Pose, Transforms and Kinematics

Shenlong Wang
UIUC

ILLINOIS

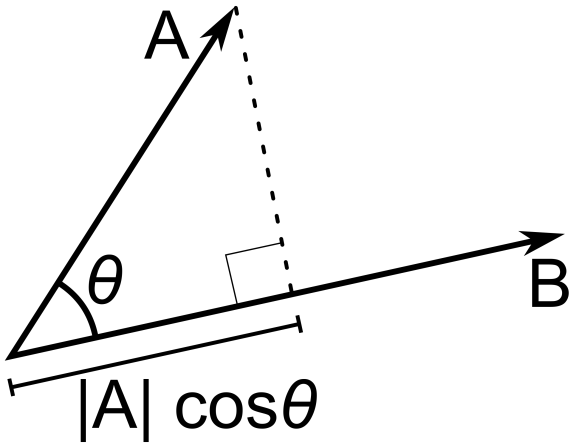
Some materials borrowed from Florian Shkurti, Kris Hauser, Vijay Kumar, Shuran Song, Kris Kitani

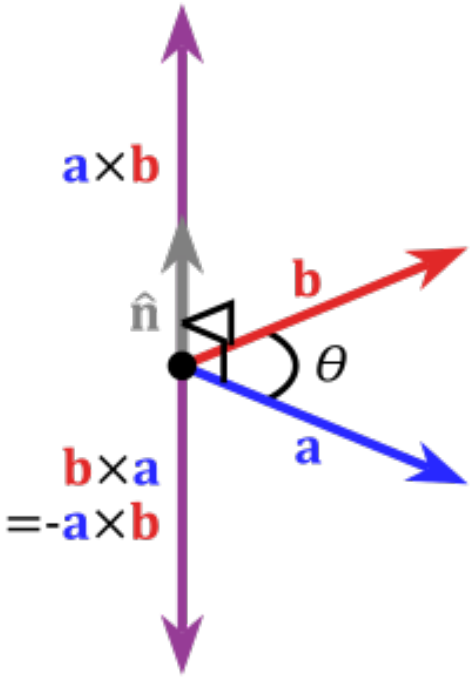
Today’s Agenda

- Coordinates and Rigid Transforms
- Coordinate Frame Composition
- 3D Rotation Representations
- Lie Group and Lie Algebra

Prerequisite

- Vector
- Matrix
- Linear Transforms
- Norm
- Dot Product
- Cross Product





Motivation

Robotics is **ALL** about management of reference frames

- **Perception** is about estimation of reference frames
- **Planning** is how to move reference frames
- **Control** is the implementation of trajectories for reference frames

The relation between references frames (described by transformations or poses) is essential to a successful system (..and where you likely spend efforts to debug).

Rigid Object

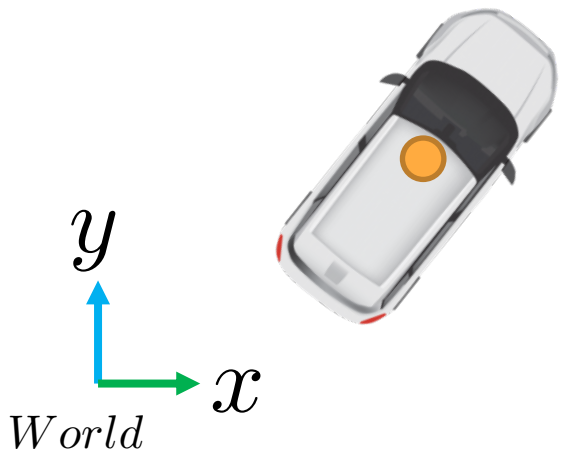
- **How would you represent the state of the vehicle quantatitively?**

Rigid Body Representation

- How would you represent the state of the vehicle?

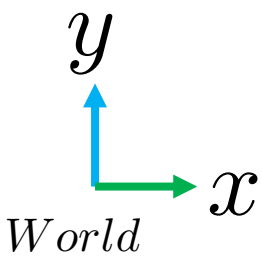
Rigid Body Representation

- How would you represent the state of the vehicle?



Rigid Body Representation

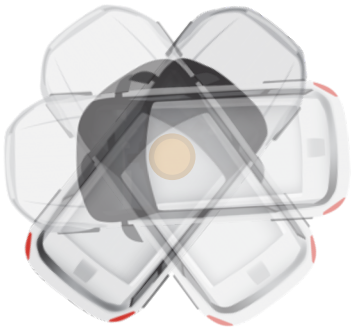
- How would you represent the state of the vehicle?

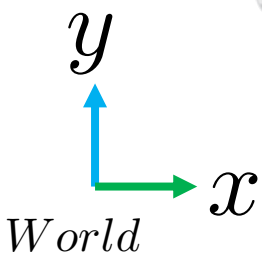


$$\mathbf{x} = (t_x, t_y)$$

Rigid Body Representation

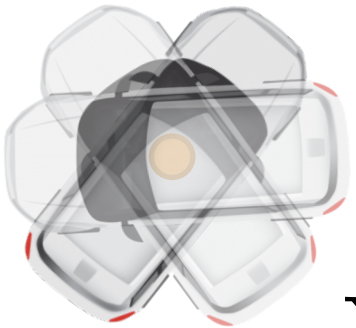
- How would you represent the state of the vehicle?

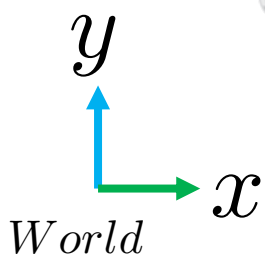




Rigid Body Representation

- How would you represent the state of the vehicle?

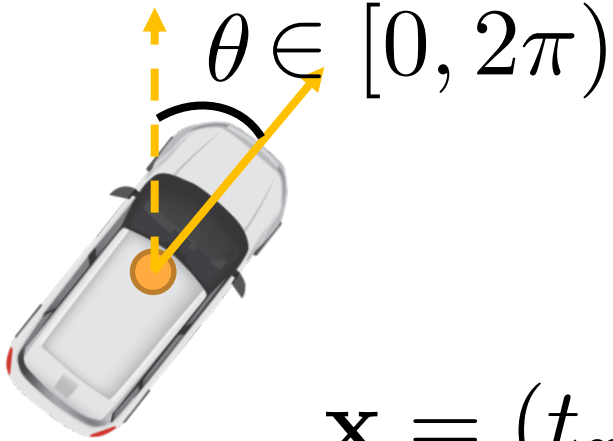




$$\mathbf{x} = (t_x, t_y, \theta)$$

Rigid Body Representation

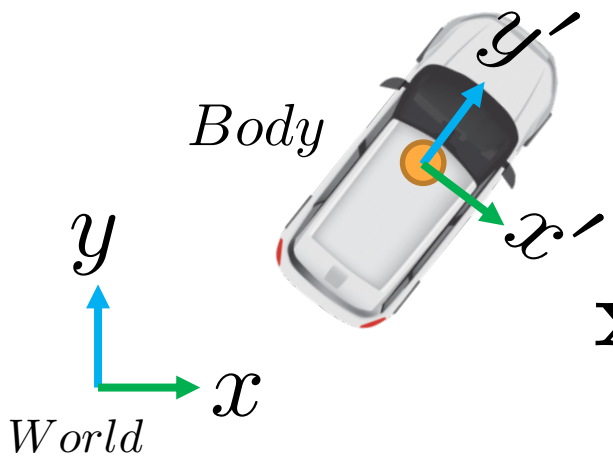
- How would you represent the state of the vehicle?
  - State of a static rigid body = (Position, Orientation)



$$\mathbf{x} = (t_x, t_y, \theta)$$

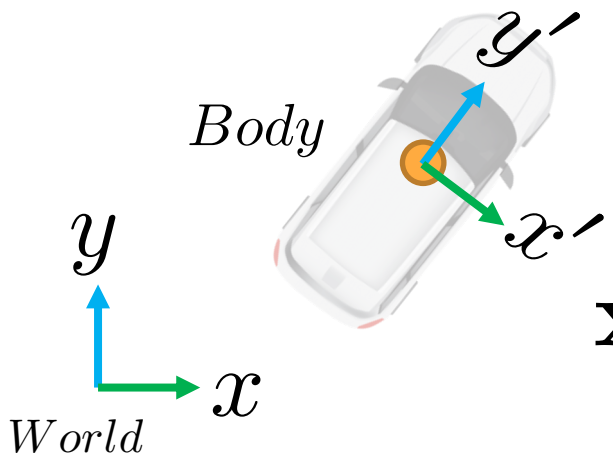
Body Frame

- Parameters of the states also defines a local coordinate frame



$$\mathbf{x} = (t_x, t_y, \theta)$$

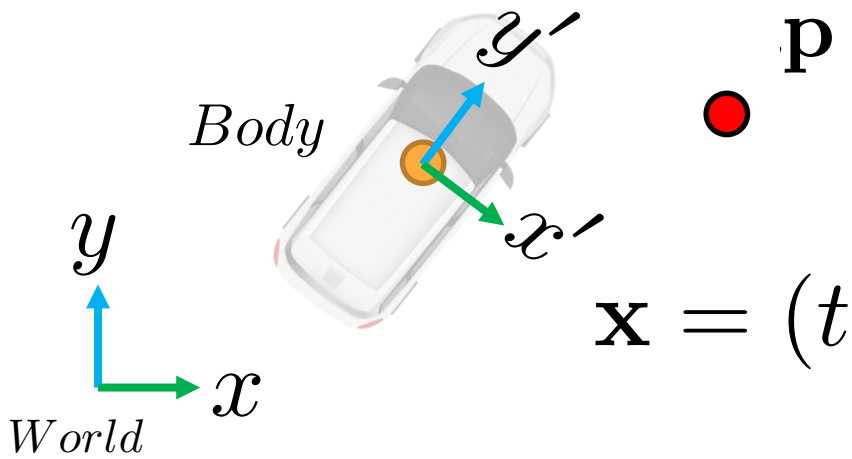
Body Frame



$$\mathbf{x} = (t_x, t_y, \theta)$$

Body Frame

- **Could you calculate the pedestrian's position in the world frame?**



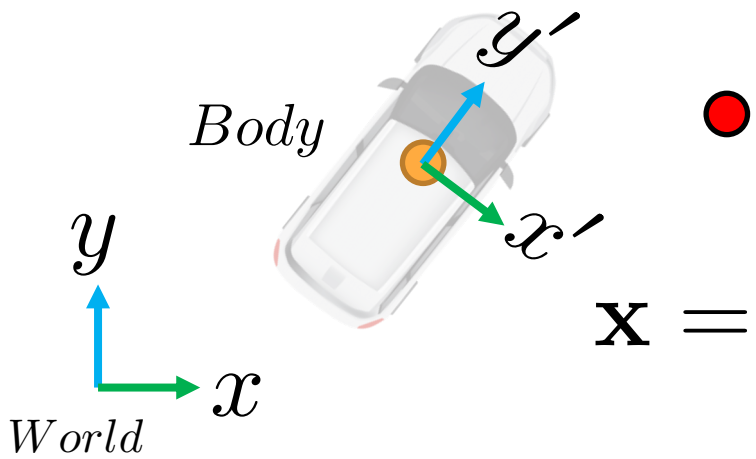
$$\mathbf{x} = (t_x, t_y, \theta)$$

Rigid Transform from Body to World

$$\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} p'_x \\ p'_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Step 1: rotate by theta

Step 2: translate



$$\mathbf{x} = (t_x, t_y, \theta)$$

Properties of Rigid Transform

$$\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} p'_x \\ p'_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Step 1: rotate by theta

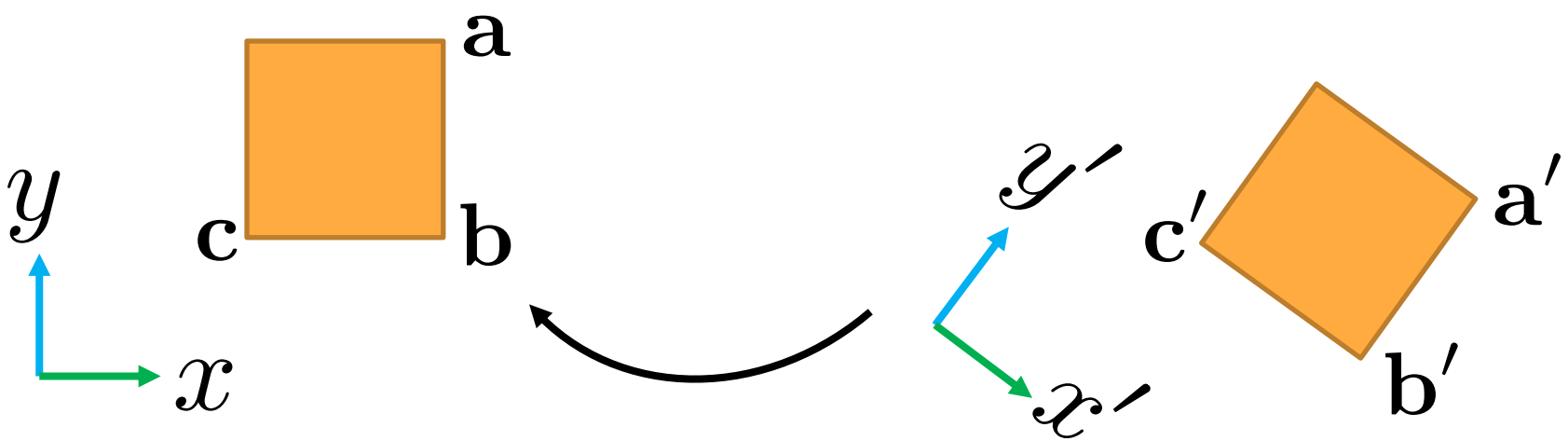
Step 2: translate

$$\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det \mathbf{R} = 1$$

Properties of Rigid Transform

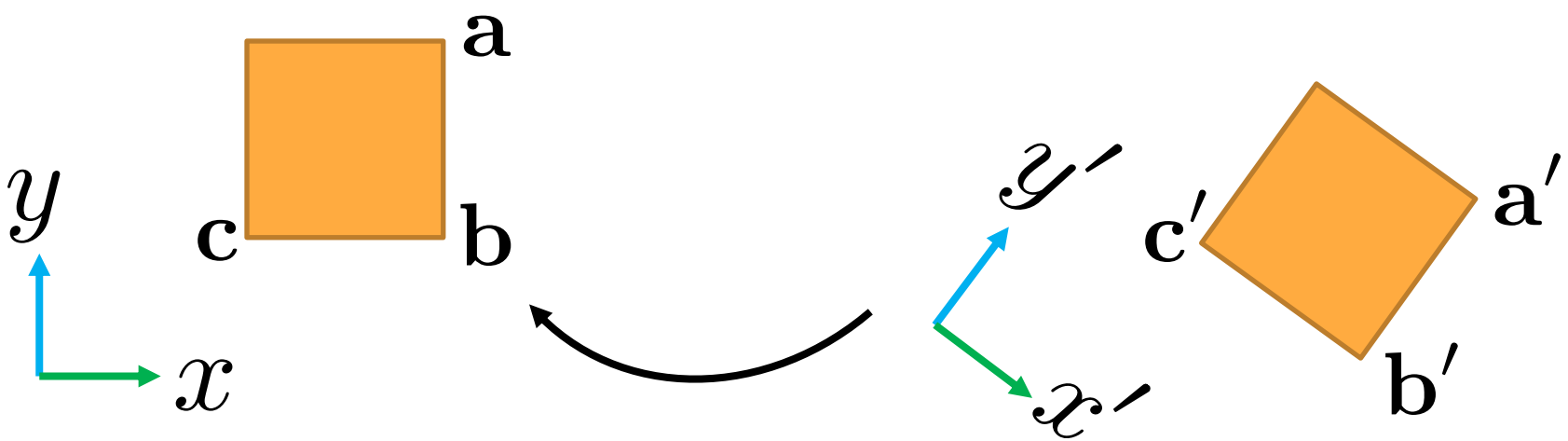
- Distance between any pair of two points is preserved:

$$(\mathbf{a}-\mathbf{b})^T(\mathbf{a}-\mathbf{b}) = (\mathbf{a}'-\mathbf{b}')^T(\mathbf{a}'-\mathbf{b}')$$



Properties of Rigid Transform

- Orientation-preserving or no reflection: any rotation between vectors is preserved:



Homogenous Coordinate

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$

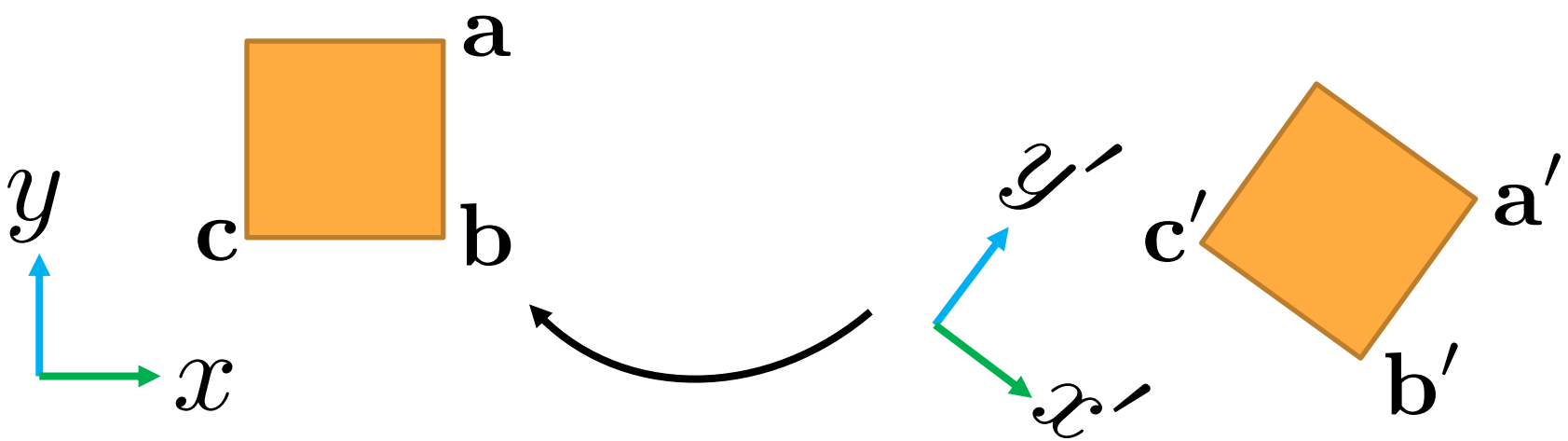
Homogenous Coordinate

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} p'_x \\ p'_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

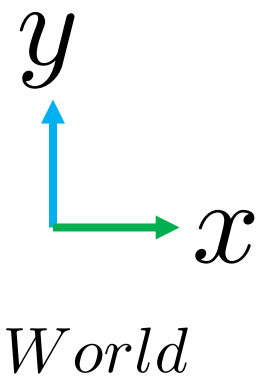
$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p'_x \\ p'_y \\ 1 \end{bmatrix}$$

Homogenous Coordinate

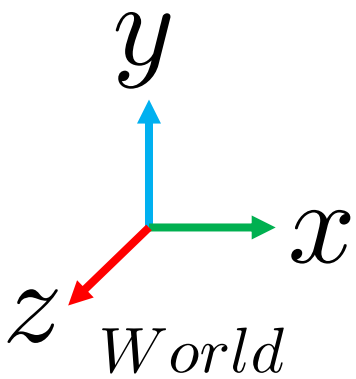
$$\hat{\mathbf{p}} = \mathbf{T}\hat{\mathbf{p}}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \hat{\mathbf{p}}'$$



3D Rigid Transform

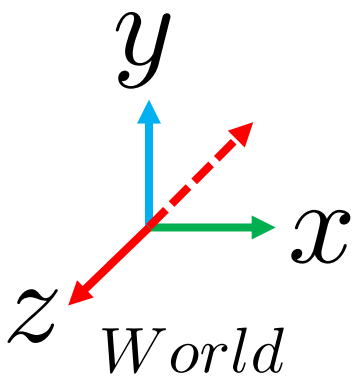


3D Rigid Transform

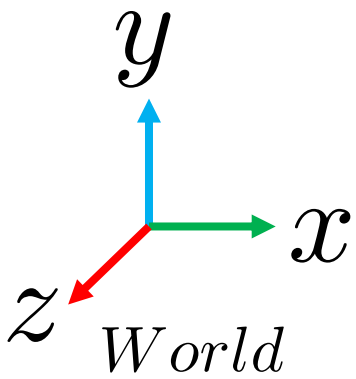


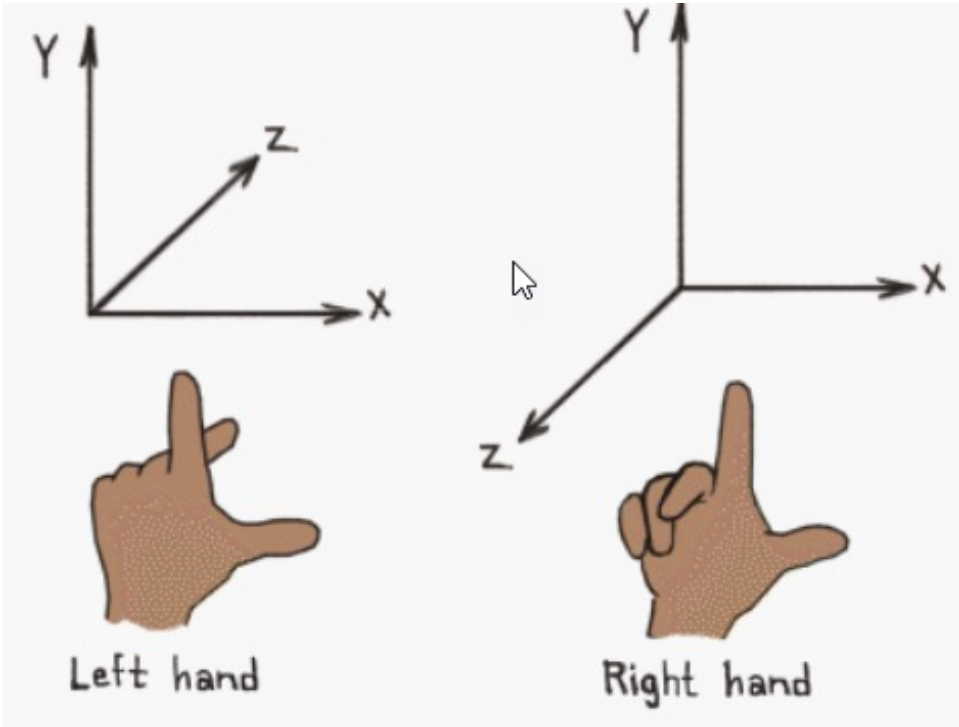
3D Rigid Transform

Which direction for z?



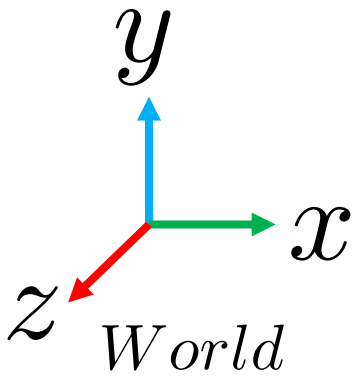
3D Rigid Transform



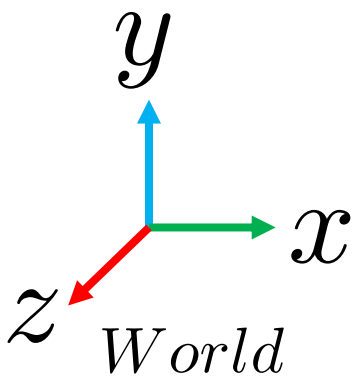


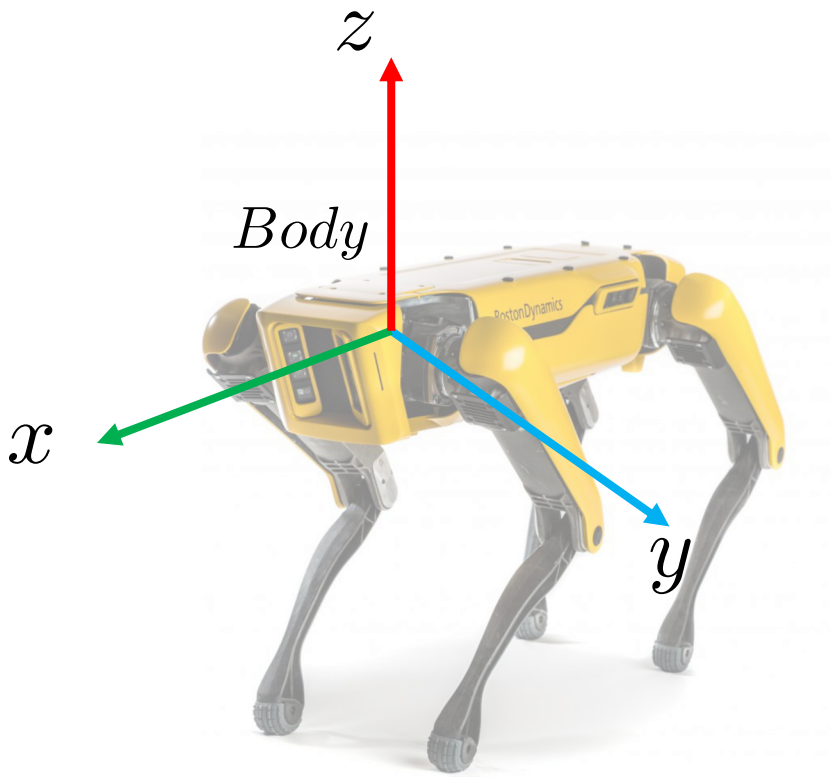
Roboticists and CVers mostly use right hand

3D Rigid Transform

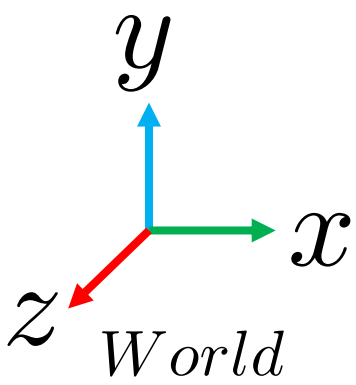


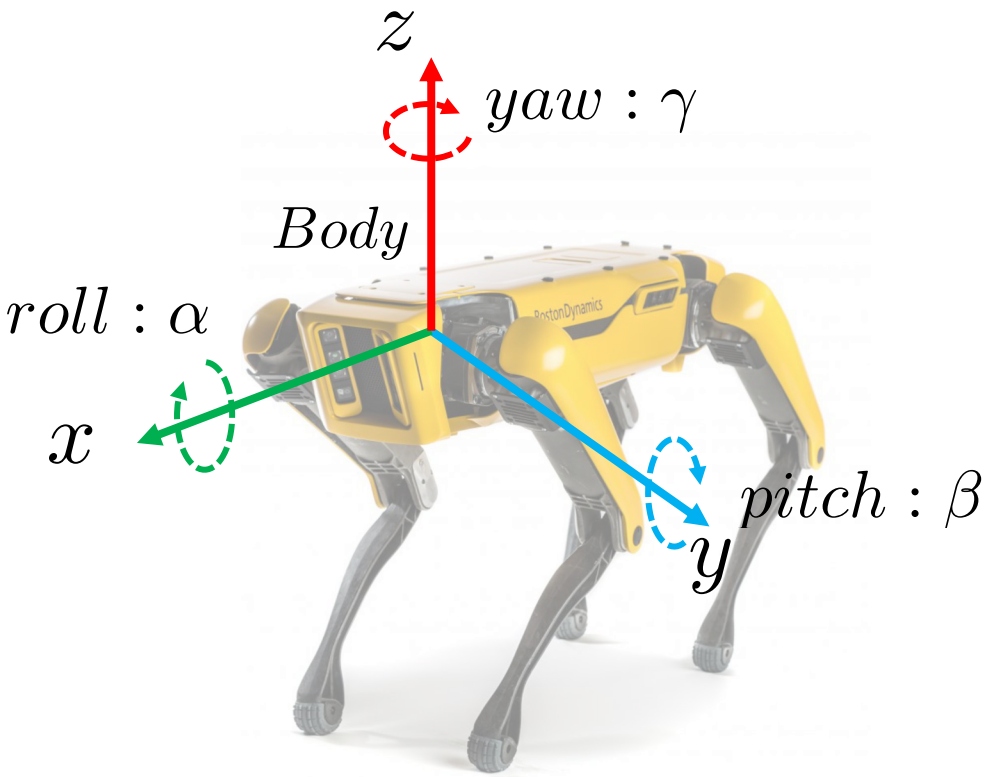
3D Rigid Transform





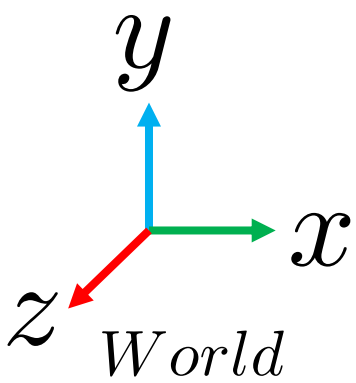
3D Rigid Transform

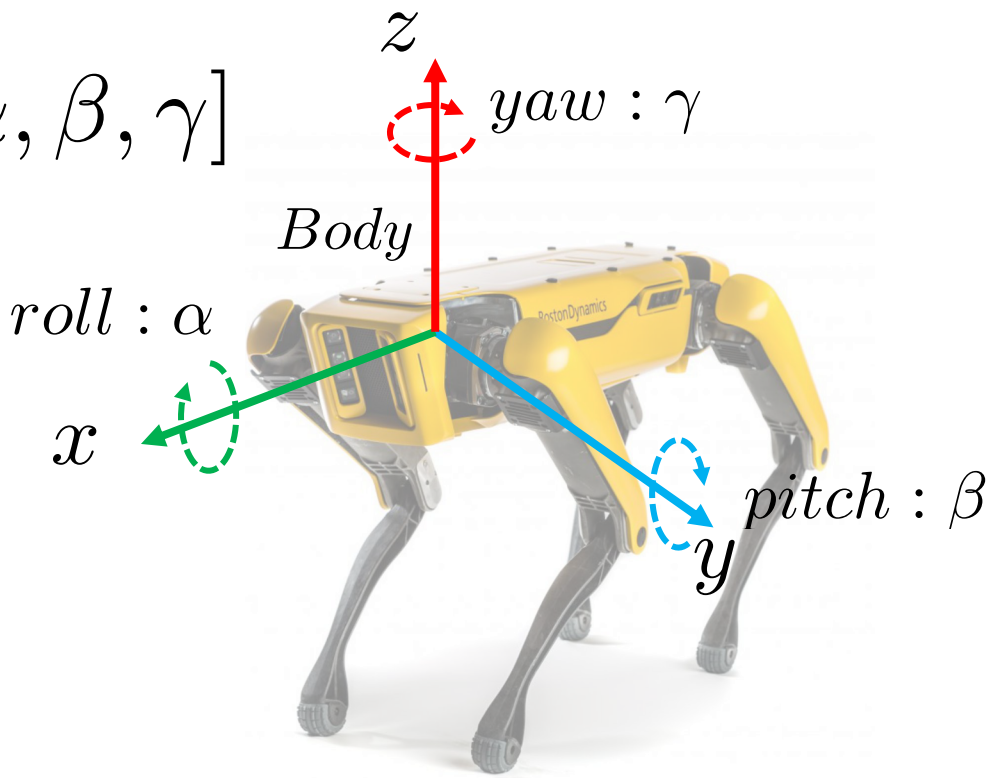




3D Rigid Transform

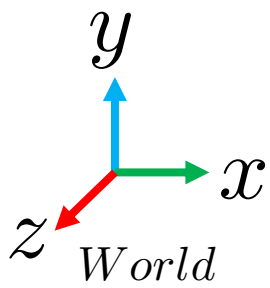
$$\mathbf{x} = [t_x, t_y, t_z, \alpha, \beta, \gamma]$$

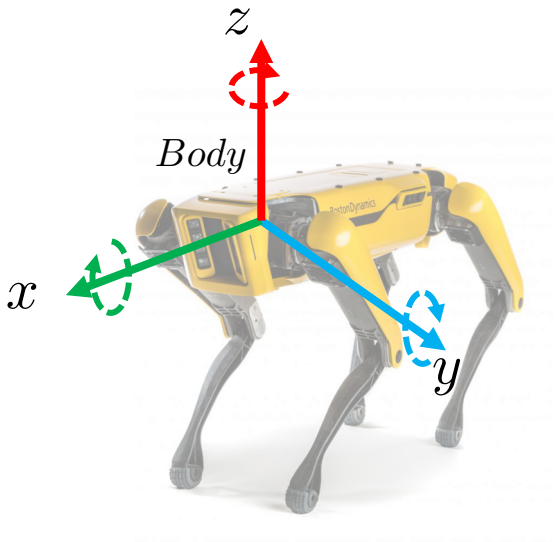




3D Rigid Transform

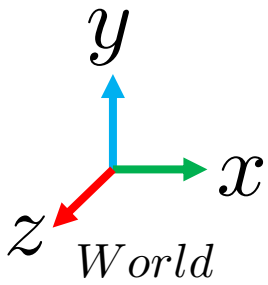
$$\hat{\mathbf{p}} = \mathbf{T}\hat{\mathbf{p}}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \hat{\mathbf{p}}'$$



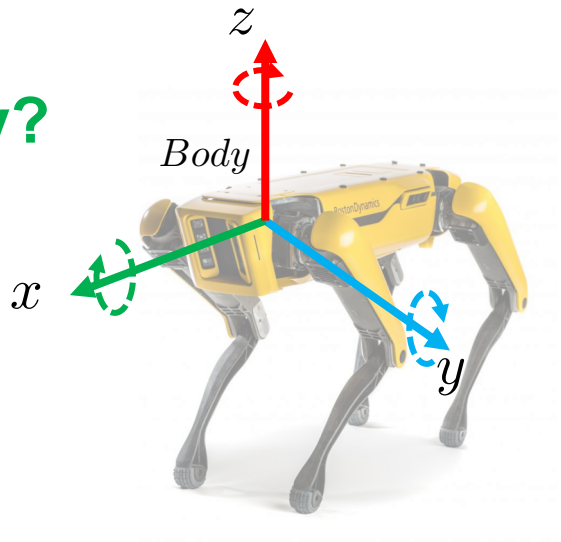


3D Rigid Transform

$$\hat{\mathbf{p}} = \mathbf{T}\hat{\mathbf{p}}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \hat{\mathbf{p}}'$$



World to Body?



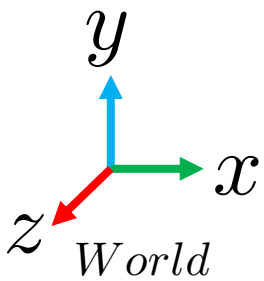
Inverse of Rigid Transform

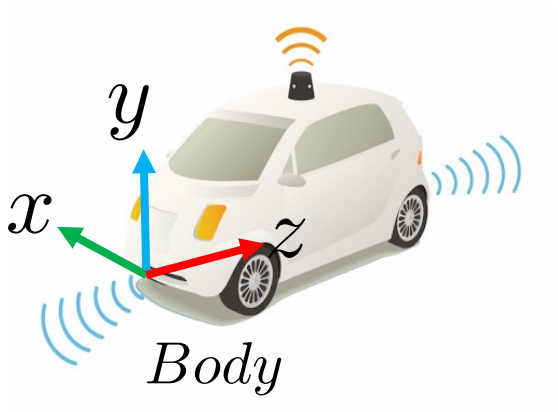
$$T^{-1} = \begin{bmatrix} R & d \\ 0_n & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0_n & 1 \end{bmatrix}$$

Try to verify the correctness!

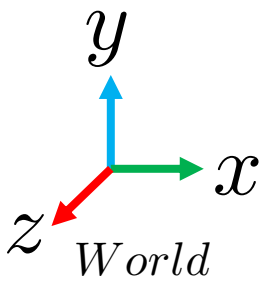
$$R^{-1} = R^T \qquad\qquad T^{-1} \neq T^T$$

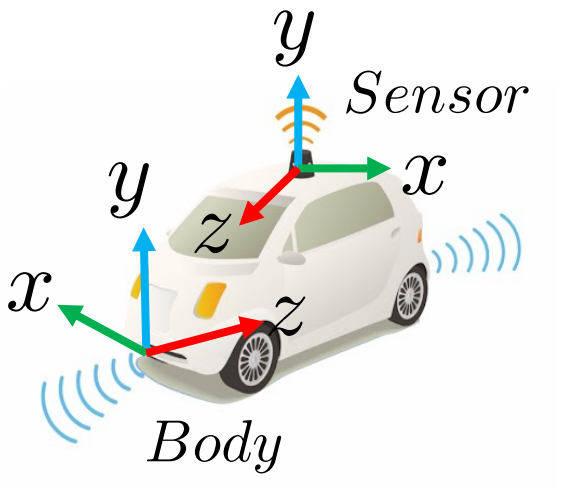
Coordinate Frames





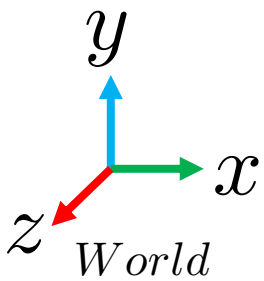
Coordinate Frames

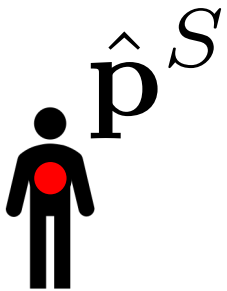


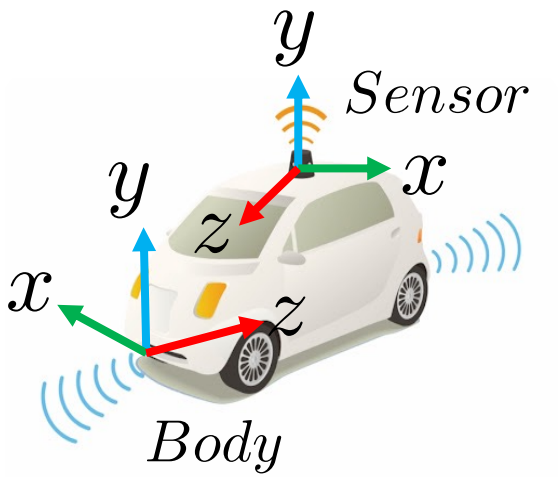


3D Rigid Transform

- Now LiDAR detects a person in its frame… (perception)

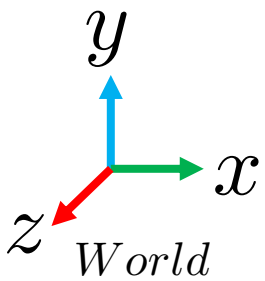


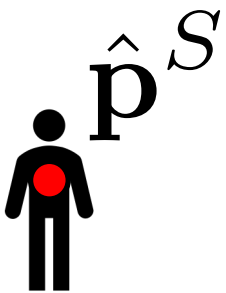


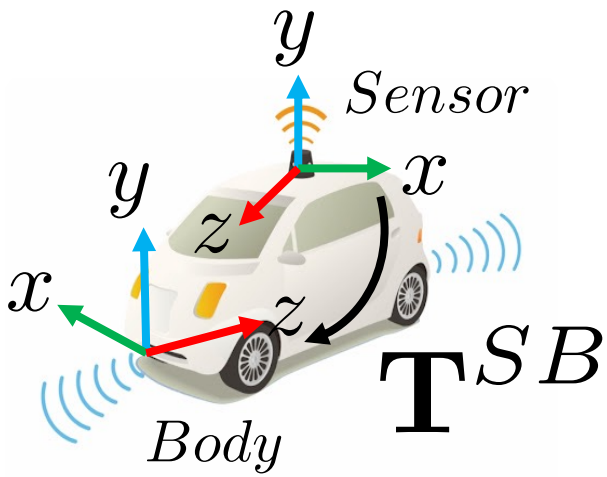


3D Rigid Transform: Composition

- Now LiDAR detects a person in its frame… (perception)
- We know the rigid transform between sensor to body (calibration)

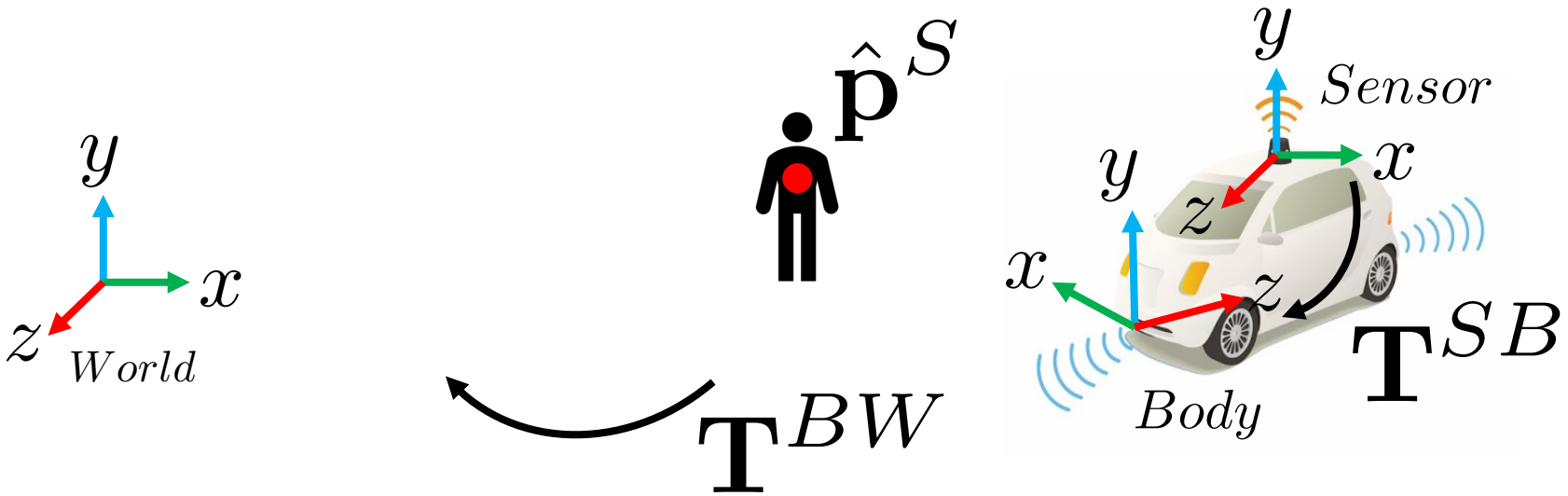






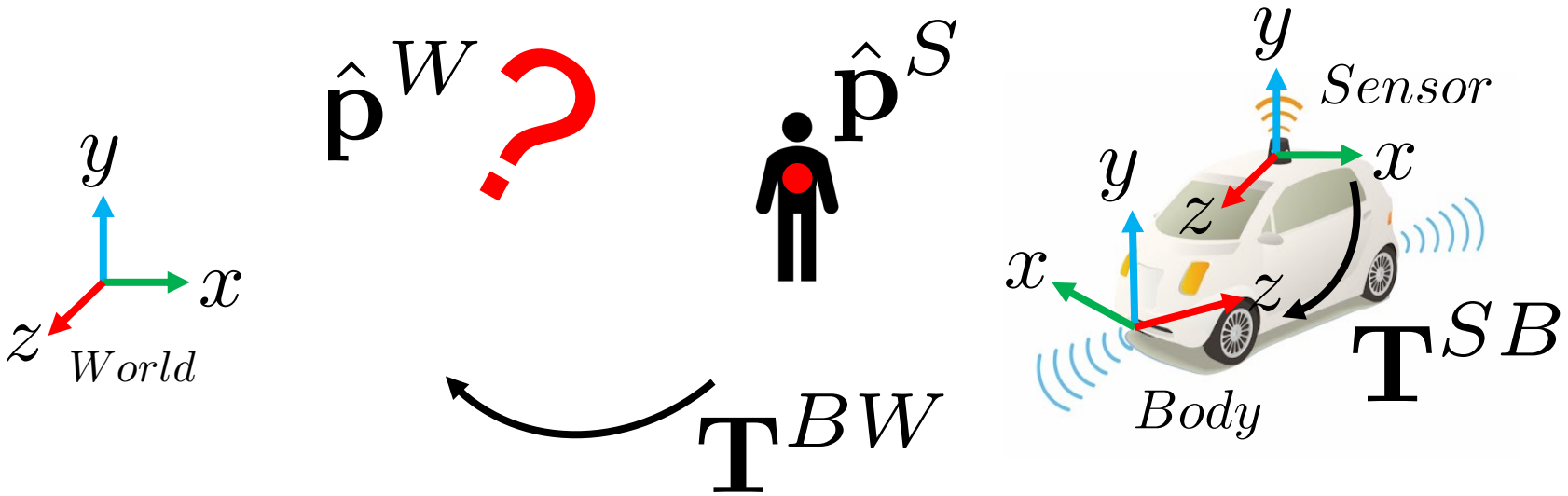
3D Rigid Transform: Composition

- Now LiDAR detects a person in its frame… (perception)
- We know the rigid transform between sensor to body (calibration)
- We estimate the rigid transform between body to world (localization)



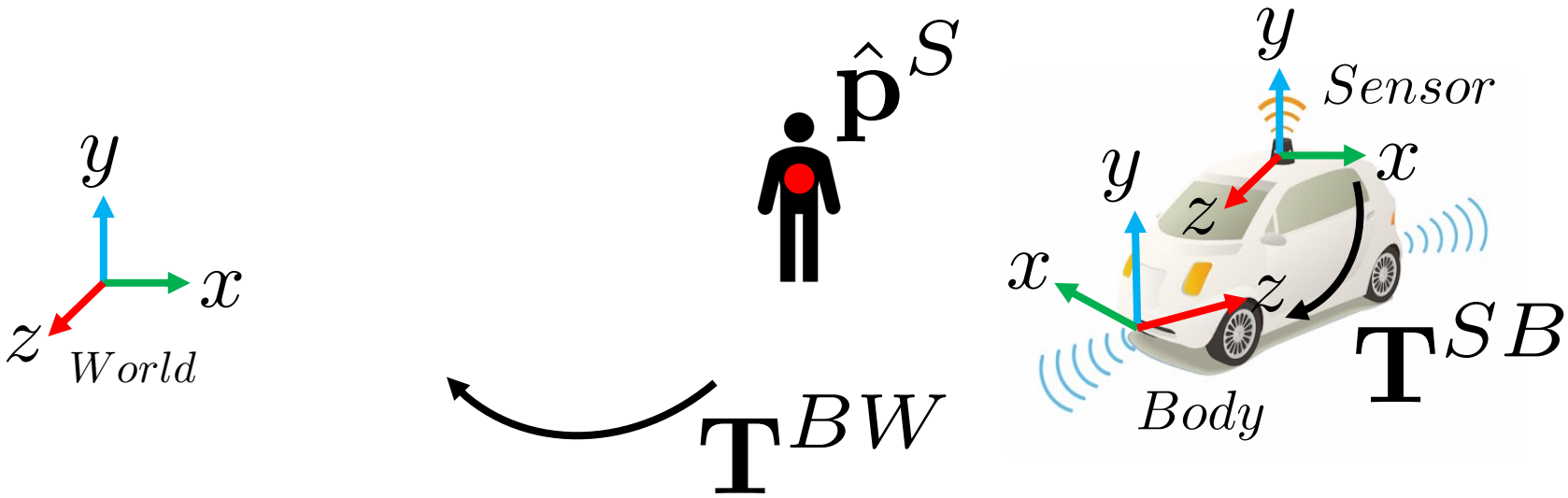
3D Rigid Transform: Composition

- Now LiDAR detects a person in its frame… (perception)
- We know the rigid transform between sensor to body (calibration)
- We estimate the rigid transform between body to world (localization)



3D Rigid Transform: Composition

$$\hat{\mathbf{p}}^{W} = \mathbf{T}^{BW}\mathbf{T}^{SB}\hat{\mathbf{p}}^{S}$$



Rotation Matrix

$$\{\mathbf{R}|\mathbf{R} \in \mathbb{R}^{3\times 3}, \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det \mathbf{R} = 1\}$$

Orthogonal

Right hand coordinate system

Rotation Matrix

$$\{\mathbf{R}|\mathbf{R} \in \mathbb{R}^{3\times 3}, \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det \mathbf{R} = 1\}$$

Orthogonal

Right hand coordinate system

- Preserving Length: $\|\mathbf{R}\mathbf{v}\| = \|\mathbf{v}\|$
- Preserving Orientation: $\mathbf{R}\mathbf{a} \times \mathbf{R}\mathbf{b} = \mathbf{R}(\mathbf{a} \times \mathbf{b})$

Could you prove these?

Rotation Matrix

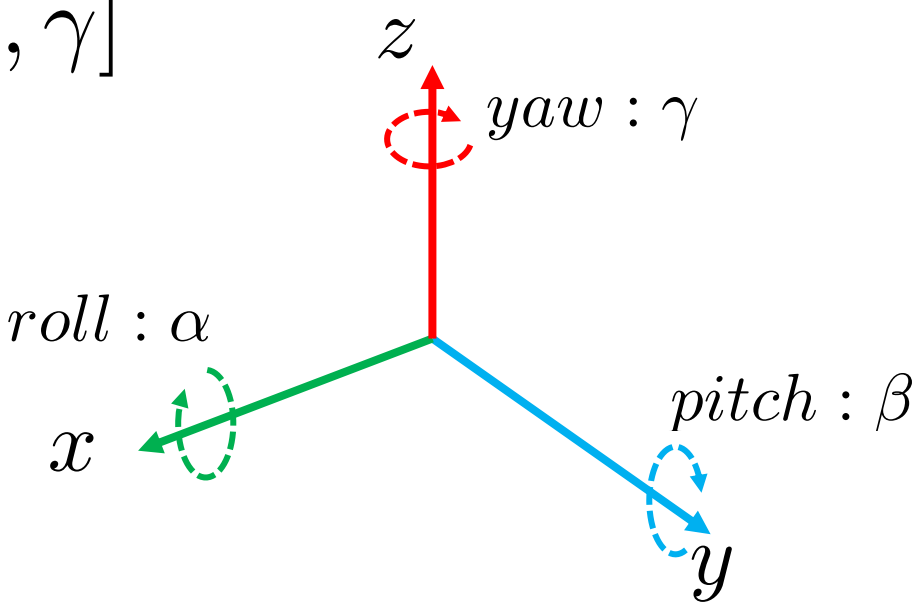
- Rotating a Vector: $$\mathbf{p}' = \mathbf{R}\mathbf{p}$$
- Composition: $$\mathbf{R}' = \mathbf{R}_2\mathbf{R}_1 \qquad \cancel{\mathbf{R}_1 + \mathbf{R}_2}$$
- Not compact: 3x3 numbers vs 3-DoF.
- Optimization/interpolation is not straightforward:

$$\min_{\mathbf{R}} f(\mathbf{R}) \text{ s.t. } \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det \mathbf{R} = 1$$

Euler Angles

- Three elemental rotations sequentially applied on each axes.

$[\alpha, \beta, \gamma]$



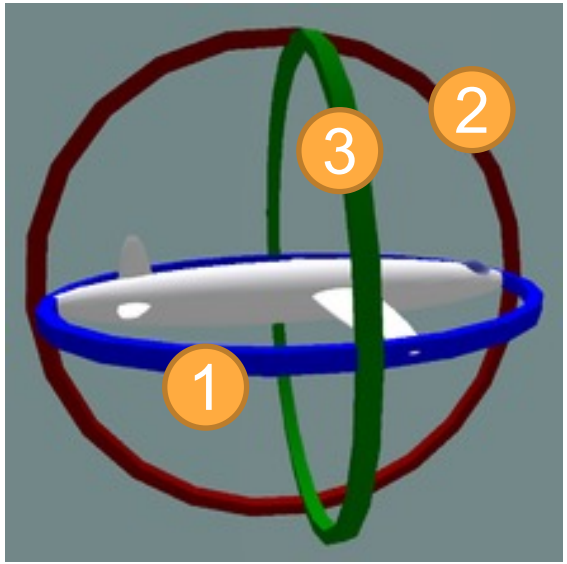
Euler Angles: Order Matters

- Need to specify the order. In total there are **twelve** valid combinations.
- (Roll, Pitch, Yaw) is a special case:

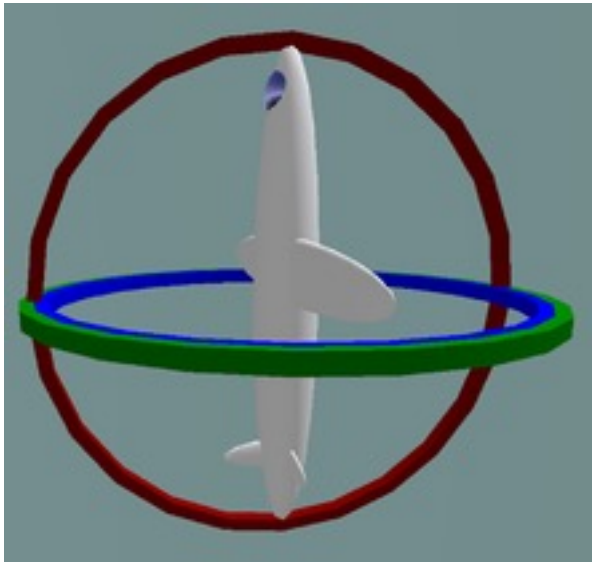
$$R = R_z(\gamma)\, R_y(\beta)\, R_x(\alpha) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

Euler Angles: Gimbal Lock

- Losses rotation powers



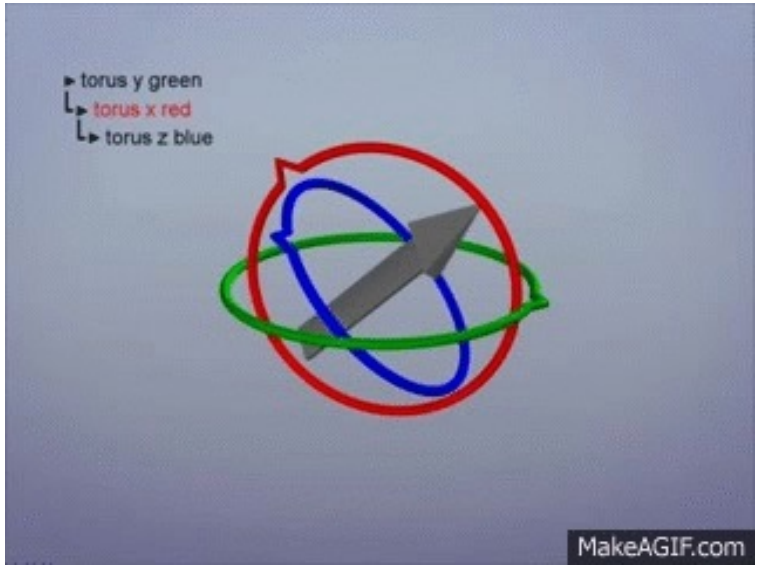
Pitch Rotates: pi/2

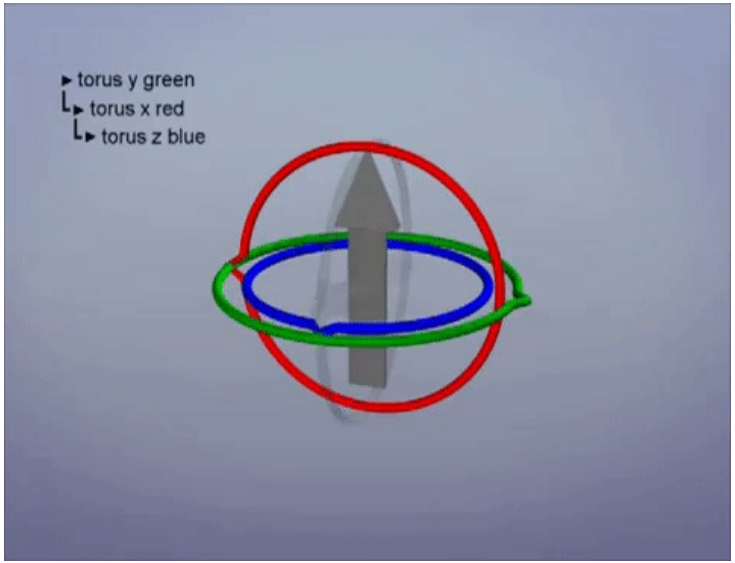


Euler Angles: Gimbal Lock

- Loss of one degree of freedom in a three-dimensional, three-gimbal mechanism

Rotation along y and z becomes the same!





Euler Angle: Singularity

When $\beta = \frac{\pi}{2}$

$$R = R_z(\gamma)\, R_y(\beta)\, R_x(\alpha) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

0 1 -1 0

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

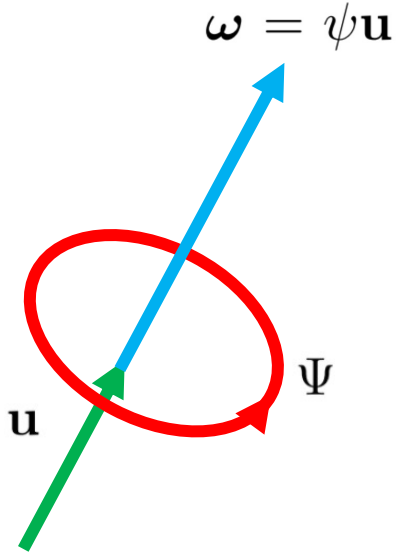
Changing alpha and gamma becomes the same!

Axis-angle

- 4-number representation (3d unit vector + 1d angle)
- Ambiguities: (-angle, -axis) is the same as (angle, axis)
- Minimal version: Euler vector (3d arbitrary vector)
- Conversion to rotation (Rodriguez formula):

$$\mathbf{R} = \mathbf{I} + [\mathbf{u}]_\times \sin\psi + [\mathbf{u}]_\times^2 (1 - \cos\psi)$$

$$[\mathbf{u}]_\times = \begin{bmatrix} 0 & -u_z, & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$



Could you derive it?

Axis-angle

- Suffer from the “edges”

$$r_1 = \begin{pmatrix} 0 \\ 0 \\ 179^\circ \end{pmatrix} \quad r_2 = \begin{pmatrix} 0 \\ 0 \\ -179^\circ \end{pmatrix} \quad r_1 - r_2 = \begin{pmatrix} 0 \\ 0 \\ 358^\circ \end{pmatrix}$$

Actual angular difference is only 2 deg.

- Interpolation and composition is hard.
- Rotating a vector is not straightforward. We have to convert it back to matrix.

Unit Quaternions

- Quaternion:

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \qquad \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

- Conjugate:

$$\mathbf{q}^{-1} = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}$$

- Hamilton Product:

$$\begin{aligned}(a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k})\,(a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) \;=\; & a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\ & + (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)\mathbf{i} \\ & + (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)\mathbf{j} \\ & + (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)\mathbf{k}\end{aligned}$$

https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

Unit Quaternions

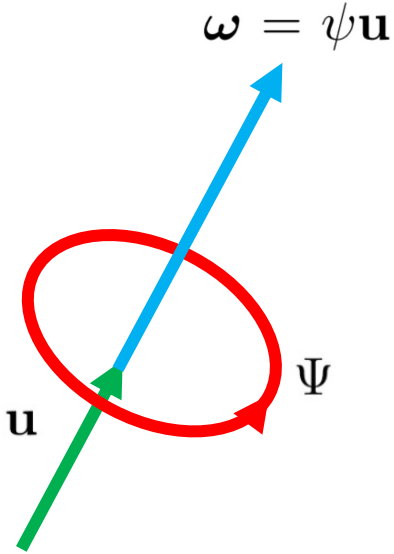
- Unit Quaternion as Rotation Representation:

$$\mathbf{q} = \left(\sin \frac{\Psi}{2} \mathbf{u}, \cos \frac{\Psi}{2} \right)$$

imagery Real

$$= \cos \frac{\Psi}{2} + (u_x \mathbf{i} + u_y \mathbf{j} + u_z \mathbf{k}) \sin \frac{\Psi}{2}$$

||q|| = 1



https://eater.net/quaternions/

Unit Quaternions

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$$
$$= \cos\frac{\Psi}{2} + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\Psi}{2}$$

Rotation along xx by yy?

(1, 0, 0, 0)

(0, 1, 0, 0)

(0, 0, 0, 1)

Unit Quaternions

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$$
$$= \cos\frac{\Psi}{2} + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\Psi}{2}$$

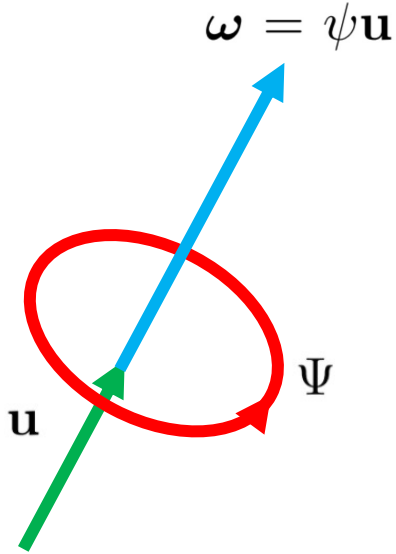
(1, 0, 0, 0)	Identity
(0, 1, 0, 0)	Rotate x axis by pi
(0, 0, 0, 1)	Rotate z axis by pi

Unit Quaternions

- Rotating a vector :

$$\mathbf{p} = (p_x, p_y, p_z) = p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$$

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



https://eater.net/quaternions/

Unit Quaternions

- Quaternion:

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

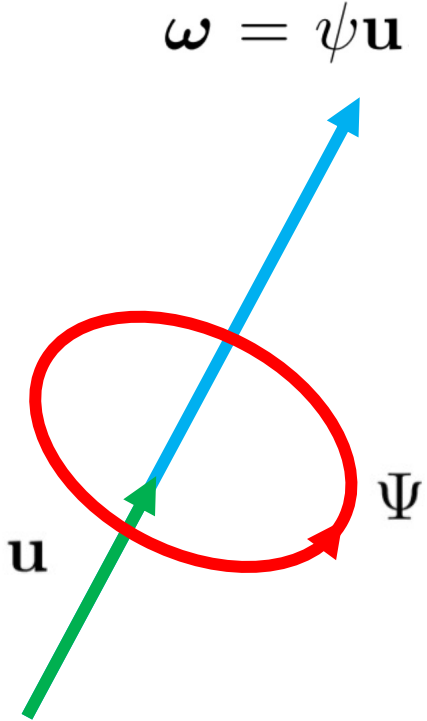
- Unit Quaternion as Rotation Representation:

$$\mathbf{q} = \left(\sin\frac{\Psi}{2}\mathbf{u}, \cos\frac{\Psi}{2}\right)$$

imagery Real

Pros:

- Continuous
- Numerically stable
- Relatively compact
- Rotating a vector is efficient



https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

Rotations Cheat Sheet

	Parameters	Singularities	Composition and Action
Matrix	9, orthogonality constraints	No	Easy
Euler Angle	3, [0, 2pi] or [0, pi]	Gimbal lock	Hard
Axis-Angle	4, unit axis vector	theta = 0	Hard
Rotation Vector	3, || v || < pi	Double representation	Hard
Quaternions	4, unit quaternion	Double representation, q -q	Easy

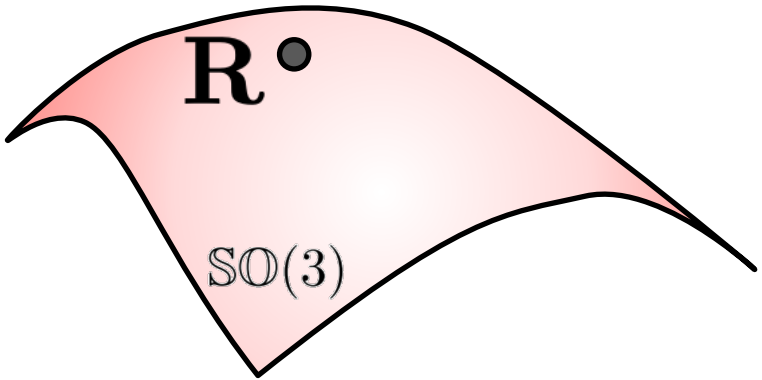
http://motion.cs.illinois.edu/RoboticSystems/3DRotations.html

IMPORTANT: Be clear about your coordinate system and rotation representation.

The Space of Rotations

Special Orthogonal Matrix Lie Group SO(3):

$$\mathbb{SO}(3) = \{\mathbf{R} | \mathbf{R} \in \mathbb{R}^{3\times 3}, \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det \mathbf{R} = 1\}$$



$$\min_{\mathbf{R} \in \mathbb{SO}(3)} f(\mathbf{R})$$

Gradient Descent?

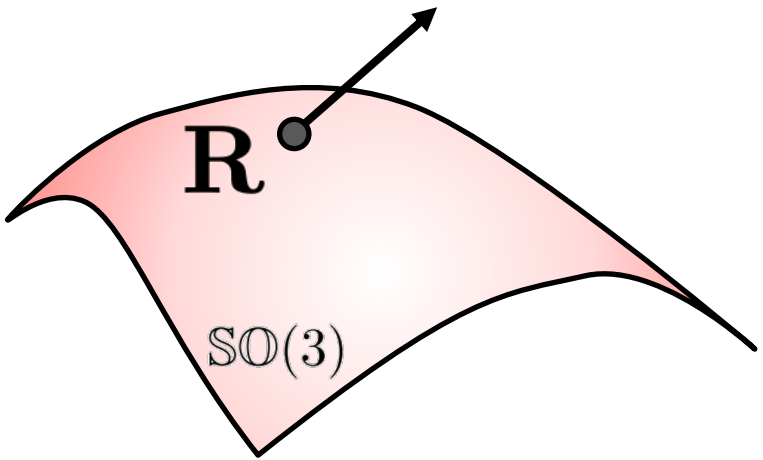
$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \gamma \nabla_{\mathbf{x}} f(\mathbf{x}^{(t)}) \longrightarrow \in \mathbb{R}^n$$

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right]$$

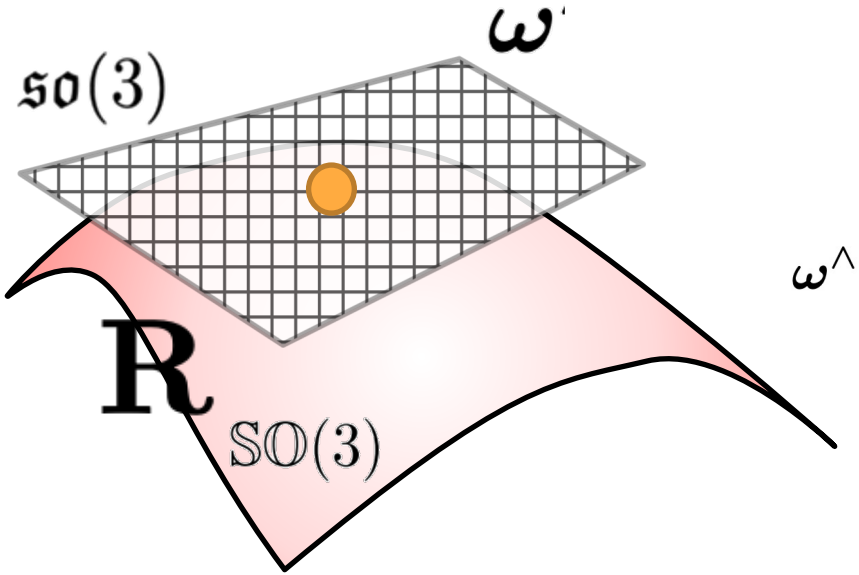
Gradient Descent?

$$\min_{\mathbf{R} \in \mathbb{SO}(3)} f(\mathbf{R})$$



so(3) Lie Algebra

- The tangent space of the SO(3) manifold.
- Coincides with the space of skew-symmetric matrix



$$\boldsymbol{\omega}^{\wedge} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathfrak{so}(3).$$

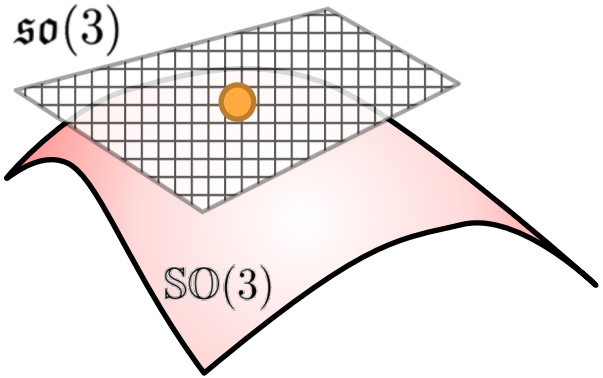
so(3) Lie Algebra

- Mapping from so(3) algebra to SO(3) group:

$$\exp(\boldsymbol{\phi}^\wedge) = \mathbf{I} + \frac{\sin(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|}\boldsymbol{\phi}^\wedge + \frac{1-\cos(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|^2}\left(\boldsymbol{\phi}^\wedge\right)^2$$

- Mapping from SO(3) group to so(3) algebra:

$$\log(\mathbf{R}) = \frac{\varphi \cdot (\mathbf{R} - \mathbf{R}^\mathsf{T})}{2\sin(\varphi)} \text{ with } \varphi = \cos^{-1}\left(\frac{\operatorname{tr}(\mathbf{R}) - 1}{2}\right)$$



SE(3) Lie Group and se(3) Lie Algebra

- SE(3) Lie Group group: the group of all rigid poses

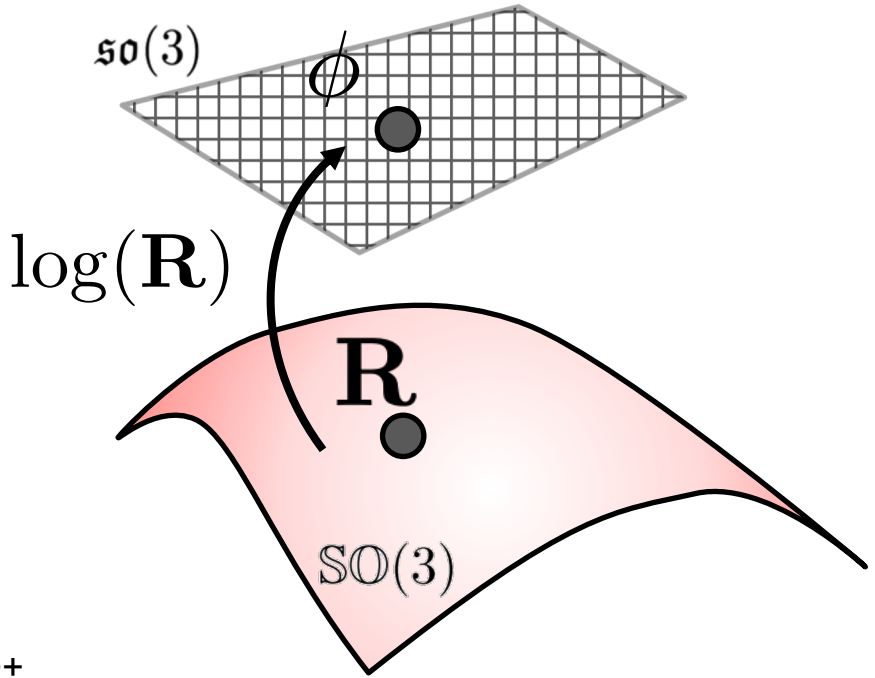
$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{SE}(3) \subset \mathbb{R}^{4\times4} \quad \text{with} \quad \mathbf{R} \in \mathbb{SO}(3), \mathbf{t} \in \mathbb{R}^3$$

- se(3) Lie algebra:

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\omega}_\times & \boldsymbol{v} \\ \mathbf{0}^T & 0 \end{bmatrix} \in \mathfrak{se}(3) \subset \mathbb{R}^{4\times4}$$

When to Use Lie Algebra?

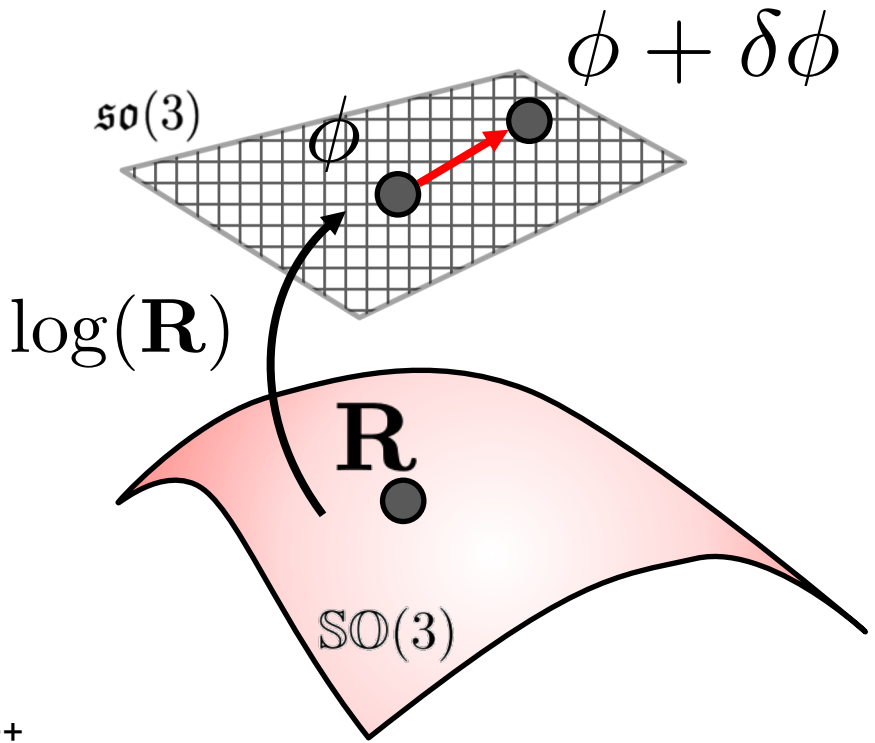
- Integration
- Smoothing
- Interpolation
- Uncertainty
- Control
- Optimization
- …



https://github.com/artivis/manif C++
https://github.com/princeton-vl/lietorch PyTorch
https://github.com/utiasSTARS/liegroups PyTorch + Numpy

When to Use Lie Algebra?

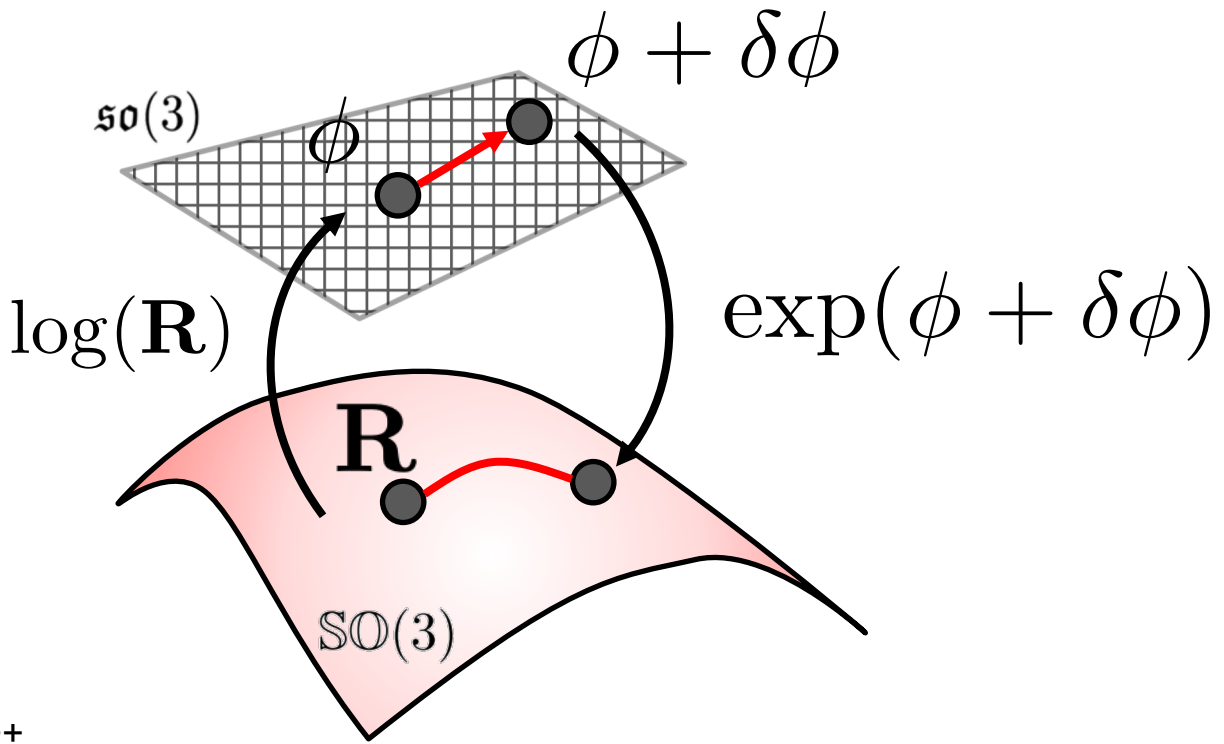
- Integration
- Smoothing
- Interpolation
- Uncertainty
- Control
- Optimization
- …



https://github.com/artivis/manif C++
https://github.com/princeton-vl/lietorch PyTorch
https://github.com/utiasSTARS/liegroups PyTorch + Numpy

When to Use Lie Algebra?

- Integration
- Smoothing
- Interpolation
- Uncertainty
- Control
- Optimization
- …



https://github.com/artivis/manif C++
https://github.com/princeton-vl/lietorch PyTorch
https://github.com/utiasSTARS/liegroups PyTorch + Numpy