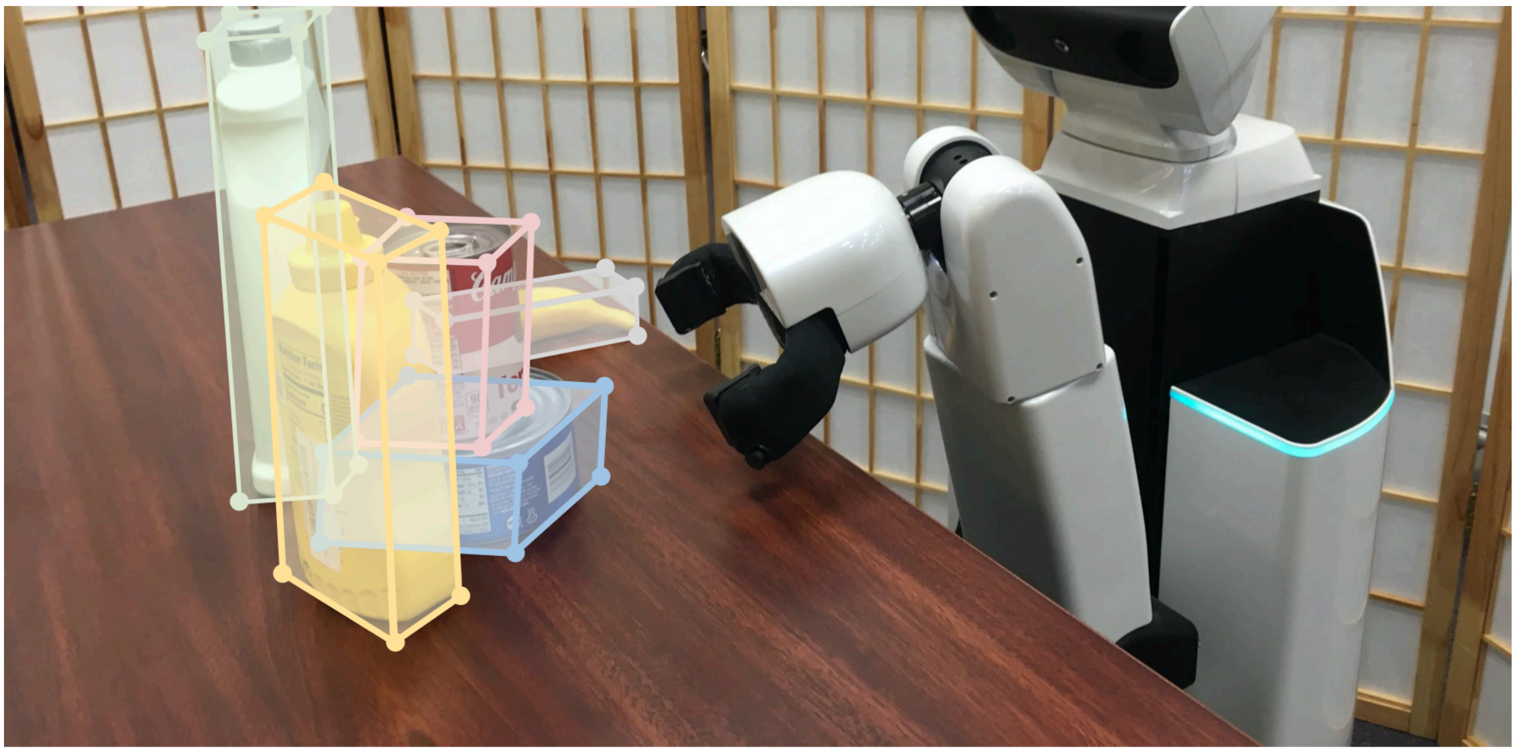
L5: Transformations

Hao Su

3D Spatial Relationships

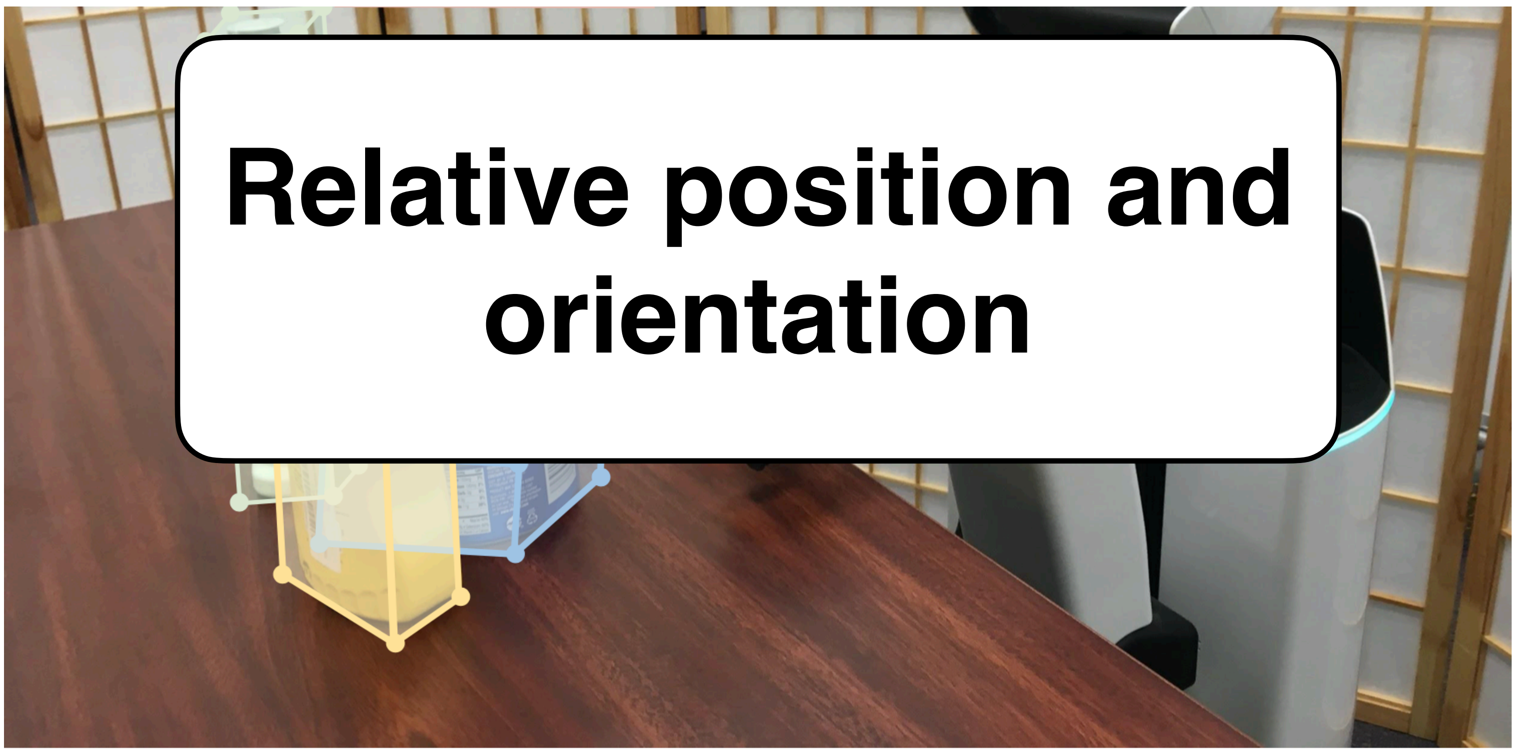
- How to represent the relationships between objects?



Wang et al., "**DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion**", *CVPR 2019*

3D Spatial Relationships

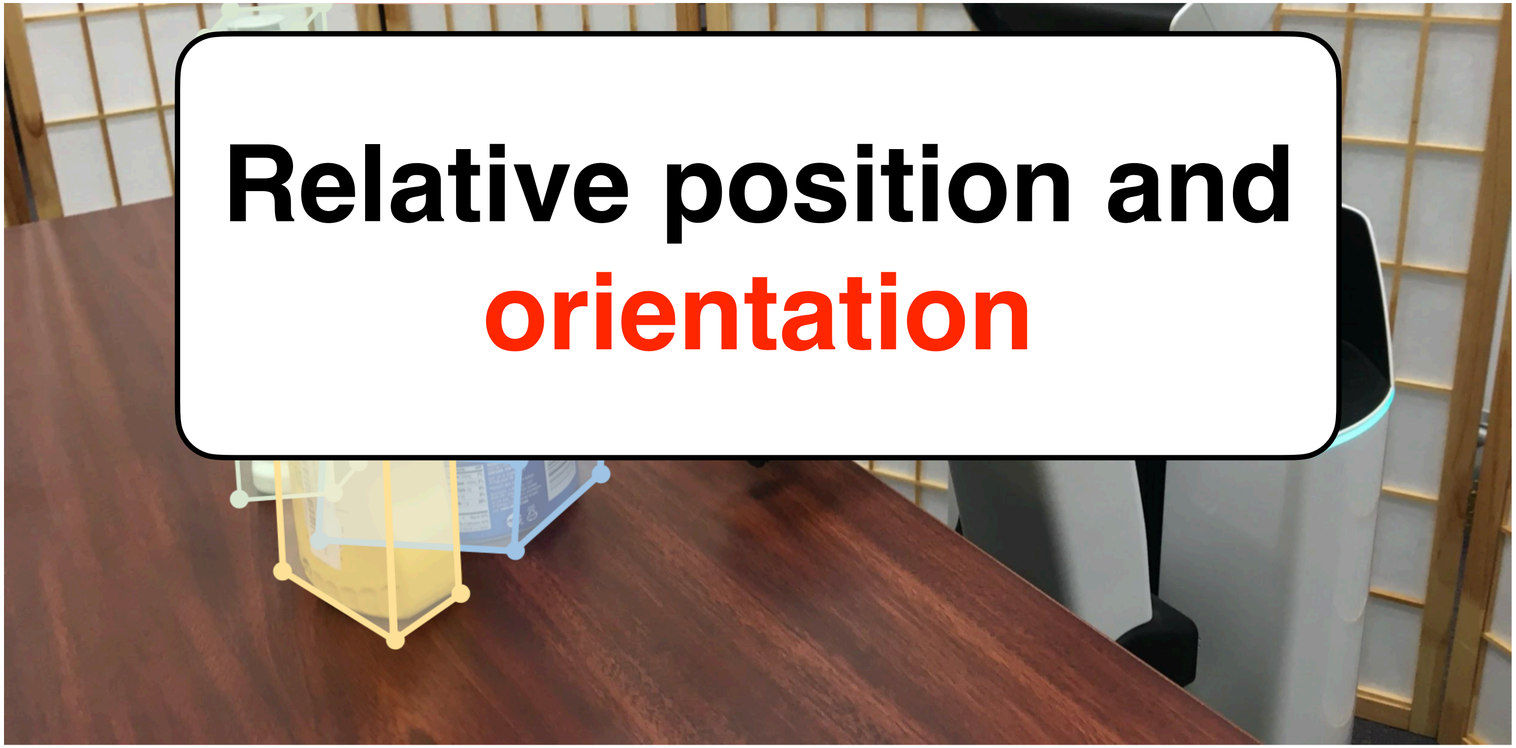
- How to represent the relationships between objects?



Wang et al., "**DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion**", *CVPR 2019*

3D Spatial Relationships

- How to represent the relationships between objects?

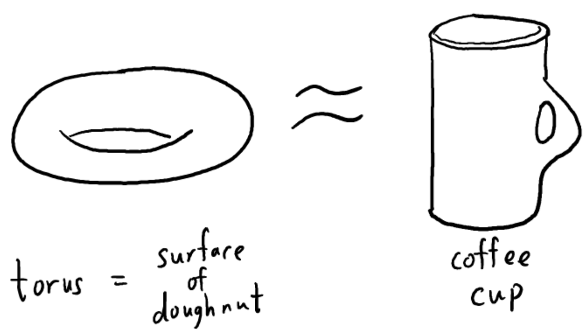


Wang et al., “**DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion**”, *CVPR 2019*

Prereq: Topology

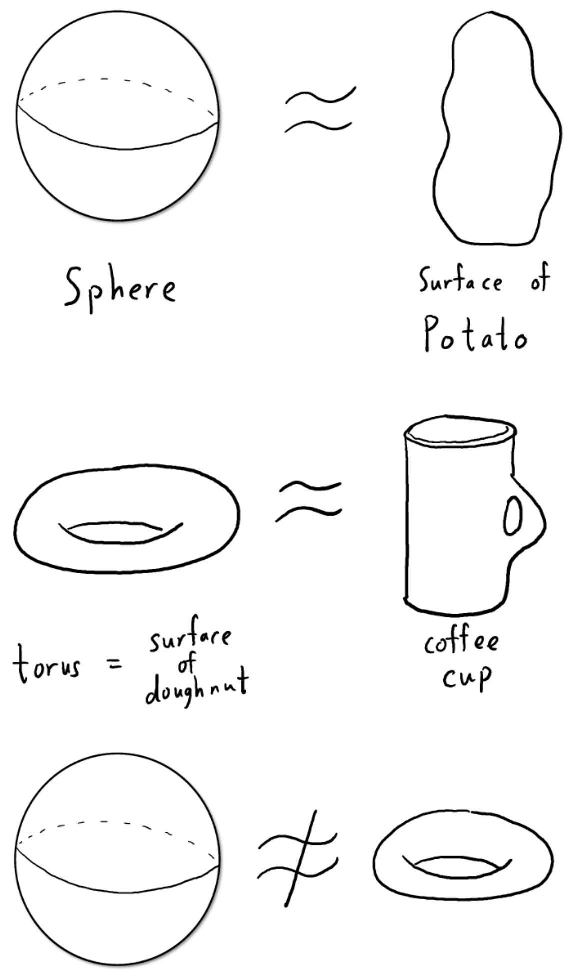
- Topology: Structural Properties of a Manifold

- Two surfaces M and N are *topologically equivalent* if there is a **differentiable bijection** between M and N



Prereq: Topology

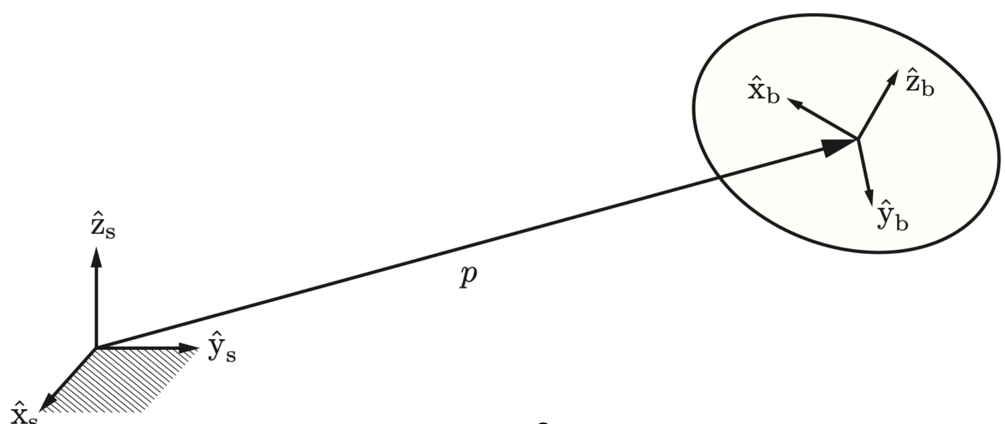
- More examples:



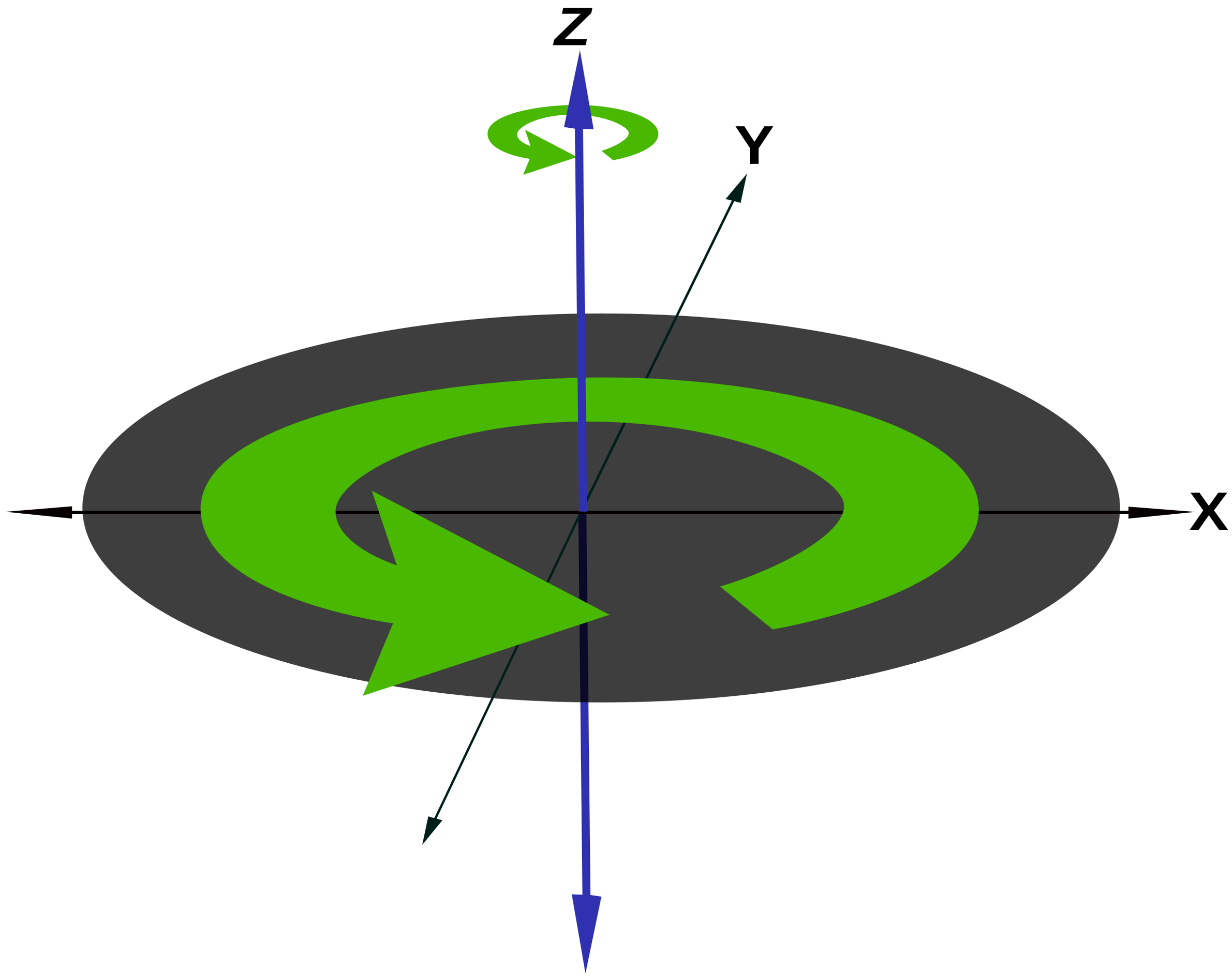
Rotation and $SO(n)$

Orientation

- We use "rotation" to represent the relative orientation between two frames
- For example,
  - Space Frame: $\{s\} = \{\hat{x}_s, \hat{y}_s, \hat{z}_s\}$
  - Body Frame: $\{b\} = \{\hat{x}_b, \hat{y}_b, \hat{z}_b\}$
  - R_{sb} rotates the frame of the space to the frame of the body



Rotation in $\mathbb{R}^2$



1 Degree of Freedom

Rotation in $\mathbb{R}^3$

3 Degree of Freedoms

The Set of Rotations

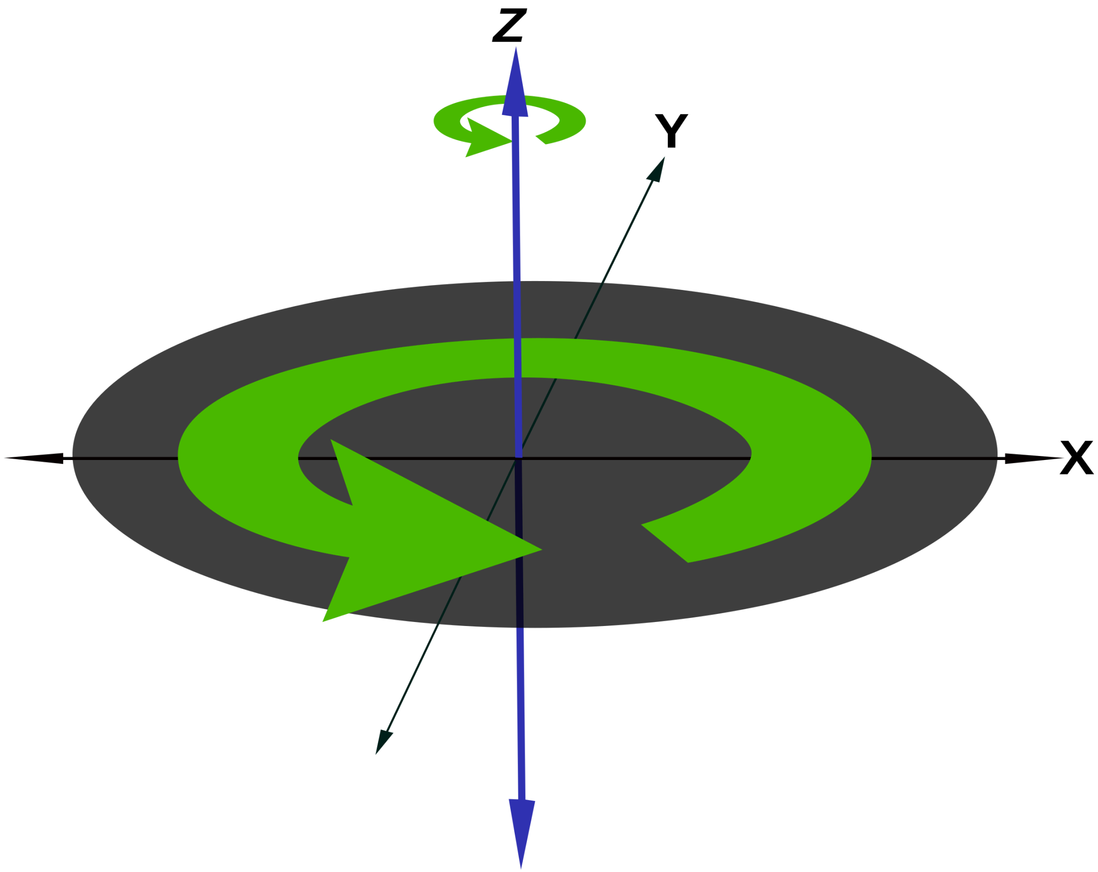
- $SO(n) = \{R \in \mathbb{R}^{n \times n} : \det(R) = 1, RR^T = I\}$
- $SO(n)$: "Special Orthogonal Group"

- "Group": a group under the *matrix multiplication*
- "Orthogonal": $RR^T = I$
- "Special": $\det(R) = 1$

- $SO(2)$: 2D rotations, 1 DoF
- $SO(3)$: 3D rotations, 3 DoF

Topology of $SO(n)$

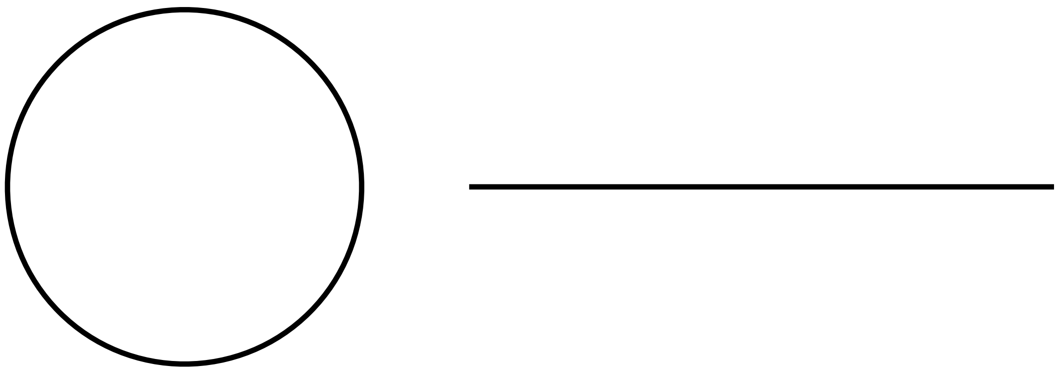
- The topology of $SO(2)$ is the same as a circle



Topology of $SO(n)$

- Circles do not have the same topology as $(-1,1)^n$

$\Longrightarrow$ No differentiable bijections (diffeomorphism) between $SO(2)$ and $(-1,1)^n$



- The topology of $SO(3)$ is also different from $(-1,1)^n$

Why do we care about the topology?

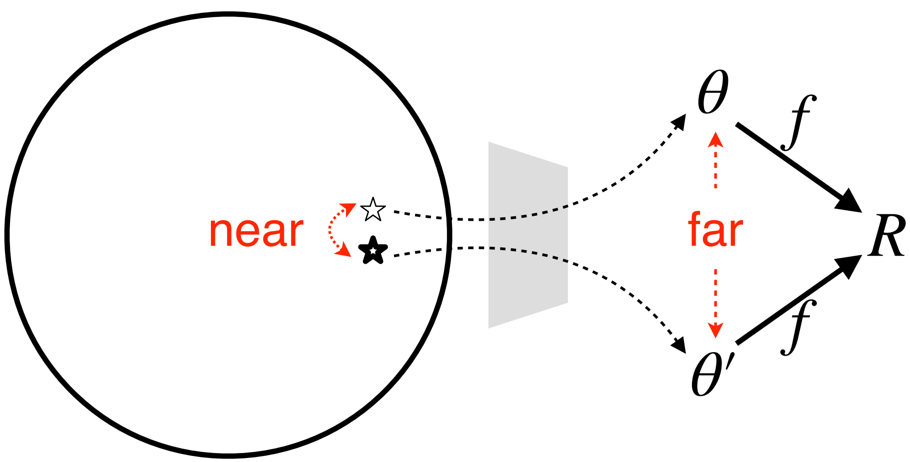
Parameterizing Rotation in Networks is Tricky

- An ideal parameterization $f(\theta) : U \mapsto SO(2)$ to use in networks:
    1. The domain is $(-l, l)^n$ (as network output)

Parameterizing Rotation in Networks is Tricky

- An ideal parameterization $f(\theta) : U \mapsto SO(2)$ to use in networks:
  1. The domain is $(-l, l)^n$
  2. f is a differentiable *bijection*

Otherwise:



- If input data points to network are close, but the θ predictions happen to be far after convergence, the network (a continuous function) will make awful predictions between the two data points!
- Need special network design to overcome the issue (will discuss in future lectures)

Parameterizing Rotation in Networks is Tricky

- An ideal parameterization $f(\theta) : U \mapsto SO(2)$ to use in networks:
    1. The domain is $(-l, l)^n$
    2. f is a differentiable *bijection*
    3. $\forall \theta \forall y \in \mathbf{T}_{f(\theta)}$ with $\|y\| = 1$, there should $\exists x \in \mathbf{T}_{\theta}$, such that $y = \mathrm{D}f[x]$ and $c + \epsilon > \|x\| > c - \epsilon$ for some constant c and small ϵ (all movement in $SO(n)$ should be achieved by movement in the domain with a near constant speed)

Parameterizing Rotation in Networks is Tricky

- An ideal parameterization $f(\theta) : U \mapsto SO(2)$ to use in networks:
  1. The domain is $(-l, l)^n$
  2. f is a differentiable *bijection*
  3. $\forall \theta \forall y \in \mathbf{T}_{f(\theta)}$ with $\|y\| = 1$, there should $\exists x \in \mathbf{T}_{\theta}$, such that $y = \mathrm{D}f[x]$ and $c + \epsilon > \|x\| > c - \epsilon$ for some constant c and small ϵ
- However, 1 and 2 are contradictory by topology!
- For 3, it also creates troubles for the $SO(3)$ case.

Euler Angles

Euler Angle is Very Intuitive

https://www.programmersought.com/article/8900590257/

Euler Angle to Rotation Matrix

- Rotation about principal axis is represented as:

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

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

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

- $R = R_x(\alpha)R_y(\beta)R_z(\gamma)$ for arbitrary rotation

Inspection from Learning Perspective

- Euler Angle is **not unique** for some rotations. For example,

$$R_z(45°)R_y(90°)R_x(45°) = R_z(90°)R_y(90°)R_x(90°)$$

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

https://www.mecademic.com/resources/Euler-angles/Euler-angles

Inspection from Learning Perspective

- Gimbal lock:
  - $\mathrm{D}f$ is rank-deficient at some θ
  - $\Rightarrow$ some movement in $\mathbf{T}_{f(\theta)}(SO(3))$ cannot be achieved

https://www.mecademic.com/resources/Euler-angles/Euler-angles

Inspection from Learning Perspective

- For example: When $\beta = \pi/2$,

$$R = R_x(\alpha)R_y(\pi/2)R_z(\gamma)$$
$$= \begin{bmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{bmatrix}$$

since changing α and γ has the same effects, a degree of freedom disappears!

https://www.mecademic.com/resources/Euler-angles/Euler-angles

Summary

- Euler angle can parameterize every rotation and has good interpretability
- It is not a unique representation at some points
- There are some points where not every change in the target space (rotations) can be realized by a change in the source space (Euler angles)

Axis-Angle

Euler Theorem

- Any rotation in $SO(3)$ is equivalent to rotation about a fixed axis $\omega \in \mathbb{R}^3$ through a positive angle θ
- $\hat{\omega}$: unit vector of rotation axis ($\|\hat{\omega}\| = 1$)
- θ: angle of rotation
- $R \in SO(3) := Rot(\hat{\omega}, \theta)$

Given $\hat{\omega}$ and θ, what is $R \in SO(3)$?

Skew-Symmetric Matrix

- A is skew-symmetric $A = -A^T$

- Skew-symmetric matrix operator:

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \ [\omega] := \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

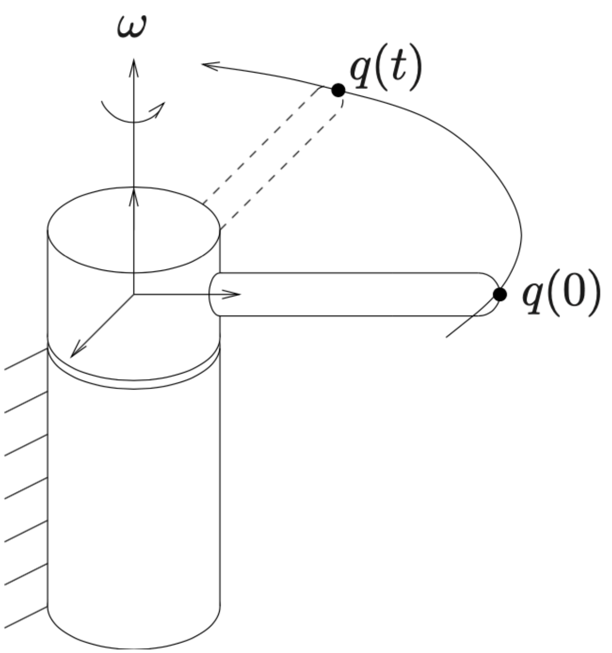
- Cross product can be a linear transformation
    - $a \times b = [a]b$

- **Lie Algebra** of 3D rotation:
    - $so(3) := \{S \in \mathbb{R}^{3\times3} : S^T = -S\}$

Given $\hat{\omega}$ and θ, what is $R \in SO(3)$?

- Consider a point q. At time $t = 0$, the position is q_0

- Rotate q with **unit** angular velocity around axis $\hat{\omega}$, i.e.,
  - $v = \hat{\omega} \times r$
  - $\dot{q}(t) = \hat{\omega} \times q(t) = [\hat{\omega}]q(t)$



Given $\hat{\omega}$ and θ, what is $R \in SO(3)$?

$$\dot{q}(t) = \hat{\omega} \times q(t) = [\hat{\omega}]q(t)$$

$\Rightarrow q(t) = e^{[\hat{\omega}]t}q_0$ (solution of the ODE)

$$\|\hat{\omega}\| = 1$$

$\Rightarrow$ the swept angle $\theta = \|\hat{\omega}t\| = t$

$$\Rightarrow q(\theta) = e^{[\hat{\omega}]\theta}q_0$$

$\Rightarrow Rot(\hat{\omega}, \theta) = e^{[\hat{\omega}]\theta}$ (exponential map)

- $\overrightarrow{\omega} = \hat{\omega}\theta$ is also called **rotation vector** or **exponential coordinate**

Given $\hat{\omega}$ and θ, what is $R \in SO(3)$?

- Definition of Matrix Exponential:

$$e^{[\hat{\omega}]\theta} = I + \theta[\hat{\omega}] + \frac{\theta^2}{2!}[\hat{\omega}]^2 + \frac{\theta^3}{3!}[\hat{\omega}]^3 + \cdots$$

- Sum of infinite series? **Rodrigues Formula**
  - Can prove that $[\hat{\omega}]^3 = -[\hat{\omega}]$
  - Then, use Taylor expansion of **sin** and **cos**
  - $e^{[\hat{\omega}]\theta} = I + [\hat{\omega}]\sin\theta + [\hat{\omega}]^2(1 - \cos\theta)$

Given $R \in SO(3)$, what is $\hat{\omega}$ and θ?

- First question: Is there a **unique** parametrization?
  - No：
    1. $(\hat{\omega}, \theta)$ and $(-\hat{\omega}, -\theta)$ give the same rotation
    2. when $R = I$, $\theta = 0$ and $\hat{\omega}$ can be arbitrary

- When 2 does not happen, and if we also restrict $\theta \in [0, \pi)$, a unique parameterization exists:
  - when $\mathrm{tr}(R) \neq -1$, can be computed by

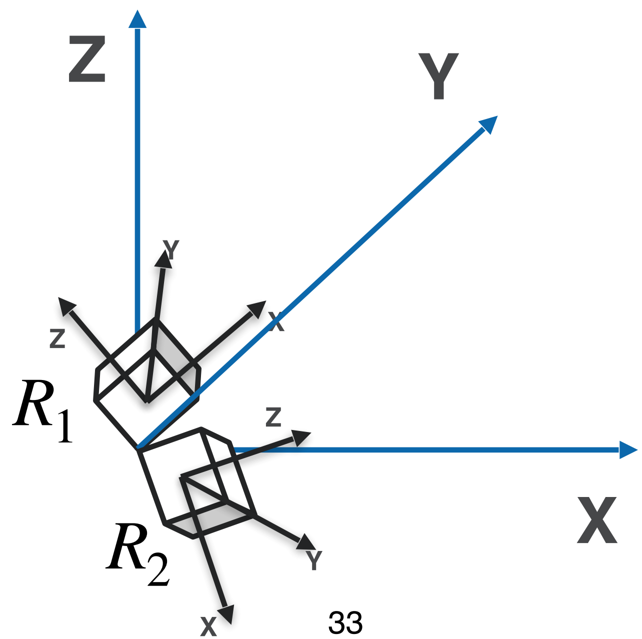
$$\theta = \arccos \frac{1}{2}[\mathrm{tr}(R) - 1], \quad [\hat{\omega}] = \frac{1}{2\sin\theta}(R - R^T)$$

  - when $\mathrm{tr}(R) = -1$, they are the cases that $\theta = \pi$ for rotations around x/y/z axis

Distance between Rotations

- How to measure the distance between rotations (R_1, R_2)?
- A natural view is to measure the (minimal) effort to rotate the body at R_1 pose to R_2 pose:

$$\because\ (R_2 R_1^T) R_1 = R_2 \qquad \therefore\ \mathrm{dist}(R_1, R_2) = \theta(R_2 R_1^T) = \arccos \frac{1}{2}[\mathrm{tr}(R) - 1]$$



Inspection from Learning Perspective

- When used in networks, one prominent issue is:
  - Suppose that you are estimating $\theta\hat{\omega}$ as a 3D vector
  - To keep a unique parameterization, you assume that $\theta \in (0,\pi]$
  - Your current solution is $\pi\hat{\omega}$
  - $(\pi - \epsilon)(-\hat{\omega})$ is mapped to a neighborhood point in $SO(3)$, but it is not in the neighborhood of the domain, hence gradient descent could not achieve it

Summary of Axis-Angle

- Axis-Angle is an intuitive rotation representation
- By adding a constraint to the domain of θ, the parameterization can be unique at most points
- Can be converted to and from rotation matrices by exponential map and its inverse (when possible)
- Induced a distance between rotations which is a metric in $SO(3)$ (independent of parameterization)

Quaternion

Mathematical Definition

- Recall the complex number $a + b\mathbf{i}$
- Quaternion is a more generalized complex number:
  $q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$
  - w is the real part and $\overrightarrow{v} = (x, y, z)$ is the imaginary part
  - Imaginary: $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$
  - anti-commutative :
    $\mathbf{ij} = \mathbf{k} = -\mathbf{ji},\ \mathbf{jk} = \mathbf{i} = -\mathbf{kj},\ \mathbf{ki} = \mathbf{j} = -\mathbf{ik}$

Properties of General Quaternions

- In vector-form, the product of two quaternions:
  For $q_1 = (w_1, \overrightarrow{v}_1)$ and $q_2 = (w_2, \overrightarrow{v}_2)$

$$q_1 q_2 = (w_1 w_2 - \overrightarrow{v}_1^T \overrightarrow{v}_2, w_1 \overrightarrow{v}_2 + w_2 \overrightarrow{v}_1 + \overrightarrow{v}_1 \times \overrightarrow{v}_2)$$

- Conjugate: $q^* = (w, -\overrightarrow{v})$

- Norm: $\|q\| = w^2 + \overrightarrow{v}^T \overrightarrow{v} = qq^* = q^* q$
- Inverse: $q^{-1} := \dfrac{q^*}{\|q\|^2}$

Unit Quaternion as Rotation

- A **unit** quaternion $\|q\| = 1$ can represent a rotation
  - Four numbers plus one constraint $\rightarrow$ 3 DoF

- Geometrically, the shell of a 4D sphere

Unit Quaternion as Rotation

- Rotate a vector $\overrightarrow{x}$ by quaternion q:
  1. Augment $\overrightarrow{x}$ to $x = (0, \overrightarrow{x})$
  2. $x' = qxq^{-1}$
- Compose rotations by quaternion:
  - $(q_2(q_1 x q_1^*) q_2^*)$: first rotate by q_1 and then by q_2
  - Since $(q_2(q_1 x q_1^*) q_2^*) = (q_2 q_1) x (q_1^* q_2^*)$, we conclude that composing rotations is as simple as multiplying quaternions!

Conversation between Quaternions and Angle-Axis

- Exponential coordinate → Quaternion:

$$q = [\cos(\theta/2), \sin(\theta/2)\hat{\omega}]$$

Quaternion is very close to angle-axis representation!

- Exponential coordinate ← Quaternion:

$$\theta = 2\arccos(w), \qquad \hat{\omega} = \begin{cases} \dfrac{1}{\sin(\theta/2)}\vec{v} & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

Conversation between Quaternion and Rotation Matrix

- Rotation $\leftarrow$ Quaternion

$$R(q) = E(q)G(q)^T$$

where $E(q) = [-\overrightarrow{q}, wI + [\overrightarrow{q}]]$ and

$$G(q) = [-\overrightarrow{q}, wI - [\overrightarrow{q}]]$$

- Rotation $\rightarrow$ Quaternion
  - Rotation $\rightarrow$ Angle-Axis $\rightarrow$ Quaternion

Inspection from Learning Perspective

- Each rotation corresponds to two quaternions ("double-covering")
- Need to normalize to unit length in networks. This normalization may cause big/small gradients in practice

More about Quaternion

- Quaternion is computationally cheap:
  - Internal representation of Physical Engine and Robot
  - Pay attention to convention (w, x, y, z) or (x, y, z, w)?
  - (w, x, y, z): SAPIEN, transforms3d, Eigen, blender, MuJoCo, V-Rep
  - (x, y, z, w): ROS, PhysX, PyBullet

Summary of Quaternion

- Very useful and popular in practice
- 4D parameterization, compact and efficient to compute
- Good numerical properties in general

Summary of Rotation Representations

	Inverse?	Composing?	Any local movement in SO(3) can be achieved by local movement in the domain?
Rotation Matrix	✔	✔	N/A
Euler Angle	Complicated	Complicated	No
Angle-axis	✔	Complicated	?
Skew-symmetrical Matrix	✔	Complicated	?
Quaternion	✔	✔	✔

? means no singularity with single exceptions

Resources

- A useful torch library that you can play with is “kornia”

- Use with cautious to its numerical properties