while walking along the Royal Canal in Dublin with his wife, and this is commemorated by a plaque on Broome bridge:

Here as he walked by on the 16th of October 1843 Sir William Rowan Hamilton in a flash of genius discovered the fundamental formula for quaternion multiplication $i^2 = j^2 = k^2 = i\,j\,k = -1$ *& cut it on a stone of this bridge.*

His original carving is no longer visible, but the bridge is a pilgrimage site for mathematicians and physicists.

$$i^2 = j^2 = k^2 = ijk = -1$$

Unit quaternions

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

$$q_0 = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}},$$

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \frac{1}{4q_0} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - 2_{12} \end{bmatrix}.$$

Unit quaternions

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \longrightarrow R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_0 q_2 + q_1 q_3) \\ 2(q_0 q_3 + q_1 q_2) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_0 q_1 + q_2 q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Vector , Angle

$$\boldsymbol{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}, \theta$$

Unit Quaternion

$$s = \cos\frac{\theta}{2} \qquad \boldsymbol{v} = \left(\sin\frac{\theta}{2}\right)\boldsymbol{n}$$

```
# Quaternions

theta = 20*np.pi/180 # angle
n = np.array([1,0,0]) # x axis, unit norm
# rotation vector
# rotation of 20 degrees along x axis
q = quaternion.from_rotation_vector(theta*n)
s = np.cos(theta/2)
v = np.sin(theta/2)*n
qm = np.quaternion(s,v[0],v[1],v[2]) # produces same result
```

$$\boldsymbol{q} = \begin{bmatrix} \cos\frac{\theta}{2} \\ n_x \sin\frac{\theta}{2} \\ n_y \sin\frac{\theta}{2} \\ n_z \sin\frac{\theta}{2} \end{bmatrix}$$

https://quaternion.readthedocs.io/en/latest/README.html#usage

```
# Quaternions
# ceate quaternion (not necesarily unit norm)
q = np.quaternion(1,2,3,4)

w1 = 20*np.pi/180 # angle
v1 = np.array([1,0,0]) # x axis, unit norm
# rotation vector
# rotation of 20 degrees along x axis
q1 = quaternion.from_rotation_vector(w1*v1)

w2 = -45*np.pi/180 # angle
v2 = np.array([1,1,0])
v2 = v2/np.sqrt(v2.dot(v2)) # normalize, make v2 unit length
q2 = quaternion.from_rotation_vector(w2*v2)

R1 = quaternion.as_rotation_matrix(q1)
q1e = quaternion.from_rotation_matrix(R1)
R2 = quaternion.as_rotation_matrix(q2)
R3 = R1.dot(R2) # composition of rotations
q3e = quaternion.from_rotation_matrix(R3)
q3 = q1*q2 # composition of rotations
R3e = quaternion.as_rotation_matrix(q3)
```

3D Homogeneous transformation matrix

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \\ 1 \end{pmatrix}$$

$$\begin{aligned} {}^{A}\tilde{\boldsymbol{p}} &= \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} {}^{B}\tilde{\boldsymbol{p}} \\ &= {}^{A}\boldsymbol{T}_{B}\,{}^{B}\tilde{\boldsymbol{p}} \end{aligned}$$

A concrete representation of relative pose ξ is $\xi \sim T \in SE(3)$ and $T_1 \oplus T_2 \mapsto T_1 T_2$ which is standard matrix multiplication.

$$T_1 T_2 = \begin{pmatrix} R_1 & t_1 \\ 0_{1\times 3} & 1 \end{pmatrix} \begin{pmatrix} R_2 & t_2 \\ 0_{1\times 3} & 1 \end{pmatrix} = \begin{pmatrix} R_1 R_2 & t_1 + R_1 t_2 \\ 0_{1\times 3} & 1 \end{pmatrix} \tag{2.20}$$

One of the rules of pose algebra from page 18 is $\xi \oplus 0 = \xi$. For matrices we know that $TI = T$, where I is the identify matrix, so for pose $0 \mapsto I$ the identity matrix. Another rule of pose algebra was that $\xi \ominus \xi = 0$. We know for matrices that $TT^{-1} = I$ which implies that $\ominus T \mapsto T^{-1}$

$$T^{-1} = \begin{pmatrix} R & t \\ 0_{1\times 3} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} R^T & -R^T t \\ 0_{1\times 3} & 1 \end{pmatrix} \tag{2.21}$$

Composition of transforms

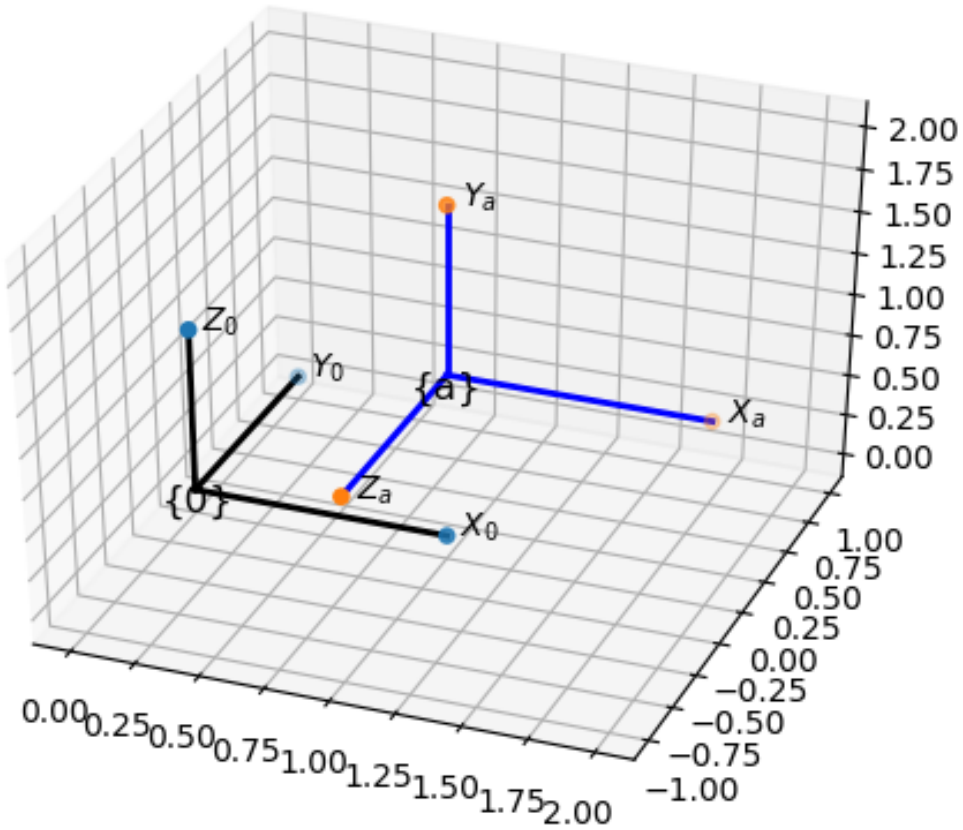
$T_A T_B$ Transform A followed by transform B along new axis

$T_B T_A$ Transform A followed by transform B along original old axis

```
T1 = rob.ttrans(np.array([1,2,3]))
T2 = rob.trotx(np.pi/4)
T3 = rob.troty(np.pi/4)
T4 = rob.trotz(np.pi/4)
```

```
T0 = np.eye(4)
Ta =
rob.ttrans(np.array([1,0,0])).dot(rob.trotx(
np.pi/2)).dot(rob.ttrans(np.array([0,1,0])))

fig = plt.figure()
ax = plt.axes(projection='3d')
rob.trplot3(ax,T0,name="0",color="k")
rob.trplot3(ax,Ta,name="a",color="b")
#ax.axis('equal')
```

$$T_a = T_{trans}(1,0,0)T_{rotx}\left(\frac{\pi}{2}\right)T_{trans}(0,1,0)$$



Exercise

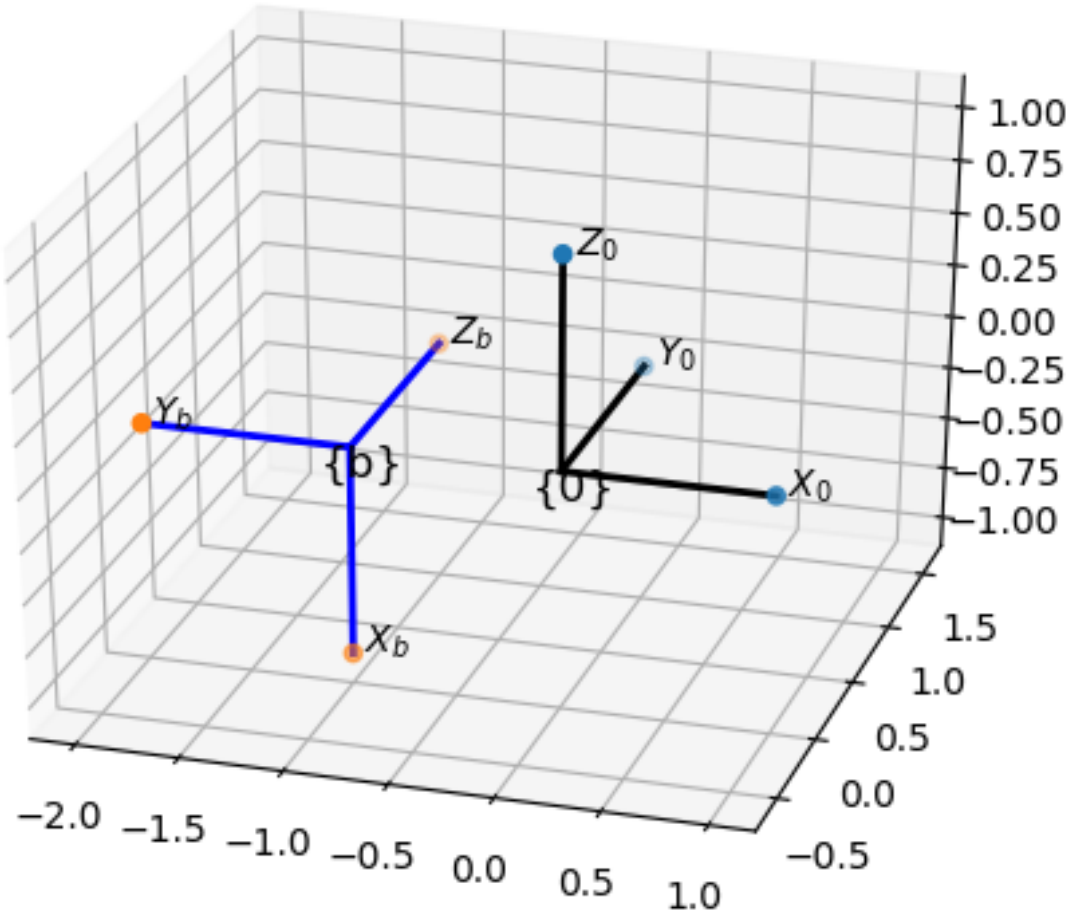
- Draw following transforms in paper to predict end-result
- Build transform matrix using python and plot result
- Verify that reults are equivalent

    - Rotation 90 degrees along Z axis
    - Translation (0,1,0) along new Y’ axis
    - Rotation 90 degrees along new Y’’ axis

$$T = T_{rotz}\left(\frac{\pi}{2}\right) T_{trans}(0,1,0) T_{roty}(\frac{\pi}{2})$$

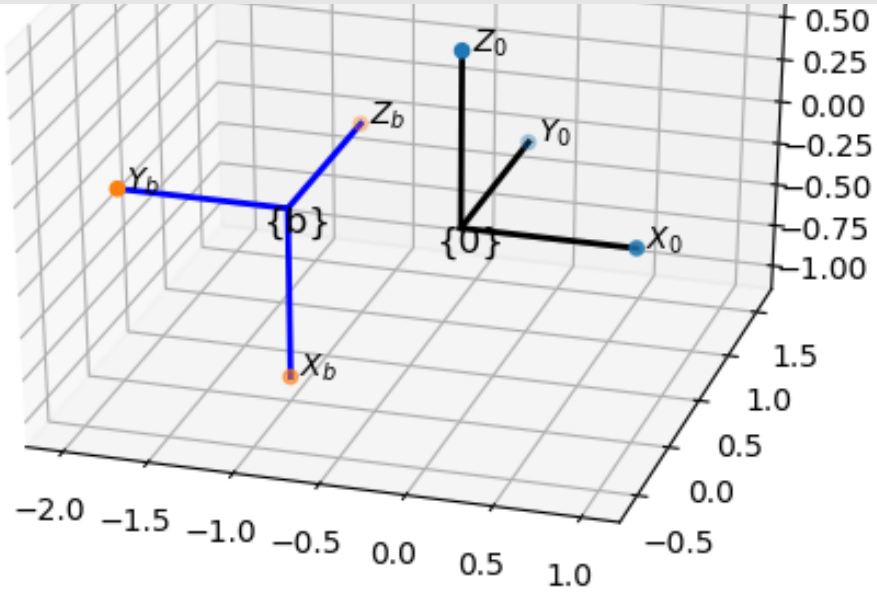
```
Tb =
rob.trotz(np.pi/2).dot(rob.ttrans(np.array([0,1,0]))).dot(rob.troty(np.pi/2))
fig = plt.figure()
ax = plt.axes(projection='3d')
rob.trplot3(ax,T0,name="0",color="k")
rob.trplot3(ax,Tb,name="b",color="b")
ax.axis('equal')
```

$$T = T_{rotz}\left(\frac{\pi}{2}\right) T_{trans}(0,1,0) T_{roty}\left(\frac{\pi}{2}\right)$$



```
To = np.eye(4)
Toa = rob.trotz(np.pi/2)
Tab = rob.ttrans(np.array([0,1,0]))
Tbc = rob.troty(np.pi/2)
Tob = Toa.dot(Tab)
Toc = Toa.dot(Tab).dot(Tbc)

fig = plt.figure()
ax = plt.axes(projection='3d')
rob.trplot3(ax,To,name="o",color="k")
rob.trplot3(ax,Toc,name="oc",color=„b“)
ax.axis('equal')
```

$$T = T_{rotz}\left(\frac{\pi}{4}\right) T_{trans}(0,1,0) T_{roty}\left(\frac{\pi}{2}\right)$$



Exercise

- Draw following transforms in paper to predict end-result
- Build transform matrix using python and plot result
- Verify that reults are equivalent

    - Rotation -45 degrees along Y axis
    - Translation (0,0,1) along new Z’ axis
    - Rotation -90 degrees along new X’’ axis

$$T = T_{roty}\left(-\frac{\pi}{4}\right) T_{trans}(0,0,1) T_{rotx}(-\frac{\pi}{2})$$

Exercise

- Draw following transforms in paper to predict end-result
- Build transform matrix using python and plot result
- Verify that reults are equivalent

    - Rotation -45 degrees along Y axis
    - Translation (0,0,1) along original Z axis
    - Rotation -90 degrees along original X axis

$$T = T_{rotx}(-\frac{\pi}{2})T_{trans}(0,0,1)T_{roty}\left(-\frac{\pi}{4}\right)$$