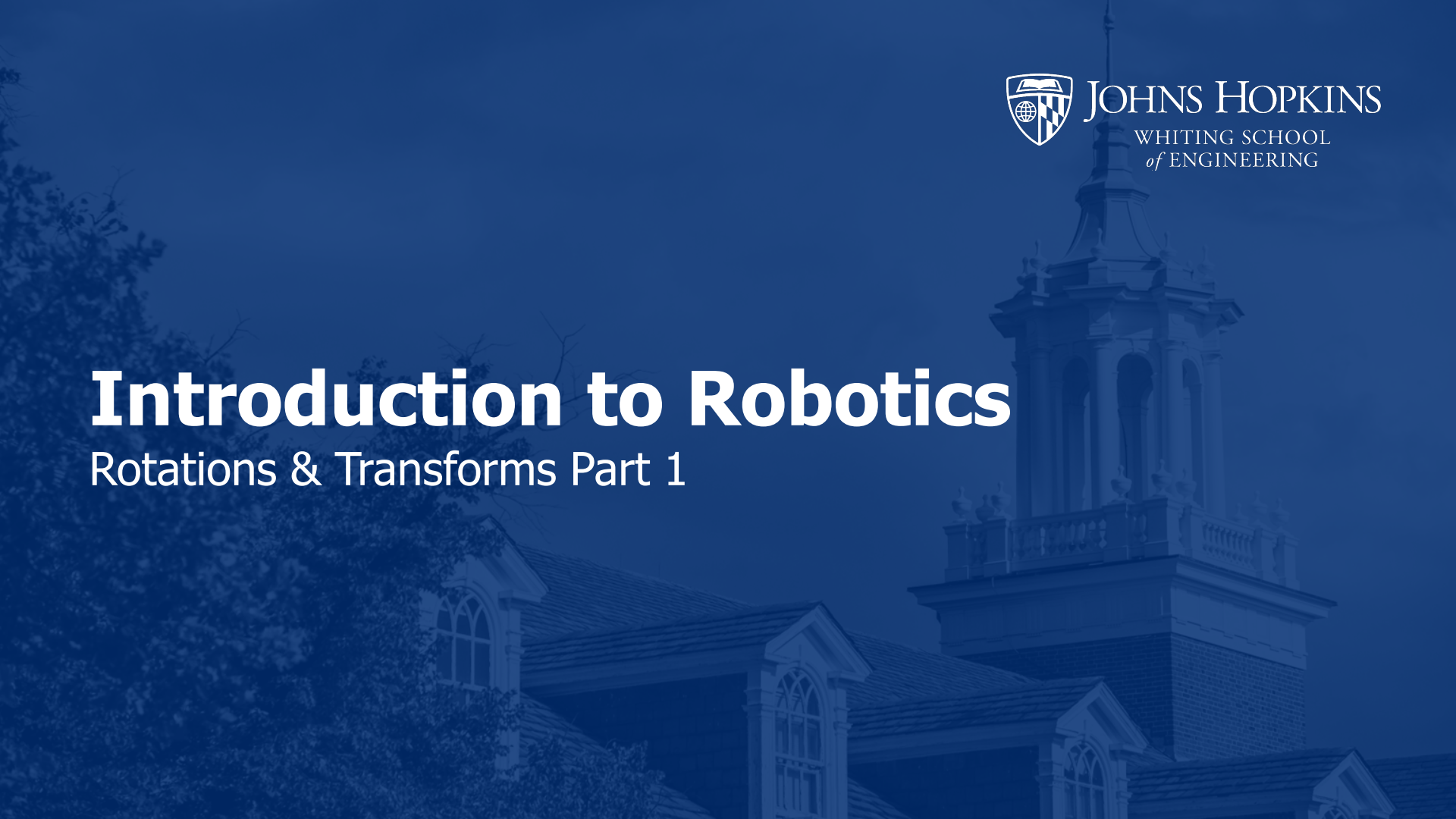
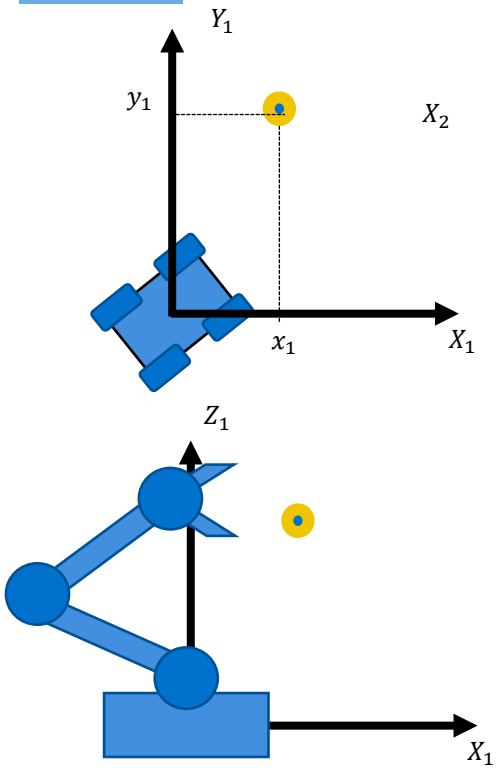


Introduction to Robotics

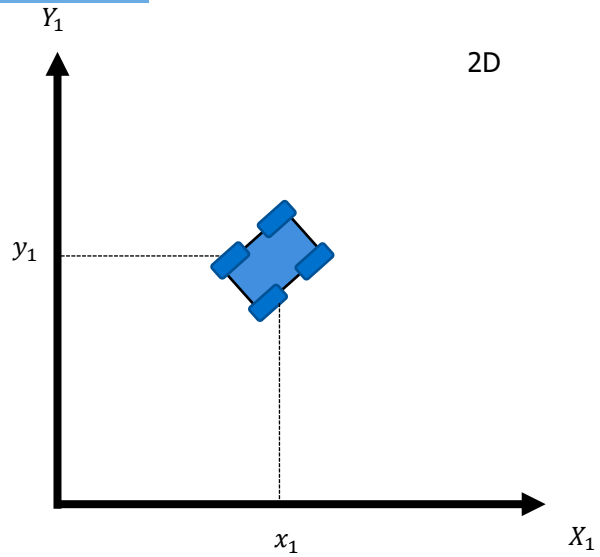
Rotations & Transforms Part 1

Frames of Reference

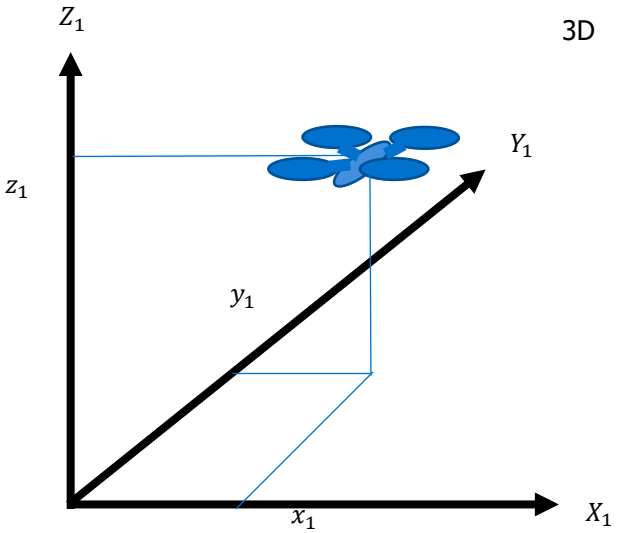


- All Robotic planning tasks require the use of multiple frames of reference.
- We must consider both global coordinate systems as well as relative coordinates to the robot and its components
- Where is the goal in relation to the robots local coordinate system?
- Where is the goal in relation to the end-effector of the robot arm?

2D & 3D Coordinate Systems



$$P_1 = \begin{bmatrix} x \\ y \end{bmatrix}_1$$



$$P_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_1$$

The Space of Rotations

- Special Orthogonal Group 3

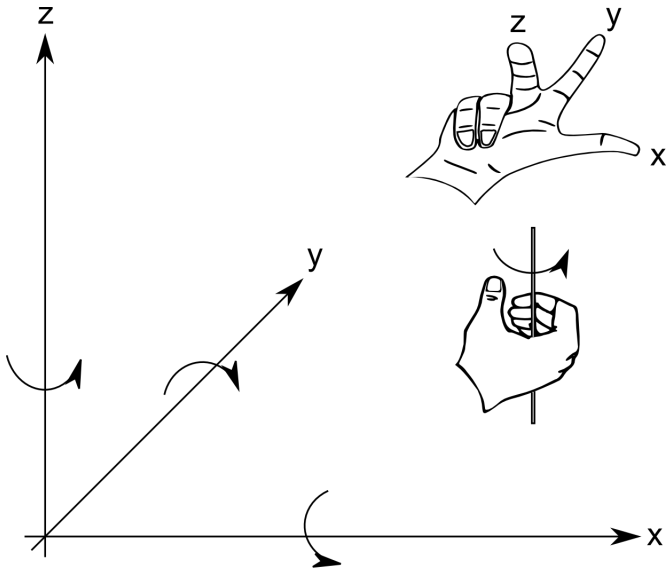
$$SO(3) = \left\{ R \in R^{3\times 3} \mid RR^T = I, \det(R) = +1 \right\}$$

- Why it's a group
  - Closed under multiplication: $R_1, R_2 \in SO(3)$ then $R_1 R_2 \in SO(3)$
  - Has an identity $\exists \mathrm{I} \in SO(3) \text{ s.t. } \mathrm{I} R_1 = R_1$
  - Has a unique inverse…
  - Is associative….

- Why orthogonal
  - Vectors in matrix are orthogonal
- Why it's special: $\det(R) = +1, \text{ not } \det(R) = \pm 1$
  - Which means it is a right-hand coordinate systems

The Right Hand Rule

- Rotation in the positive direction can be determined by the right hand rule.
  - With the right thumb in the direction of the axis the fingers of the hand curl in the direction of the positive rotation



A coordinate system indicating the direction of the coordinate axes and rotation around them. These directions have been derived using the right-hand rules.

3D Rotation Representations

You need at least three numbers to represent an arbitrary rotation in $SO(3)$

- Euler Angles
  - ZYZ
  - ZYX (roll, pitch, yaw)
- Axis-Angle

There exists one four-number representation:

- Quaternions

Euler Angle Representation

Common convention has the positive X axis aligned of the vehicle's direction of motion and positive Y axis orthogonal to that (on the left side)

- Roll (x axis)

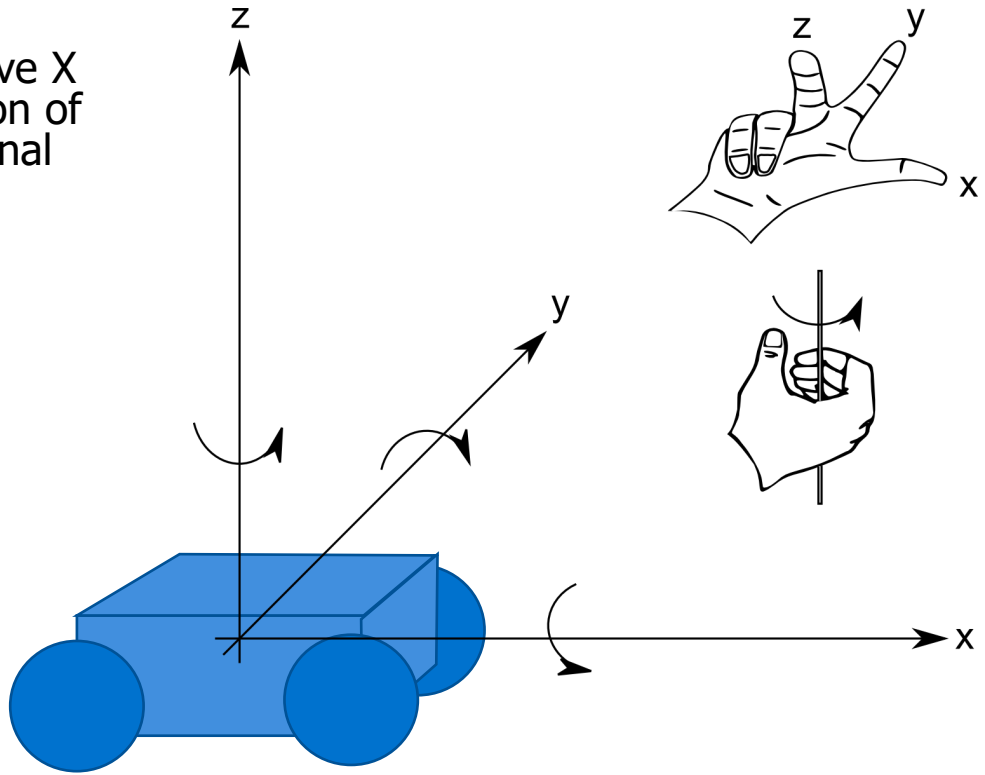
$$R_x(\alpha)$$

- Pitch (y axis)

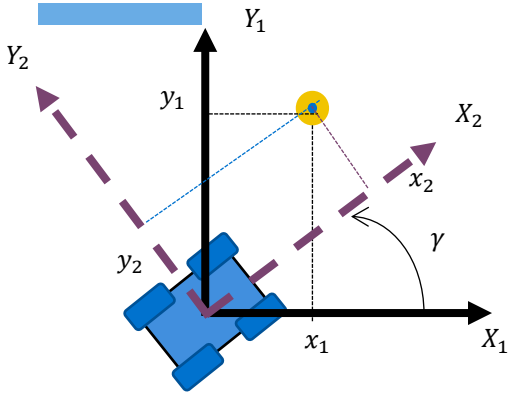
$$R_y(\beta)$$

- Yaw (z axis)

$$R_z(\gamma)$$



Rotations



In Euler angles, each rotation is imagined to be represented in the post-rotation coordinate frame of the last rotation

$$P_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_1 = R_z P_2 = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_2$$

$$= \begin{bmatrix} \cos\gamma * x - \sin\gamma * y \\ \sin\gamma * \mathrm{x} + \cos\gamma * y \\ z \end{bmatrix}_2$$

Roll

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

Pitch

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

Yaw

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

Order of Rotations

- Roll, Pitch, Yaw

$$R_{ZYX}(\gamma,\beta,\alpha)=\begin{bmatrix}\cos\gamma & -\sin\gamma & 0\\ \sin\gamma & \cos\gamma & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\beta & 0 & \sin\beta\\ 0 & 1 & 0\\ -\sin\beta & 0 & \cos\beta\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\alpha & -\sin\alpha\\ 0 & \sin\alpha & \cos\alpha\end{bmatrix}$$

$$R_{ZYX}(\gamma,\beta,\alpha)=\begin{bmatrix}\cos\gamma & -\sin\gamma & 0\\ \sin\gamma & \cos\gamma & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\beta & \sin\beta\sin\alpha & \sin\beta\cos\alpha\\ 0 & \cos\alpha & -\sin\alpha\\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha\end{bmatrix}$$

$$R_{ZYX}(\gamma,\beta,\alpha)=\begin{bmatrix}\cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha-\sin\gamma\cos\alpha & \cos\gamma\sin\beta\cos\alpha+\sin\gamma\sin\alpha\\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha+\cos\gamma\cos\alpha & \sin\gamma\sin\beta\cos\alpha-\cos\gamma\sin\alpha\\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha\end{bmatrix}$$

Rotation Matrix Back to Euler Angles

$$R_{ZYX}(\gamma,\beta,\alpha)=\begin{bmatrix}\cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha-\sin\gamma\cos\alpha & \cos\gamma\sin\beta\cos\alpha+\sin\gamma\sin\alpha\\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha+\cos\gamma\cos\alpha & \sin\gamma\sin\beta\cos\alpha-\cos\gamma\sin\alpha\\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha\end{bmatrix}$$

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

$$\alpha=\operatorname{atan}2\left(R_{32},R_{33}\right)$$

$$\beta=\operatorname{atan}2\left(-R_{31},\sqrt{R_{32}^2+R_{33}^2}\right)$$

$$\gamma=\operatorname{atan}2\left(R_{21},R_{11}\right)$$

We use atan2 rather than $\tan^{-1}$ so we can determine the quadrant

Inverse of Rotations

- Rotation matrices are **Orthogonal** which means the transpose is equal to the inverse

$$\cos\theta = \cos -\theta \qquad \sin -\theta = -\sin\theta$$

$$R_x(\alpha)^{-1} = R_x(\alpha)^T$$

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

$$R_x(-\alpha)R_x(a) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & (\cos\alpha)^2 + (\sin\alpha)^2 & \cos\alpha\sin\alpha - \cos\alpha\sin\alpha \\ 0 & \cos\alpha\sin\alpha - \cos\alpha\sin\alpha & (\sin\alpha)^2 + (\cos\alpha)^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inverse of Rotations (cont.)

- While applying a negative angle may work for a single axis it does not work in the 3D case.
- Instead each rotation must be undone in reverse order
- Or just use the Transpose!

$$R_{ZYX}(\gamma,\beta,\alpha)^{-1} \neq R_{ZYX}(-\gamma,-\beta,-\alpha)$$

$$R_{ZYX}(\gamma,\beta,\alpha)^{-1} = R_{ZYX}(\gamma,\beta,\alpha)^{T} = R_{XYZ}(-\alpha,-\beta,-\gamma)$$

$$R_{ZYX}(\gamma,\beta,\alpha)^{-1} = \begin{bmatrix} \cos\gamma\cos\beta & \sin\gamma\cos\beta & -\sin\beta \\ \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \sin\gamma\sin\beta\sin\alpha + \cos\gamma\cos\alpha & \cos\beta\sin\alpha \\ \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha & \cos\beta\cos\alpha \end{bmatrix}$$

$$R_{ZYX}(\gamma,\beta,\alpha) = \begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha + \cos\gamma\cos\alpha & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix}$$

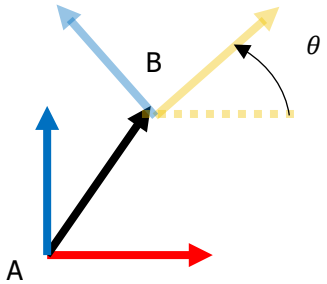
2D Rigid Motion

Special Orthogonal (SO)

Combine position and orientation:

Special Euclidean Group: $SE(2)$

$$SE(2) = \left\{(t, R) : t \in \mathbb{R}^2, R \in SO(2)\right\} = \mathbb{R}^2 \times SO(2)$$



$t_B^A \in \mathbb{R}^2$ is the translation between A and B

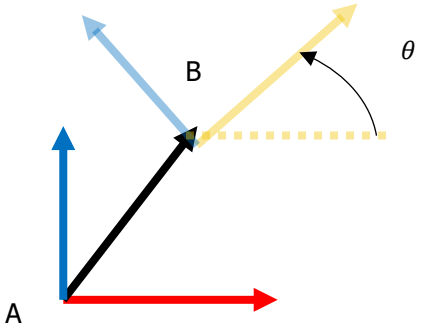
$R_B^A \in SO(2)$ Is the rotation between A and B

If $R \in SO(2)$ then $R \in \mathbb{R}^{2\times 2}, RR^T = I$ and $\det(R) = 1$

$$R_B^A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Basic Transformation

- Converts a point from Frame B into Frame A
- First apply rotation, then apply the translation



$$P^A = R_B^A P^B + t_B^A$$

Homogeneous Representation

- A 2D point is represented by appending a "1" to yield a vector in $\mathbb{R}^2, P = [x, y, 1]^T$
- A 3D point is represented by appending a "1" to yield a vector in $\mathbb{R}^3, P = [x, y, z, 1]^T$
- They are called homogenous coordinates
- The affine transformation of a point

$$P^A = R_B^A P^B + t_B^A$$

Is represented by a linear transformation using a homogenous coordinates

$$\begin{bmatrix} P^A \\ 1 \end{bmatrix} = \begin{bmatrix} R_B^A & t_B^A \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} P^B \\ 1 \end{bmatrix}$$

Affine vs Linear Transforms

- Affine Transformations may seem more immediately intuitive but are more complicated to represent transformations between multiple coordinate systems.

Affine Transformations

$$P^A = R_B^A P^B + t_B^A$$

$$P^B = R_C^B P^C + t_C^B$$

$$P^A = R_B^A \left(R_C^B P^C + t_C^B \right) + t_B^A$$

Cumbersome representation

Linear Transformations

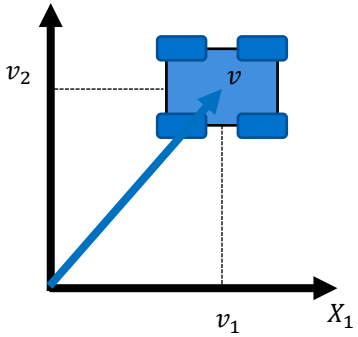
$$\begin{bmatrix} P^A \\ 1 \end{bmatrix} = \begin{bmatrix} R_B^A & t_B^A \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P^B \\ 1 \end{bmatrix} = \mathrm{E}_{\mathrm{B}}^{\mathrm{A}} \begin{bmatrix} P^B \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} P^B \\ 1 \end{bmatrix} = \begin{bmatrix} R_C^B & t_C^B \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P^C \\ 1 \end{bmatrix} = \mathrm{E}_{C}^{\mathrm{B}} \begin{bmatrix} P^C \\ 1 \end{bmatrix}$$

$$P^A = E_B^A E_C^B \begin{bmatrix} P^C \\ 1 \end{bmatrix}$$

Compact representation

Translation



Translation

$$T_v = \begin{bmatrix} 1 & 0 & 0 & v_1 \\ 0 & 1 & 0 & v_2 \\ 0 & 0 & 1 & v_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

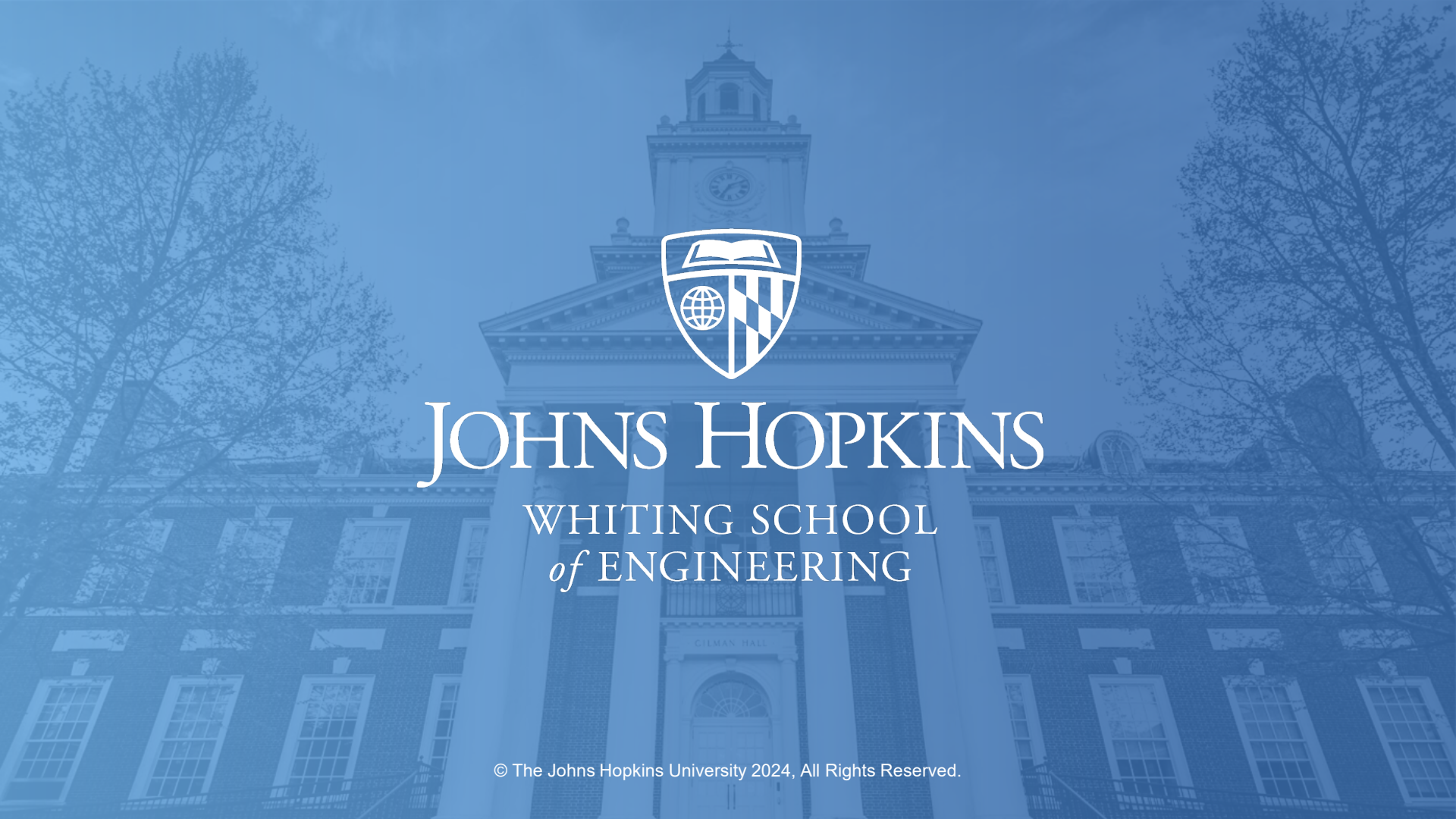
$$P_1 = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_1 = T_z P_2 = \begin{bmatrix} 1 & 0 & 0 & v_1 \\ 0 & 1 & 0 & v_2 \\ 0 & 0 & 1 & v_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_2 = \begin{bmatrix} x+v_1 \\ y+v_2 \\ z+v_3 \\ 1 \end{bmatrix}_2$$

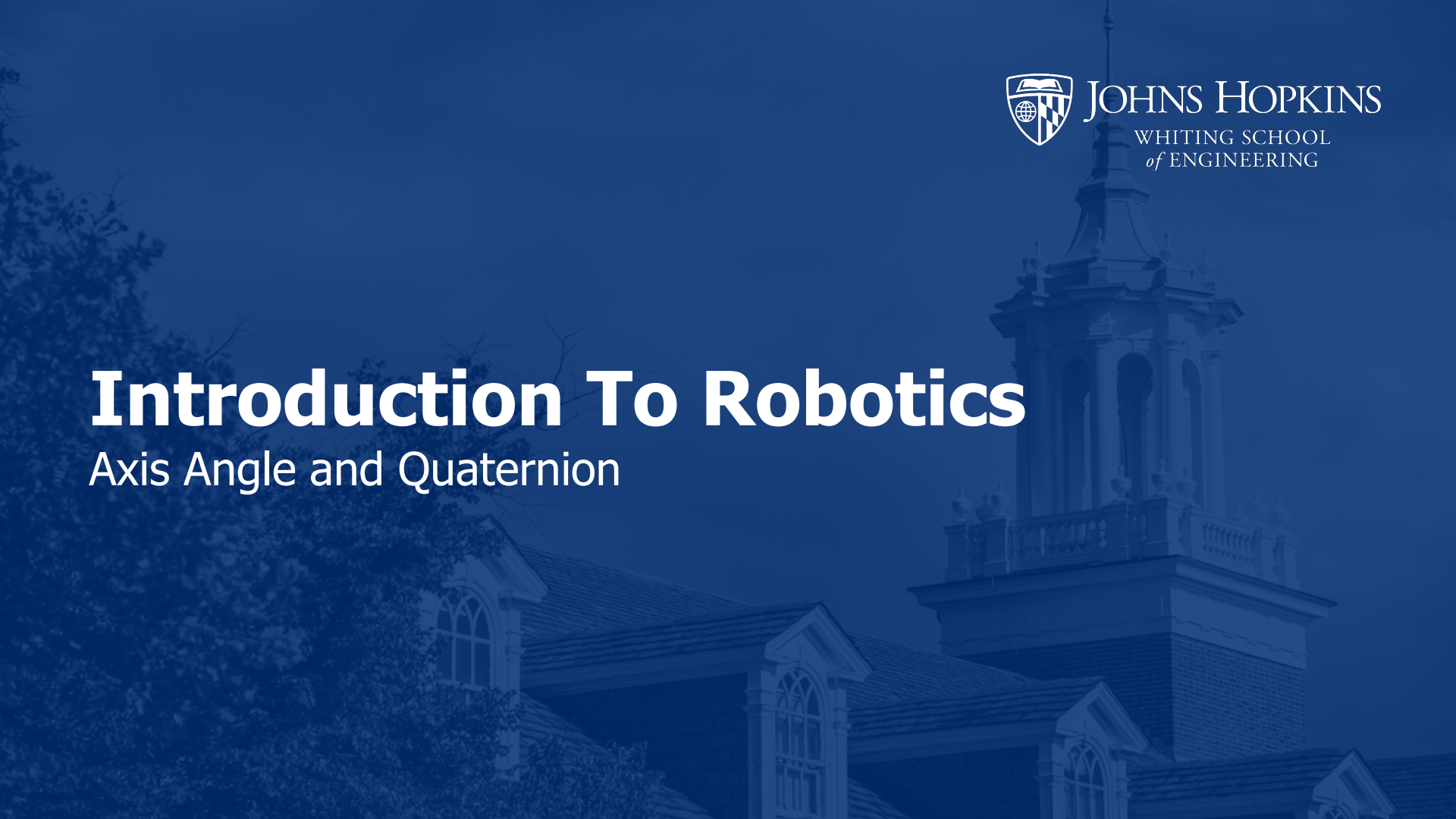
To apply a translation transformation we must write our 3D vector as 4 homogenous coordinates.

$$P_1 = [x, y, z] \rightarrow [x, y, z, 1]$$

Inverse Translation

$$T_v^{-1} = T_{-v}$$



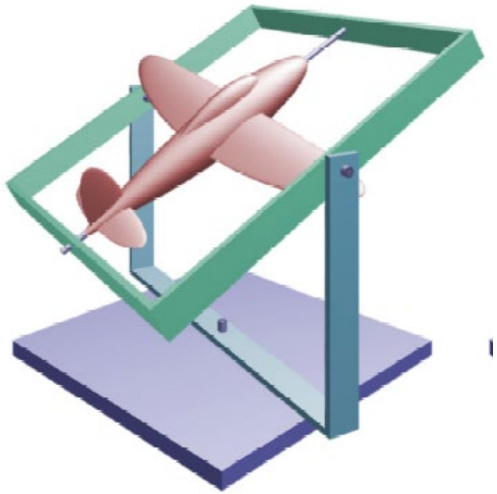


Introduction To Robotics

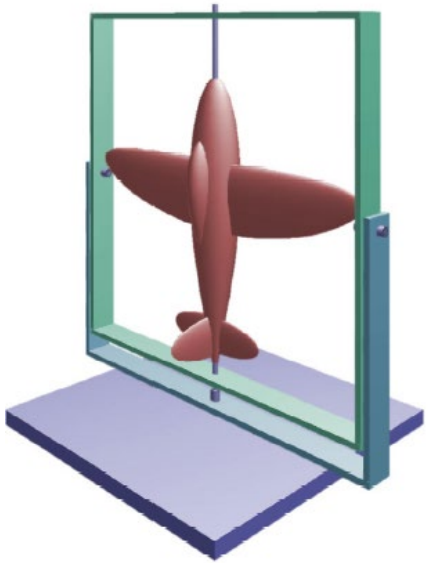
Axis Angle and Quaternion

Issue with Euler Angles – Gimbal Lock

Roll, pitch, yaw



Gimbal lock: reduced DOF due to overlapping axes



Ref: http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

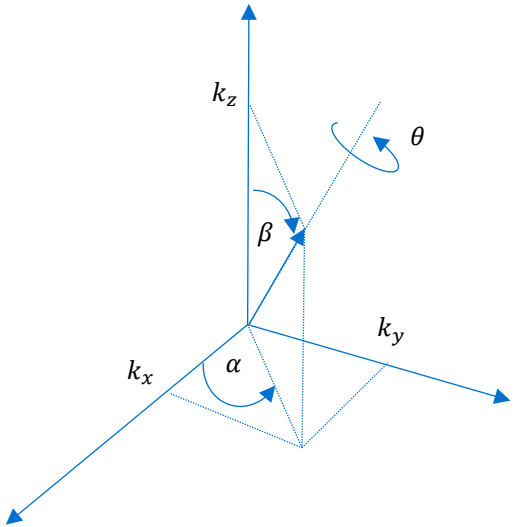
Axis-Angle Representation

Theorem: (Euler) Any orientation, $R \in SO(3)$

is equivalent to a rotation about a fixed axis

$k \in \mathbb{R}^3$ through angle $\theta \in [0, 2\pi)$

Axis: $k = \begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix}$ Angle: Θ



Rotating by $R_{k,\theta}$

is equivalent to we rotating vector k to align the Z axis, rotating by Θ then rotating it back

$$R_{k,\theta} = R_{z,\alpha} R_{y,\beta} R_{z,\theta} R_{y,-\beta} R_{z,-\alpha}$$

Rodrigues' Rotation Formula

$$p = p_{\parallel} + p_{\perp}$$

$$p_{\parallel} = (k \cdot p)k$$

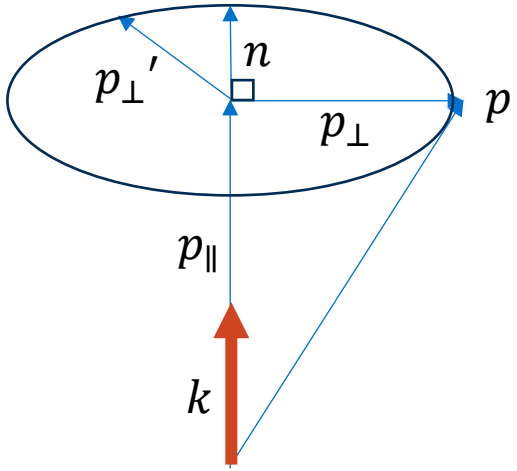
$$p_{\perp} = p - p_{\parallel}$$

$$p_{\perp} = p - (k \cdot p)k$$

$$p' = p_{\parallel} + p_{\perp}'$$

$$n = k \times p$$

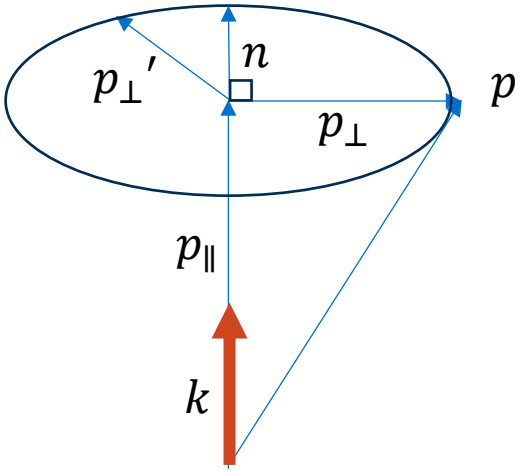
$$p_{\perp}' = \cos\theta\, p_{\perp} + \sin\theta\, (k \times p)$$



assume k is a unit vector

Rodrigues' Rotation Formula

$$p' = p_{\parallel} + p_{\perp}' \qquad n = k \times p \qquad p_{\parallel} = (k \cdot p) \cdot k$$



Convert k into a skew matrix

$$\hat{k} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$$

$$n = k \times p = \hat{k}p \qquad p_{\perp} = -k \times n = -\hat{k}n$$

$$p_{\perp} = -\hat{k}^2 p \qquad p_{\parallel} = (k^T p)^T k$$

assume k is a unit vector

Rodrigues' Rotation Formula

$$p' = p_{\parallel} + p_{\perp}' \qquad p_{\parallel} = (k^T p)k \qquad p_{\perp} = -\hat{k}^2 p$$

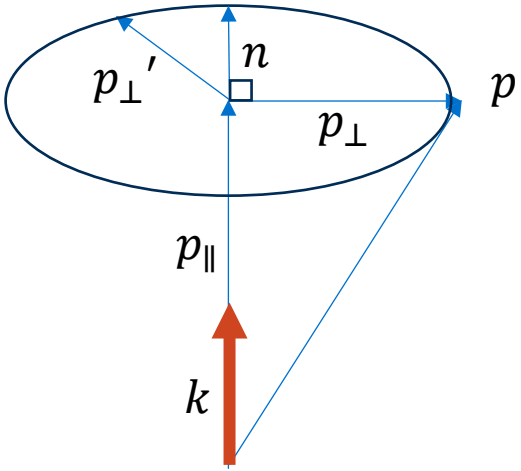
$$p_{\perp}' = \cos\theta\, p_{\perp} + \sin\theta\, n$$

$$p_{\perp}' = -\cos\theta\, \hat{k}^2 p + \sin\theta\, \hat{k} p$$

$$p' = (k^T p)k - \cos\theta\, \hat{k}^2 p + \sin\theta\, \hat{k} p \qquad p_{\parallel} = p - p_{\perp}$$

$$p' = \left(p + \hat{k}^2 p\right) - \cos\theta\, \hat{k}^2 p + \sin\theta\, \hat{k} p$$

$$p' = \left[I + \sin\theta\, \hat{k} + (1 - \cos\theta)\hat{k}^2\right]p = Rp$$



assume k is a unit vector

Rodrigues' Rotation Formula

$$p' = \left[I + \sin\theta\,\hat{k} + (1-\cos\theta)\hat{k}^2\right]p = Rp$$

$$R = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} + \sin\theta \begin{bmatrix}0 & -k_z & k_y\\ k_z & 0 & -k_x\\ -k_y & k_x & 0\end{bmatrix} + (1-\cos\theta)\begin{bmatrix}k_x^2-1 & k_xk_y & k_xk_z\\ k_xk_y & k_y^2-1 & k_yk_z\\ k_xk_z & k_yk_z & k_z^2-1\end{bmatrix}$$

$$R_{k,\theta} = \begin{bmatrix} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k_z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k_z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{bmatrix} \qquad v_\theta = 1 - c_\theta$$

Axis-Angle Representation (3)

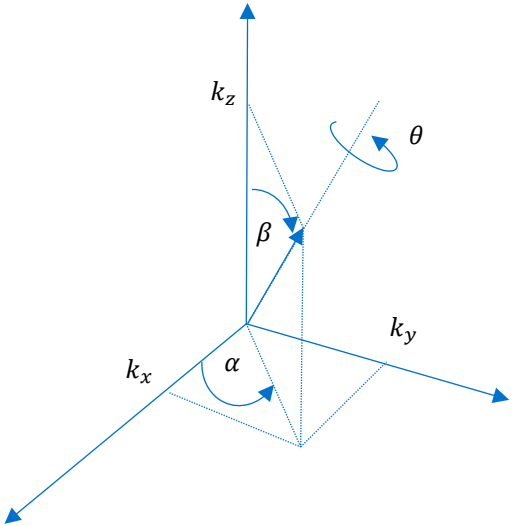
We can now solve for Θ and *k* if we have already have a rotation matrix

(i.e., if we want to convert from a Euler angle rotation matrix)

$$\theta = \cos^{-1}\left(\frac{\text{trace}(R)-1}{2}\right)$$

$$k = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32}-R_{23} \\ R_{13}-R_{31} \\ R_{21}-R_{12} \end{bmatrix}$$

$$\text{trace}(R) = R_{11}+R_{22}+R_{33}$$

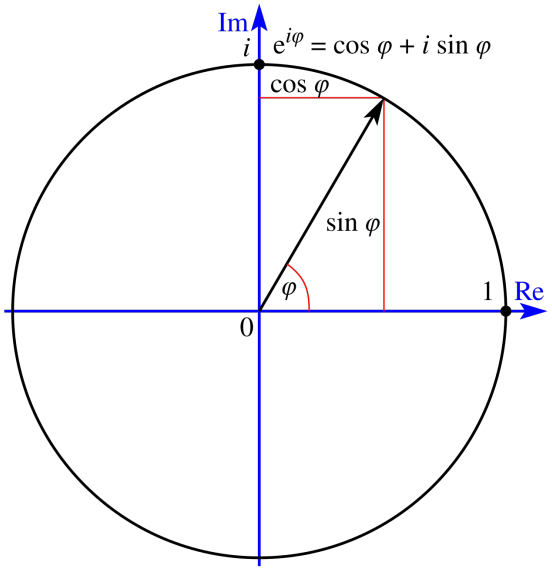


Quaternions

- Quaternions have nice geometrical interpretation.
- Quaternions have advantages in representing rotation.
- Quaternions are robust to rounding errors during computation
  - Quaternion: Round-off errors just lead to a slightly different angle
  - Rotation Matrix: Round-off errors lead to a matrix that is not orthogonal
- Quaternions are the default method of representing transforms in ROS

Complex Numbers vs Quaternions

- Complex Numbers
  - Define basis elements 1 and *i*
  - Elements have form: $x + i\,y$
  - Where $i^2 = -1$
  - Euler's Formula: $e^{i\psi} = \cos\psi + \sin\psi$



- Quaternions
  - Basis elements: 1, *i, j, k*
  - Elements have form: $q_0 + q_1\,i + q_2\,j + q_3\,k$
  - Where $i^2 = j^2 = k^2 = ijk = -1$

Quaternion Notation

We can write a quaternion in several ways

$$q = q_0 + q_1\ i + q_2\ j + q_3\ k$$

$$q = (q_0, q_1, q_2, q_3)$$

$$q = (q_0, \mathbf{q}) = q_0 + \mathbf{q}$$

A point p in 3D space can be represented in quaternion space as (more on that later)

$$p = (0, \mathbf{p})$$

Properties of Quaternions

For our purposes we only consider the set of unit quaternions

$$Q^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

This is a point on a four-dimensional hyper-sphere i.e., S^3

To combine two rotations (x_0, x) and (y_0, y) together you must multiply them

The identity quaternion is $\mathrm{Q}_I = (1, 0)$

The complex conjugate is also the inverse

$$Q^* = (q_0, q)^* = (q_0, -q) = Q^{-1}$$

Given a unit axis $\hat{k}$ and an angle Θ (just like axis angle)

The associated quaternion is $Q_{\hat{k},\theta} = \left(\cos\left(\frac{\theta}{2}\right), \hat{k} \sin\left(\frac{\theta}{2}\right) \right)$

Basic Element Multiplication

Given the the orthogonal vectors *i, j, k* s.t.

$$i\ j\ k = -1$$
$$i(i\ j\ k) = i(-1)$$
$$-j\ k = -i$$
$$j\ k = i$$

Which allows us to derive the following

$$i\ j = k,\ j\ i = -k$$
$$j\ k = i,\ k\ j = -i$$
$$k\ i = j,\ i\ k = -j$$

Quaternion products *i, j, k* behave like cross product

Quaternion Product

Say we want to multiple two quaternions together

$$(z_0, z) = (x_0, x)(y_0, y)$$

$$z_0 + z_1 i + z_2 j + z_3 k = (x_0 + x_1 i + x_2 j + x_3 k)(y_0 + y_1 i + y_2 j + y_3 k)$$

$$= x_0(y_0 + y_1 i + y_2 j + y_3 k) + x_1 i(y_0 + y_1 i + y_2 j + y_3 k) + x_2 j(y_0 + y_1 i + y_2 j + y_3 k) + x_3 k(y_0 + y_1 i + y_2 j + y_3 k)$$

$$z_0 = x_0\, y_0 - x_1\, y_1 - x_2\, y_2 - x_3\, y_3$$
$$z_1 = x_0\, y_1 + x_1\, y_0 + x_2\, y_3 - x_3\, y_2$$
$$z_2 = x_0\, y_2 + x_2\, y_0 + x_3\, y_1 - x_1\, y_3$$
$$z_3 = x_0\, y_3 + x_3\, y_0 + x_1\, y_2 - x_2\, y_1$$

Alternatively we can say that

$$z_0 = x_0 y_0 - x^T y$$
$$z = x_0 y + y_0 x + x \times y$$

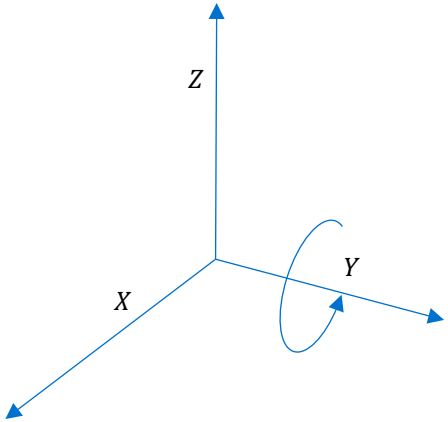
Where × is the cross product operator

Quaternion Rotations

- Take vector in $p_i \in \mathbb{R}^3$ represented as the quaternion $P_i = (0, p_i)$
- You can rotate P_a from frame a to b: $P_b = Q_{ba} P_a Q_{ba}^*$
- Composition $Q_{ca} = Q_{cb}\ Q_{ba}$
- Inversion $Q_{cb} = Q_{ca} Q_{ba}^{-1}$

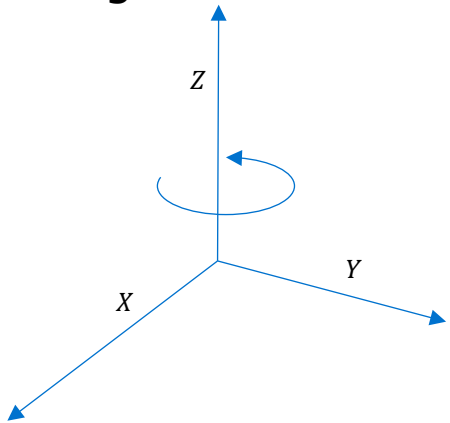
Quaternion Composition Example

Rotate 90 degrees around Y (pitch)



$$q_a = \cos\frac{\pi}{4} + j\sin\frac{\pi}{4}$$

Rotate 90 degrees around Z (yaw)



$$q_b = \cos\frac{\pi}{4} + k\sin\frac{\pi}{4}$$

$$q_{ab} = q_a q_b$$

Quaternion Composition Example

$$\cos\frac{\pi}{4} = \sin\frac{\pi}{4} = \frac{\sqrt{2}}{2} \qquad q_a = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2} \qquad q_b = \frac{\sqrt{2}}{2} + k\frac{\sqrt{2}}{2}$$

$$q_{ab} = q_a q_b = \left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)\left(\frac{\sqrt{2}}{2} + k\frac{\sqrt{2}}{2}\right)$$

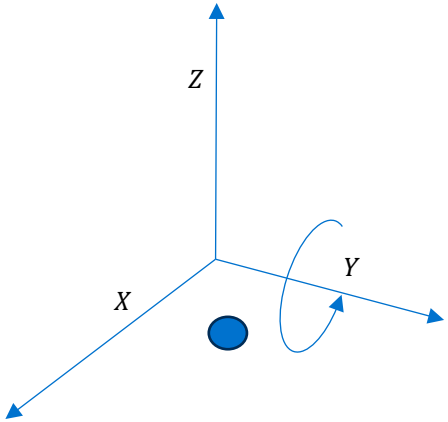
$$= \frac{\sqrt{2}}{2}\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\frac{\sqrt{2}}{2}j + \frac{\sqrt{2}}{2}\frac{\sqrt{2}}{2}k + \frac{\sqrt{2}}{2}\frac{\sqrt{2}}{2}jk = \frac{1}{2}(1 + j + k + i)$$

$$\cos\frac{\theta}{2} = \frac{1}{2} \qquad \theta = \frac{2\pi}{3} \qquad \sin\frac{\theta}{2} = \frac{\sqrt{3}}{2}$$

$$q_{ab} = \frac{1}{2} + \frac{\sqrt{3}}{2}\left(\frac{1}{\sqrt{3}}i + \frac{1}{\sqrt{3}}j + \frac{1}{\sqrt{3}}k\right)$$

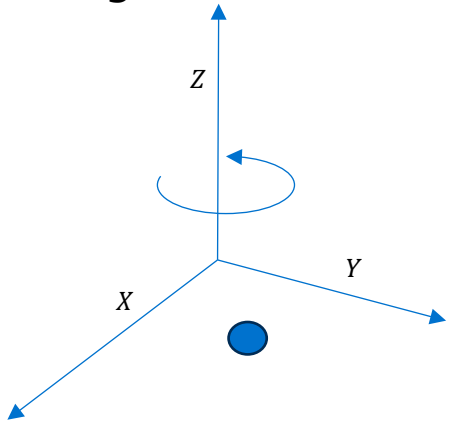
Quaternion Rotation

Rotate 90 degrees around Y (pitch)



$$q_a = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$$

Rotate 90 degrees around Z (yaw)



$$q_b = \frac{\sqrt{2}}{2} + k\frac{\sqrt{2}}{2}$$

Rotate the following point

$$p = [1,1,0]$$

Quaternion Rotation

$$p = [1,1,0] = 0 + i + j \qquad q_b = \frac{\sqrt{2}}{2} + k\frac{\sqrt{2}}{2} \qquad q_b^* = \frac{\sqrt{2}}{2} - k\frac{\sqrt{2}}{2}$$

$$p_b' = q_b p q_b^* = \left(\frac{\sqrt{2}}{2} + k\frac{\sqrt{2}}{2}\right)(0 + i + j)\left(\frac{\sqrt{2}}{2} - k\frac{\sqrt{2}}{2}\right)$$

$$= \left(i\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i\right)\left(\frac{\sqrt{2}}{2} - k\frac{\sqrt{2}}{2}\right)$$

$$= (j\sqrt{2})\left(\frac{\sqrt{2}}{2} - k\frac{\sqrt{2}}{2}\right) \qquad = 0 + \mathrm{j} - \mathrm{i} \qquad = [-1,1,0]$$

Quaternion Rotation

$$p = [1,1,0] = 0 + i + j \qquad q_a = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2} \qquad q_a^* = \frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}$$

$$p'_a = q_a p q_a^* = \left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)(0 + i + j)\left(\frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}\right)$$

$$= \left(i\frac{\sqrt{2}}{2} - k\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}\right)\left(\frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}\right)$$

$$= \left(i\frac{\sqrt{2}}{2} - k\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}\right)\left(\frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}\right) \qquad = \frac{1}{2}(i - k + j - 1 - k - i + 1 + j)$$

$$= 0 + j - k \qquad = [0,1,-1]$$

Quaternion Rotation

$$p = [1,1,0] = 0 + i + j$$

$$q_a = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$$

$$q_b = \frac{\sqrt{2}}{2} + k\frac{\sqrt{2}}{2}$$

$$q_{ab} = \frac{1}{2} + \frac{\sqrt{3}}{2}\left(\frac{1}{\sqrt{3}}i + \frac{1}{\sqrt{3}}j + \frac{1}{\sqrt{3}}k\right)$$

$$q_{ab}^* = \frac{1}{2} - \frac{\sqrt{3}}{2}\left(\frac{1}{\sqrt{3}}i + \frac{1}{\sqrt{3}}j + \frac{1}{\sqrt{3}}k\right)$$

EXERCISE: Compute the following

$$p' = q_{ab} p q_{ab}^*$$

Quaternion to Rotation Matrix

Taking the axis-angle representation $Q_{\hat{k},\theta} = \left(\cos\left(\frac{\theta}{2}\right), \hat{k}\sin\left(\frac{\theta}{2}\right) \right)$

$$= \cos\left(\frac{\theta}{2}\right) + \hat{k}\sin\left(\frac{\theta}{2}\right) + \hat{k}\sin\left(\frac{\theta}{2}\right)j + \hat{k}\sin\left(\frac{\theta}{2}\right)k$$

$$q_w = \cos\frac{\theta}{2}$$

$$\theta = 2\cos^{-1} q_w$$

$$\hat{k} = \frac{[q_x, q_y, q_z]}{\sin\frac{\theta}{2}} = \frac{[q_x, q_y, q_z]}{\sin(\cos^{-1} q_w)}$$

$$k_x = \frac{q_x}{\sin(\cos^{-1} q_w)}$$

$$k_y = \frac{q_y}{\sin(\cos^{-1} q_w)}$$

$$k_z = \frac{q_y}{\sin(\cos^{-1} q_w)}$$

Quaternion to Rotation Matrix

$$R = \begin{bmatrix} 1 + (1 - \cos\theta)(-k_y^2 - k_z^2) & -\sin\theta\, k_z + (1 - \cos\theta)k_x k_y & \sin\theta\, k_y + (1 - \cos\theta)k_x k_z \\ \sin\theta\, k_z + (1 - \cos\theta)k_x k_y & 1 + (1 - \cos\theta)(-k_x^2 - k_z^2) & -\sin\theta\, k_x + (1 - \cos\theta)k_y k_z \\ -\sin\theta\, k_y + (1 - \cos\theta)k_x k_z & \sin\theta\, k_x + (1 - \cos\theta)k_y k_z & 1 + (1 - \cos\theta)(-k_x^2 - k_y^2) \end{bmatrix}$$

$$m_{11} = 1 + (1 - \cos\theta)\left(-k_y^2 - k_z^2\right)$$

$$= 1 + (1 - \cos(2\cos^{-1} q_w)\left(-\left(\frac{q_y}{\sin(\cos^{-1} q_w)}\right)^2 - \left(\frac{q_z}{\sin(\cos^{-1} q_w)}\right)^2\right)$$

$$= 1 + \frac{(1 - \cos(2\cos^{-1} q_w)}{\sin^2(\cos^{-1} q_w)}\left(-q_y^2 - q_z^2\right)$$

Quaternion to Rotation Matrix

$$m_{11} = 1 + \frac{(1 - \cos(2\cos^{-1} q_w)}{\sin^2\ (\cos^{-1} q_w)}\left(-q_y^2 - q_z^2\right)$$

$$m_{11} = 1 + \frac{(1 - (1 - 2\sin^2(\cos^{-1} q_w))}{\sin^2(\cos^{-1} q_w)}\left(-q_y^2 - q_z^2\right)$$

$$m_{11} = 1 + \frac{-2\sin^2(\cos^{-1} q_w)}{\sin^2(\cos^{-1} q_w)}\left(-q_y^2 - q_z^2\right)$$

$$m_{11} = 1 + 2\left(-q_y^2 - q_z^2\right)$$

Remember the trig identity

$$\cos 2\theta = \cos^2\theta - \sin^2\theta = 2\cos^2\theta - 1 = 1 - 2\sin^2\theta$$

Quaternion Rotation Matrix

Both formulations are correct, the top version is simply more common.

$$R(Q) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix}$$

$$R(Q) = \begin{bmatrix} 2(q_0^2+q_1^2)-1 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 2(q_0^2+q_1^2)-1 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 2(q_0^2+q_3^2)-1 \end{bmatrix}$$

$$1-2(q_2^2+q_3^2) = 2(q_0^2+q_1^2)-1$$

$$2 = 2(q_0^2+q_1^2+q_2^2+q_3^2)$$

$$1 = (q_0^2+q_1^2+q_2^2+q_3^2)$$

Definition of a unit quaternion

Axis angle representation (k, θ)

$$q = \left[\cos\left(\frac{\theta}{2}\right), k_x\sin\left(\frac{\theta}{2}\right), k_y\sin\left(\frac{\theta}{2}\right), k_z\sin\left(\frac{\theta}{2}\right)\right]$$

$$1 = (q_0^2+q_1^2+q_2^2+q_3^2)$$

$$1 = \left(\cos^2\left(\frac{\theta}{2}\right) + k_x^2\sin^2\left(\frac{\theta}{2}\right) + k_y^2\sin^2\left(\frac{\theta}{2}\right) + k_z^2\sin^2\left(\frac{\theta}{2}\right)\right)$$

$$1 = \left(\cos^2\left(\frac{\theta}{2}\right) + \sin^2\left(\frac{\theta}{2}\right)(k_x^2+k_y^2+k_z^2)\right)$$

Rotation Matrix to Quaternion

- For rotation matrices R such that $\text{trace}(R) \neq \pm 1$

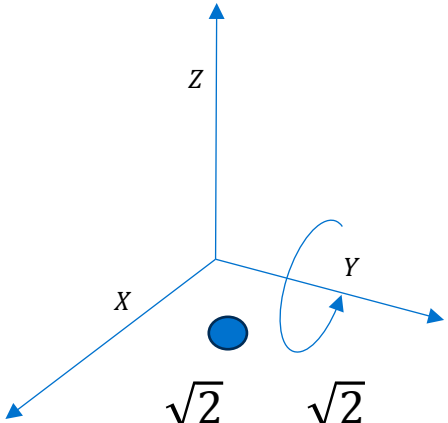
  it is possible to solve for the quaternion values from the rotation matrix in the same way we do for the axis-angle values

$$q_0 = \frac{1}{2}\sqrt{1 + \text{trace}(R)}$$

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

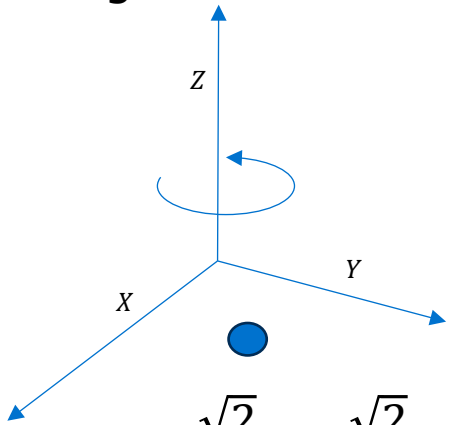
Suggested Exercise

Rotate 90 degrees around Y (pitch)



$$q_a = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$$

Rotate 90 degrees around Z (yaw)



$$q_b = \frac{\sqrt{2}}{2} + k\frac{\sqrt{2}}{2}$$

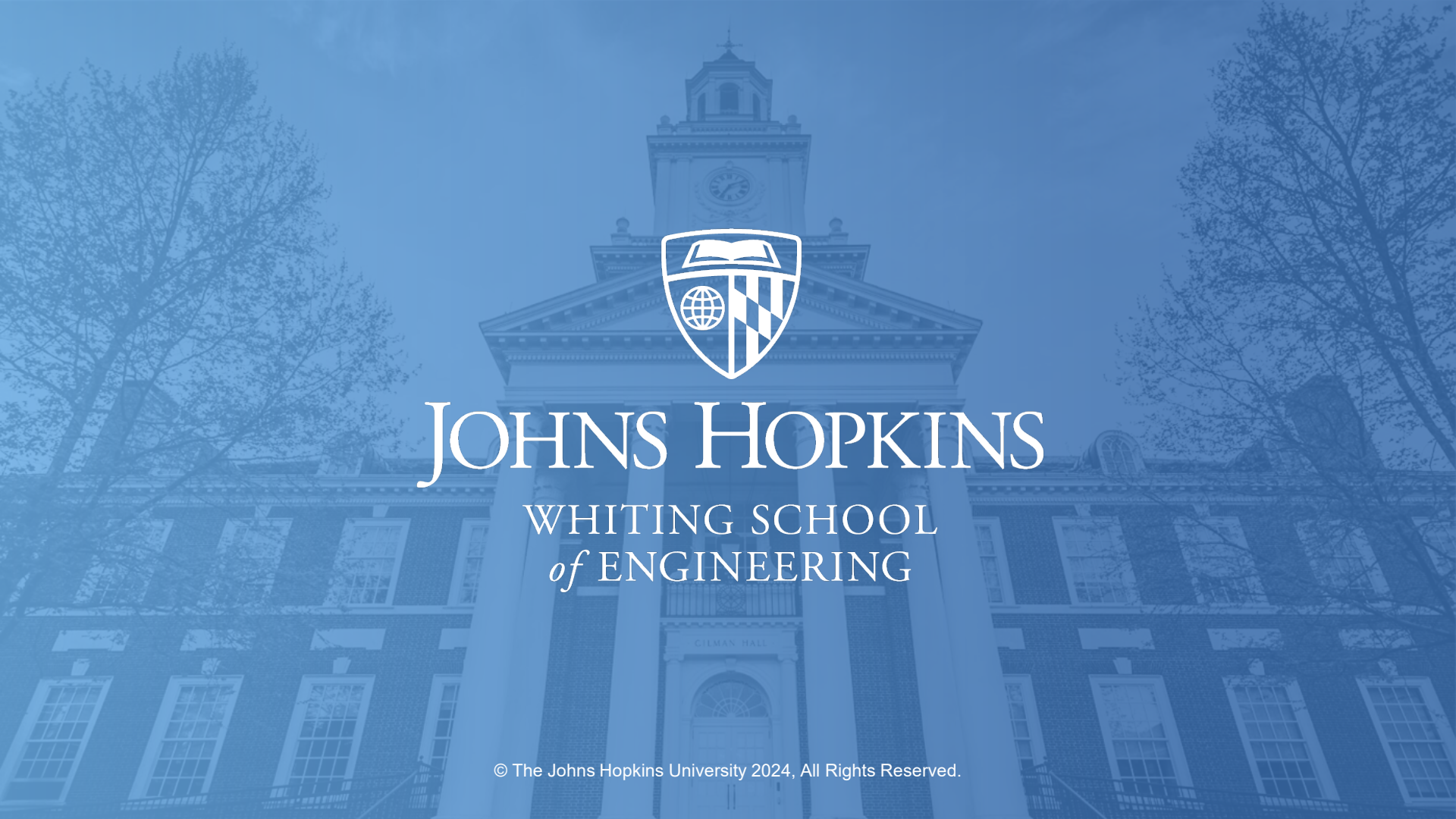
EXERCISE:

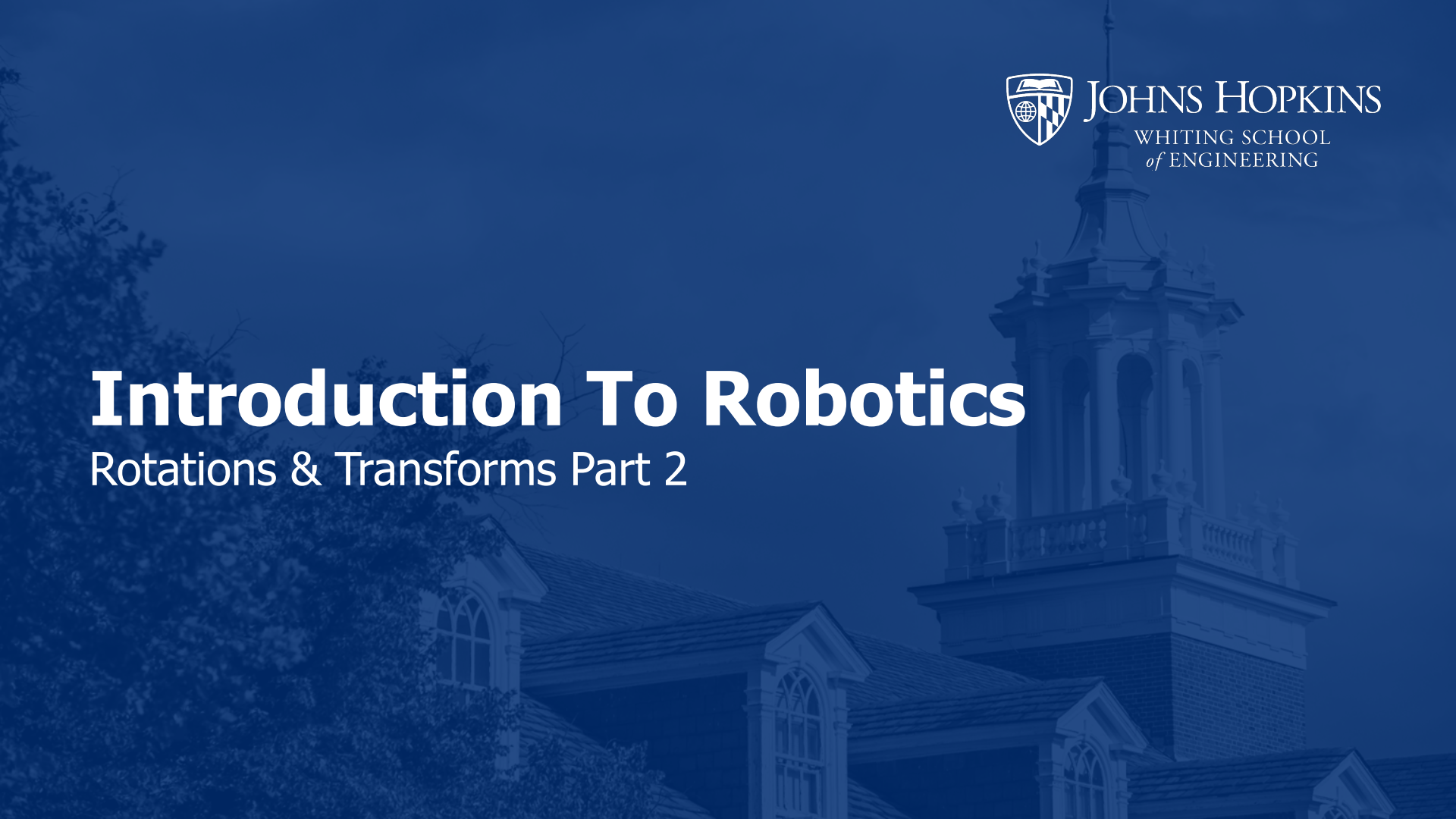
Confirm $\mathrm{R}(q_a) = R_{ZYX}\left(\frac{\pi}{2}, 0, 0\right)$, $\mathrm{R}(q_b) = R_{ZYX}\left(0, \frac{\pi}{2}, 0\right)$, and $\mathrm{R}(q_{ab}) = R_{ZYX}\left(\frac{\pi}{2}, \frac{\pi}{2}, 0\right)$

Transformations with Quaternions

- Given an
  - original point p with orientation (as a quaternion) q
  - A translation t_a and a rotation (as a quaternion) q_a
- Compute a new position p'_a and orientation q' as follows:

$$p'_a = q_a p q_a^* + t_a$$
$$q' = q_a q$$



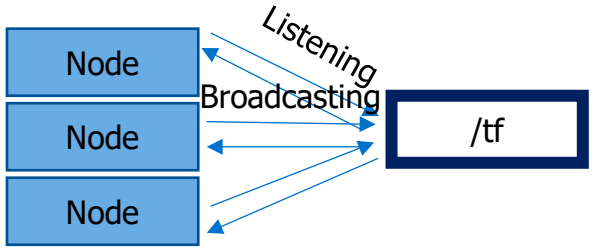


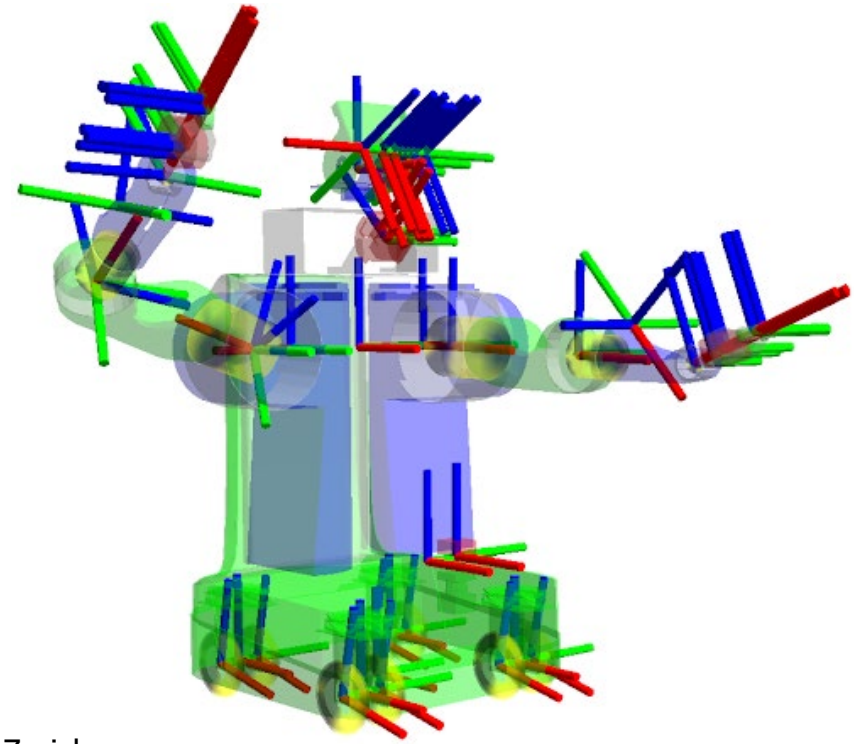
Introduction To Robotics

Rotations & Transforms Part 2

TF Transformation System

- Tool for keeping track of coordinate frames over time
- Maintains relationship between coordinate frames in a tree structure buffered in time
- Let's the user transform points, vectors, etc. between coordinate frames at desired time
- Implemented as a publisher/subscriber model on topics /tf and /tf_static





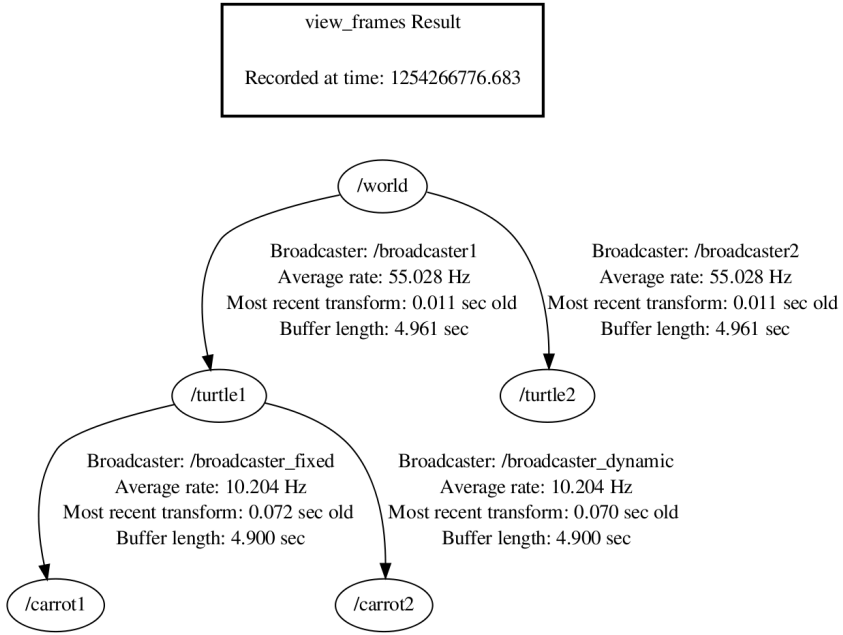
Adapted from Programming for Robotics, Bjelonic et.al ETH Zurich

Transform Tree

- TF listeners user a buffer to listen to all broadcasted transforms
- Query for specific transforms from the transform tree

tf2_msgs/TFMessage.msg

```
geometry_msgs/TransformStamped[] transforms
      std_msgs/Header header
            uint32 seqtimestamp
            string frame_id
            string child_frame_id
      geometry_msgs/Transform transform
            geometry_msgs/Vector3 translation
            geometry_msgs/Quaternion rotation
```



TF Tools

Command Line+

Print information on the current transformation tree

```
ros2 run tf2_ros tf2_monitor
```

Print information about the transform between two frames

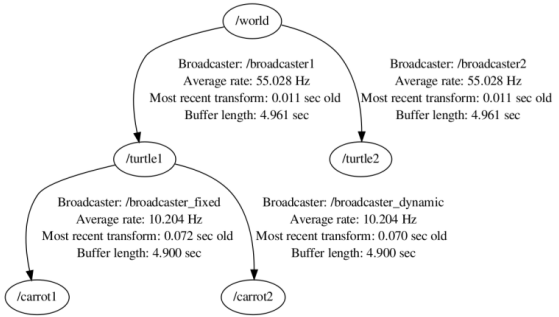
```
ros2 run tf2_ros tf2_echo foo bar
```

At time 0.0
- Translation: [1.000, 2.000, 3.000]
- Rotation: in Quaternion [-0.475, -0.076, 0.240, 0.843]

View Frames

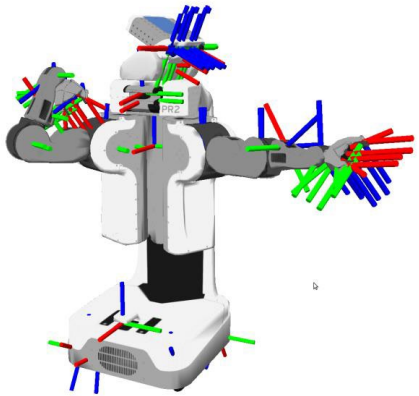
Creates a visual graph (PDF) of the transform tree

```
ros2 run tf2_tools
view_frames.py
```



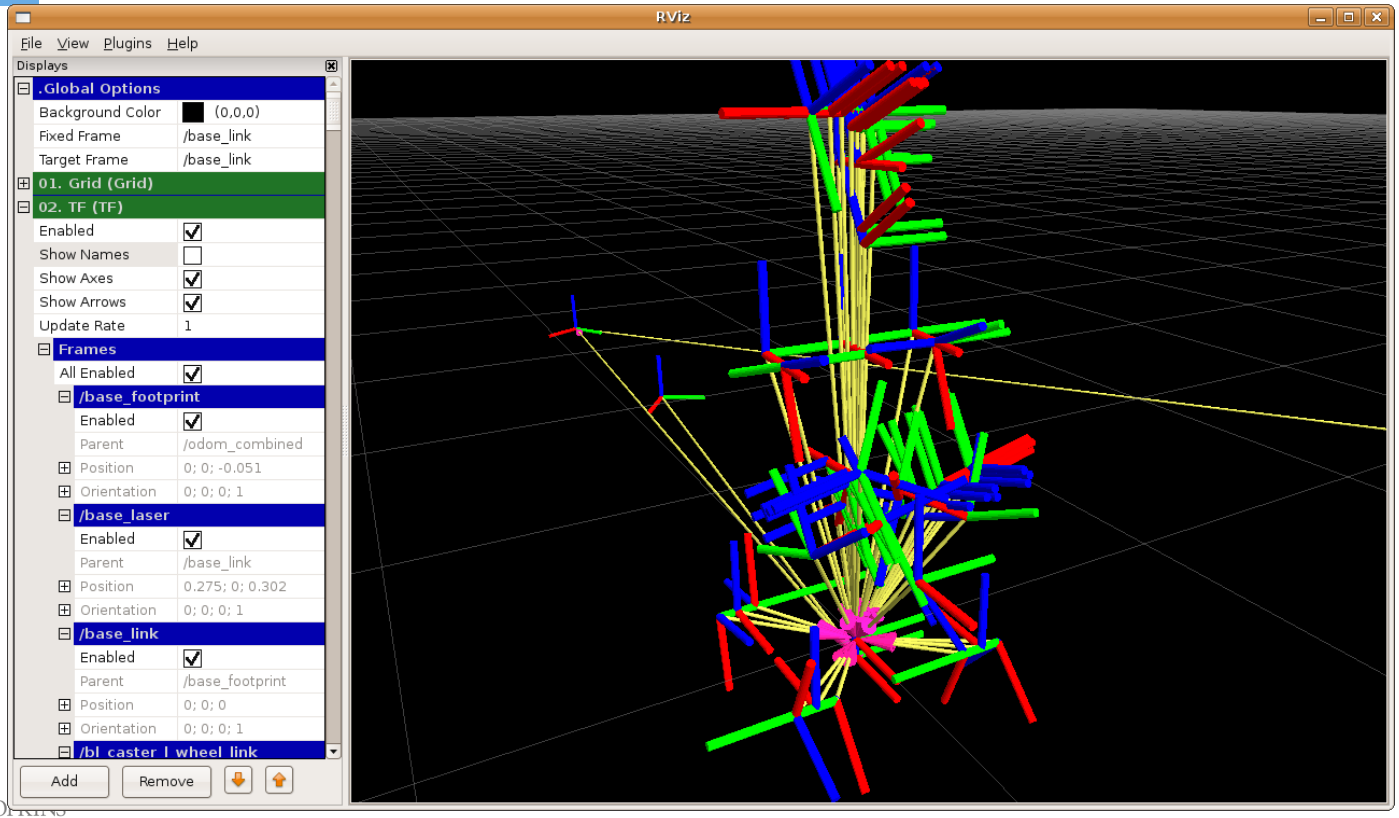
Rviz

3D visualization of the transforms



Note: At the time of this lecture tf2 support in ROS2 is still preliminary

RVIZ Plugin


RViz
File View Plugins Help
Displays
.Global Options
Background Color (0,0,0)
Fixed Frame /base_link
Target Frame /base_link
01. Grid (Grid)
02. TF (TF)
Enabled
Show Names
Show Axes
Show Arrows
Update Rate 1
Frames
All Enabled
/base_footprint
Enabled
Parent /odom_combined
Position 0; 0; -0.051
Orientation 0; 0; 0; 1
/base_laser
Enabled
Parent /base_link
Position 0.275; 0; 0.302
Orientation 0; 0; 0; 1
/base_link
Enabled
Parent /base_footprint
Position 0; 0; 0
Orientation 0; 0; 0; 1
/bl_caster_l_wheel_link
Add
Remove

TF2 Listener Python3 API for ROS2

- Create a TF Listener to fill up a buffer. ROS2 requires a node as input.

```
node=rclpy.create_node('tf2_rrbot_listener')
tf_buff = tf2_ros.Buffer()
listener = tf2_ros.TransformListener(tf_buff,node)
```

- To look up transforms use

```
geometry_msgs::TransformStampedtransformStamped=
tf2_ros.lookup_transform(target_frame_id,source_frame_id, time);
```

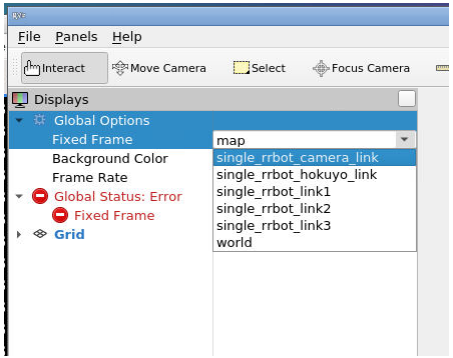
- For time use the clock for the node hosting the listener

```
when = rclpy.time.Time()
```

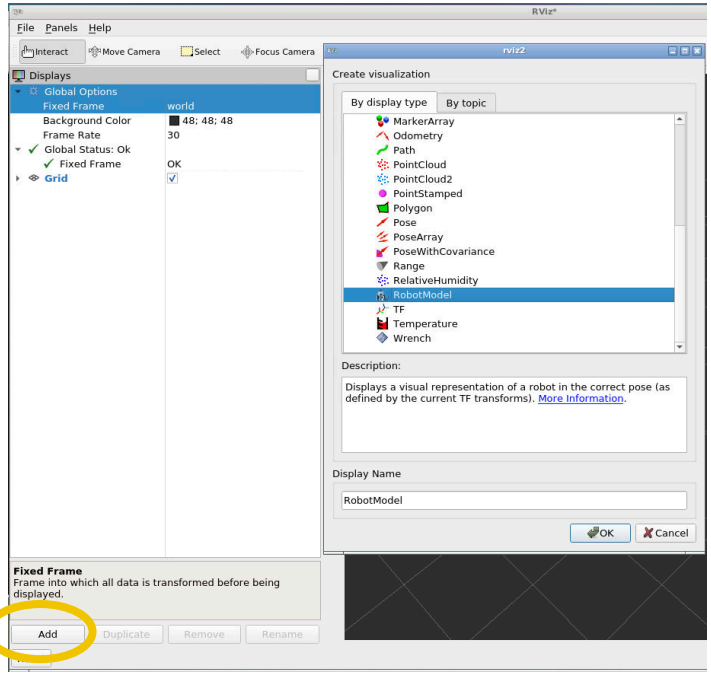
```
import tf2_ros
import rclpy

node=rclpy.create_node('tf2_rrbot_listener')

tf_buff = tf2_ros.Buffer()
listener = tf2_ros.TransformListener(tf_buff,node)

frame1_name = 'world'
frame2_name = 'single_rrbot_camera_link'

while True:
    try:
        when = rclpy.time.Time()
        trans = tf_buff.lookup_transform(to_frame,
    from_frame,
                when, timeout=Duration(seconds=5.0))
    except tf2_ros.LookupException:
        node.get_logger().info('Waiting for transform...')
         sleep(1)
```

Viewing Robots in RVIZ

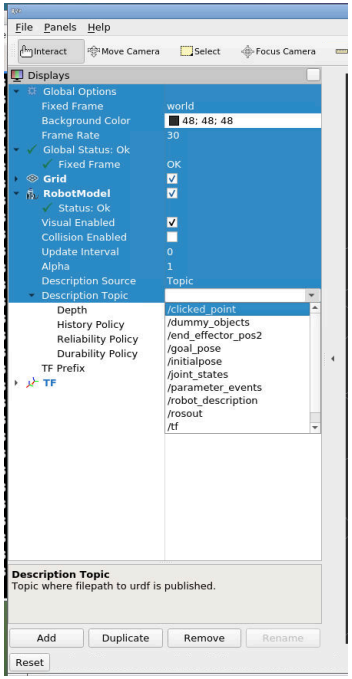
Step 1.
Select the correct frame

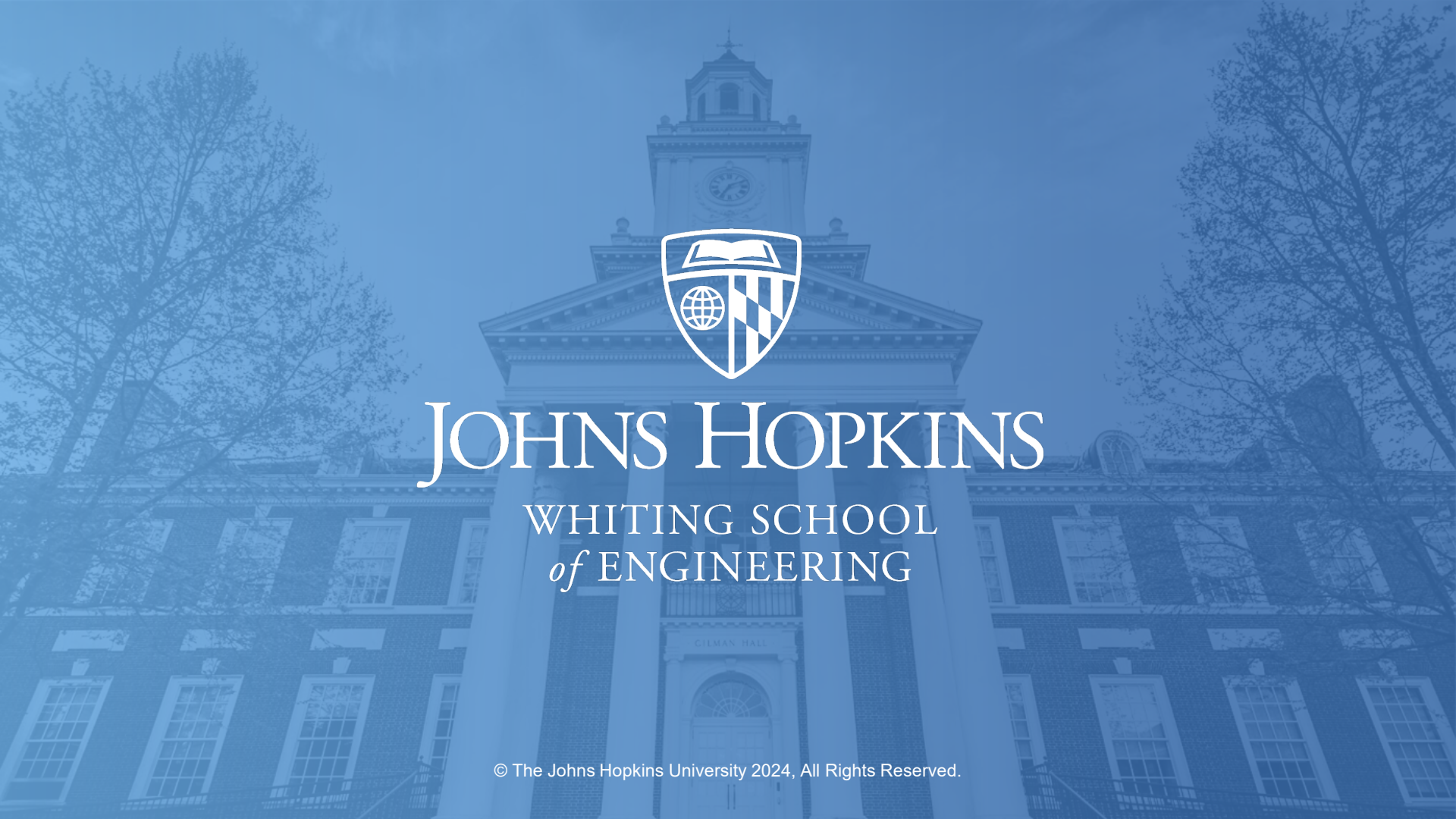


Step 2.
Add RobotModel and TF elements to the display



Step 3.
Select the topics to display (e.g. /robot_description)





Assignment 2

- You are provided with a ROS2 Node that subscribes to TF2 and /dummy_object/pose and publishes the object position in the base frame
- Complete all the functions in assignment2.py
  - euler_rotation_matrix
  - quaternion_rotation_matrix
  - quaternion_multiply
  - quaternion_to_euler
  - rotate
  - inverse_rotation
  - transform_pose