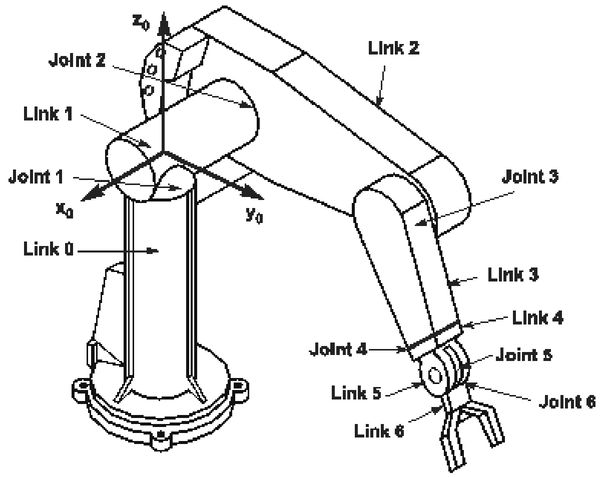
Lecture 5: Other Orientation Representations

Iñigo Iturrate
Assistant Professor
SDU Robotics,
The Maersk McKinney Moller Institute,
University of Southern Denmark

Ø27-604-3
inju@mmmi.sdu.dk



SDU ROBOTICS

What did we learn last time?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

What are we doing today?

1. CAD Modelling in Autodesk Inventor – *Guest Lecture by Aljaz Kramberger*
2. CAD Assemblies in Autodesk Inventor – *Guest Lecture by Aljaz Kramberger*
3. Introduction to Robotics & Recap of Linear Algebra and Mathematical Notation
4. Translations & Rotation Matrices
5. **Other Representations for Orientation *(Today)***
6. Transformation Matrices
7. DH Parameters & Forward Kinematics
8. Analytical Forward Kinematics & Kinematic Simulation
9. Inverse Kinematics
10. Velocity Kinematics & the Jacobian Matrix
11. More about the Jacobian & Trajectory Generation
12. Manipulability, More on the Robotic Systems Toolbox

Topics for Today

Part I:

- The **”problem”** with **orientation**
- **”Warm-up” exercise**/discussion
- **Overview** of orientation/rotation representations

Part II:

- **Angle-Set Conventions**
  - **Fixed angles**
  - **Euler angles**
- **Gimbal Lock**

Part III:

Equivalent **angle-axis** (Euler vector)

Part I: What is this whole fuss about orientation?

(or why we are using more than one lecture on it)

The Concept of Orientation

How would you represent an orientation:

- If we lived in a **1D world**?
- If we lived in a **2D world**?
- If we lived in a **3D world**?

Why is Orientation Difficult?

Position is **all nice and easy** for our brains.

- We like $\mathbb{R}$. It is easy to relate to:

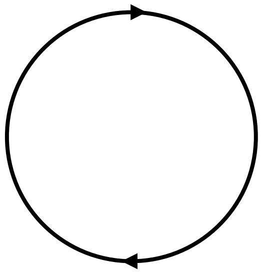
$$-\infty \longleftrightarrow +\infty$$

$$+\infty \neq -\infty$$

☺

Orientation is a ***funky concept*** to your brain, in terms of mathematics.

- Orientation does not live in $\mathbb{R}$. It is related to circles and spheres and lives in $SO(3)$ (in 3D). Funny things happen:



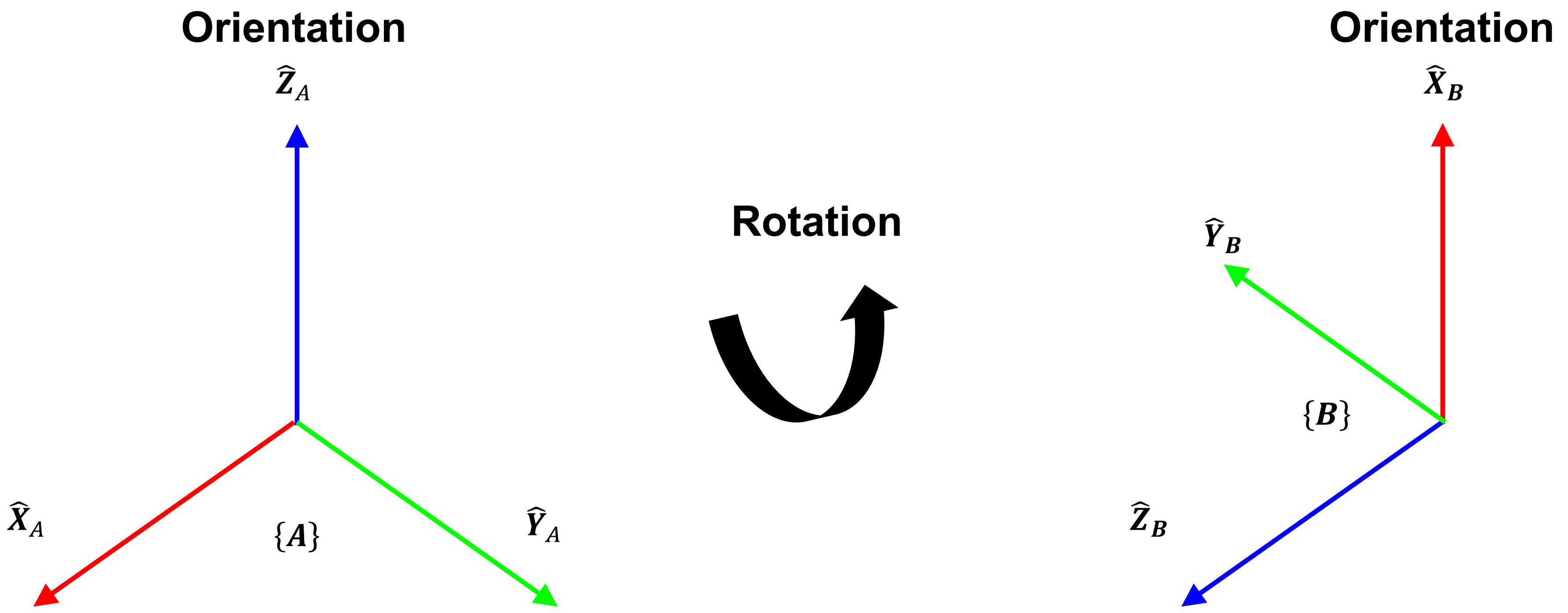
$$0 = 2\pi$$

?!?!

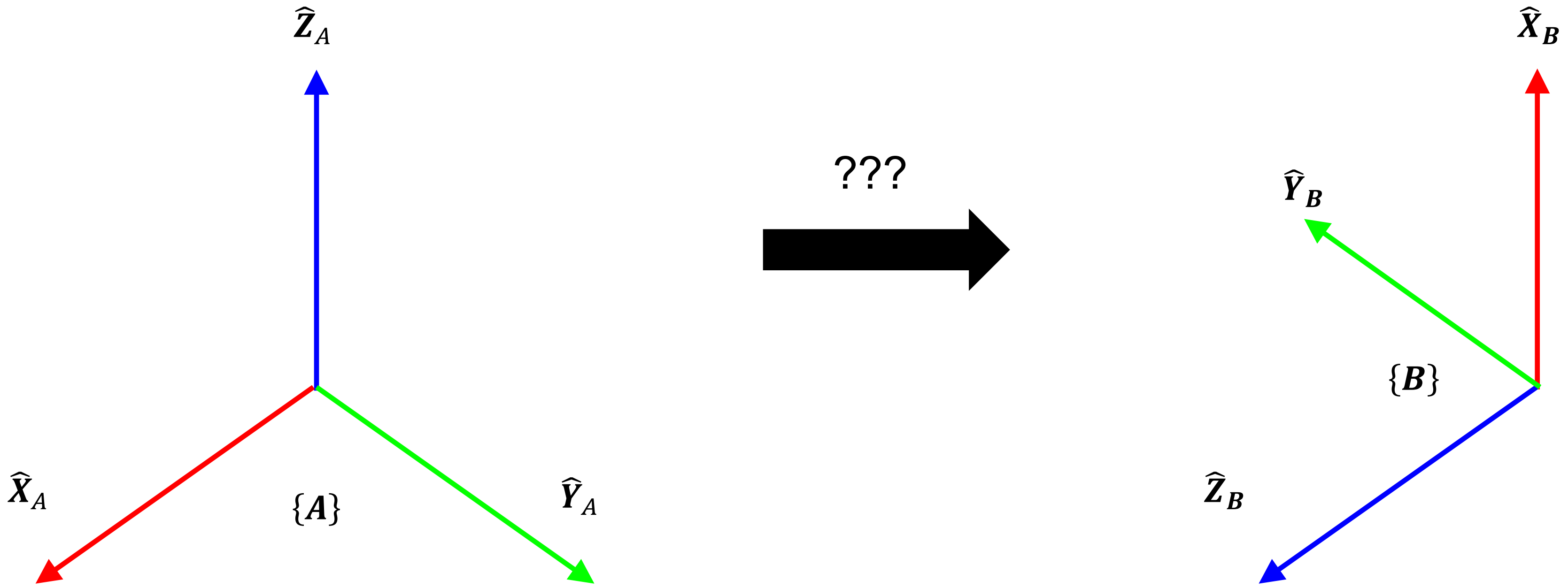
☹

Rotation vs. Orientation

- **Orientation describes a state**.
- **Rotation** is an **operation**, i.e. it describes how to get from one orientation to another.



How do you get from {A} to {B}?

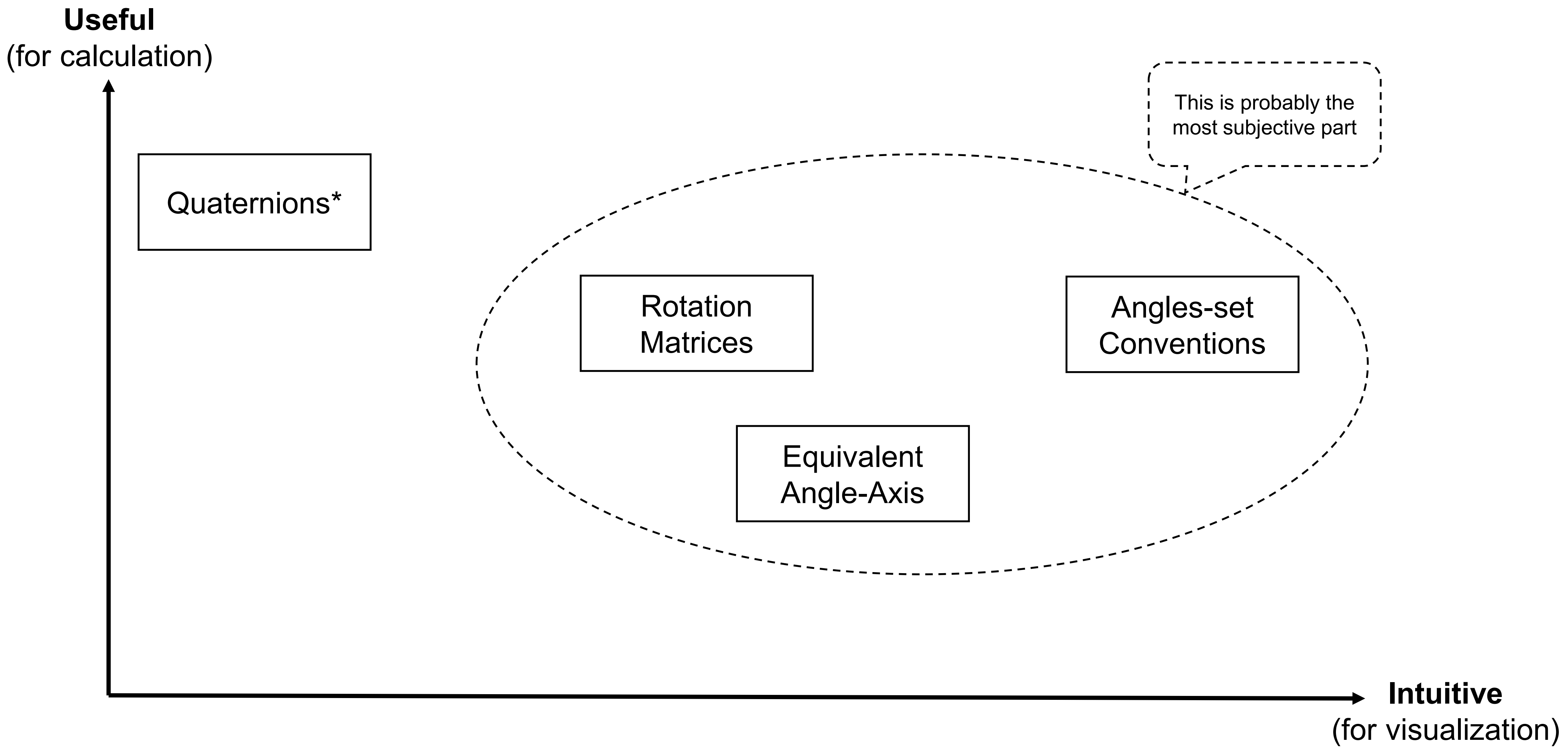


Splitting the **3D problem** into a **sequence of rotations** is one **solution***:

- You may think **this is easy**…
- …so why did I say orientation is difficult?
- Well, this **solution carries other problems**, as we will see.

Note: Called **Euler angles.*

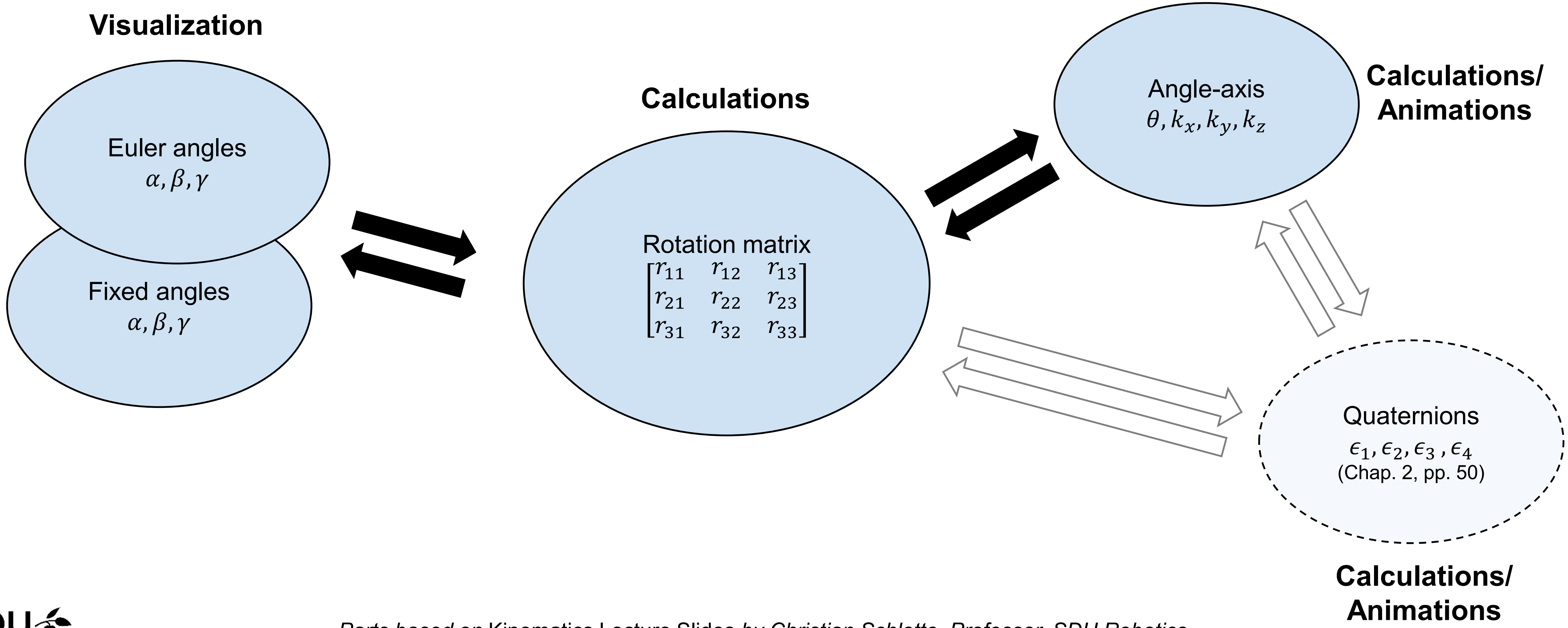
(Subjective) Overview of Orientation Representations



*Quaternions are **wonderful** and extremely powerful.
Also, they live in a **4-dimensional space** and will therefore make your brain explode if you think too much about them.*

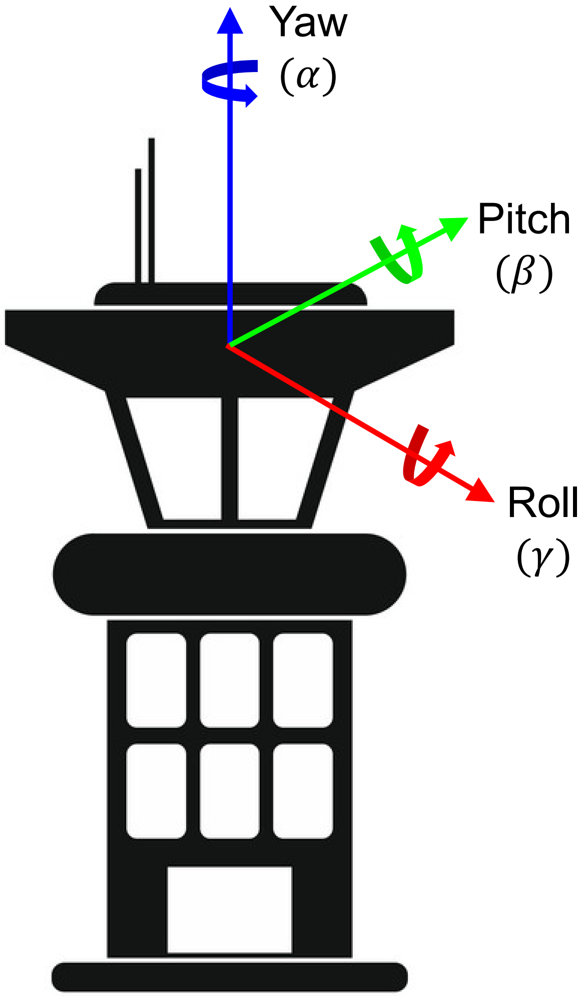
SDU ROBOTICS

Orientation/Rotation Representations: Use-Cases

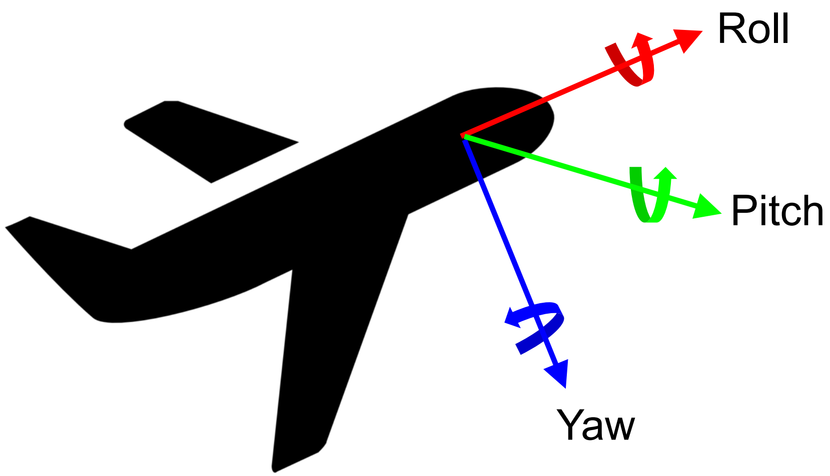


Part II: Angle-Set Conventions

Angle-Set Conventions



Fixed Angles



Euler Angles

Angle-set conventions represent orientation as **three separate rotations around different axes**.

- For **fixed angles**, the **coordinate system stays fixed/still**.
- For **Euler angles**, the **coordinate system rotates**.

The rotations can be applied in **many orders**.

SDU ROBOTICS

All 24 Angle-Set Conventions

12 Euler angles sets

$\boldsymbol{R}_{X'Y'Z'}$	$\boldsymbol{R}_{X'Z'Y'}$	$\boldsymbol{R}_{Y'X'Z'}$	$\boldsymbol{R}_{Y'Z'X'}$	$\boldsymbol{R}_{Z'X'Y'}$	$\boldsymbol{R}_{Z'Y'X'}$
$\boldsymbol{R}_{X'Y'X'}$	$\boldsymbol{R}_{X'Z'X'}$	$\boldsymbol{R}_{Y'X'Y'}$	$\boldsymbol{R}_{Y'Z'Y'}$	$\boldsymbol{R}_{Z'X'Z'}$	$\boldsymbol{R}_{Z'Y'Z'}$

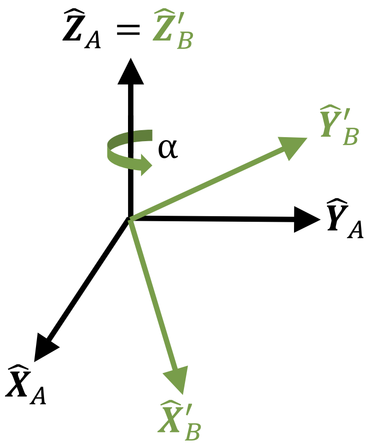
12 Fixed angle sets

$\boldsymbol{R}_{XYZ}$	$\boldsymbol{R}_{XZY}$	$\boldsymbol{R}_{YXZ}$	$\boldsymbol{R}_{YZX}$	$\boldsymbol{R}_{ZXY}$	$\boldsymbol{R}_{ZYX}$
$\boldsymbol{R}_{XYX}$	$\boldsymbol{R}_{XZX}$	$\boldsymbol{R}_{YXY}$	$\boldsymbol{R}_{YZY}$	$\boldsymbol{R}_{ZXZ}$	$\boldsymbol{R}_{YZY}$

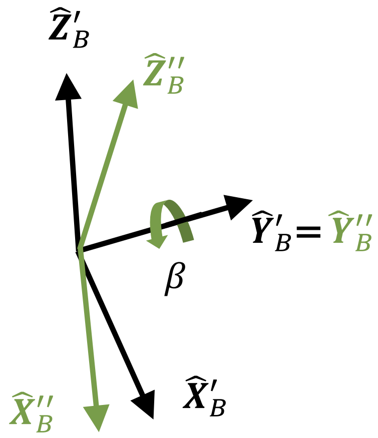
Euler Angles Z-Y-X

Euler angles: Each rotation is performed about an axis of the **coordinate system from the last step**.

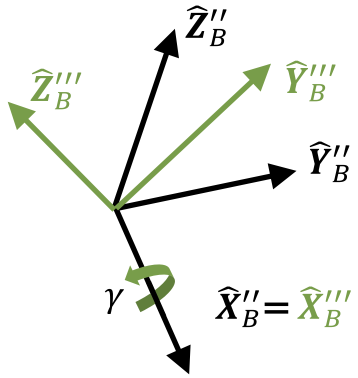
1. Start with a known frame $\{\boldsymbol{A}\}$. Rotate first about $\hat{\boldsymbol{Z}}_A$ by angle α.



2. Then, rotate about the resulting $\hat{\boldsymbol{Y}}'_B$ by angle β.

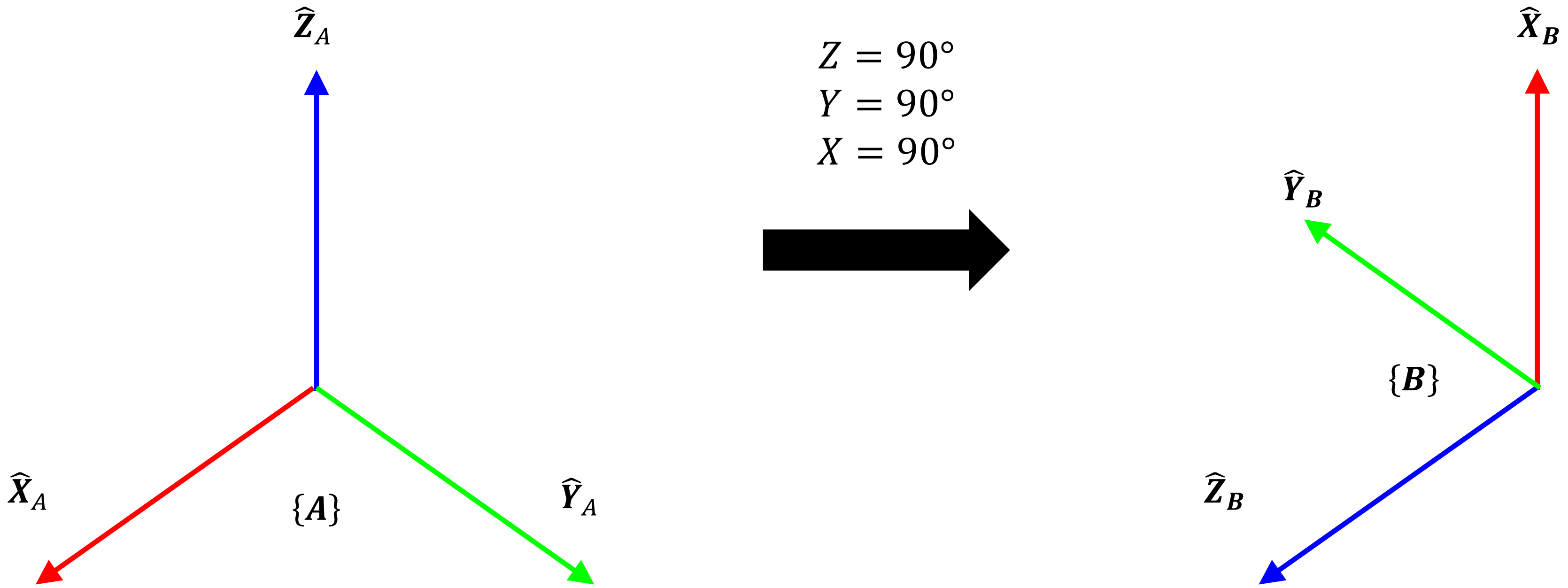


3. Finally, rotate about the resulting $\hat{\boldsymbol{X}}''_B$ by angle γ.



Euler Angles Z-Y-X to Rotation Matrix

You have **already done this**! Just **right-multiply three rotation matrices around each axis in order**:



$$ {}^{A}_{B}\boldsymbol{R}_{Z'Y'X'}(\alpha,\beta,\gamma) = \boldsymbol{R}_Z(\alpha)\boldsymbol{R}_Y(\beta)\boldsymbol{R}_X(\gamma) = \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix} = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} $$

Stack from left to right
(i.e. multiply each new rotation on the right)

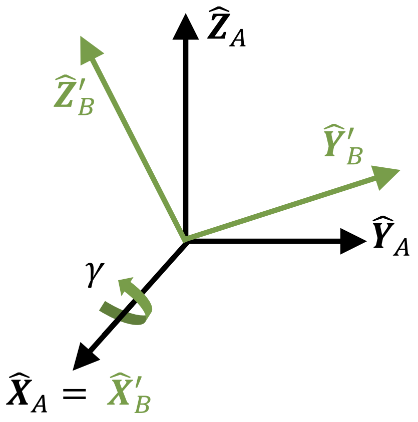
$$ c\alpha = \cos\alpha\,, \qquad s\alpha = \sin\alpha $$

SDU ROBOTICS

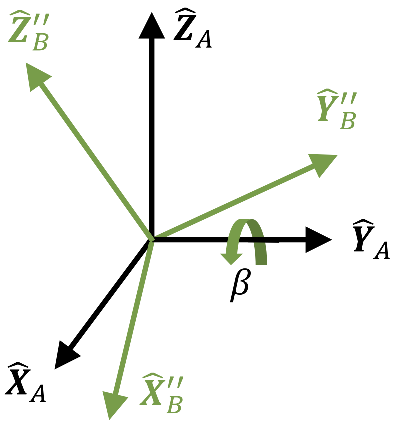
Fixed Angles X-Y-Z (Roll-Pitch-Yaw)

Fixed angles: Each rotation is performed about an axis of the same **fixed reference coordinate system**.

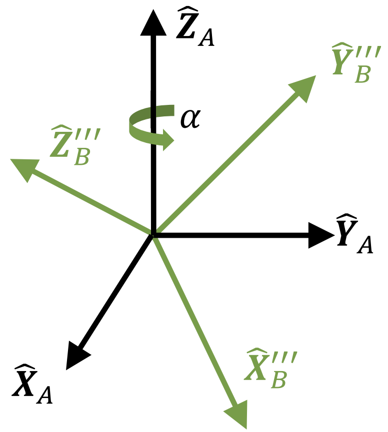
1. Start with a known frame $\{\boldsymbol{A}\}$. Rotate first about $\hat{\boldsymbol{X}}_A$ by angle γ.



2. Then, rotate about the $\hat{\boldsymbol{Y}}_A$ by angle β.



3. Finally, rotate about the $\hat{\boldsymbol{Z}}_A$ by angle α.



Fixed Angles X-Y-Z to Rotation Matrix

Opposite of Euler angles: **left-multiply three rotation matrices around each axis in order**:

$$ {}^{A}_{B}\boldsymbol{R}_{XYZ}(\gamma, \beta, \alpha) = \boldsymbol{R}_Z(\alpha)\boldsymbol{R}_Y(\beta)\boldsymbol{R}_X(\gamma) = \textbf{???} $$

Stack from right to left
(i.e. multiply each new rotation on the left)

In MATLAB, try the following:

Using combinations of your three functions *rotx()*, *roty()*, *rotz():*

1. Calculate:
   a) The rotation matrix for the Euler Angles Z-Y-X, for $\alpha = 35°, \beta = 70°, \gamma = 20°$.
   b) The rotation matrix for the Fixed Angles X-Y-Z (RPY), for $\gamma = 20°, \beta = 70°, \alpha = 35°$.

2. Calculate:
   a) The rotation matrix for the Euler Angles Z-Y-Z, for $\alpha = 80°, \beta = 125°, \gamma = -70°$.
   b) The rotation matrix for the Fixed Angles Z-Y-Z, for $\gamma = -70°, \beta = 125°, \alpha = 80°$.

Notice anything interesting?

SDU ROBOTICS

Relation between Fixed and Euler Angles

Fixed angles for a certain order are equal to Euler angles with the opposite order!

$$ {}^{A}_{B}\boldsymbol{R}_{XYZ}(\gamma, \beta, \alpha) = {}^{A}_{B}\boldsymbol{R}_{Z'Y'X'}(\alpha, \beta, \gamma) $$

$$ {}^{A}_{B}\boldsymbol{R}_{YZX}(\gamma, \beta, \alpha) = {}^{A}_{B}\boldsymbol{R}_{X'Z'Y'}(\alpha, \beta, \gamma) $$

$$ {}^{A}_{B}\boldsymbol{R}_{ZXY}(\gamma, \beta, \alpha) = {}^{A}_{B}\boldsymbol{R}_{Y'X'Z'}(\alpha, \beta, \gamma) $$

etc.

What is happening?

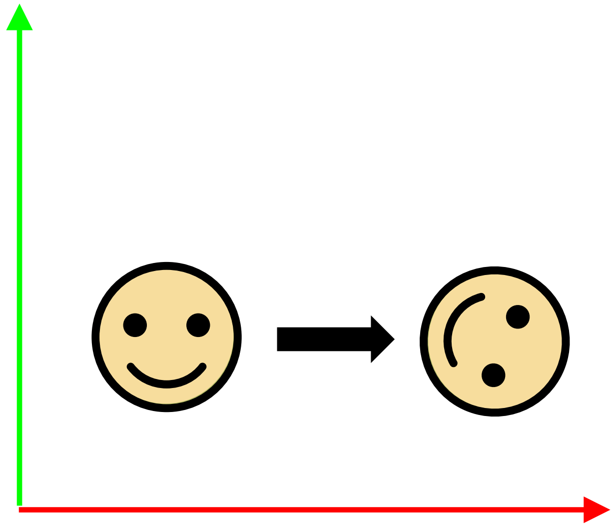
How can we interpret the two approaches geometrically?

Local vs. Global Coordinate Frames
(and Pre/Post-multiplication)

It comes down to: are we **moving** or **fixing** the reference frame?

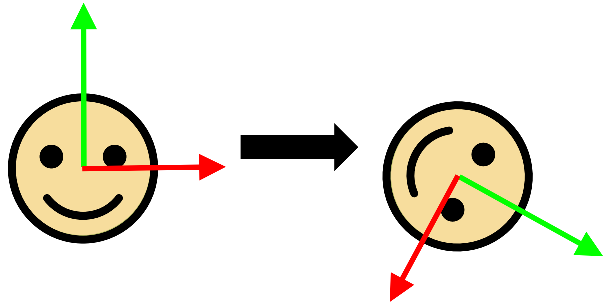
Fixed Angles

- **Global coordinate frame,** does not move
- **Left-multiplied**
- For a robot: the **base** is a **fixed frame**

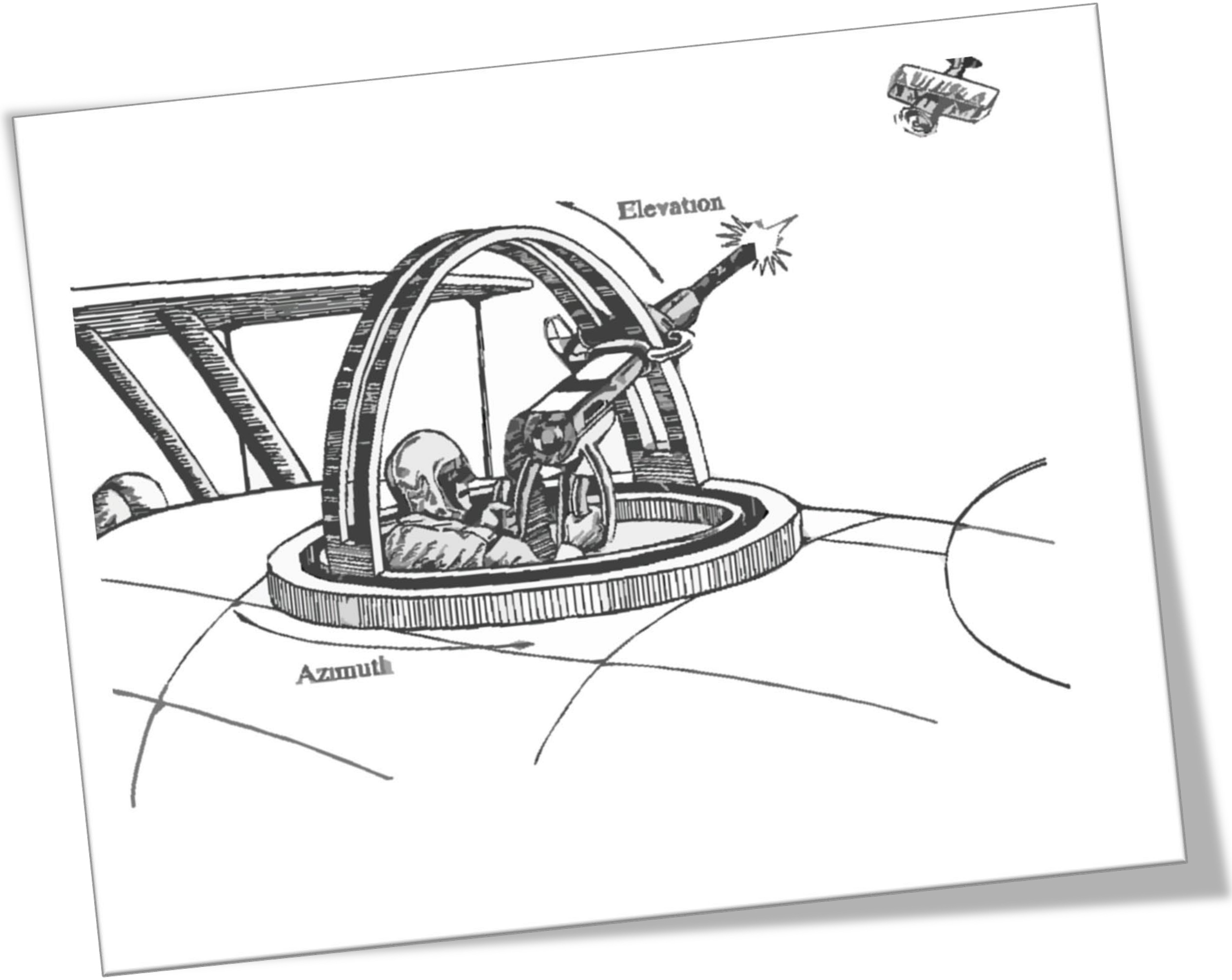


Euler Angles

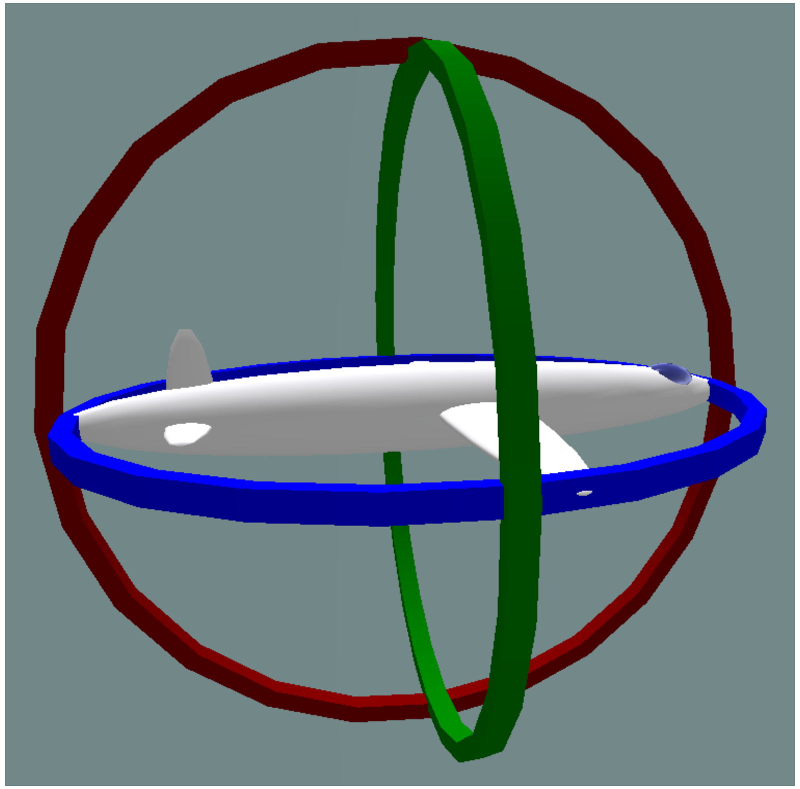
- **Local coordinate frame,** moves
- **Right-multiplied**
- For a robot, the **tool** is a **moving frame**

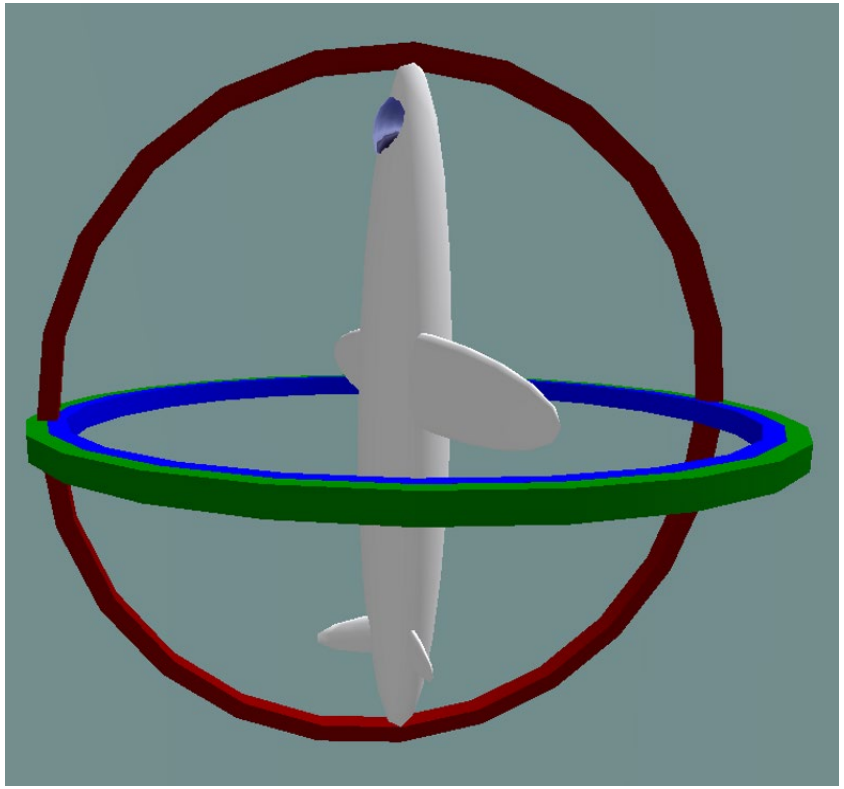


An Issue with Angle-Sets: Gimbal Lock (Example)



Gimbal Lock





Gimbal Lock: When does it happen?

Gimbal lock can happen for **two different reasons**:

- **Mathematically**: in representations that map 3D orientation to sets of three angles (i.e. **angle-sets**).
- **Physically:**
  - In **mechanisms** that decouple orientation into several angles.
    (e.g. the airplane gunner example).
  - In **measurement** systems that decouple quantities related to orientation into several angles.
    (e.g. Inertial Measurement units → see Apollo 11)

Rotation Matrix to Fixed Angles X-Y-Z (I)

$$ {}^{A}_{B}\boldsymbol{R}_{Z'Y'X'}(\alpha,\beta,\gamma) = {}^{A}_{B}\boldsymbol{R}_{XYZ}(\gamma,\beta,\alpha) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

$$ \cos\beta = \sqrt{{r_{11}}^2 + {r_{21}}^2}, $$

$$ \sin\beta = -r_{31} \longrightarrow \tan\beta = \frac{\sin\beta}{\cos\beta} \longrightarrow \beta = \text{atan}\frac{-r_{31}}{\sqrt{{r_{11}}^2 + {r_{21}}^2}} $$

$$ \sqrt{(\cos\alpha\cos\beta)^2 + (\sin\alpha\cos\beta)^2} = \sqrt{(\cos\beta)^2\left((\cos\alpha)^2 + (\sin\alpha)^2\right)} = \cos\beta $$

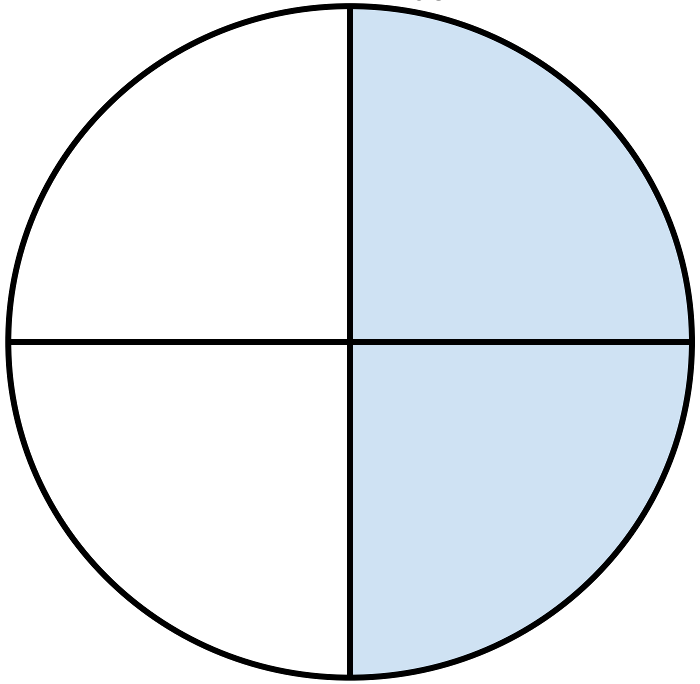
$$ \beta = \text{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right) $$

atan2 is a **better version** of atan that **takes into account the full circle.**

SDU ROBOTICS

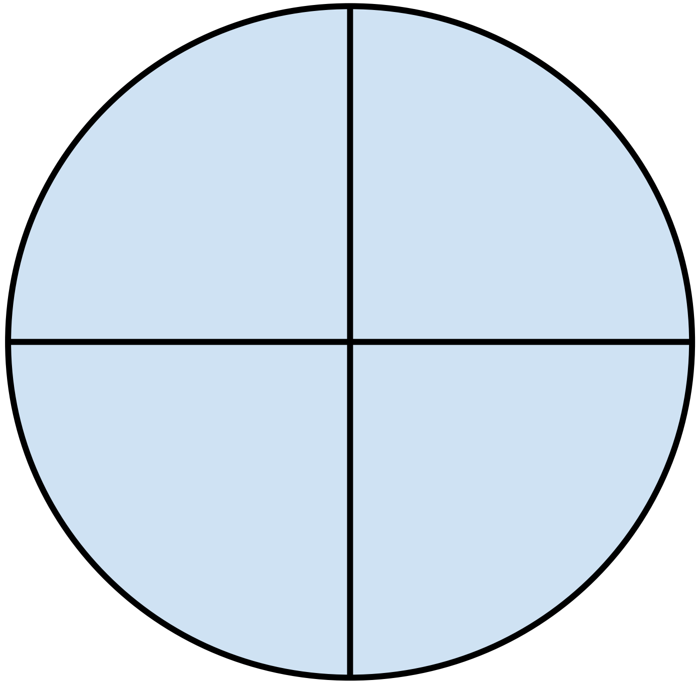
atan vs atan2

$\mathbf{atan}\left(\frac{y}{x}\right)$



Returns angles between $\frac{\pi}{2}$ and $-\frac{\pi}{2}$.

$\mathbf{atan2}(y, x)$



Returns angles in the full circle (0 to 2π).

Rotation Matrix to Fixed Angles X-Y-Z (II)

$$ {}^{A}_{B}\boldsymbol{R}_{Z'Y'X'}(\alpha,\beta,\gamma) = {}^{A}_{B}\boldsymbol{R}_{XYZ}(\gamma,\beta,\alpha) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

$$ r_{21} = \sin\alpha\cos\beta \longrightarrow \sin\alpha = \frac{r_{21}}{\cos\beta} $$

$$ r_{11} = \cos\alpha\cos\beta \longrightarrow \cos\alpha = \frac{r_{11}}{\cos\beta} $$

$$ \alpha = atan2\left(\frac{r_{21}}{\cos\beta}, \frac{r_{11}}{\cos\beta}\right) $$

$$ r_{32} = \cos\beta\sin\gamma \longrightarrow \sin\gamma = \frac{r_{32}}{\cos\beta} $$

$$ r_{33} = \cos\beta\cos\gamma \longrightarrow \cos\gamma = \frac{r_{33}}{\cos\beta} $$

$$ \gamma = atan2\left(\frac{r_{32}}{\cos\beta}, \frac{r_{33}}{\cos\beta}\right) $$

SDU ROBOTICS

Rotation Matrix to Other Angle-Sets

We have seen how to convert a **rotation matrix** to **fixed angle X-Y-Z**.

How do we do it for **other kinds of angle sets**?

- We can **use Fixed angle to Euler angle equivalences**.
  - e.g. Fixed angle X-Y-Z = Euler angle Z-Y-X

- We can use **similar logic** to the previous slides **for other angle combinations**:
  1. **Isolate sine** and **cosine** for one angle.
  2. **Use atan2** to compute the angle.
  3. **Repeat** for other angles.

Part III: Equivalent Angle-Axis

Equivalent Angle-Axis (Euler Vector)

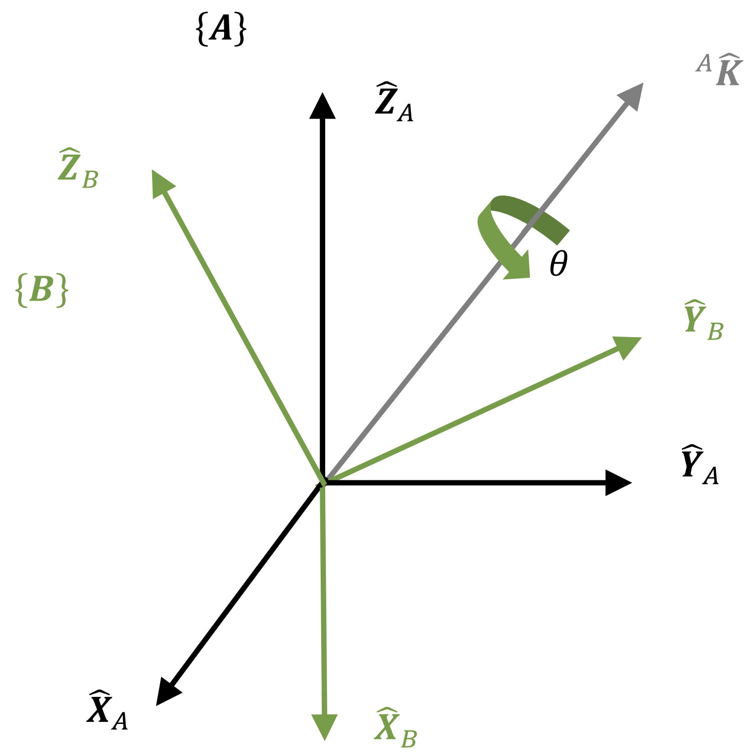
Combination of:

- **Axis**: **unit vector** that represents the **direction around which we rotate**.
- **Angle**: **scalar** that represents **how much** we rotate, using the **right-hand rule**.

Can be **given as**:

$$\theta,\ \hat{\boldsymbol{K}} = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \qquad \text{or} \qquad \boldsymbol{K} = \theta\hat{\boldsymbol{K}} = \begin{bmatrix} \theta k_x \\ \theta k_y \\ \theta k_z \end{bmatrix}$$

separate **combined***



*Universal Robots uses this representation in their Graphical User Interface

Equivalent Angle-Axis to Rotation Matrix

When we defined $\boldsymbol{R}_X(\theta),\ \boldsymbol{R}_Y(\theta),\ \boldsymbol{R}_Z(\theta)$ in Lecture 4, we were already doing this.

We can do this for **any arbitrary angle:**

$$
{}^{A}_{B}\boldsymbol{R}_K(\theta) = \begin{bmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{bmatrix}, \text{ with } v\theta = 1 - c\theta
$$

Deriving this is **quite tedious**. You can try this as an exercise (2.6).

SDU ROBOTICS

Rotation Matrix to Equivalent Angle-Axis

Given a rotation matrix:

$$ {}^{A}_{B}\boldsymbol{R}_K(\theta) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

An angle-axis pair can be computed as:

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

$$ \widehat{\boldsymbol{K}} = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} $$

Note that:

$$ \left(\theta, \widehat{\boldsymbol{K}}\right) = \left(-\theta, -\widehat{\boldsymbol{K}}\right) $$

Recap: What have we discussed today

- The **problems** with **mathematically modeling** 3D **orientations/rotations**.
- **Angle-set conventions**
  - **Fixed angles**
  - **Euler angles**
  - **Gimbal Lock**
- **Equivalent Angle-Axis**
- **Conversions** between representations

Take home message:

There are **various ways to represent orientation** in 3D space.

When **choosing** one or another, it is important to keep in mind the **application** (e.g. calculations or visualization) and the **limitations of the representation** (e.g. **Gimbal lock**).

SDU ROBOTICS

Thank you for today.

Iñigo Iturrate

Ø27-604-3

inju@mmmi.sdu.dk

SDU ROBOTICS