L3: Rotation

Hao Su

Agenda

- Multi-Link Rigid-Body Geometry
- Concepts of $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$
- Angle-Axis Parameterization of Rotations
- Quaternions
- Local Structure of $\mathbb{SO}(3)$

Multi-Link Rigid-Body Geometry

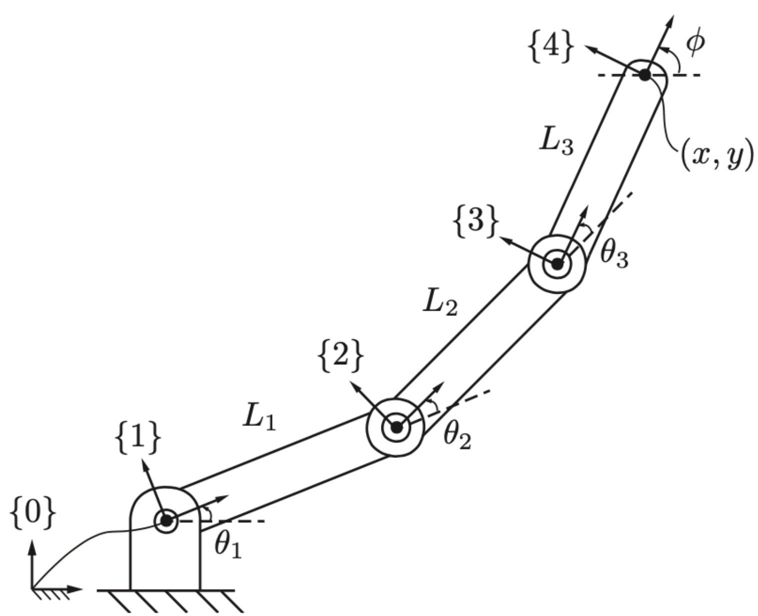
Link and Joint

Link:

- **Links** are the rigid-body connected in sequence

Joint:

- **Joints** are the connectors between links. They determine the DoF of motion between adjacent links

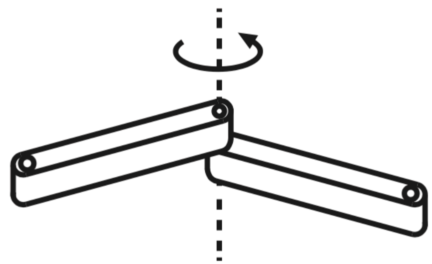


Base Link and End-Effector Link

- Base link / root link:
  - The 0-th link of the robot
  - Regarded as the “fixed” reference
  - The spatial frame $\mathscr{F}_s$ is attached to it
- End-effector link
  - The last link
  - e.g., the gripper
  - A frame $\mathscr{F}_e$ is attached to it

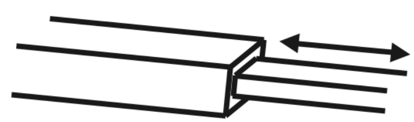
Two Common Joint Types

- Revolute/Hinge/Rotational joint



Revolute
(R)

- Prismatic/Translational joint



Prismatic
(P)

Kinematics: The Basic Geometry Calculation Task

- Kinematics: Describing the motion of bodies (position and velocity)
- Kinematics **does not consider** how to achieve motion via force

Kinematics Configuration

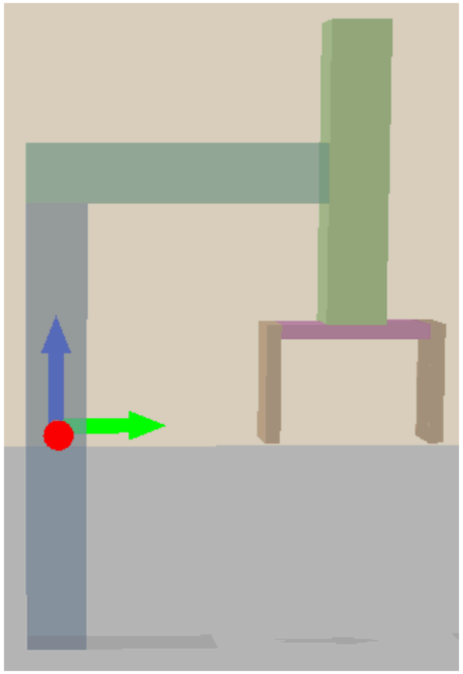
- Assuming frames are assigned to each link, we can parameterize **the pose of each joint**
    - Using the relative **angle** and **translation** between adjacent frames
- Two representations of the pose of the end-effector
    - **Joint space:** The space in which each coordinate is a vector of joint poses (**angles** around **joint axis**)
    - **Cartesian space:** The space of the rigid transformations of the end-effector by $(R_{s\to e}, \mathbf{t}_{s\to e})$, where $\mathcal{F}_e$ is the end-effector frame

Kinematics Equations

- Map the joint space coordinate $\theta \in \mathbb{R}^n$ to a transformation matrix T:

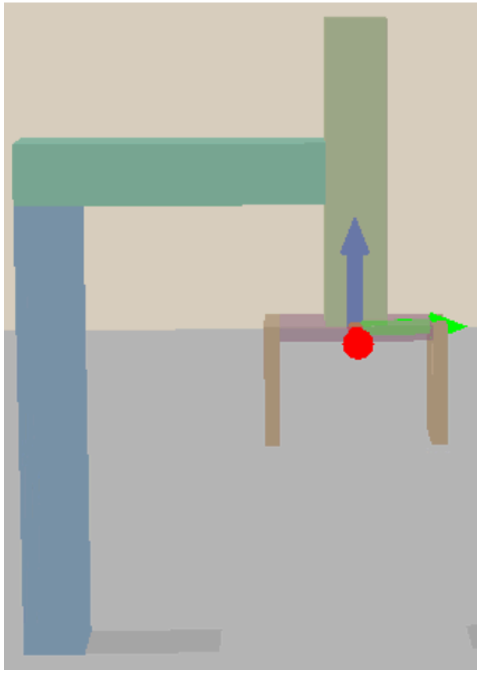
$$T_{s \to e} = f(\theta)$$

- Calculated by composing transformations along the kinematic chain



base

→



end_effector

$$T^0_{0\to3} = T^0_{0\to1}T^1_{1\to2}T^2_{2\to3} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & -\sin\theta_1(l_2+l_3) \\ \sin\theta_1 & \cos\theta_1 & 0 & \cos\theta_1(l_2+l_3) \\ 0 & 0 & 1 & l_1 - l_4 + \theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Kinematics

- Given the forward kinematics $T_{s \to e}(\theta)$ and the target pose $T_{target} = \mathbb{SE}(3)$, find solutions θ that satisfy $T_{s \to e}(\theta) = T_{target}$

How to Relate the Motion in Joint Space and Cartesian Space?

- Q1: If the robot moves by $\Delta\theta$ in the joint space, how much will it move in the Cartesian space? (Forward Kinematics)
- Q2: If the robot would move the end-effector by Δx in the Cartesian space, how shall it change its joint poses? (Inverse Kinematic)
- Suppose we parameterize (R, t) by the **angles** around **axis**, we need to derive the differential map

- We will study the differentiability of rotation and rigid transformations

$\mathbb{SO}(3)$ and $\mathbb{SE}(3)$

$\mathbb{SO}(3)$: The Space of Rotations

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

- “Group”: roughly, closed under matrix multiplication
- “Orthogonal”: $RR^T = I$
- “Special”: $\det(R) = 1$

- $\mathbb{SO}(2)$: 2D rotations, 1 DoF
- $\mathbb{SO}(3)$: 3D rotations, 3 DoF

$\mathbb{SE}(3)$: The Space of Rigid Transformations

- $\mathbb{SE}(3) := \left\{ T = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix}, R \in \mathbb{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\}$
- $\mathbb{SE}(3)$: “Special Euclidean Group”

- “Group”: roughly, closed under matrix multiplication
- “Euclidean”: R and $\mathbf{t}$
- “Special”: $\det(R) = 1$

- 6 DoF

- We need some theoretical understanding of $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$
  - The topological structure
  - The parameterization
  - The differentiable properties

Angle-Axis Parameterization of Rotations

Euler's Theorem

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

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

Skew-Symmetric Matrix

- A is skew-symmetric $A = -A^T$
- Skew-symmetric matrix operator:

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, [a] := \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

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

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

- We can show that, for any $x \in \mathbb{R}^3$

$$\begin{aligned}\mathrm{Rot}(\hat{\omega}, \theta)x &= x + (\sin\theta)\hat{\omega} \times x + (1 - \cos\theta)\hat{\omega} \times (\hat{\omega} \times x) \\ &= \{I + [\hat{\omega}]\sin\theta + [\hat{\omega}]^2(1 - \cos\theta)\}x \qquad (1)\end{aligned}$$

Derivation: https://en.wikipedia.org/wiki/Rotation_matrix

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

- We can show that, for any $x \in \mathbb{R}^3$
$$\mathrm{Rot}(\hat{\omega}, \theta)x = x + (\sin\theta)\hat{\omega} \times x + (1 - \cos\theta)\hat{\omega} \times (\hat{\omega} \times x)$$
$$= \{I + [\hat{\omega}]\sin\theta + [\hat{\omega}]^2(1 - \cos\theta)\}x \qquad (1)$$
- By Taylor's expansion of **sin**, **cos**, $[\hat{\omega}]^3 = -[\hat{\omega}]$, and above
$$\mathrm{Rot}(\hat{\omega}, \theta)x = (I + \theta[\hat{\omega}] + \frac{\theta^2}{2!}[\hat{\omega}]^2 + \frac{\theta^3}{3!}[\hat{\omega}]^3 + \cdots)x$$

Derivation: https://en.wikipedia.org/wiki/Rotation_matrix

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

- We can show that, for any $x \in \mathbb{R}^3$
$$\mathrm{Rot}(\hat{\omega}, \theta)x = x + (\sin\theta)\hat{\omega} \times x + (1 - \cos\theta)\hat{\omega} \times (\hat{\omega} \times x)$$
$$= \{I + [\hat{\omega}]\sin\theta + [\hat{\omega}]^2(1 - \cos\theta)\}x \qquad (1)$$
- By Taylor's expansion of **sin**, **cos**, $[\hat{\omega}]^3 = -[\hat{\omega}]$, and above
$$\mathrm{Rot}(\hat{\omega}, \theta)x = (I + \theta[\hat{\omega}] + \frac{\theta^2}{2!}[\hat{\omega}]^2 + \frac{\theta^3}{3!}[\hat{\omega}]^3 + \cdots)x$$
- Recall Taylor's expansion of exponential,
$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots$$

Derivation: https://en.wikipedia.org/wiki/Rotation_matrix

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

- We can show that, for any $x \in \mathbb{R}^3$
$$\mathrm{Rot}(\hat{\omega}, \theta)x = x + (\sin\theta)\hat{\omega} \times x + (1 - \cos\theta)\hat{\omega} \times (\hat{\omega} \times x)$$
$$= \{I + [\hat{\omega}]\sin\theta + [\hat{\omega}]^2(1 - \cos\theta)\}x \qquad (1)$$
- By Taylor's expansion of **sin**, **cos**, $[\hat{\omega}]^3 = -[\hat{\omega}]$, and above
$$\mathrm{Rot}(\hat{\omega}, \theta)x = (I + \theta[\hat{\omega}] + \frac{\theta^2}{2!}[\hat{\omega}]^2 + \frac{\theta^3}{3!}[\hat{\omega}]^3 + \cdots)x$$
- Recall Taylor's expansion of exponential,
$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots$$
- Formally, we have:
$$\mathrm{Rot}(\hat{\omega}, \theta)x = e^{[\hat{\omega}]\theta}x, \forall x \in \mathbb{R}^3$$

Derivation: https://en.wikipedia.org/wiki/Rotation_matrix

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

- By $\text{Rot}(\hat{\omega}, \theta)x = e^{[\hat{\omega}]\theta}x, \forall x \in \mathbb{R}^3$,

$$\text{Rot}(\hat{\omega}, \theta) \equiv e^{[\hat{\omega}]\theta}$$

- This is under such a **Definition of Matrix Exponential**:

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

- By (1) in the last slide, we obtain **Rodrigues** formula:

$$e^{[\hat{\omega}]\theta} = I + [\hat{\omega}]\sin\theta + [\hat{\omega}]^2(1 - \cos\theta)$$

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

- In the angle-axis representation of $\mathrm{Rot}(\hat{\omega}, \theta) = e^{[\hat{\omega}]\theta}$
- $\overrightarrow{\theta} = \hat{\omega}\theta$ is also called the **rotation vector**, or **exponential coordinate**

Rodrigues Formula

- 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**
  - $e^{[\hat{\omega}]\theta} = I + [\hat{\omega}]\sin\theta + [\hat{\omega}]^2(1 - \cos\theta)$

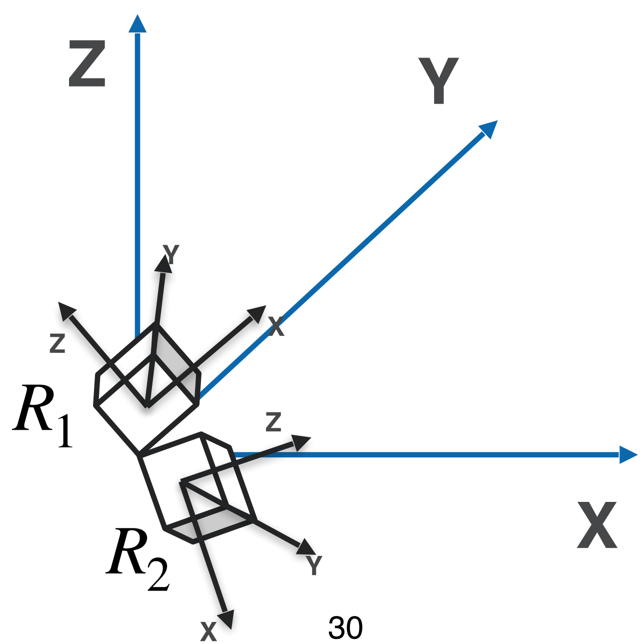
Given $R \in \mathbb{SO}(3)$, what is $\hat{\omega}$ and θ?

- 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
    3. $(\hat{\omega}, \pi)$ and $(-\hat{\omega}, \pi)$ give the same rotation $(\mathrm{tr}(R) = -1)$
- If we restrict $\theta \in (0,\pi)$, a unique parameterization exists:
  - $\theta = \arccos \frac{1}{2}[\mathrm{tr}(R) - 1], \quad [\hat{\omega}] = \frac{1}{2\sin\theta}(R - R^T)$

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:

$$\because (R_2R_1^T)R_1 = R_2 \qquad \therefore \operatorname{dist}(R_1, R_2) = \theta(R_2R_1^T) = \arccos \frac{1}{2}[\operatorname{tr}(R_2R_1^T) - 1]$$



Quaternion

Note: In this section, $\overrightarrow{x} \in \mathbb{R}^3$ and $q \in \mathbb{R}^4$

Quaternion is a “Number”

- 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

- Vector form: $q = (w, \overrightarrow{v})$
- Product:
  - For $q_1 = (w_1, \overrightarrow{v}_1)$ and $q_2 = (w_2, \overrightarrow{v}_2)$,
    $q_1 q_2 = (w_1 w_2 - \overrightarrow{v}_1^T \overrightarrow{v}_2, w_1 \overrightarrow{v}_2 + w_2 \overrightarrow{v}_1 + \overrightarrow{v}_1 \times \overrightarrow{v}_2)$
  - Not commutable (note that $\overrightarrow{v}_1 \times \overrightarrow{v}_2 \neq \overrightarrow{v}_2 \times \overrightarrow{v}_1$)
- Conjugate: $q^* = (w, -\overrightarrow{v})$
- Norm: $\|q\|^2 = w^2 + \overrightarrow{v}^T \overrightarrow{v} = qq^* = q^* q$
- 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

Build Rotation Quaternion

- Exponential coordinate → Quaternion:

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

- Quaternion is very close to angle-axis representation!
- Exponential coordinate ← Quaternion:

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

Unit Quaternion as Rotation

- Rotate a vector $\overrightarrow{x}$ by a 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 $(q_2(q_1 x q_1^*) q_2^*) = (q_2 q_1) x (q_1^* q_2^*)$, composing rotations is as simple as multiplying quaternions!

Conversation between Quaternion and Rotation Matrix

- Rotation ← Quaternion

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

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

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

- Rotation → Quaternion
  - Rotation → Angle-Axis → Quaternion

Inspection

- 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

Local Structure of $\mathbb{SO}(3)$

Local Structure of $\mathbb{SO}(3)$

- Definition of Matrix Exponential:

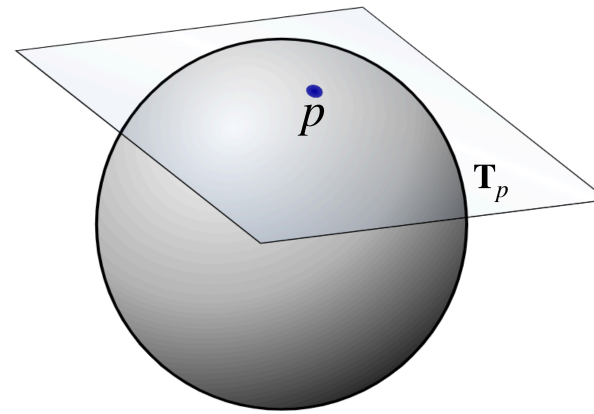
$$e^{[\hat{\omega}]\theta} = I + \theta[\hat{\omega}] + \frac{\theta^2}{2!}[\hat{\omega}]^2 + \frac{\theta^3}{3!}[\hat{\omega}]^3 + \cdots$$

- Note:
  - $e^{A+B} = e^A e^B$ only when $AB - BA = 0$
- When $\theta \approx 0$, $e^{[\hat{\omega}]\theta} = I + \theta[\hat{\omega}] + o(\theta[\hat{\omega}])$

Local Structure of $\mathbb{SO}(3)$

- By $e^{[\hat{\omega}]\theta} = I + \theta[\hat{\omega}] + o(\theta[\hat{\omega}])$ when $\theta \approx 0$,

$$e^{[\overrightarrow{\theta}]} - I = [\overrightarrow{\theta}] + o([\overrightarrow{\theta}])$$



- Interpretation:
  - $[\overrightarrow{\theta}]$ is a linear subspace of $\mathbb{R}^{3\times 3}$
  - $e^{[\overrightarrow{\theta}]} \to I$ as $[\overrightarrow{\theta}] \to 0$
  - **Any** local movement in $\mathbb{SO}(3)$ around I, which is $\approx e^{[\overrightarrow{\theta}]} - I$, can be approximated by $[\overrightarrow{\theta}]$
- The set of $[\overrightarrow{\theta}]$ forms the tangent space of $\mathbb{SO}(3)$ at I

Lie algebra $\mathfrak{so}(3)$ of $\mathbb{SO}(3)$

- The set of $[\overrightarrow{\theta}]$ is the tangent space of $\mathbb{SO}(3)$ at $R = I$
  - Ex: What is the tangent space at any $R \in \mathbb{SO}(3)$?
    - $\because e^{[\overrightarrow{\theta}]} - I = [\overrightarrow{\theta}] + o([\overrightarrow{\theta}]),\ \therefore e^{[\overrightarrow{\theta}]}R - R = [\overrightarrow{\theta}]R + o([\overrightarrow{\theta}])$
    - i.e., $\forall R' \in \mathbb{SO}(3)$ near R, $\exists\, \overrightarrow{\theta} \in \mathbb{R}^3$ such that $R' \approx R + [\overrightarrow{\theta}]R$
    - So the tangent space at R is $\{SR : S \in \mathbb{R}^{3\times3}, S^T = -S\}$
- We give this set a name, the "Lie algebra of $\mathbb{SO}(3)$"
  - $\mathfrak{so}(3) := \{S \in \mathbb{R}^{3\times3} : S^T = -S\}$

Why called "algebra"?

- Introducing Lie bracket $[A, B] = AB - BA$, and the set of skew-symmetric matrices are closed under this binary operator
- Then, the set of skew-symmetric matrices form an "algebra", because
  - The set is closed under Lie bracket
  - Left and right distributive law are satisfied under Lie bracket

Read by Yourself

$\dot{R}$

- Let us first parameterize the rotation of a body frame by time:
    - An observer associated to $\mathcal{F}_o$ records the motion as $R^o_{s'\to b(t)}$, where the body frame is at $\mathcal{F}_{b(t)}$.

$\dot{R}$

$$
\begin{aligned}
R^o_{s'\to b(t+\Delta t)} - R^o_{s'\to b(t)} &= R^o_{b(t)\to b(t+\Delta t)} R^o_{s'\to b(t)} - R^o_{s'\to b(t)} \\
&= e^{[\chi^o_{b(t)\to b(t+\Delta t)}]} R^o_{s'\to b(t)} - R^o_{s'\to b(t)} \\
&\approx [\chi^o_{b(t)\to b(t+\Delta t)}] R^o_{s'\to b(t)}
\end{aligned}
$$

- Divided by Δt and take the limit, we have

$$
\begin{aligned}
\dot{R}^o_{s'\to b(t)} &= \lim_{\Delta t \to 0} \left[\frac{\overrightarrow{\theta}^o_{b(t)\to b(t+\Delta t)}}{\Delta t} \right] R^o_{s'\to b(t)} \\
&= [\omega^o_{b(t)}] T^o_{s'\to b(t)}
\end{aligned}
$$

- $\omega^o_{b(t)} := \lim_{\Delta t \to 0} \dfrac{\overrightarrow{\theta}^o_{b(t)\to b(t+\Delta t)}}{\Delta t}$ is the instant angular velocity

Basic Challenge in Parameterizing $\mathbb{SO}(3)$

$\mathbb{SO}(3)$: The Space of Rotations

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

- "Group": roughly, closed under matrix multiplication
- "Orthogonal": $RR^T = I$
- "Special": $\det(R) = 1$

- $\mathbb{SO}(2)$: 2D rotations, 1 DoF
- $\mathbb{SO}(3)$: 3D rotations, 3 DoF

Read by Yourself

Parameterization

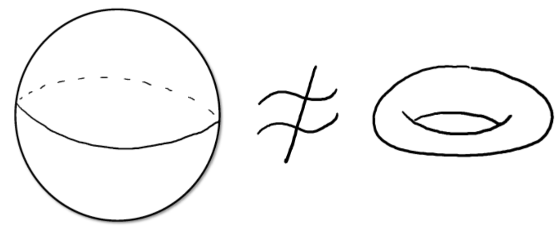
- Record $R \in \mathbb{SO}(3)$ by real numbers
- In other words, we look for mappings between $\mathbb{R}^d$ and $\mathbb{SO}(3)$:

$$f(\theta) = R_\theta$$

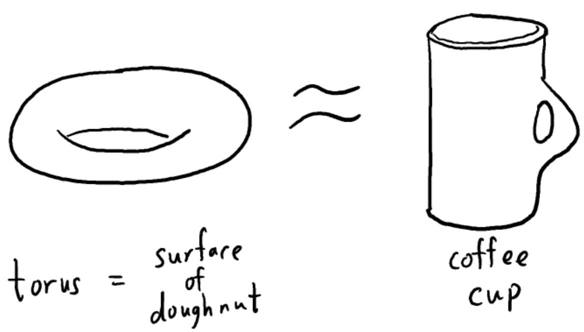
Read by Yourself

Prereq.: Topology

- Topology: Structural Properties of a Manifold



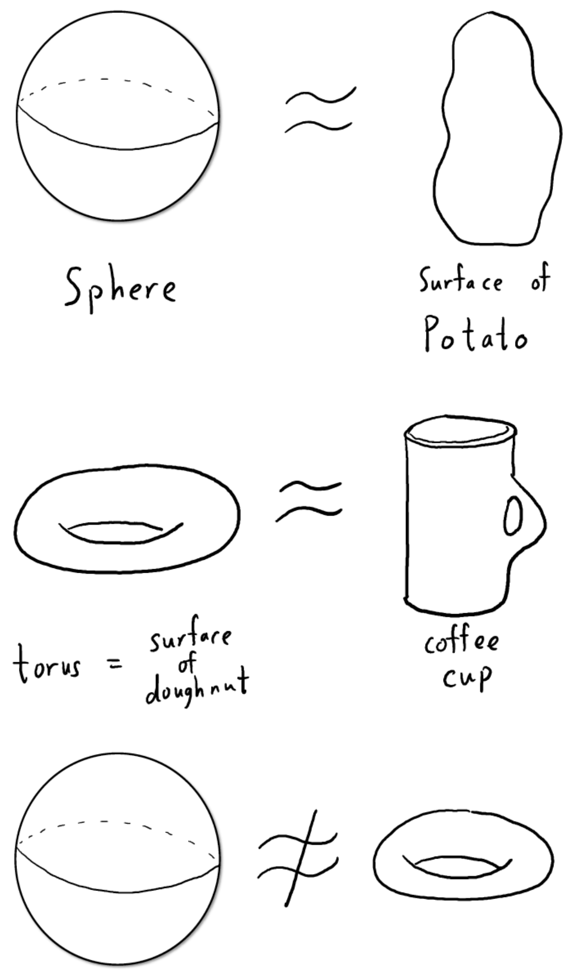
- Two surfaces M and N are *topologically equivalent* if there is a **differentiable bijection** between M and N



Read by Yourself

Prereq: Topology

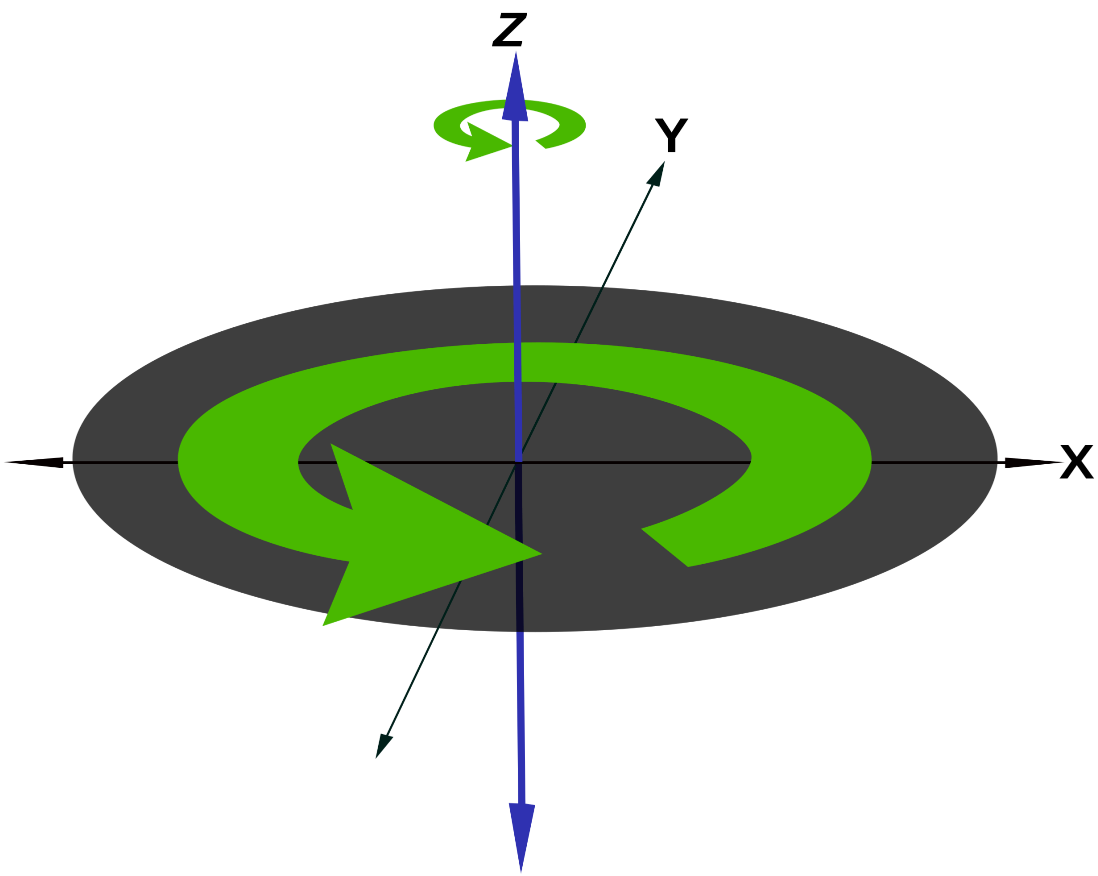
- More examples:



Read by Yourself

Topology of $\mathbb{SO}(n)$

- The topology of $\mathbb{SO}(2)$ is the same as a circle

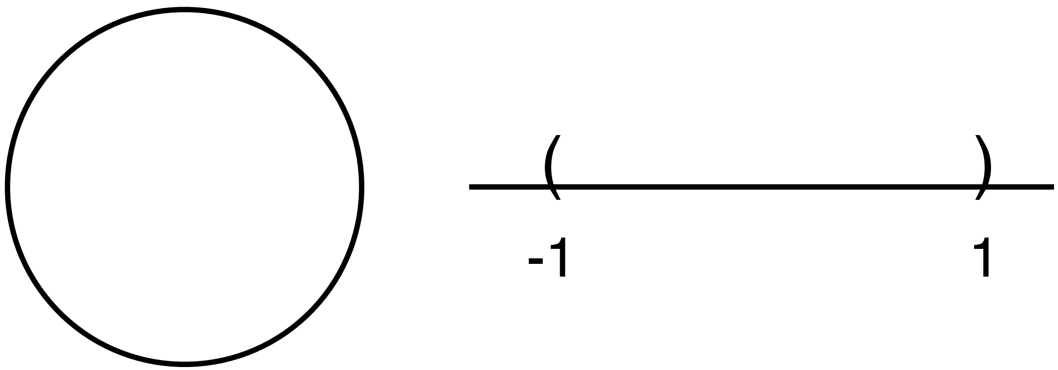


Read by Yourself

Topology of $\mathbb{SO}(n)$

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

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



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

Read by Yourself

Parameterizing Rotations is Tricky

- Although $\mathbb{SO}(3)$ only has 3 DoF, you cannot build a differentiable bijection between $\mathbb{SO}(3)$ and any subset of $\mathbb{R}^3$

- Even parameterizing $\mathbb{SO}(3)$ by $\mathbb{R}^d$ with $d > 3$,
  - we cannot build differentiable bijections with $(-1,1)^d$
  - we have to either introduce constraints, or bear with singularities and the "multi-cover" issue

- The challenge brings a lot of trouble to optimization and learning

Read by Yourself

Recent Progress on the Theoretical Understanding of Rotation Parameterization

- Revisiting the Continuity of Rotation Representations in Neural Networks, Xiang et al.

Read by Yourself

Euler Angle is Very Intuitive

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

Read by Yourself

Euler Angle to Rotation Matrix

- Rotation about principal axis is represented as:

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

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

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

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

Read by Yourself

Inspection

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

$$R_z(45^\circ)R_y(90^\circ)R_x(45^\circ) = R_z(90^\circ)R_y(90^\circ)R_x(90^\circ)$$

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

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

Read by Yourself

Inspection

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

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

Read by Yourself

Inspection

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

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

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

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

Read by Yourself

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)

Read by Yourself

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	?
Quaternion	✔	✔	✔

? means no singularity with single exceptions

Summary of Rotation Representations

- It is quite often that, we use
  - rotation matrices to define concepts
  - Euler angles to visualize rotations
  - angle-axis representation to visualize rotations and calculate derivatives
  - quaternion to write fast codes

Resources

- In Python, you could use the transforms3d library
- For differentiable transformations, you can play with is "Kornia", but use with cautious to its numerical properties
- "ceres" is a C++ library that is quite useful