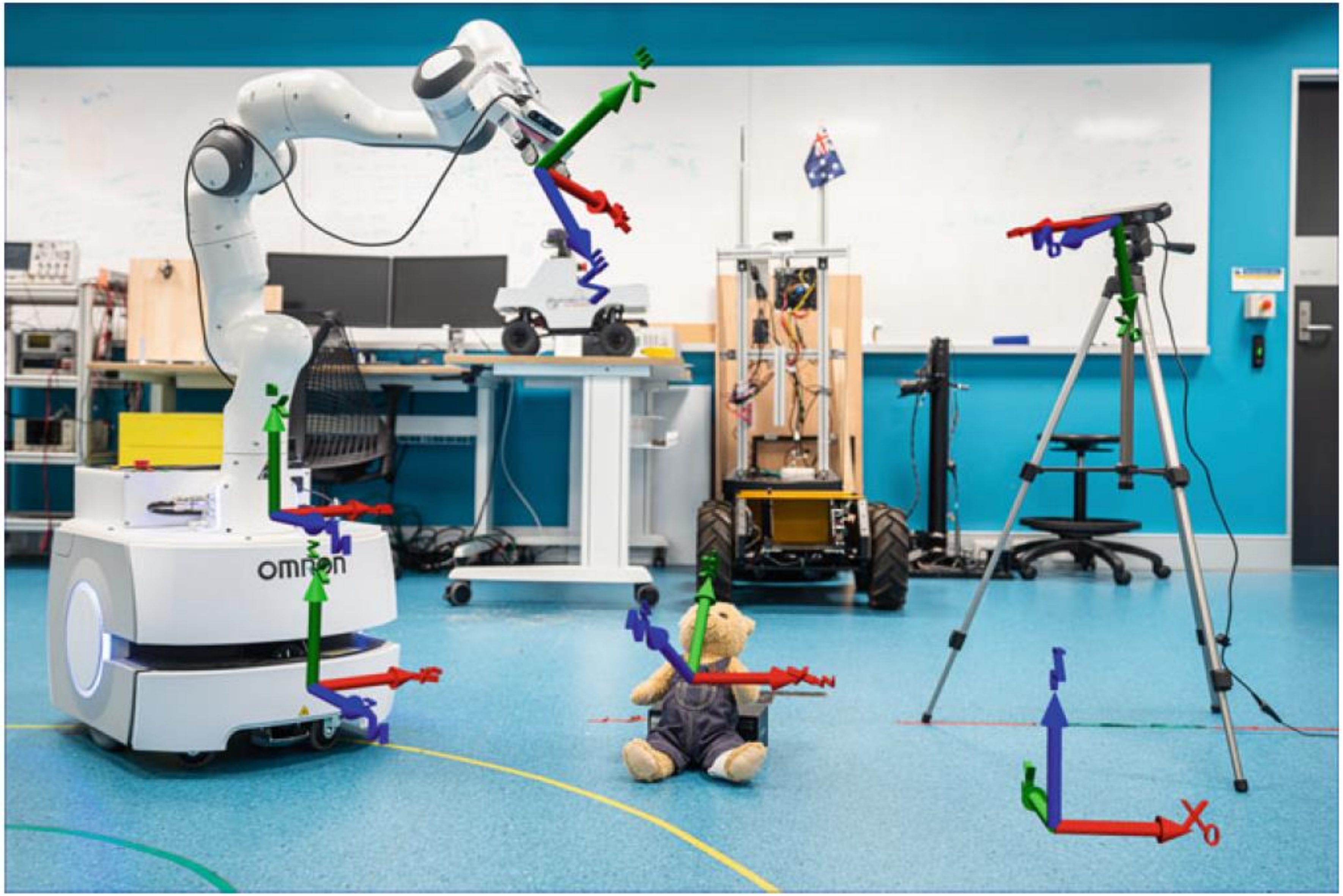
Frame Coodinate Transformation

심주용
숙명여자대학교
기계시스템학부

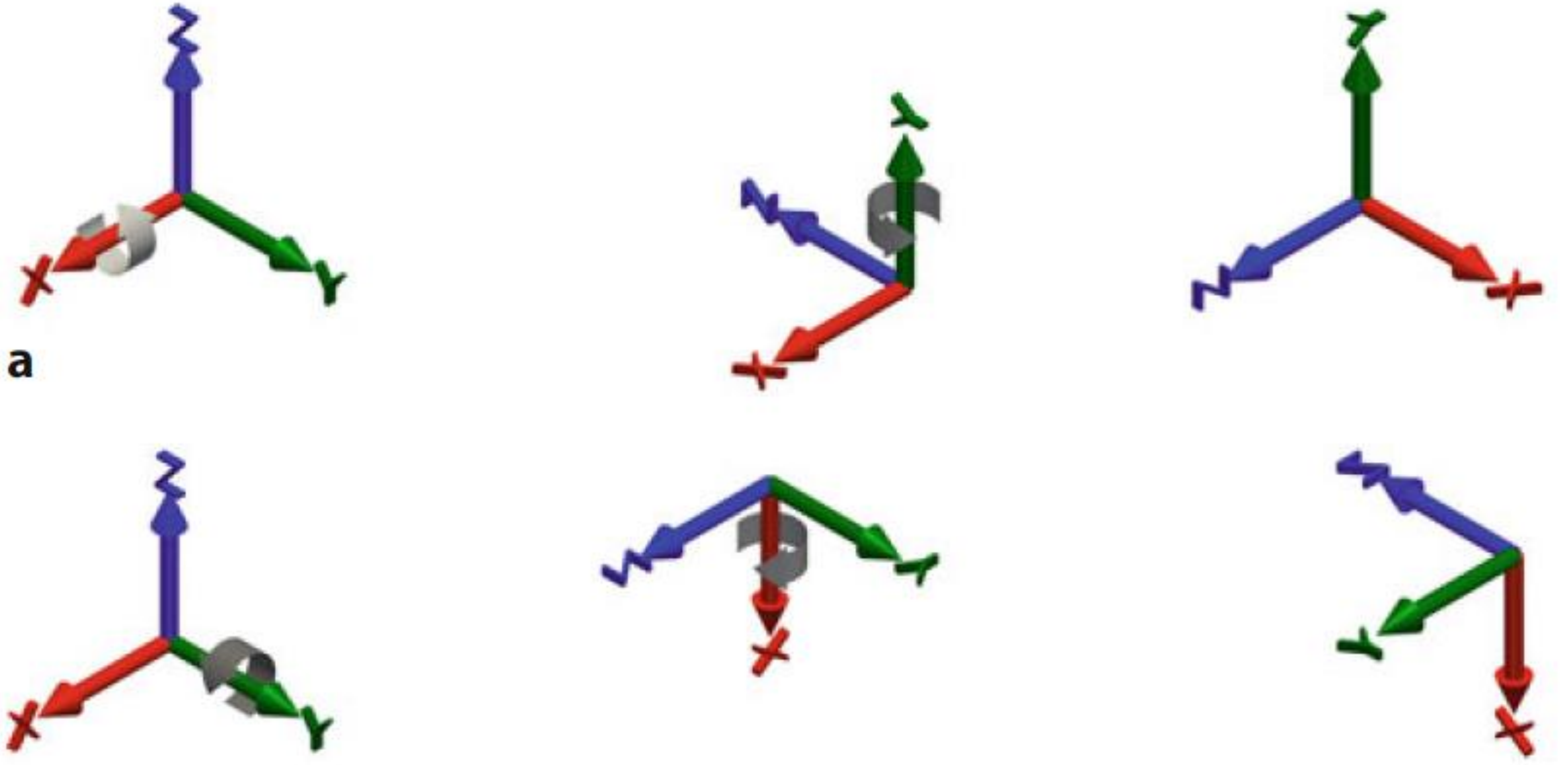
Coordinate Frames



Peter Corke - Robotics, Vision and Control_ Fundamental Algorithms in Python, 3rd Edition. 146-Springer (2023)

3D Coordinate Frames

Noncommutativity of rotation



b original pose | after first rotation | affter second rotation

3D Rotation Matrix

Frame B to Frame A

$$\begin{pmatrix} {}^{A}p_x \\ {}^{A}p_y \\ {}^{A}p_z \end{pmatrix} = {}^{A}\mathbf{R}_B \begin{pmatrix} {}^{B}p_x \\ {}^{B}p_y \\ {}^{B}p_z \end{pmatrix}$$

$$R = \begin{array}{ccc} \boxed{\begin{matrix} 1 \\ 0 \\ 0 \end{matrix}} & \boxed{\begin{matrix} 0 \\ 0 \\ 1 \end{matrix}} & \boxed{\begin{matrix} 0 \\ -1 \\ 0 \end{matrix}} \\ \text{new x-axis} & \text{new y-axis} & \text{new z-axis} \end{array}$$

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

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

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

Homogeneous TransformationMatrix

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

$$T = \begin{matrix} 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 \end{matrix}$$

new origin

new x-axis

new y-axis

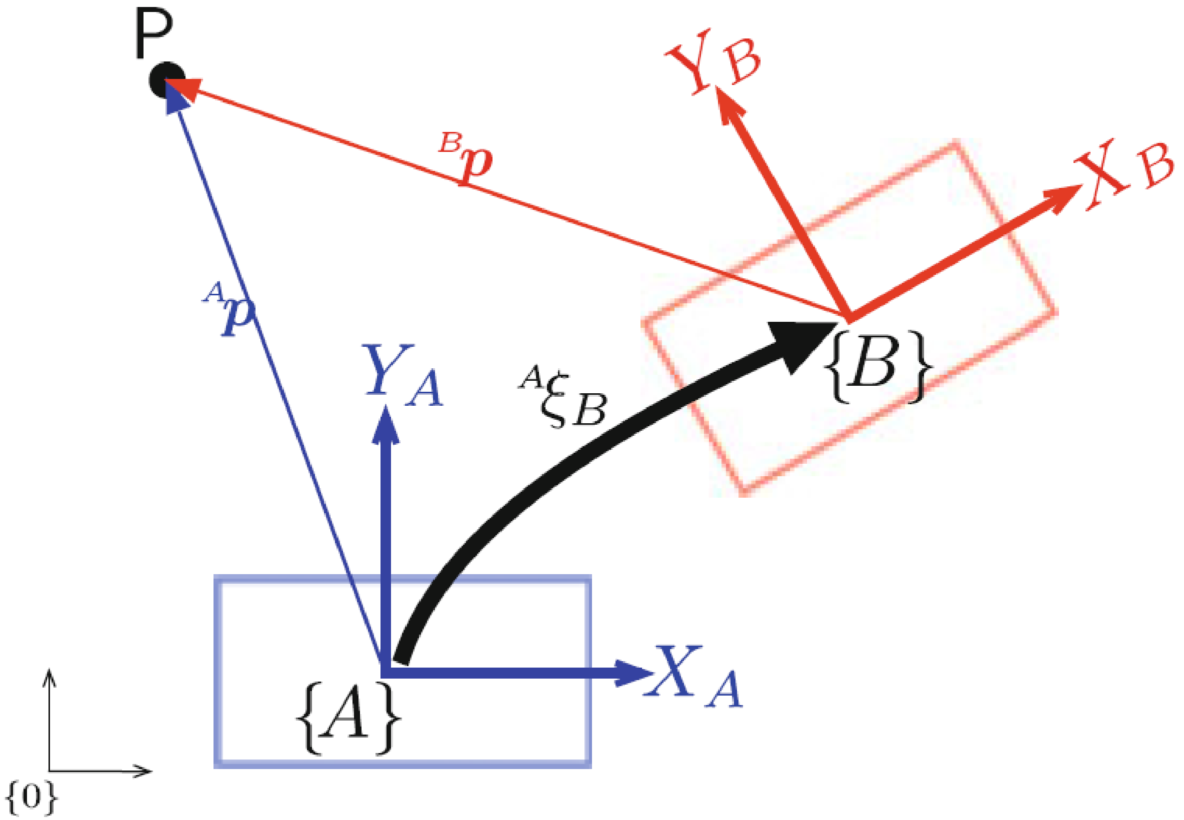
new z-axis

Coordinate Frames

Point P can be described by coordinate vectors expressed in frame {A} or {B}

$$^{A}\boldsymbol{p} = {}^{A}\boldsymbol{\xi}_{B} \cdot {}^{B}\boldsymbol{p}$$

$^{A}\xi_{B}$: the pose of {B} relative to {A}



$$^{X}\boldsymbol{\xi}_{Z} = {}^{X}\boldsymbol{\xi}_{Y} \oplus {}^{Y}\boldsymbol{\xi}_{Z}$$

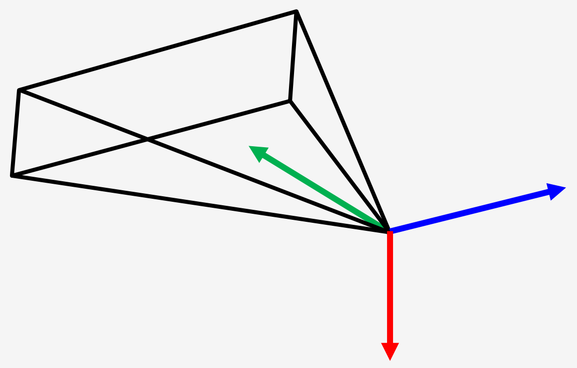
$$^{X}\xi_{Y} \oplus {}^{Y}\xi_{Z} = {}^{X}\xi_{Z}$$

same

Peter Corke - Robotics, Vision and Control_ Fundamental Algorithms in Python, 3rd Edition. 146-Springer (2023)

Frame Coordinate Transformation

```python
import numpy as np
from scipy.spatial.transform import Rotation as R
from numpy.linalg import inv

def create_pose_matrix(x, y, z, heading, pitch, roll):
    """Create a 4x4 pose matrix from translation and Euler angles."""
    rot = eul2rot(heading, pitch, roll)
    pose_matrix = np.eye(4)
    pose_matrix[:3, :3] = rot
    pose_matrix[:3, 3] = [x, y, z]
    return pose_matrix

# Example DataFrames with data (please ensure prame_data and sel_data are loaded properly)
# Generate pose matrices for each dataset
team1_poses = [create_pose_matrix(row['x'], row['y'], row['z'], row['heading'], row['pitch'], row['roll']) for index, row in team1_data.iterrows()]
sim_poses = [create_pose_matrix(row['x'], row['y'], row['z'], row['heading'], row['pitch'], row['roll']) for index, row in sim_data.iterrows()]

# Compute individual transformations
transformations = []

for team1_pose, sim_pose in zip(team1_poses, sim_poses):
    # Compute the transformation T_i for each pair (T_i * sim_pose = team1_pose)
    # Rearrange to T_i = team1_pose * inv(sim_pose)
    T_i = team1_pose @ inv(sim_pose)
    transformations.append(T_i)

# Average the transformations to find the final matrix
average_transformation = sum(transformations) / len(transformations)
```



Reality Capture Frame Coordinate

Reality Capture Frame Coordinate

```python
import numpy as np
from scipy.spatial.transform import Rotation as R
from numpy.linalg import inv

def create_pose_matrix(x, y, z, heading, pitch, roll):
    """Create a 4x4 pose matrix from translation and Euler angles
    rot = eul2rot(heading, pitch, roll)
    pose_matrix = np.eye(4)
    pose_matrix[:3, :3] = rot
    pose_matrix[:3, 3] = [x, y, z]
    return pose_matrix

# Example DataFrames with data (please ensure prame_data and sel_
# Generate pose matrices for each dataset
team1_poses = [create_pose_matrix(row['x'], row['y'], row['z'], r
sim_poses = [create_pose_matrix(row['x'], row['y'], row['z'], row

# Compute individual transformations
transformations = []

for team1_pose, sim_pose in zip(team1_poses, sim_poses):
    # Compute the transformation T_i for each pair (T_i * sim_pos
    # Rearrange to T_i = team1_pose * inv(sim_pose)
    T_i = team1_pose @ inv(sim_pose)
    transformations.append(T_i)

# Average the transformations to find the final matrix
average_transformation = sum(transformations) / len(transformatio
```