清華大學 交叉信息研究院
Tsinghua University Institute for Interdisciplinary Information Sciences

ISR Lab
Intelligent Systems and Robotics Lab

Advanced Topics for Robotics
智能机器人前沿探究

Lecture 2: Robot Modeling

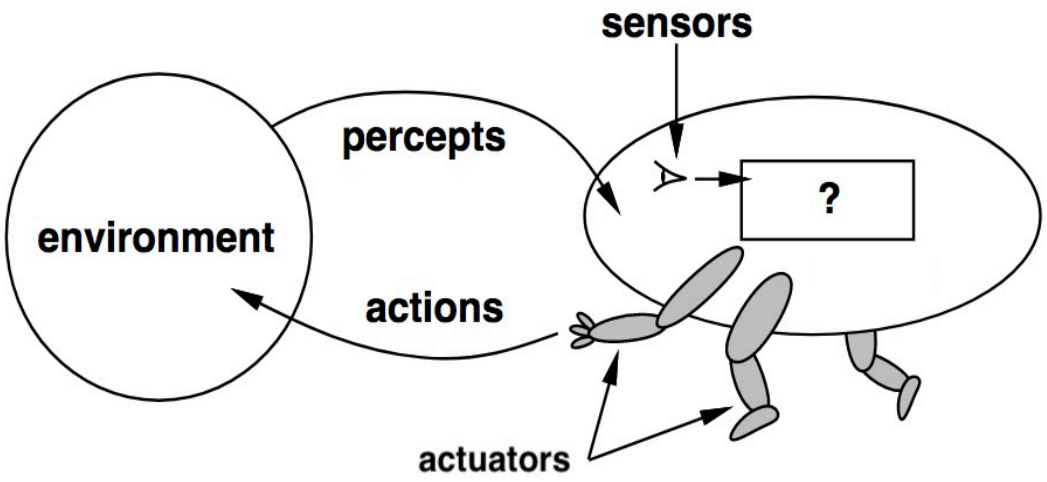
Jianyu Chen (陈建宇)

Institute for Interdisciplinary Information Sciences

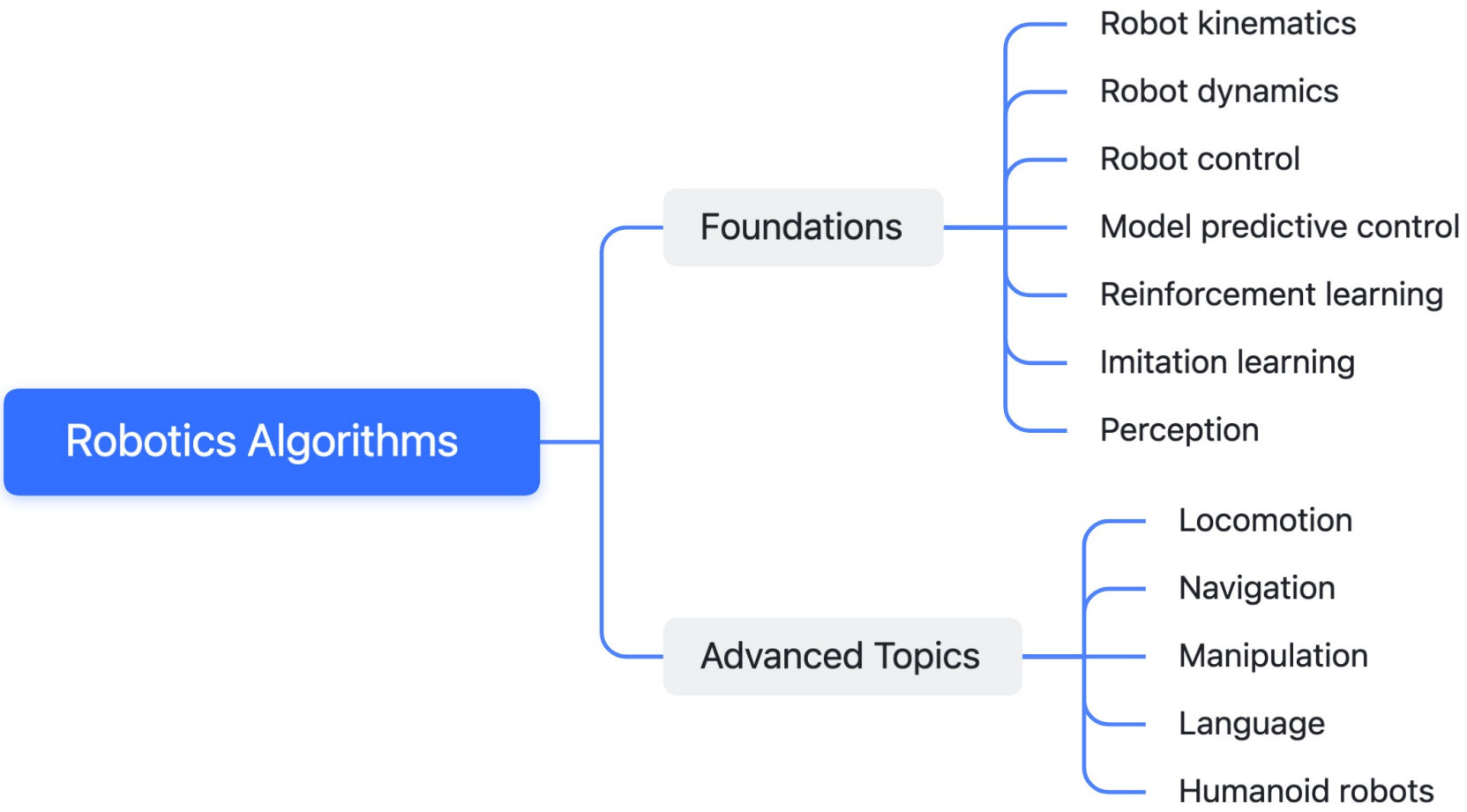
Tsinghua University

Robot

- Definition of a robot
  - A robot is an automatic machine with **physical body** and **intelligence** which **interact with real world environment physically**

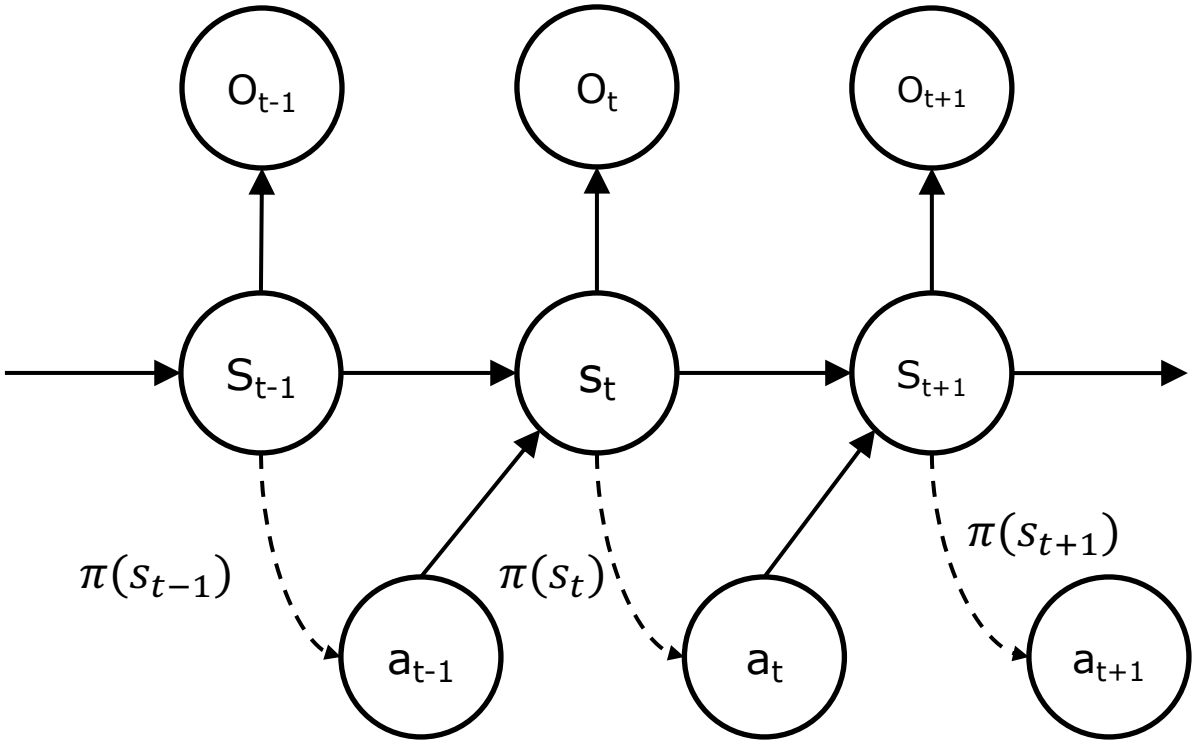


Lecture topics



A General Problem formulation

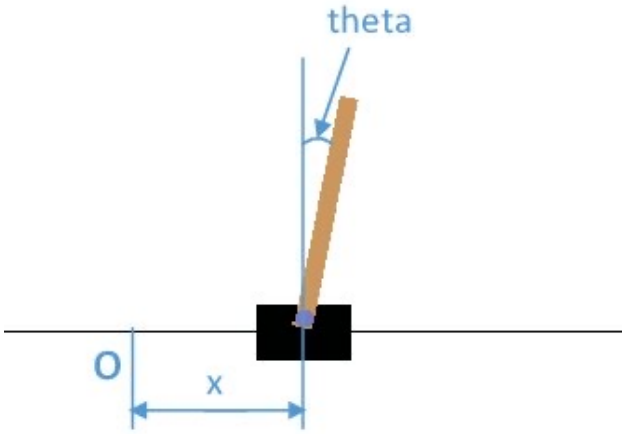
- Let's interpret it more clearly mathematically



s: state
a: action
o: observation
π : policy

State space

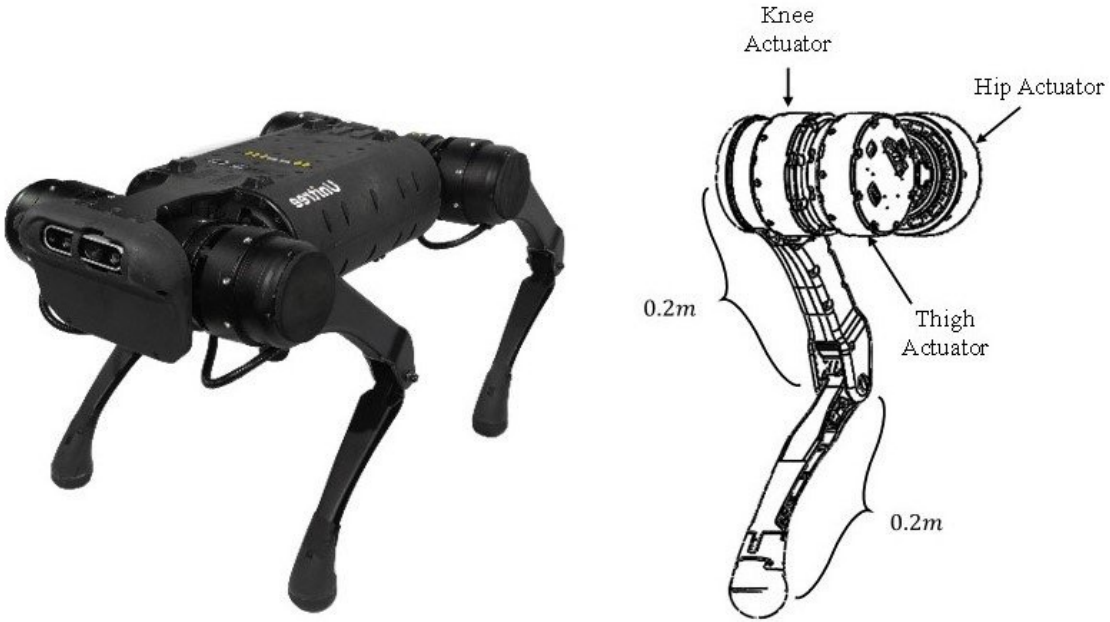
- **State variable:**
  - The smallest possible subset of system variables
  - that can represent the entire state of the system at any given time
  - such that the future evolvement of the system only depends on the current state, not the past (Markov).



- What is the state variable of the cart-pole?
- How about a robot?

State space of a robot

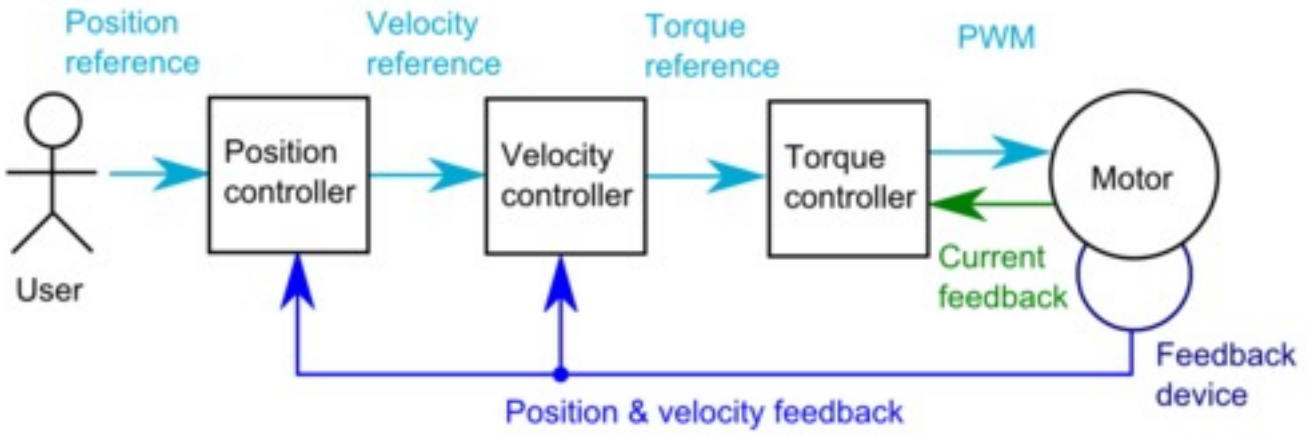
- Joint positions
- Joint velocities
- Base position (position and orientation)
- Base velocity (linear and angular)



Unitree A1

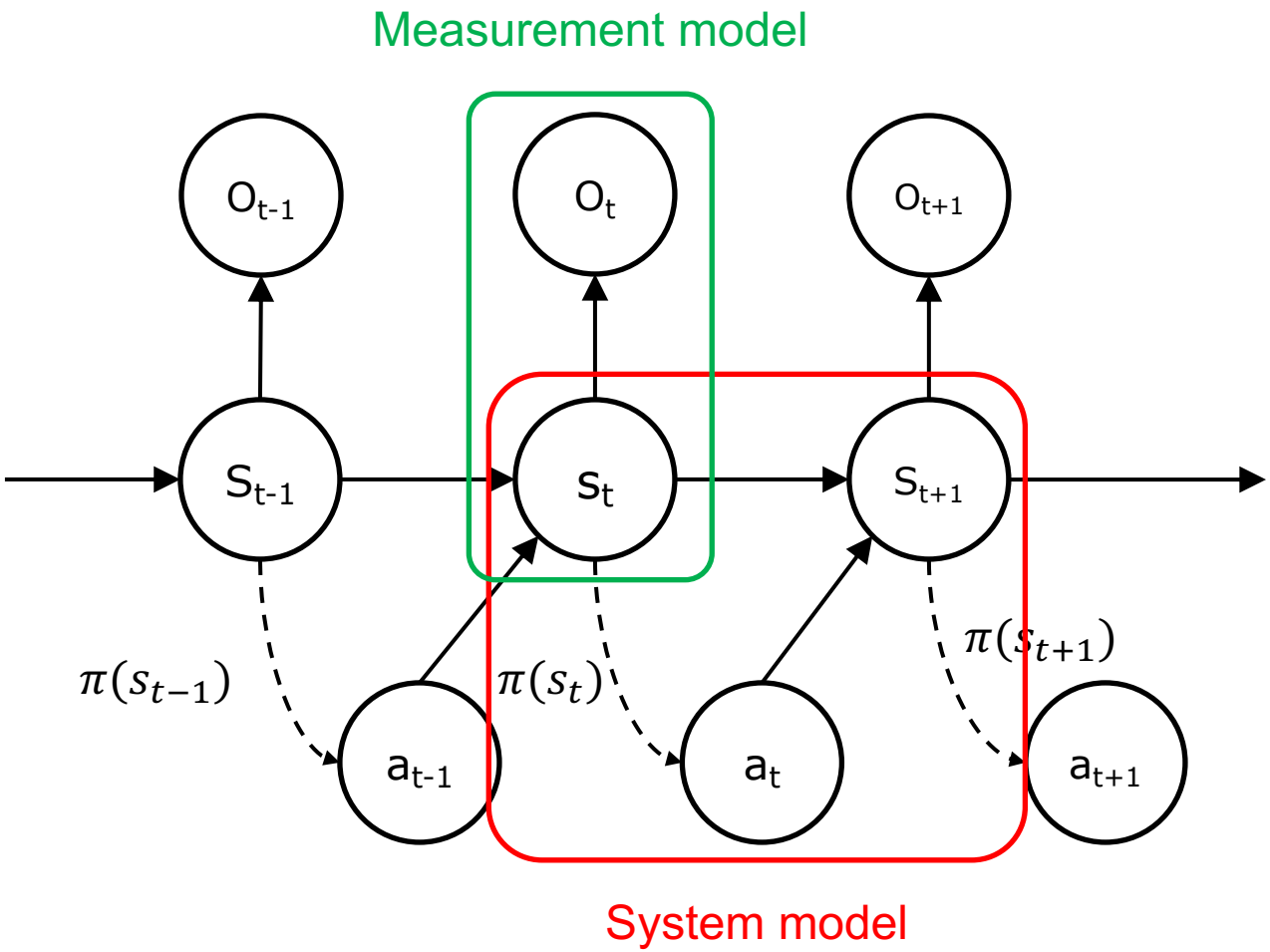
Action space

- Different choices of action space
  - Torque
  - Velocity
  - Position
  - Mixed



Modeling

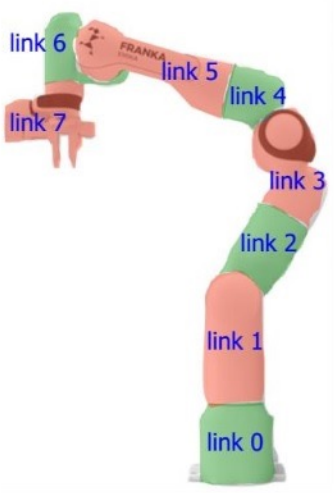
- How to model a robot generally

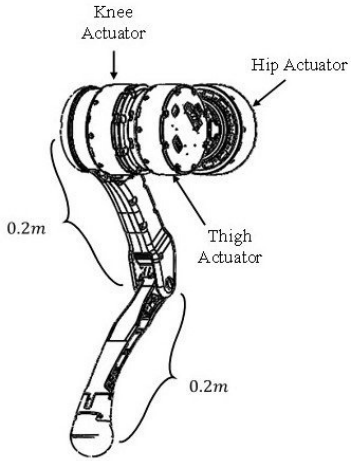


s: state
a: action
o: observation
π : policy

Robot kinematics

- Describe the motion of the robot
  - Positions and orientations
  - Linear and angular velocities
  - Scale to high-dof robots

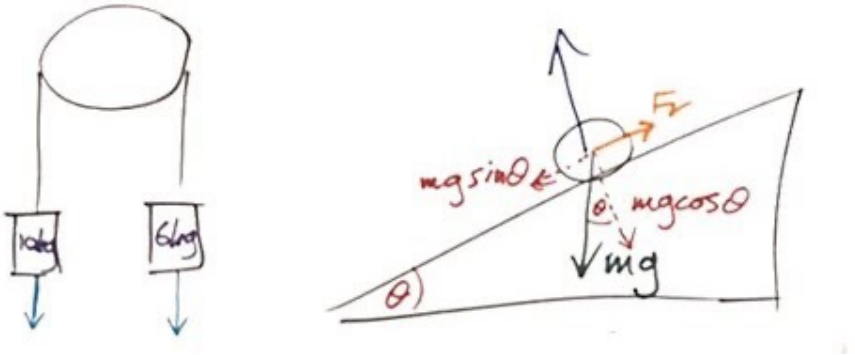




Robot dynamics

- Relationships between **force** and **accelerations** for robots

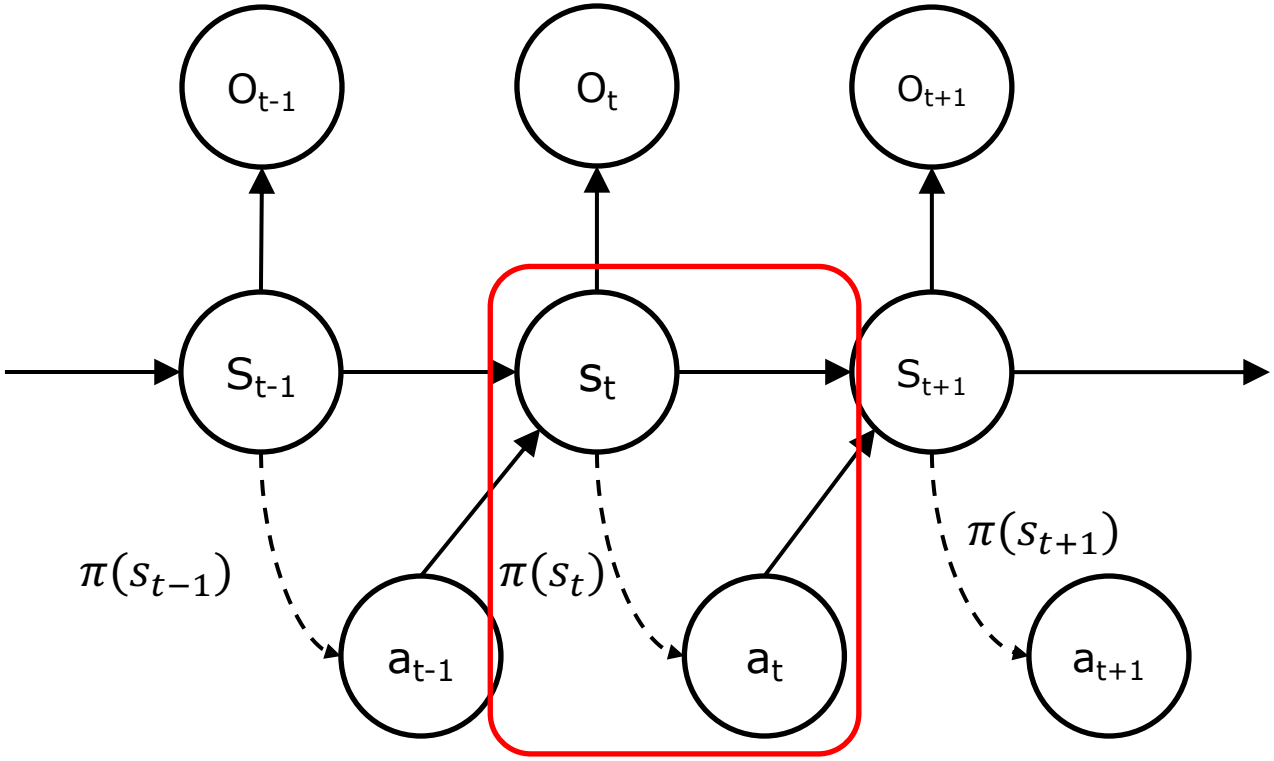
$$F = m\,a$$



$$\tau \Longleftrightarrow \theta,\ \dot{\theta},\ \ddot{\theta}$$

$$\tau = M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) + J^{\mathrm{T}}(\theta)\mathcal{F}_{\mathrm{tip}}$$

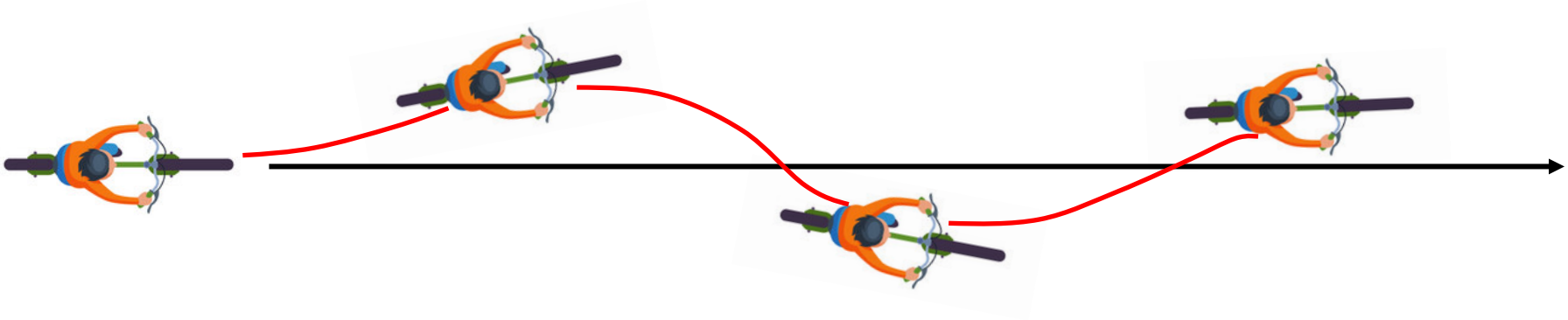
Policy

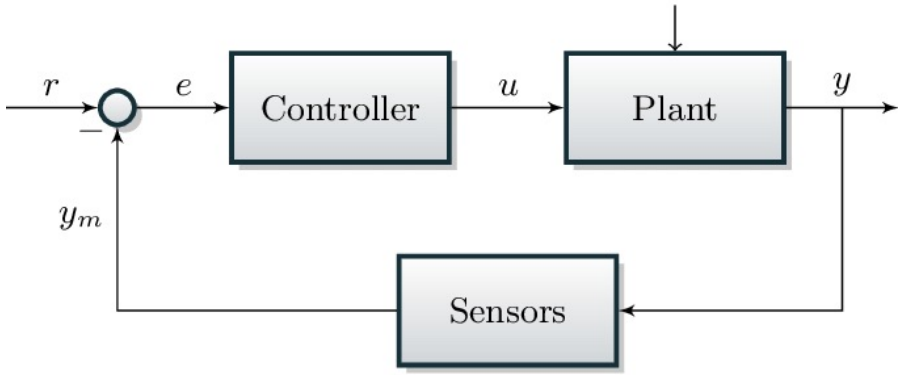


s: state
a: action
o: observation
π : policy

Feedback control

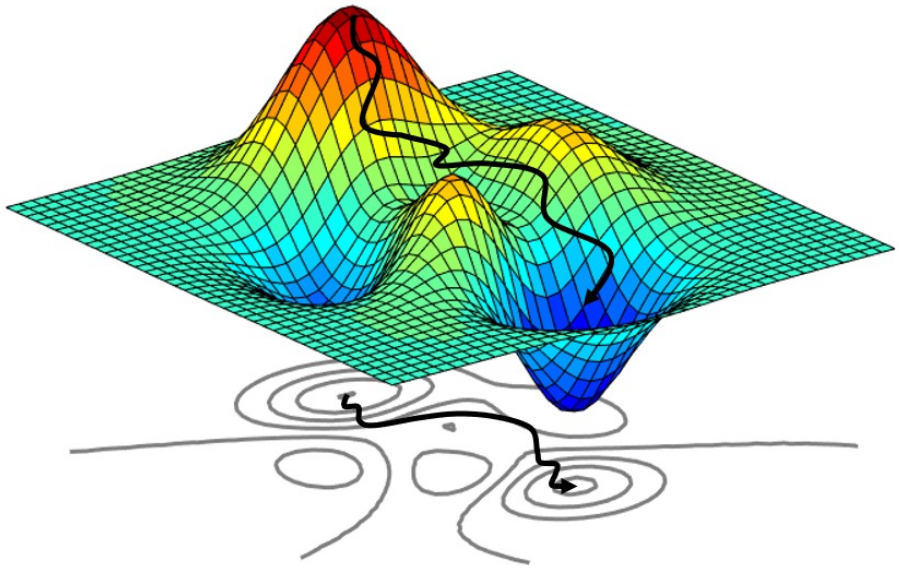
- Principles for **feedback control** of robots





Optimization-based control

- Planning and control using **optimization** based on robot models



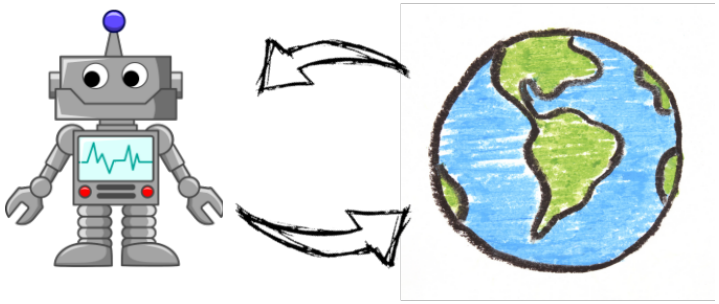
$$u(\cdot), x(\cdot) = \underset{u(\cdot),x(\cdot)}{\operatorname{argmin}} \; l_T(x(T)) + \int_{t=0}^{T} l(x(t), u(t))$$

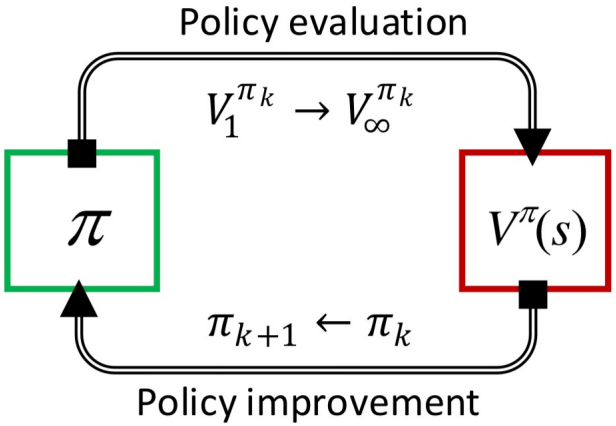
$$\text{s.t.} \quad \dot{x}(t) = f(x(t), u(t))$$

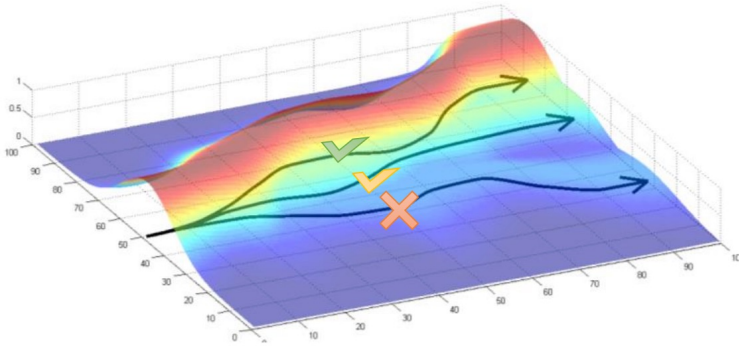
$$x(t) = x_0$$

Reinforcement learning

- Learn closed robot controller through **trial-and-error interaction with the environments**

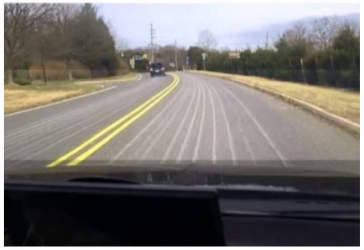




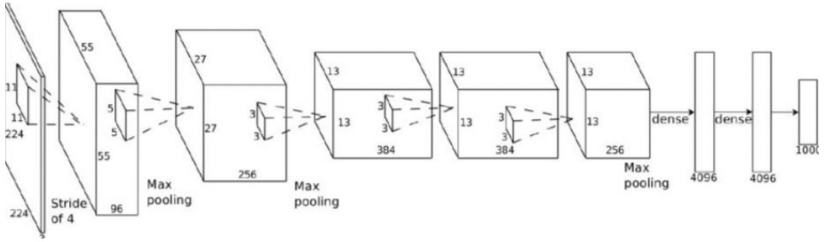


Imitation learning

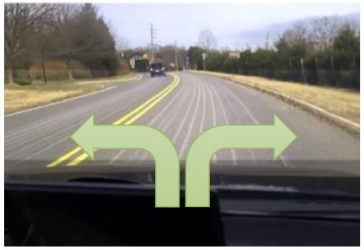
- Learning with guidance from expert demonstrations



$\mathbf{o}_t$

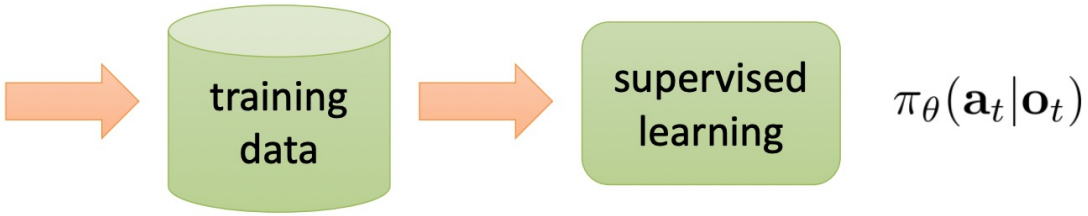


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$



$\mathbf{a}_t$

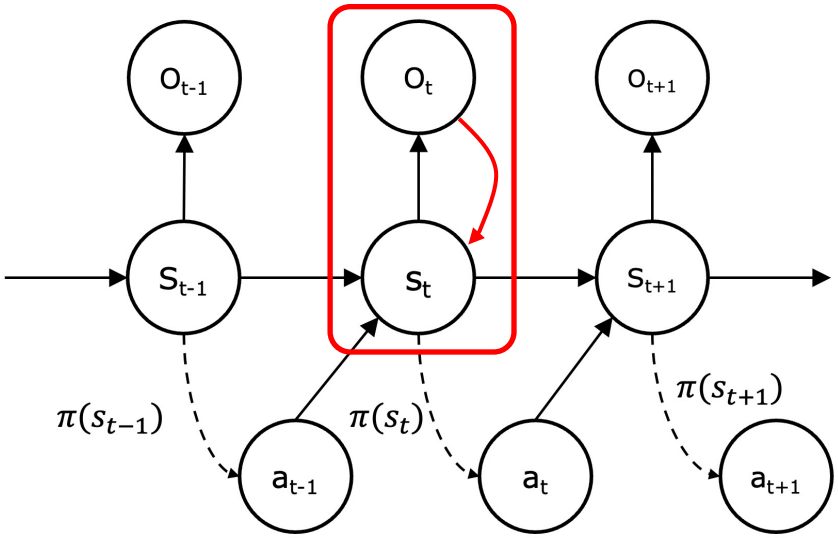
$\mathbf{o}_t$
$\mathbf{a}_t$

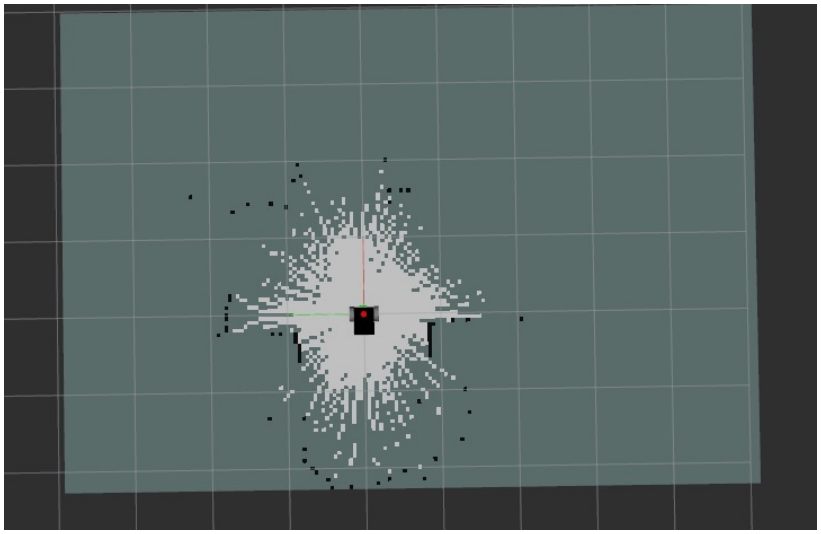


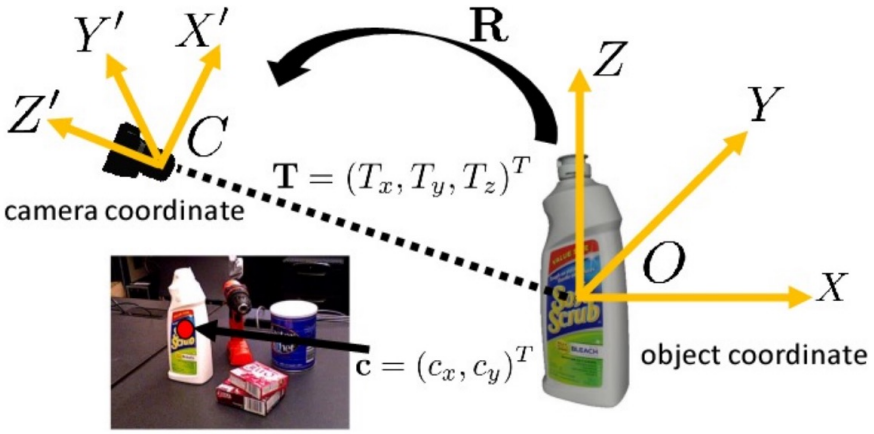
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

Perception

- Estimate proprioceptive
  - Base and joint states
- Detect external
  - Object pose and vel

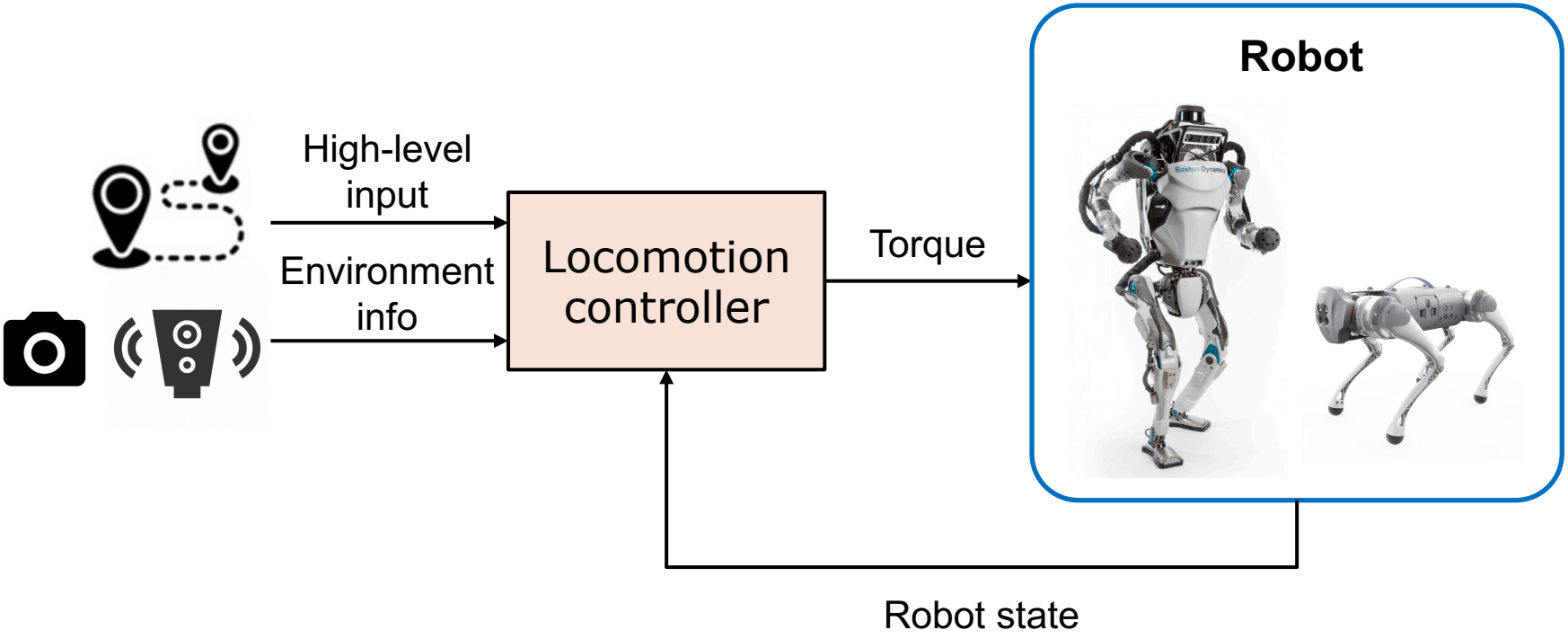






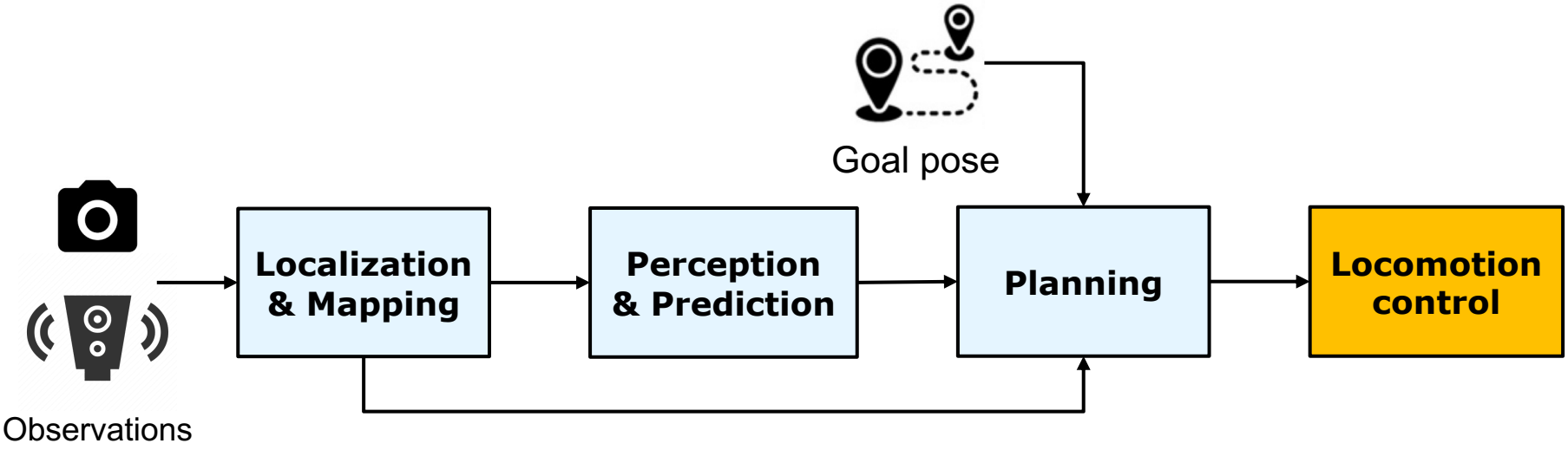
Robot locomotion

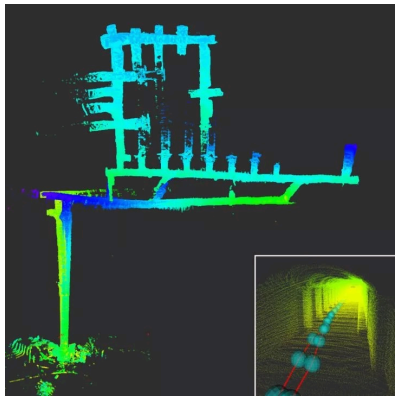
- Control the robot to reach target state in the future

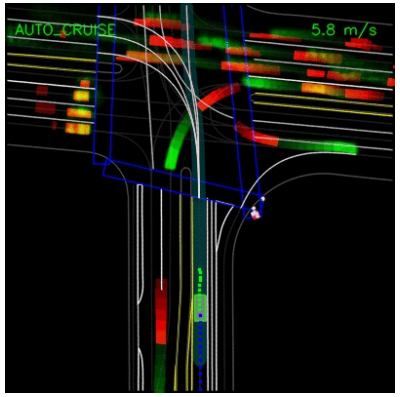


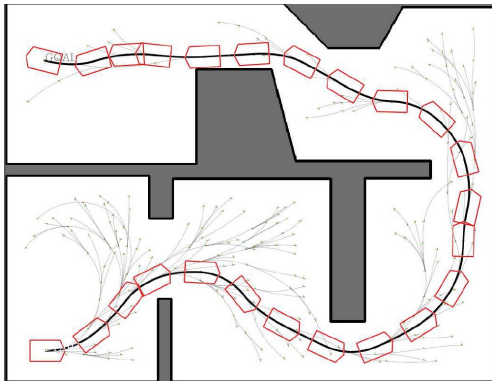
Robot navigation

- Navigate the robot to target destination



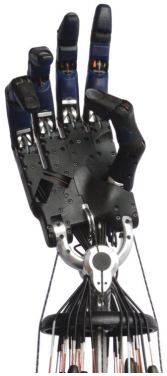






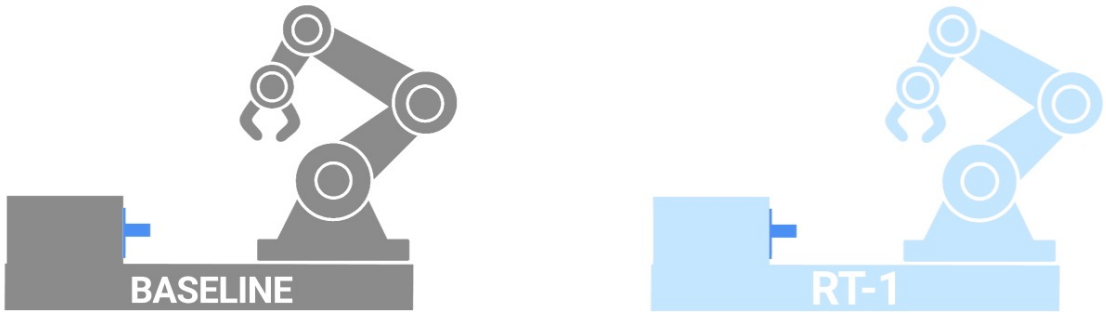
Robotic manipulation

- Manipulate environmental objects with end-effectors



Robot foundation model

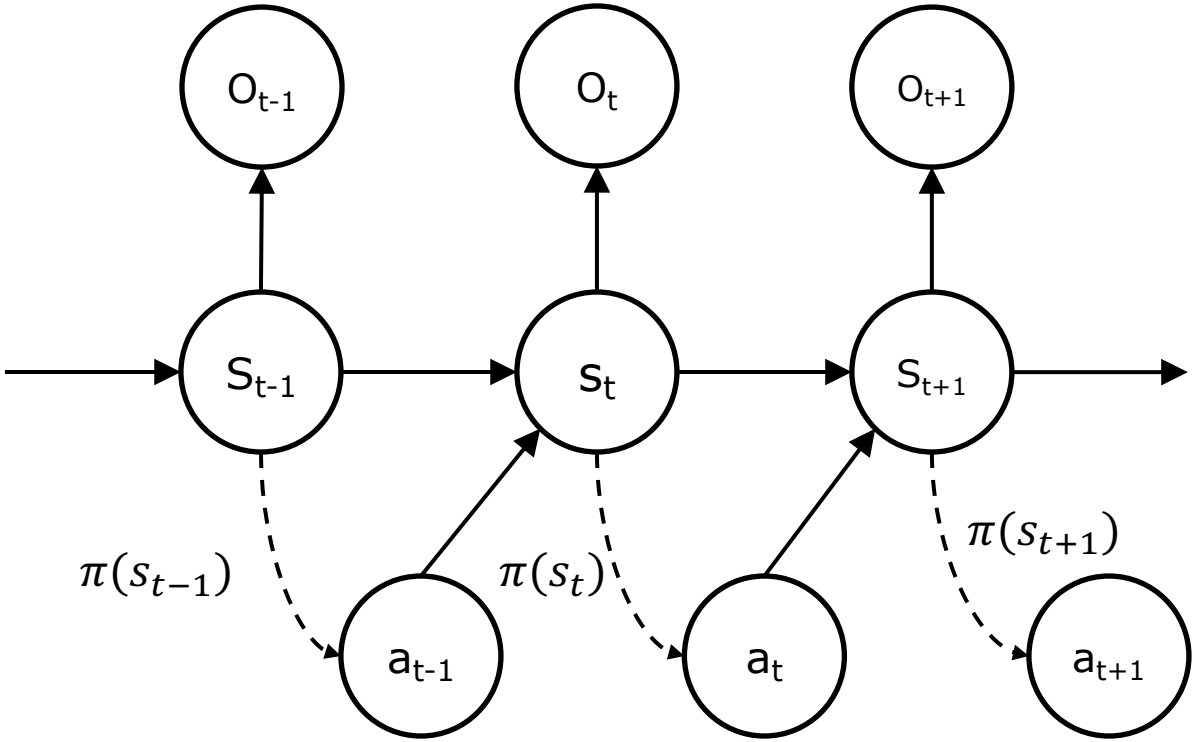
- Scaling up robot models for generalization



Contents

A General Problem formulation

- Let's interpret it more clearly mathematically



s: state
a: action
o: observation
π : policy

- A unified world model
- The bitter lesson (Richard Sutton): simple but effective!

SORA and world model

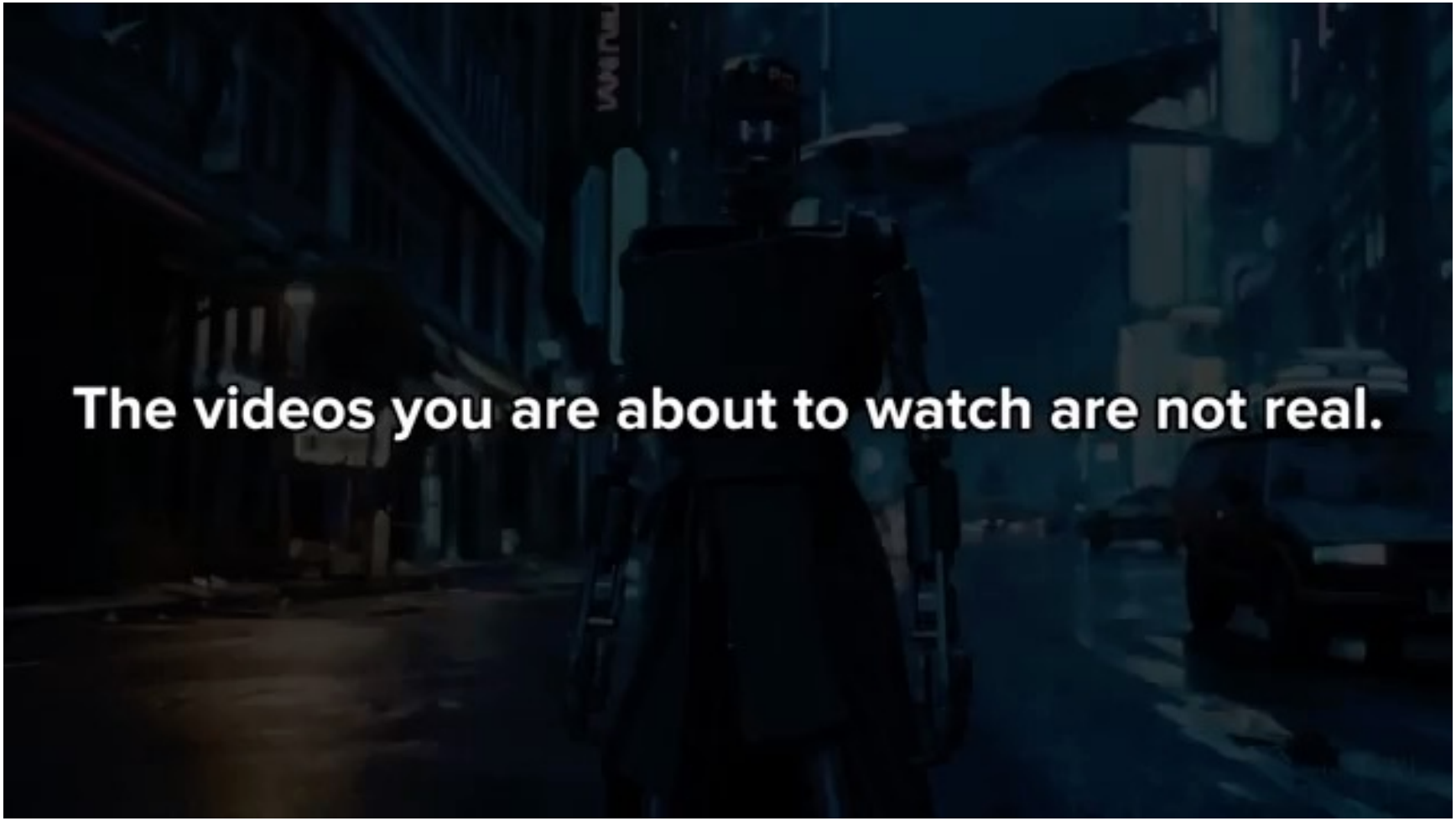
OpenAI Research Products Safety Company

February 15, 2024

Video generation models as world simulators

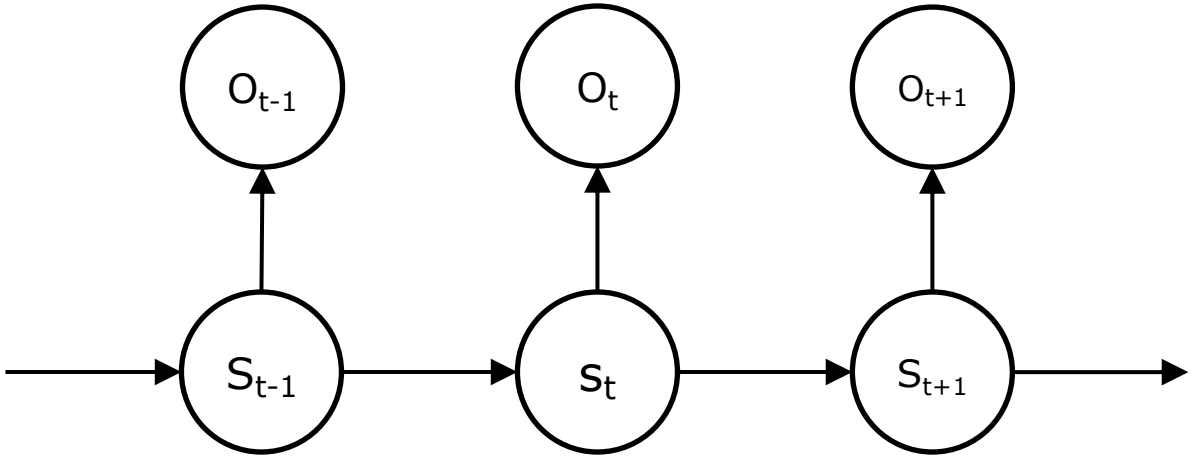
View Sora overview

SORA and world model



Sora and world model

- What is sora doing?



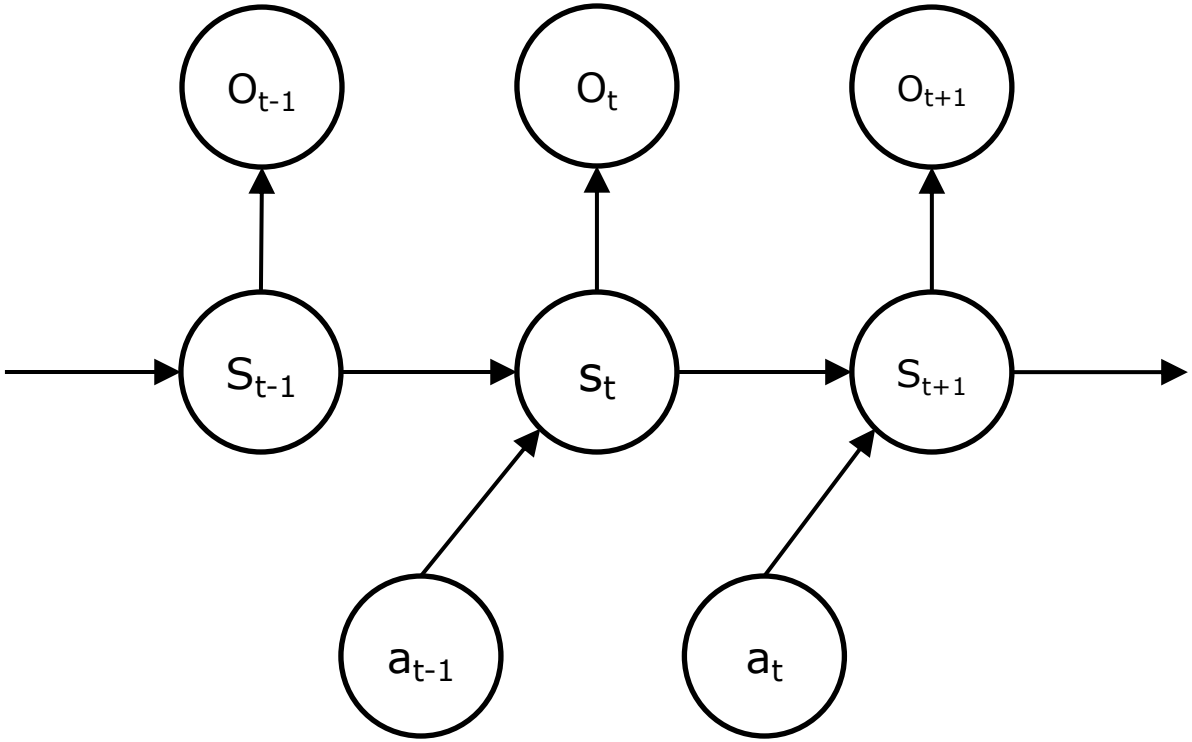
s: state
a: action
o: observation
π : policy

- What's missing?
  - Actions!

1X robotics and world model

1X robotics and world model

- What is 1X doing?

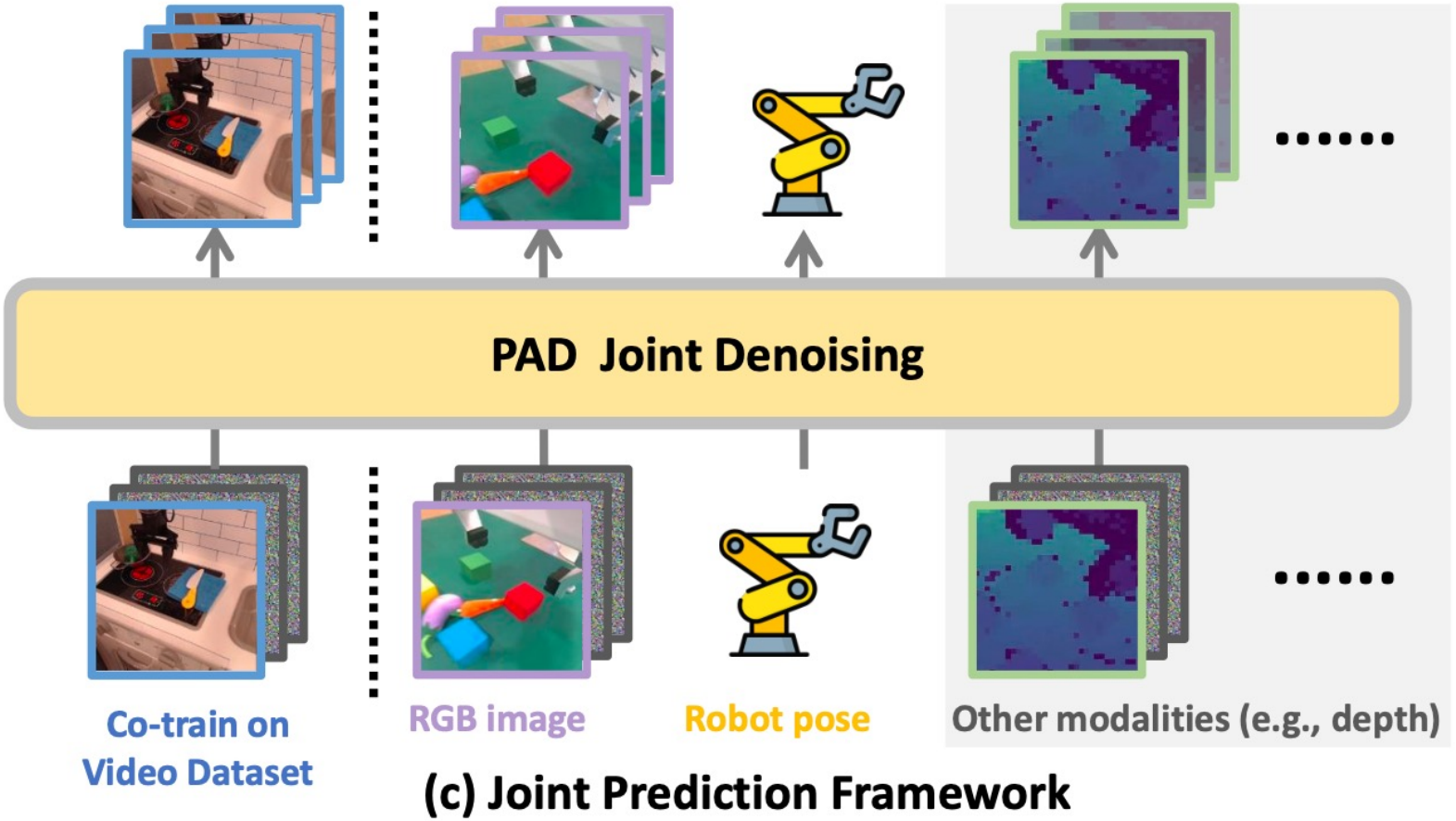


s: state
a: action
o: observation
π : policy

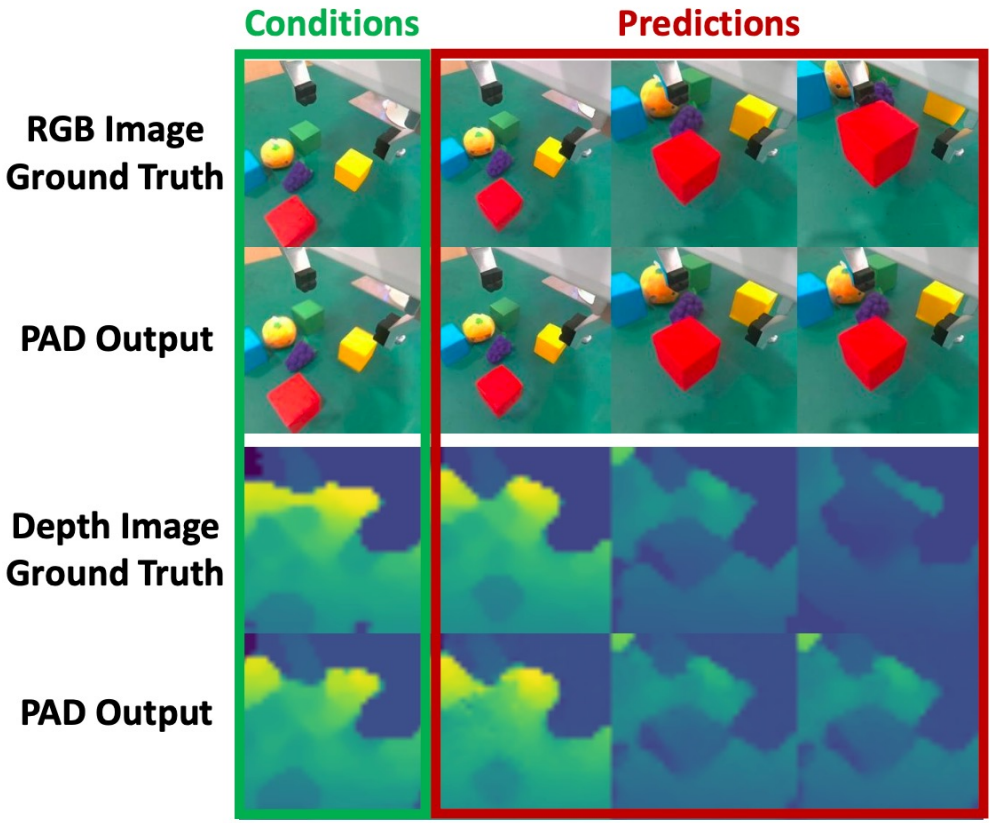
- What's missing?
  - Policy!

Prediction with actions

- Jointly model world and policies



Prediction with actions



Pure learning-based world models

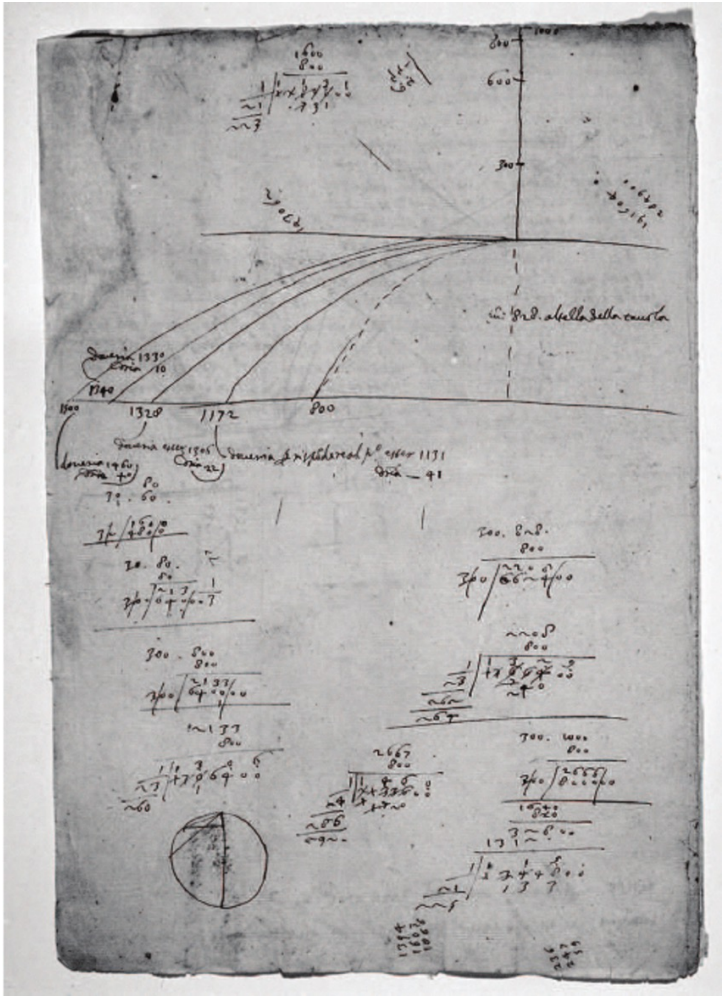
- Do these really work now?
- Failure cases for sora

Pure learning-based world models

- Do these really work now?
- Failure cases for 1X

Should everything be data-driven?

- How did Newton, Kepler, and Galileo model the world?
- Physics!
- Maybe not everything must be data-driven



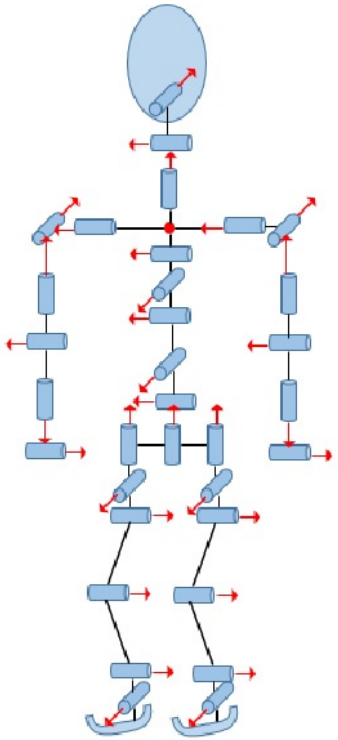
Galileo's notes

Can everything be physics-driven?

- Indeed, the world might be too complex to model directly with physics

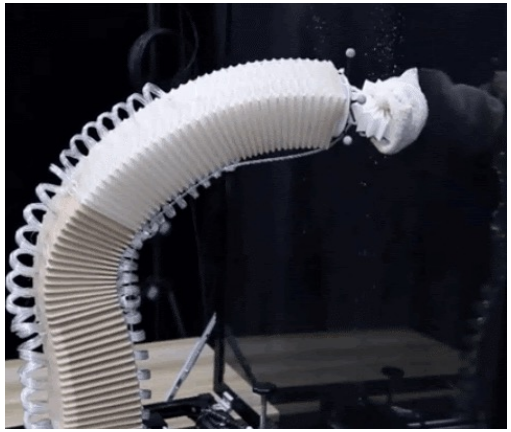
What can we model with physics?

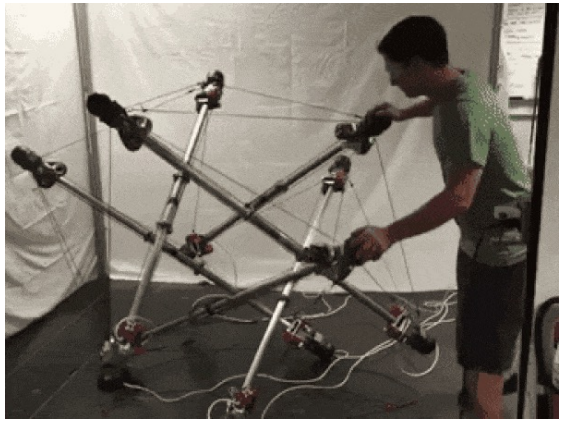
- However, the robot itself, is relatively easy to model precisely with physics
- We built the robot ourselves, so we know everything about it



What can we model with physics?

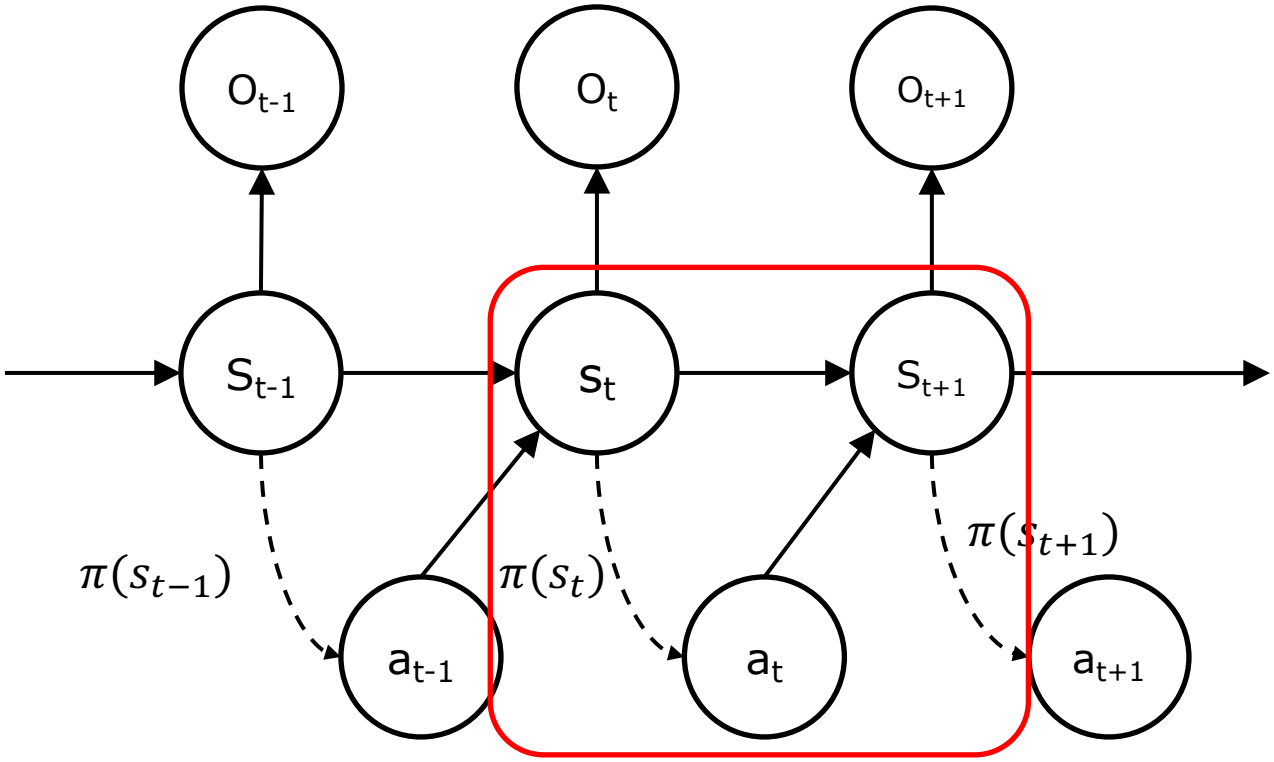
- We will concentrate on modeling rigid robots which are easier to model, instead of soft robots





What will we learn about robot modeling?

- Robot modeling consists part of the world model

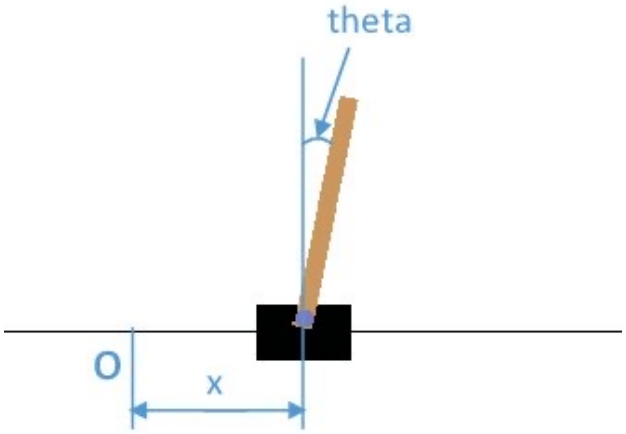


s: state
a: action
o: observation
π : policy

Robot model inside

State space

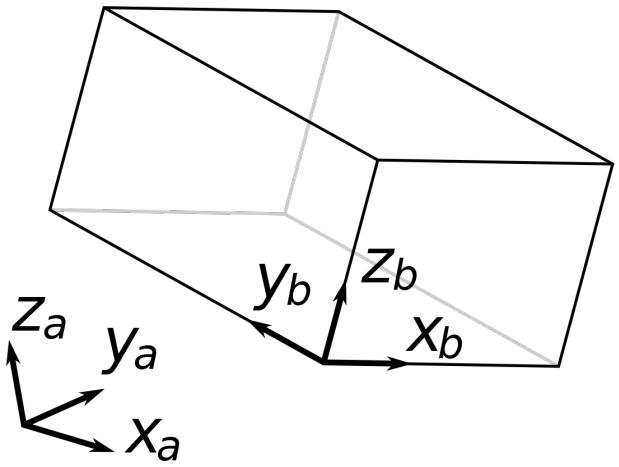
- **State variable:**
  - The smallest possible subset of system variables
  - that can represent the entire state of the system at any given time
  - such that the future evolvement of the system only depends on the current state, not the past (Markov).

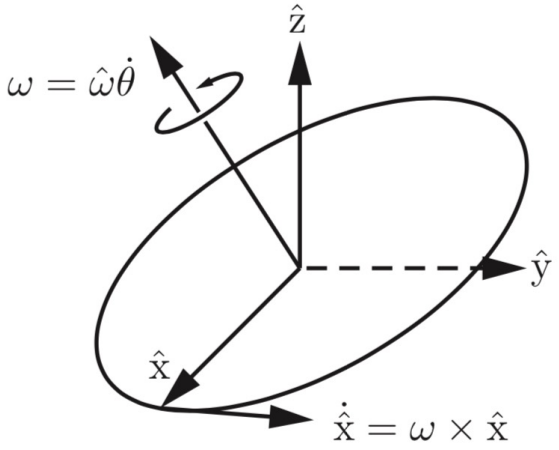


- What is the state variable of the cart-pole?
- How about a robot?

Rigid body motion

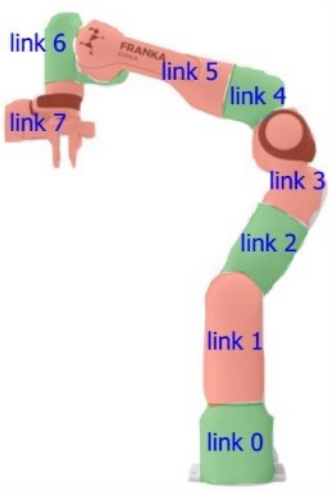
- Describe the motion of a single rigid body
  - Positions and orientations
  - Linear and angular velocities

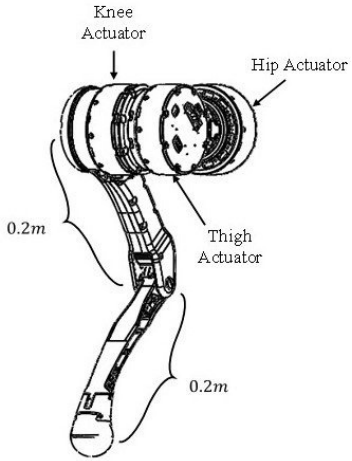




Robot kinematics

- Describe the motion of the robot
  - Positions and orientations
  - Linear and angular velocities
  - Scale to high-dof robots

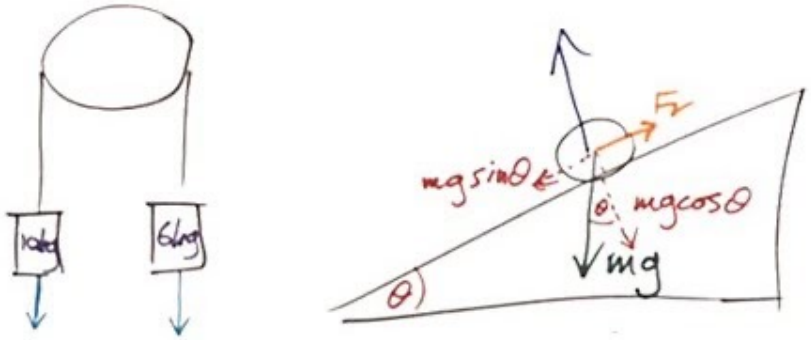




Robot dynamics

- Relationships between **force** and **accelerations** for robots

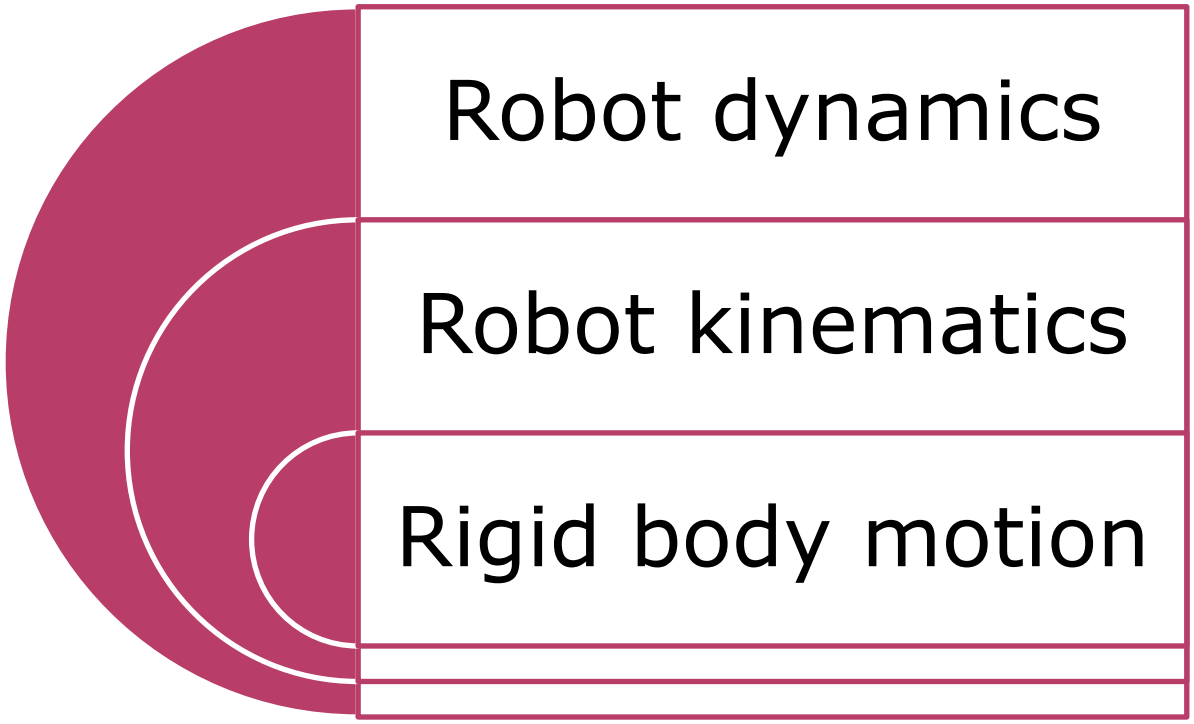
$$F = m\,a$$



$$\tau \Longleftrightarrow \theta,\ \dot{\theta},\ \ddot{\theta}$$

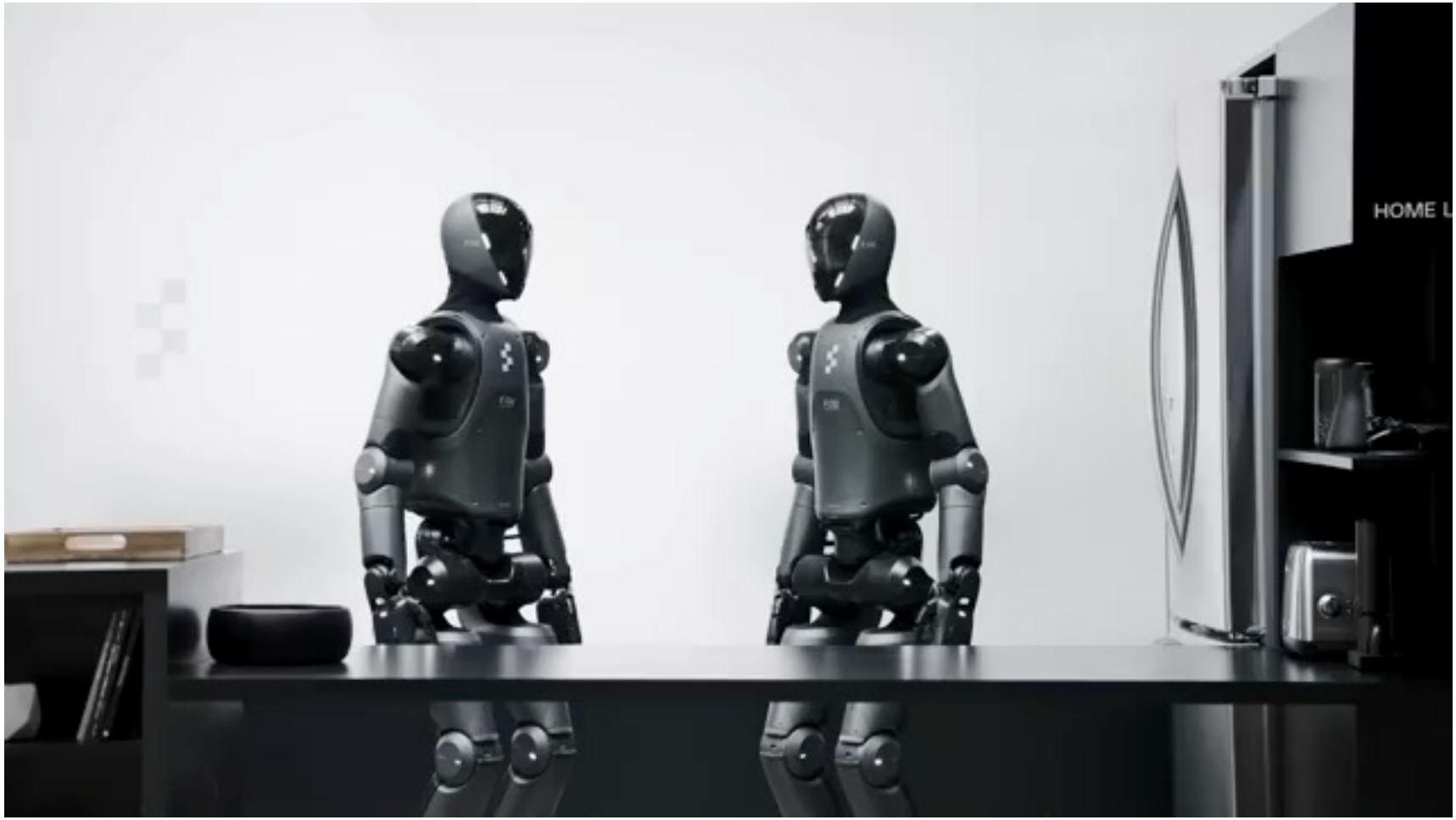
What will we learn about robot modeling?

- The relationships between rigid body motion, robot kinematics, and robot dynamics



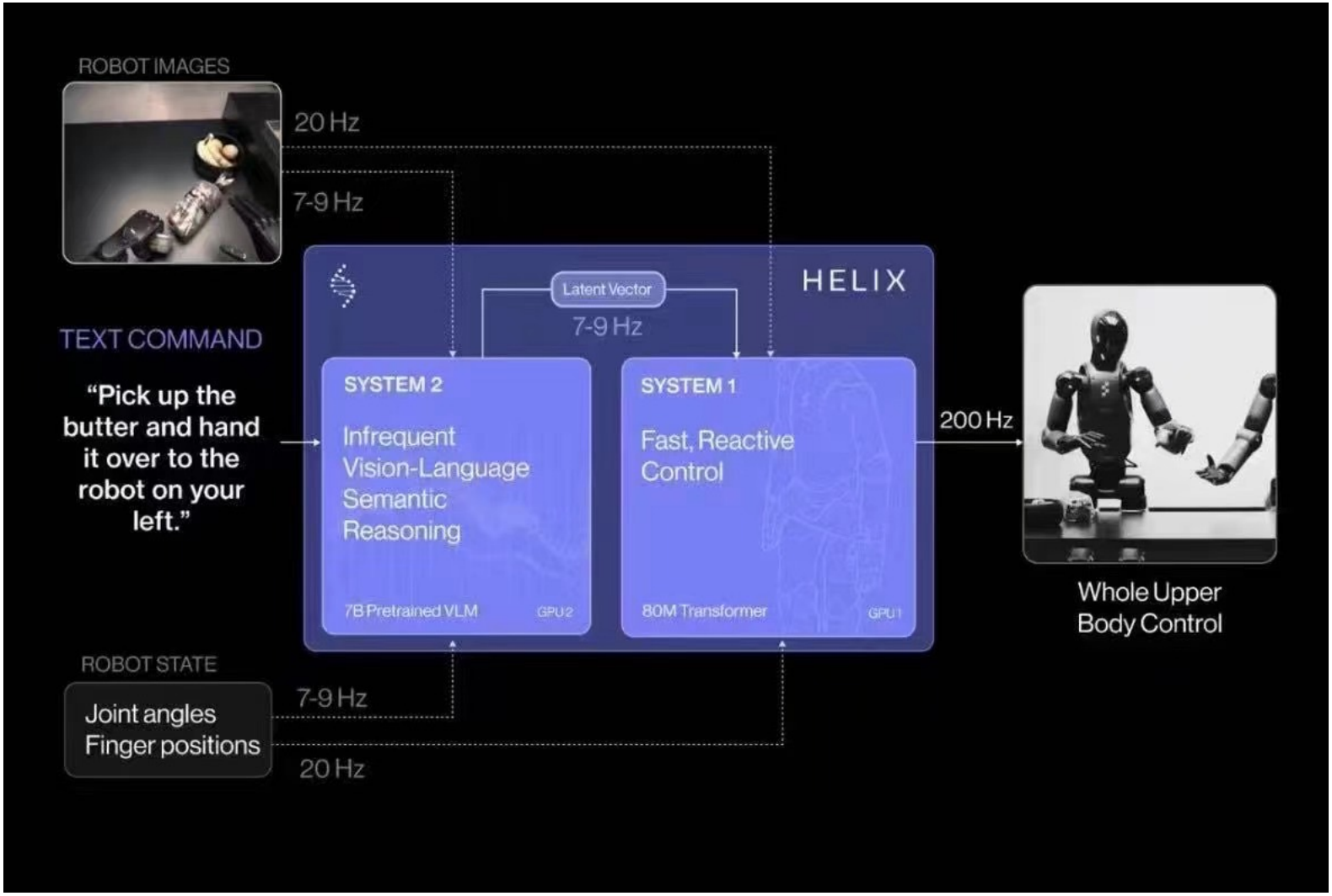
Why study robot modeling?

- The Figure AI demo



Why study robot modeling?

- Helix architecture



Why study robot modeling?

- HiRT architecture

HiRT: Enhancing Robotic Control with Hierarchical Robot Transformers

Jianke Zhang*[1], Yanjiang Guo*[1], Xiaoyu Chen[1],
Yen-Jen Wang[2], Yucheng Hu[1], Chengming Shi[1], Jianyu Chen†[1,3]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[2]University of California, Berkeley

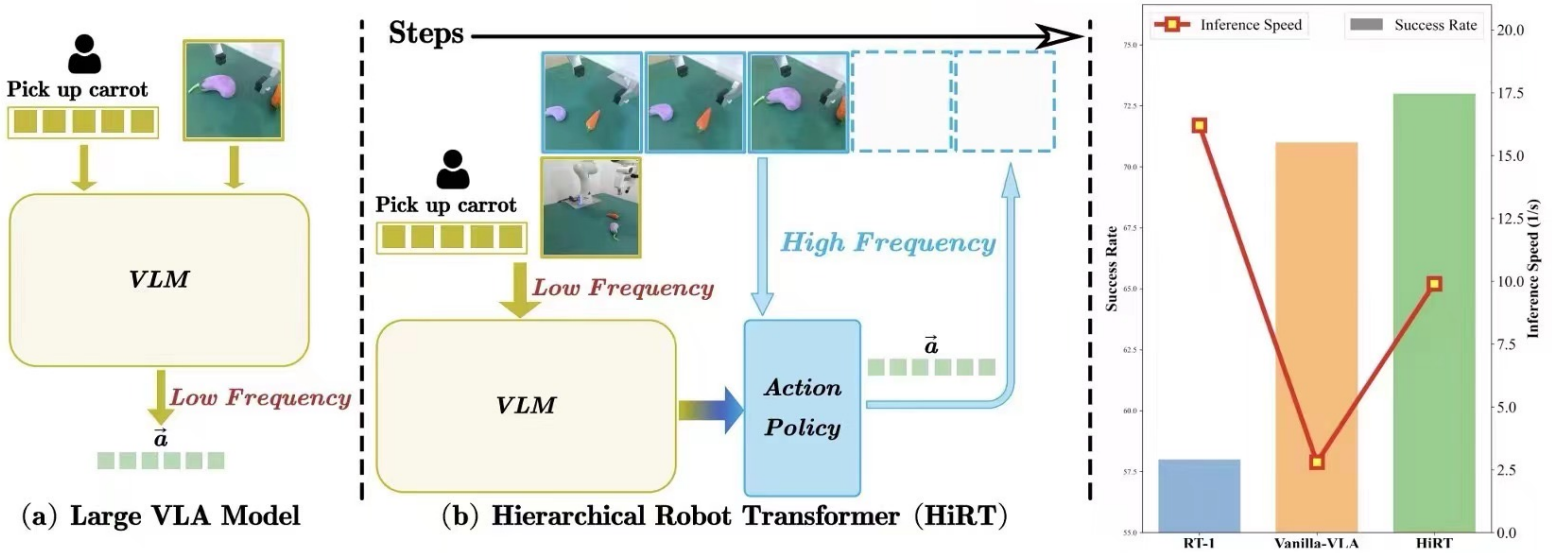
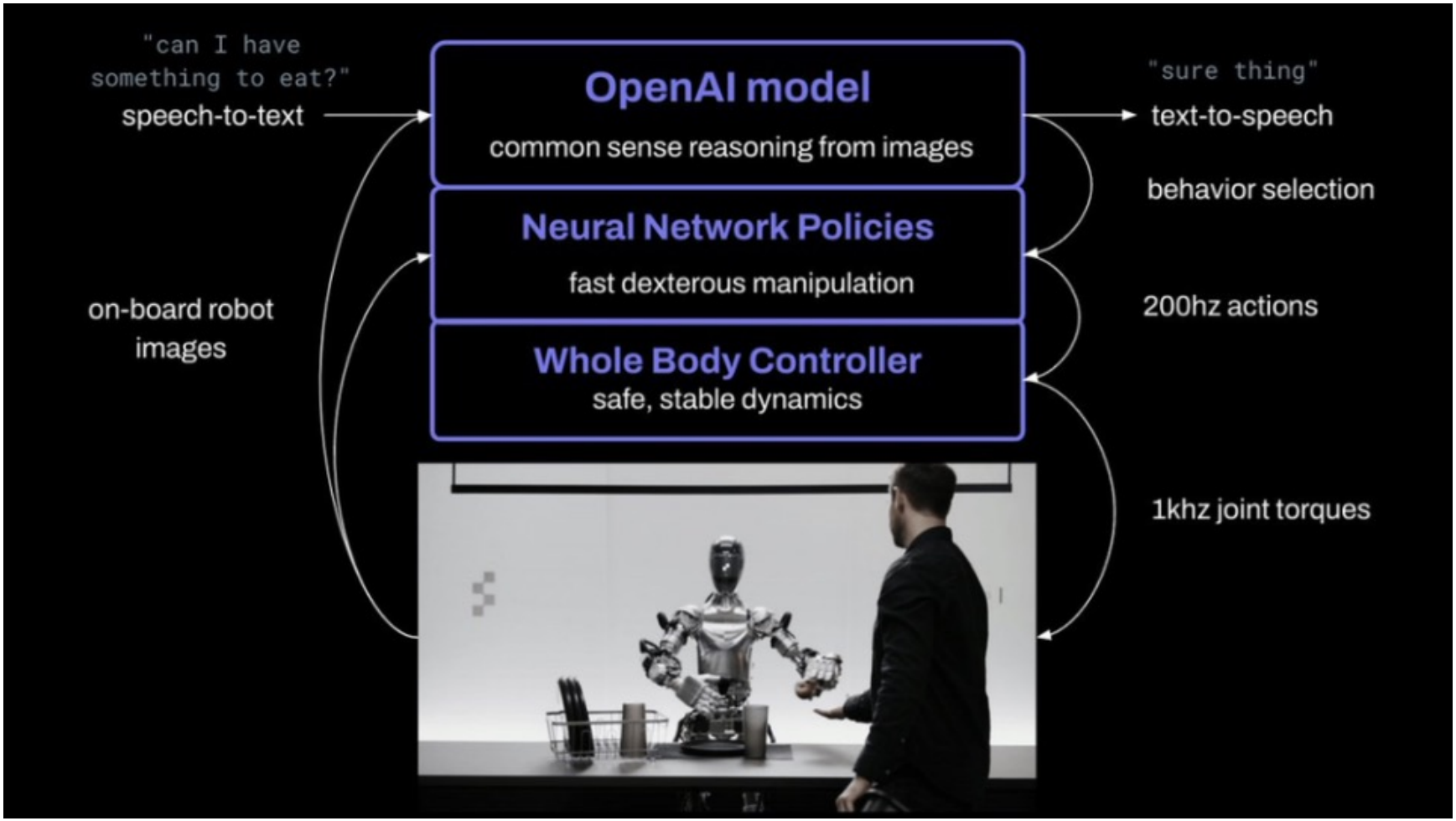


Figure 1: Illustration of our proposed HiRT high-level architecture. (a) Unlike large VLA models

Why study robot modeling?

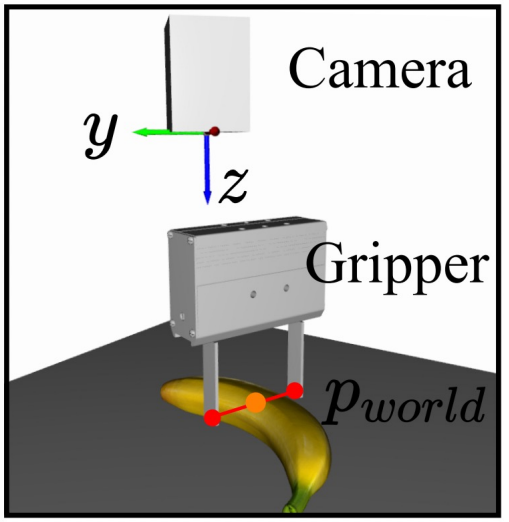
- Robot modeling is the basics for robot control



Why study robot modeling?

- Provide necessary supports for robotics functionalities

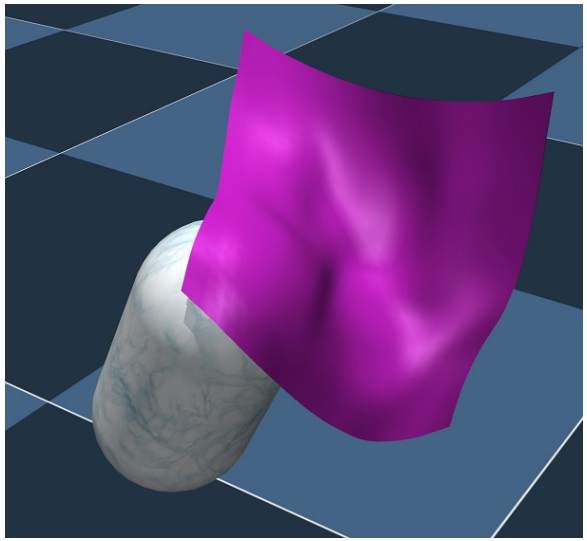
Tele-operation

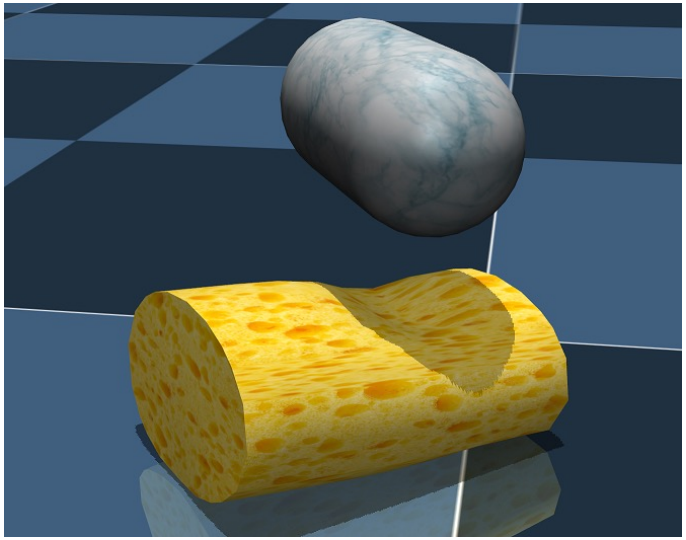


Robot grasping

Why study robot modeling?

- The techniques in robot modeling also consist the basics for robot simulations



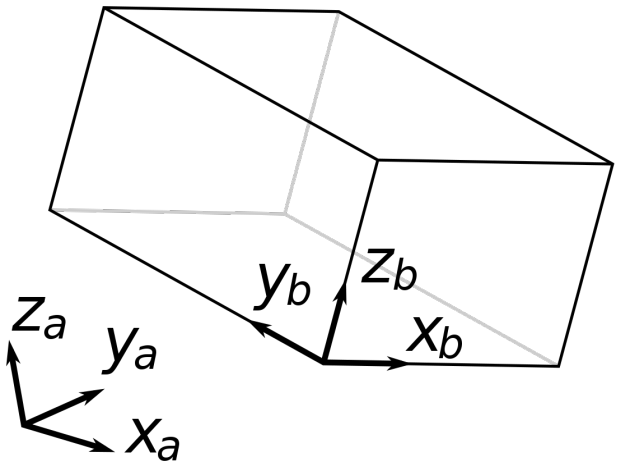


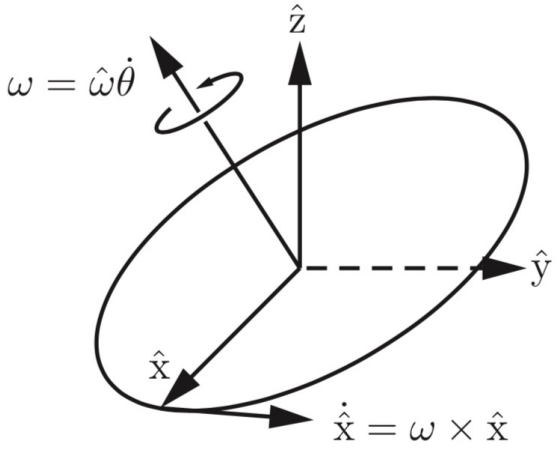
$$M\dot{v} + c = \tau + J^T f$$

Contents

Rigid body motion

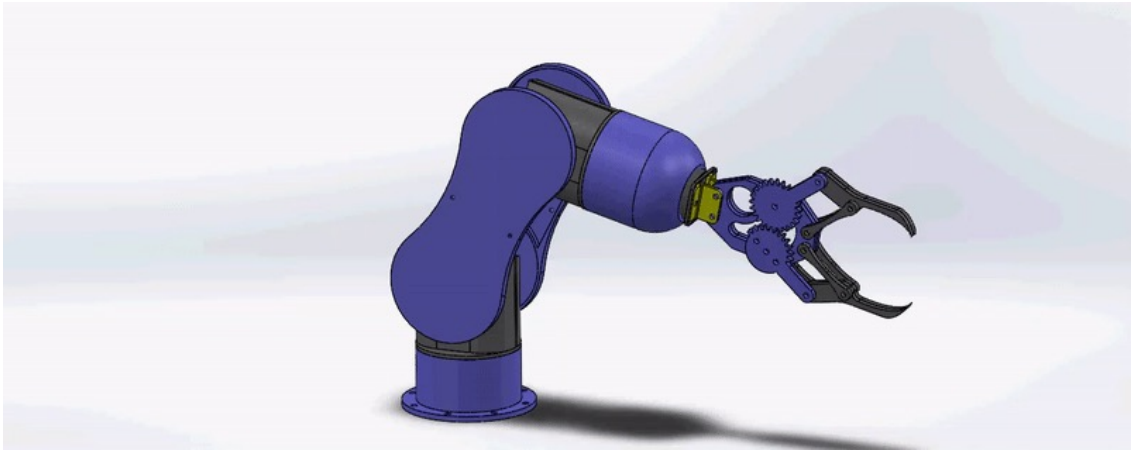
- Describe the motion of a single rigid body
  - Positions and orientations
  - Linear and angular velocities





Translation and rotation

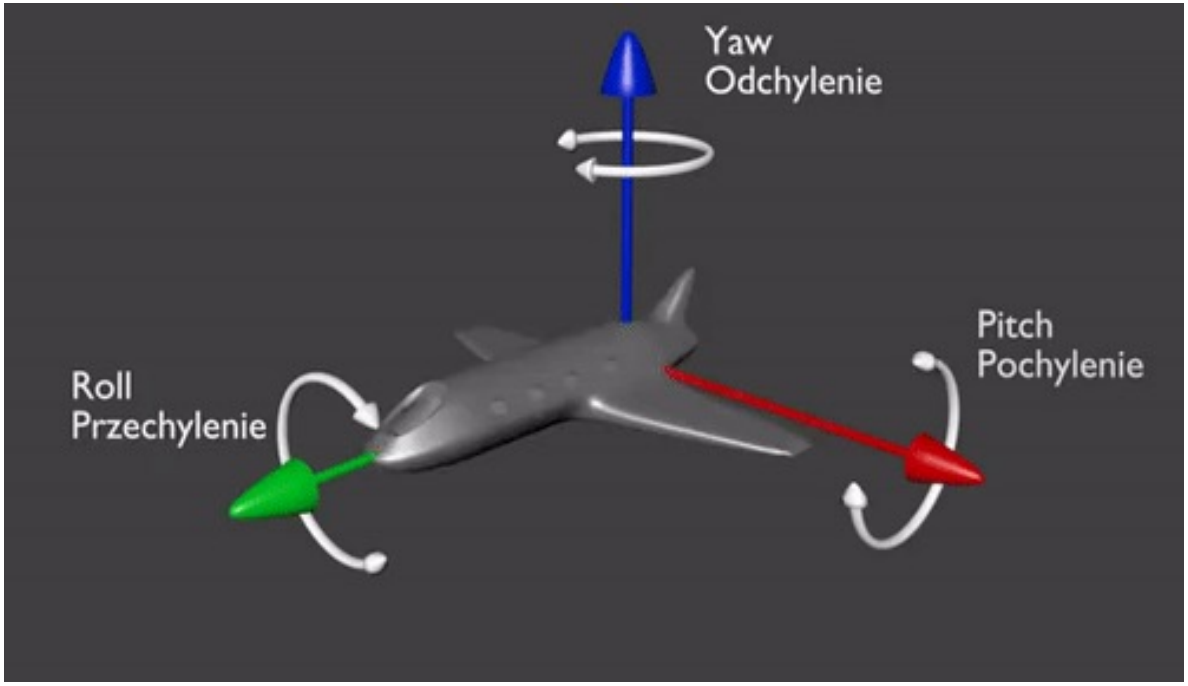
- The **pose** of a rigid body is composed of a translation with a rotation



- Translation is relatively easy, let's talk about rotation
- There are three common ways to represent rotation: **Euler angle**, **quaternion**, **rotation matrix**

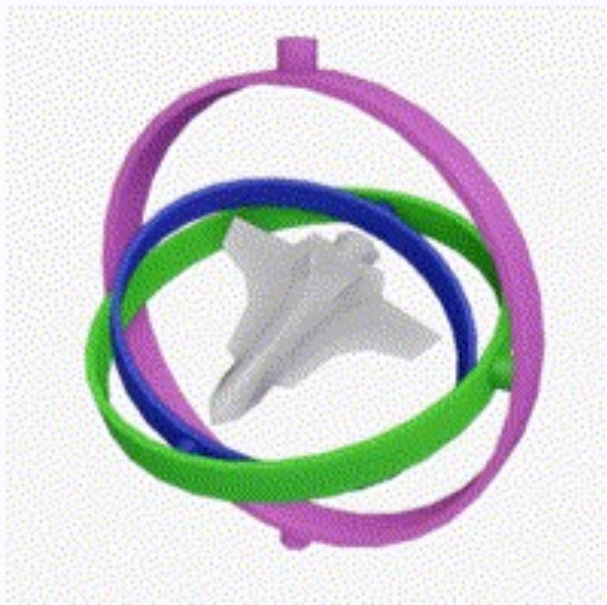
Euler Angle

- **Yaw-Pitch-Roll**
- The most intuitive representation of rotation



Gimbal lock

- However, Euler angle representation suffers from the **Gimbal lock**: Yaw and roll are at the same axis when Pitch is 90 degree, thus lose one dimension



- Euler angle are often used as an **intuitive interface** for human users, but not for algorithm

Quaternion

- **Quaternion**

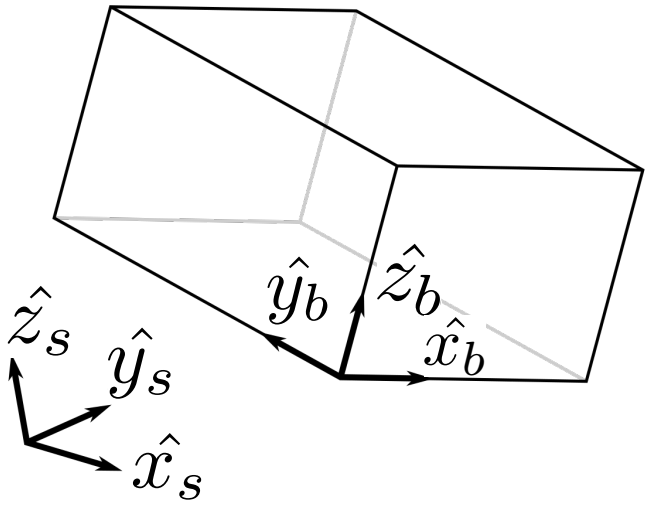
$$q = w + xi + yj + zk$$

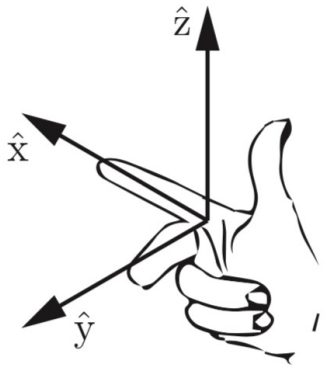
$$q = (w, x, y, z) \quad \text{or} \quad (x, y, z, w)$$

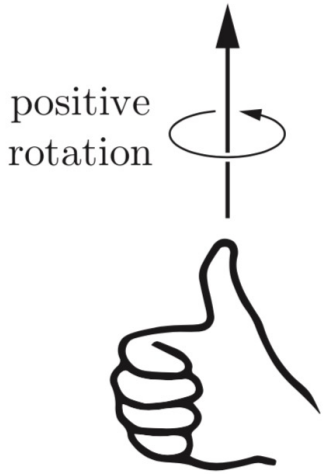
- Quaternion is usually used for **single rigid body**, e.g., flight control
- But for **high dof articulated robot** modeling, rotation matrix is more convenient
- Quaternion only need **4 variables** to represent rotation, cheaper than **rotation matrix (9 variables)**

Frames

- Before talking about rotation matrix, let's define **frames**
- **Space frame {s}:** A stationary reference frame {s}.
- **Body frame {b}:** a stationary frame that is coincident with the body-attached frame **at any instant**.
  - We can describe the pose of the rigid body by describing this frame







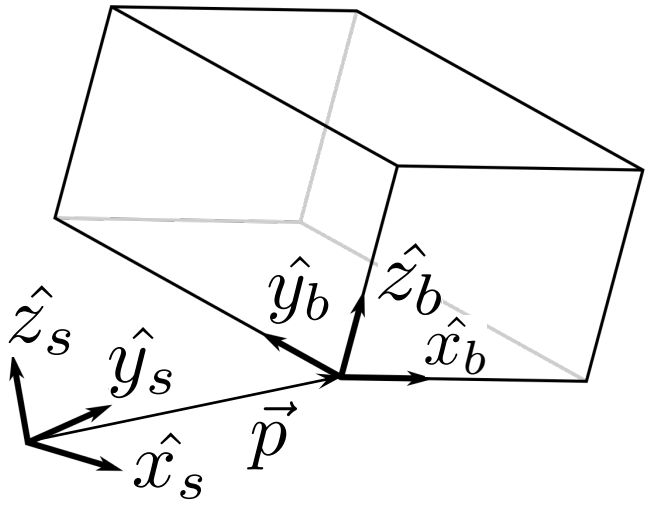
Represent body frame in space frame

- Represent the **position** of body frame's **origin**

$$\vec{p} = p_1 \hat{\mathrm{x}}_\mathrm{s} + p_2 \hat{\mathrm{y}}_\mathrm{s} + p_3 \hat{\mathrm{z}}_\mathrm{s}$$

- Represent body frame's **unit axes** (orientation)

$$\begin{aligned}
\hat{\mathrm{x}}_\mathrm{b} &= r_{11}\hat{\mathrm{x}}_\mathrm{s} + r_{21}\hat{\mathrm{y}}_\mathrm{s} + r_{31}\hat{\mathrm{z}}_\mathrm{s}, \\
\hat{\mathrm{y}}_\mathrm{b} &= r_{12}\hat{\mathrm{x}}_\mathrm{s} + r_{22}\hat{\mathrm{y}}_\mathrm{s} + r_{32}\hat{\mathrm{z}}_\mathrm{s}, \\
\hat{\mathrm{z}}_\mathrm{b} &= r_{13}\hat{\mathrm{x}}_\mathrm{s} + r_{23}\hat{\mathrm{y}}_\mathrm{s} + r_{33}\hat{\mathrm{z}}_\mathrm{s}.
\end{aligned}$$



Rotation matrices

- Write as coordinates

$$\vec{p} = [\hat{x}_s \quad \hat{y}_s \quad \hat{z}_s] \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}}_{p}$$

$$[\hat{x}_b \quad \hat{y}_b \quad \hat{z}_b] = [\hat{x}_s \quad \hat{y}_s \quad \hat{z}_s] \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}}_{R}$$

- (R, p) pair can be used to describe the frame
- R is called the **rotation matrix**
  - 3 dof of orientation, 9 variables in R, 6 constraints

Rotation matrices

- Unit norm condition of axes $\{\hat{\mathrm{x}}_\mathrm{b}, \hat{\mathrm{y}}_\mathrm{b}, \hat{\mathrm{z}}_\mathrm{b}\}$ constraints

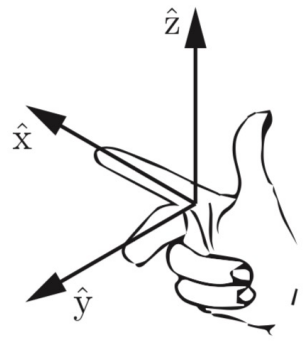
$$\begin{aligned} r_{11}^2 + r_{21}^2 + r_{31}^2 &= 1, \\ r_{12}^2 + r_{22}^2 + r_{32}^2 &= 1, \\ r_{13}^2 + r_{23}^2 + r_{33}^2 &= 1. \end{aligned}$$

- Orthogonal constraints $\hat{\mathrm{x}}_\mathrm{b} \cdot \hat{\mathrm{y}}_\mathrm{b} = \hat{\mathrm{x}}_\mathrm{b} \cdot \hat{\mathrm{z}}_\mathrm{b} = \hat{\mathrm{y}}_\mathrm{b} \cdot \hat{\mathrm{z}}_\mathrm{b} = 0$

$$\begin{aligned} r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32} &= 0, \\ r_{12}r_{13} + r_{22}r_{23} + r_{32}r_{33} &= 0, \\ r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33} &= 0. \end{aligned}$$

- More compactly, $R^\mathrm{T} R = I$

Special orthogonal group

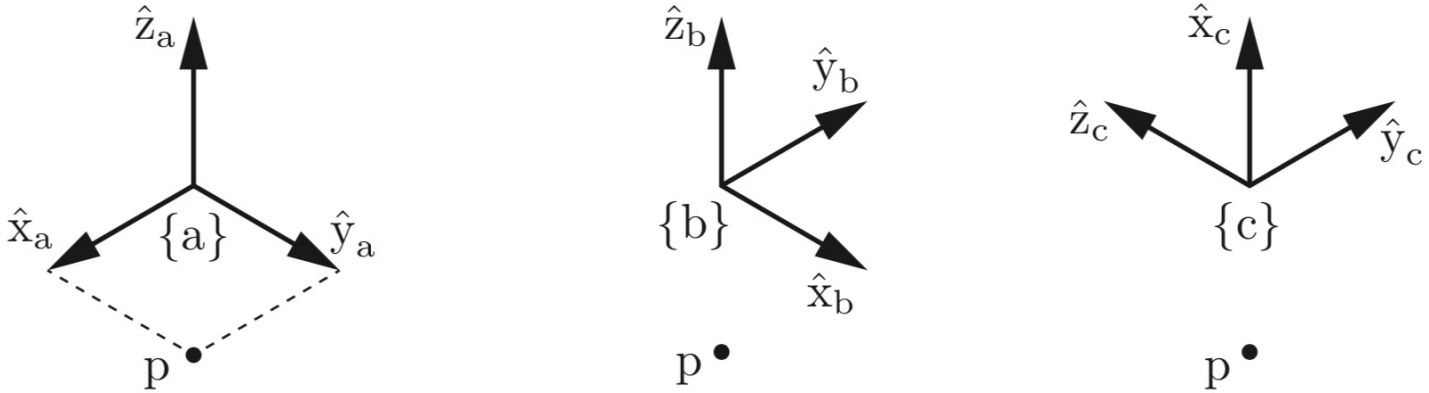


- Note the right-handed condition
- Rotation matrices form **Special orthogonal group SO(3)**: the set of all 3x3 matrices R that satisfy
  - $R^{\mathrm{T}} R = I$
  - $\det R = 1$
- Similarly, we have SO(2) and SO(n).
- **SO(2)**:

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Example

- Three frames, {a} coincides with {s}.
  - What are the rotation matrices?



$$R_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_b = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_c = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

Homogeneous Transformation matrices

- We have seen the pair (R, p) can be used to describe a frame
- Now define **homogeneous transformation matrices** or **special Euclidean group SE(3)**: the set of 4x4 matrices

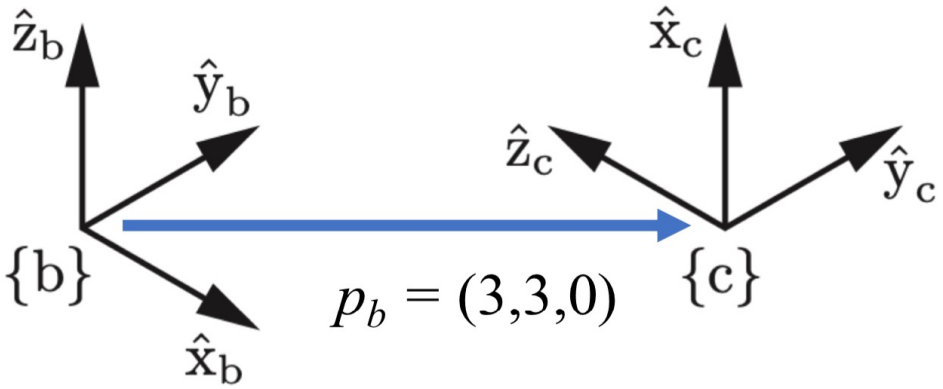
$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with $R \in SO(3)$ and $p \in \mathbb{R}^3$.

- Packaging (R, p) into SE(3) helps a lot in operations
- Similarly, we have SE(2) with $R \in SO(2), p \in \mathbb{R}^2$

Example

- Write T_{bc}



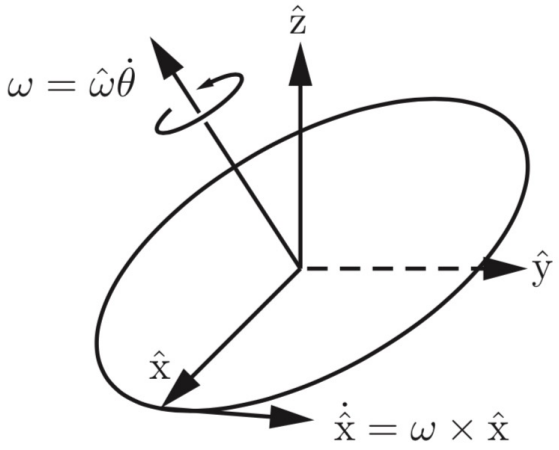
$$T_{bc} = \begin{bmatrix} 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation matrix and axis-angle

- It is not intuitive to obtain the rotation matrix
- If a frame starts from coincident with {s}, and rotates around the rotation axis $\hat{\omega}$ for an angle θ, then it must get to some orientation R.
- **Angular velocity**: rotating with axis $\hat{\omega}$ with speed $\dot{\theta}$
- connection between the rotation matrix representation and the **axis-angle representation**:

$$\hat{\omega}\ \theta \Longleftrightarrow R$$

$$\hat{\omega}\ \dot{\theta} \Longleftrightarrow \dot{R}$$



Rodrigues' formula

- We have the **Rodrigues' formula**:

$$R(\hat{\omega}, \theta) = e^{[\hat{\omega}]\theta} = I + \sin\theta\,[\hat{\omega}] + (1 - \cos\theta)[\hat{\omega}]^2 \in SO(3)$$

- $\hat{\omega}\theta \in \mathbb{R}^3$ is called the **exponential coordinates** of $R \in SO(3)$
- $e^{[\hat{\omega}]\theta}$ is called the **matrix exponential** of $[\hat{\omega}\theta] \in so(3)$
- where we define skew-symmetric matrix representation of a 3D vector $x = [x_1\ x_2\ x_3]^{\mathrm{T}} \in \mathbb{R}^3$:

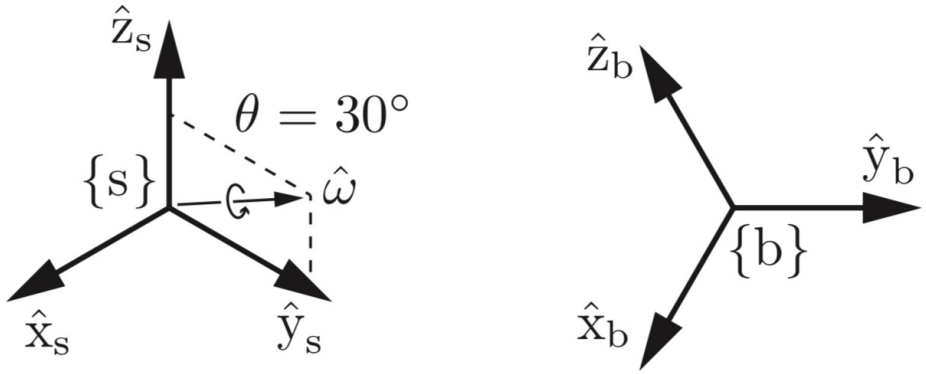
$$[x] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

$$[x] = -[x]^{\mathrm{T}}$$

Rotation operation with exponential coordinates

- Rotation matrix can be used as a **rotation operation**

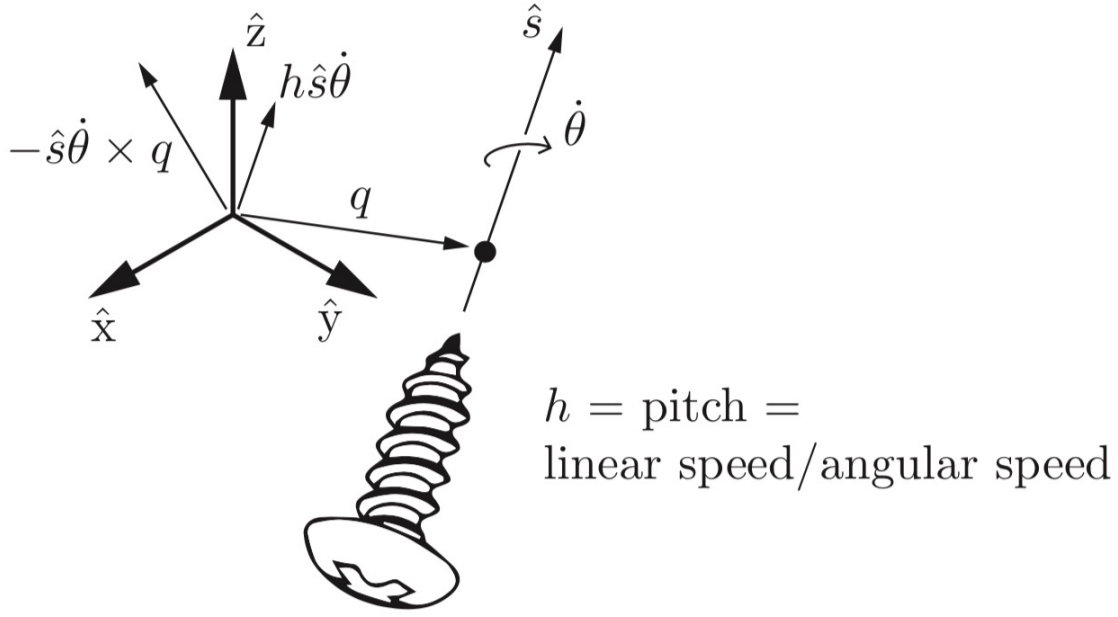
$$R_{sb'} = R(\hat{\omega}_s, \theta) R_{sb}$$



$$\begin{aligned}
R &= e^{[\hat{\omega}_1]\theta_1} \\
&= I + \sin\theta_1 [\hat{\omega}_1] + (1 - \cos\theta_1)[\hat{\omega}_1]^2 \\
&= I + 0.5 \begin{bmatrix} 0 & -0.5 & 0.866 \\ 0.5 & 0 & 0 \\ -0.866 & 0 & 0 \end{bmatrix} + 0.134 \begin{bmatrix} 0 & -0.5 & 0.866 \\ 0.5 & 0 & 0 \\ -0.866 & 0 & 0 \end{bmatrix}^2 \\
&= \begin{bmatrix} 0.866 & -0.250 & 0.433 \\ 0.250 & 0.967 & 0.058 \\ -0.433 & 0.058 & 0.899 \end{bmatrix}.
\end{aligned}$$

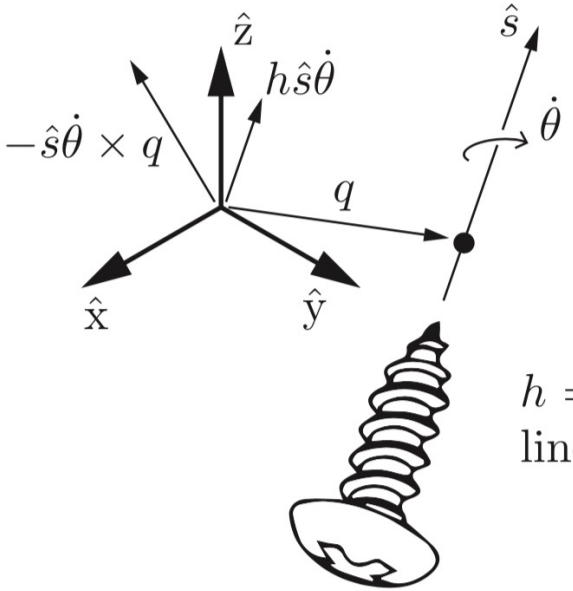
Rigid body motion

- From the **Chasles' theorem**, we have: spatial rigid body motion can be viewed as **rotating about an axis while also translating along the axis**, just like a screw.
- This screw motion is defined by a **screw axis** $\mathcal{S}$, which is the collection of $\{q, \hat{s}, h\}$.



Spatial velocity

- The screw axis extends a rotation axis, but with linear translation and the axis might not cross the origin
- Combined with the rotation velocity $\dot{\theta}$, we can obtain the **spatial velocity (twist):**
  - w: angular velocity of the axis $\hat{s}$
  - v: linear velocity of the reference frame's origin (imaging expand the rigid body to cover the origin!)

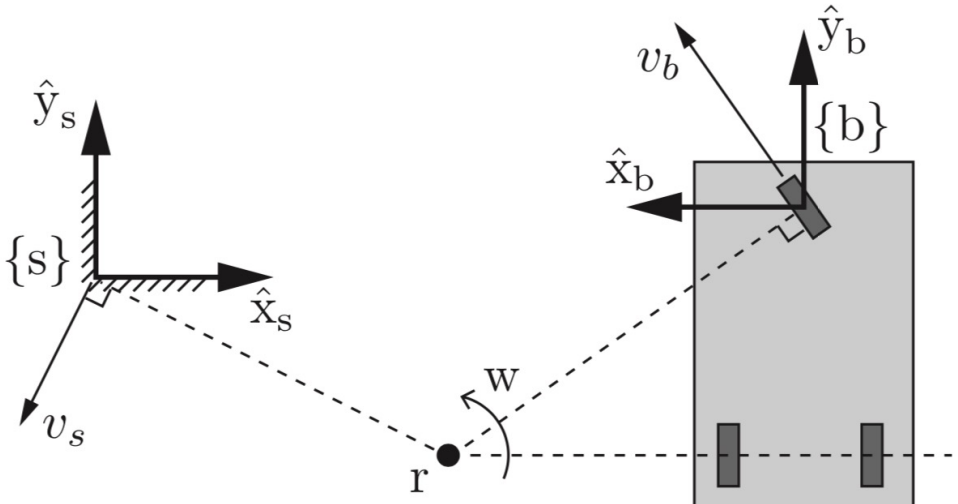


$$\mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix} = \begin{bmatrix} \hat{s}\dot{\theta} \\ -\hat{s}\dot{\theta} \times q + h\hat{s}\dot{\theta} \end{bmatrix}$$

h = pitch = linear speed/angular speed

Example

- Top view of a car in 3D space, $\hat{z}_b$ is into the page and $\hat{z}_s$ is out of the page. At current time the car's motion can be viewed as rotating around r with w = 2 rad/s. We have $r_s = (2, -1, 0)$ and $r_b = (2, -1.4, 0)$, what is $\mathcal{V}_s$?



$$\omega_s = (0, 0, 2)$$

$$v_s = \omega_s \times (-r_s) = r_s \times \omega_s = (-2, -4, 0),$$

$$\mathcal{V}_s = \begin{bmatrix} \omega_s \\ v_s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ -2 \\ -4 \\ 0 \end{bmatrix}$$

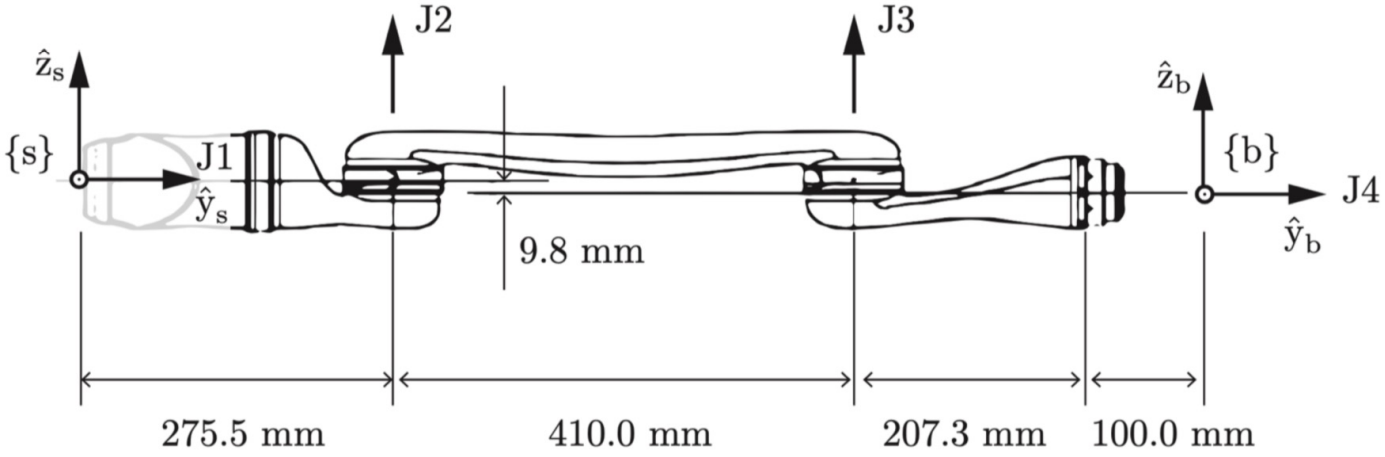
Screw axis

- Unlike rotation axis, the screw axis might have some ill conditions
  - If no rotation (e.g, angular speed=0), then the pitch h is infinite.
- We can use a better representation of the **screw axis** by **normalizing the twist**:
  - $\|\omega\| = 1$ and $v = -\omega \times q + h\omega$
  - $\omega = 0$ and $\|v\| = 1$

$$\mathcal{S} = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6$$

Example

- Kinova lightweight 4-dof arm
  - For J4, what is the screw axis in {s} ?
  - How about J2 ?



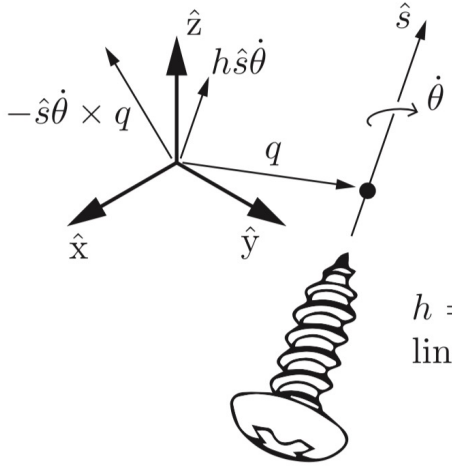
$$S_s = [0,\ 1,\ 0,\ 9.8mm/s,\ 0,\ 0]^T \quad S_s = [0,\ 0,\ 1,\ 275.5,\ 0,\ 0]^T$$

Transformation matrix and screw-angle

- Similarly to rotation matrix and axis-angle representation
- If a frame starts from coincident with {s}, and rotates around the screw axis $\hat{s}$ for an angle θ then it must get to some orientation R.
- **Spatial velocity**: rotating with axis $\hat{s}$ with speed $\dot{\theta}$
- connection between the transformation matrix and the **screw-angle representation**:

$$\hat{s}\ \theta \Longleftrightarrow T$$

$$\hat{s}\ \dot{\theta} \Longleftrightarrow \dot{T}$$



Transformation matrix and screw-angle

- Exponential coordinates

$$T(\mathcal{S}, \theta) = e^{[\mathcal{S}]\theta}$$

  - **Exponential coordinates** of $T \in SE(3)$: $\mathcal{S}\theta \in \mathbb{R}^6$
  - **Matrix exponential** of $[\mathcal{S}]\theta \in se(3)$:

$$e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} & G(\theta)v \\ 0 & 1 \end{bmatrix} \qquad [\mathcal{S}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}$$

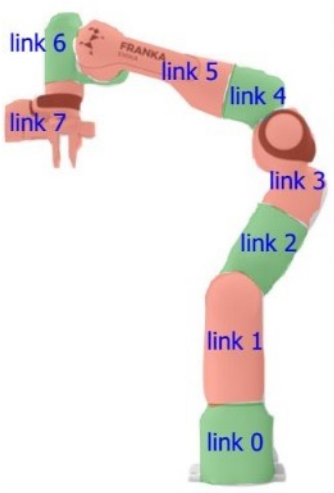
- **Displacement** operation:

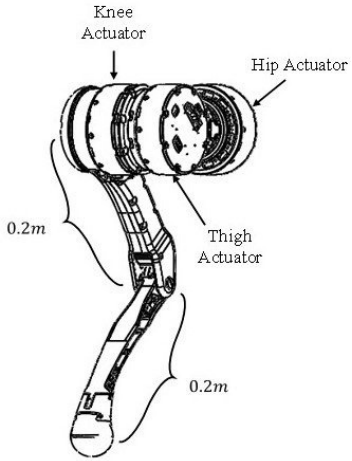
$$T_{sb'} = T(\mathcal{S}, \theta) T_{sb}$$

Contents

Robot kinematics

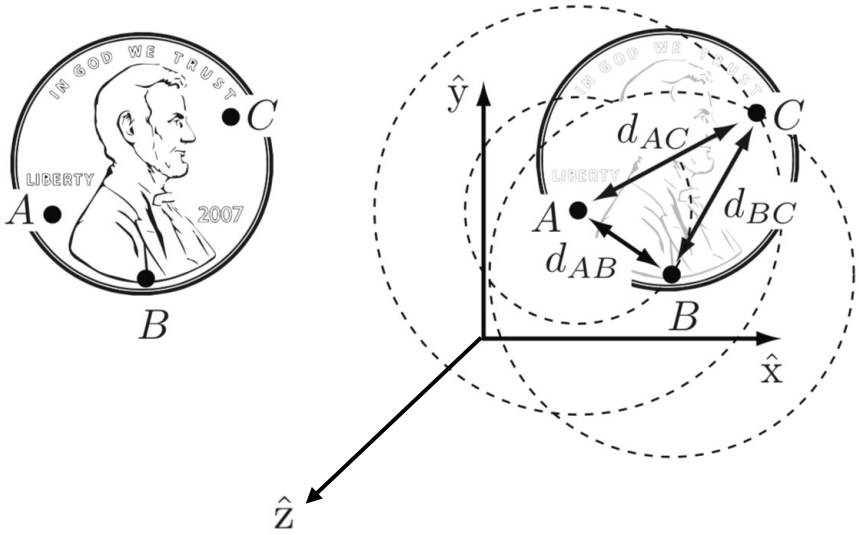
- Describe the motion of the robot
  - Positions and orientations
  - Linear and angular velocities
  - Scale to high-dof robots





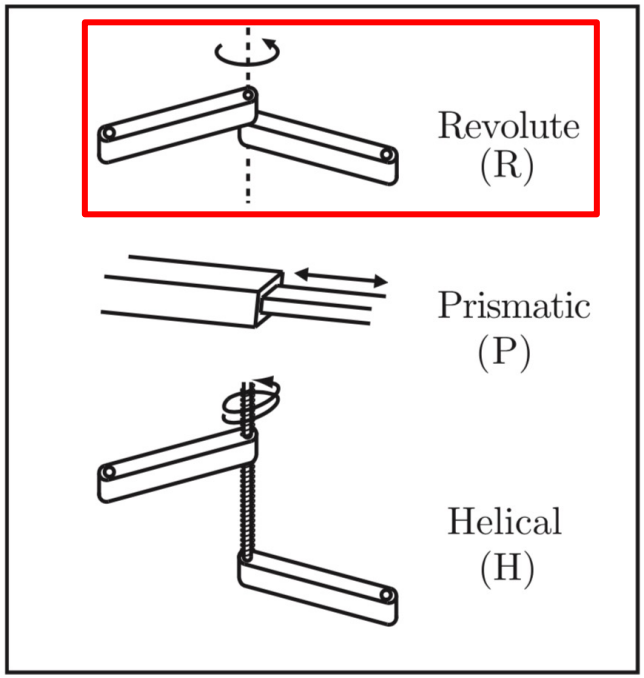
Degree of freedom

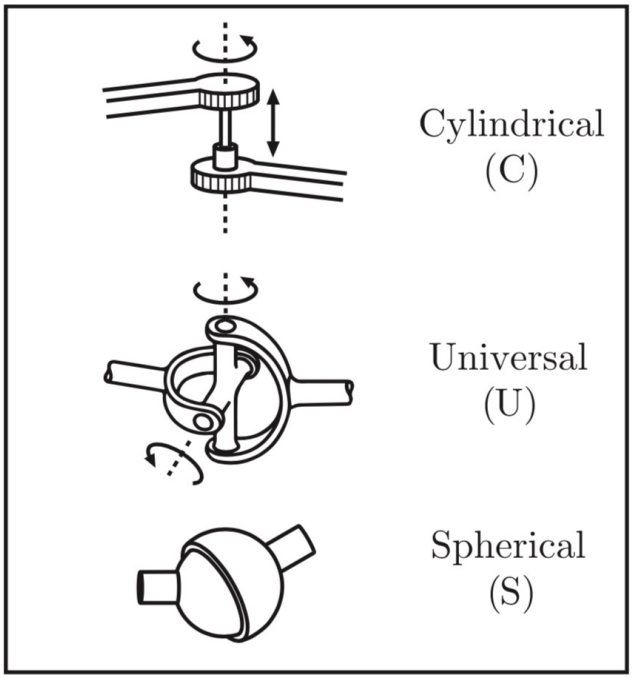
- **Degree of freedom (dof):** smallest number of real-valued coordinates needed to represent the configuration.
- Dof of a 3D rigid body
  - 6 dof



Joints

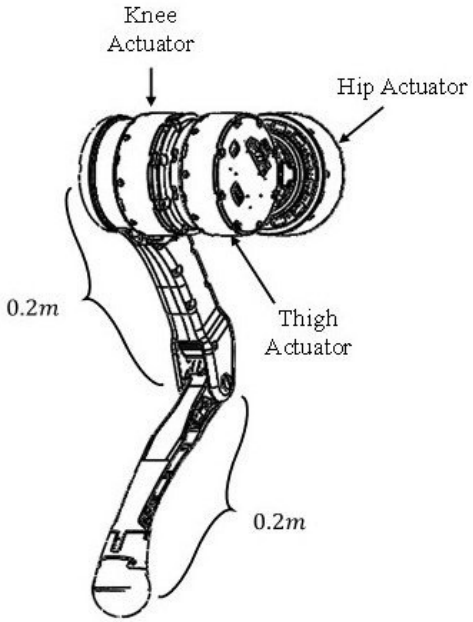
- Robots are composed of multiple rigid body links connected by **joints**
- Existing types of joints
  - Most robots are composed of **revolute joints**





Dof of a robot

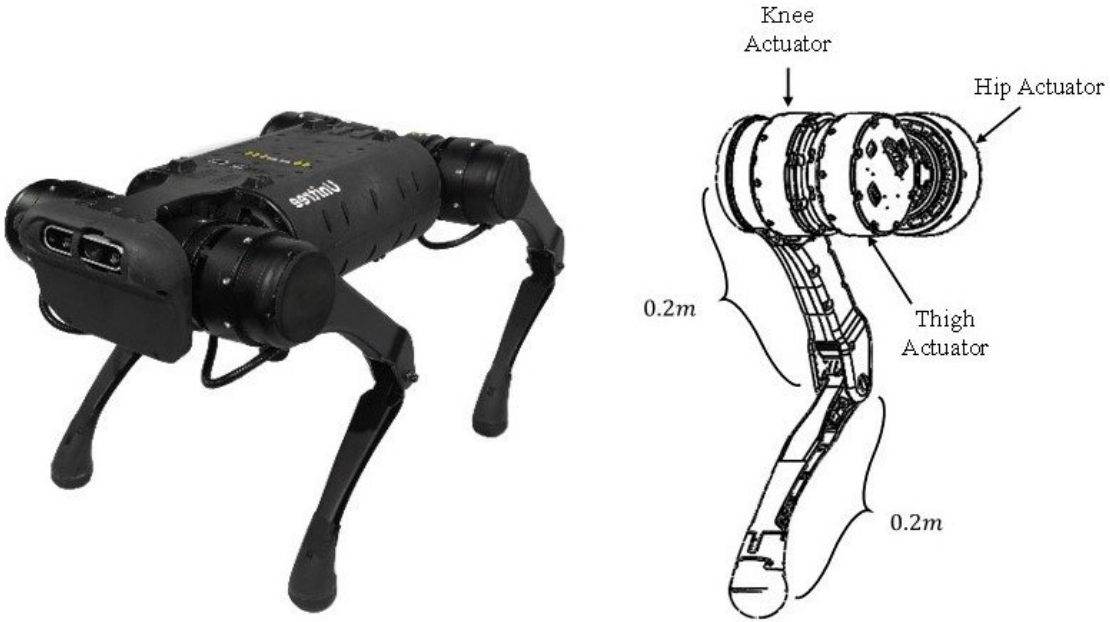
- Question
  - How many degree of freedom is the robot dog?
    - Legs: 3*4
    - Base: 3 (position) + 3 (orientation)



Unitree A1

State space of a robot

- Joint positions
- Joint velocities
- Base position (position and orientation)
- Base velocity (linear and angular)



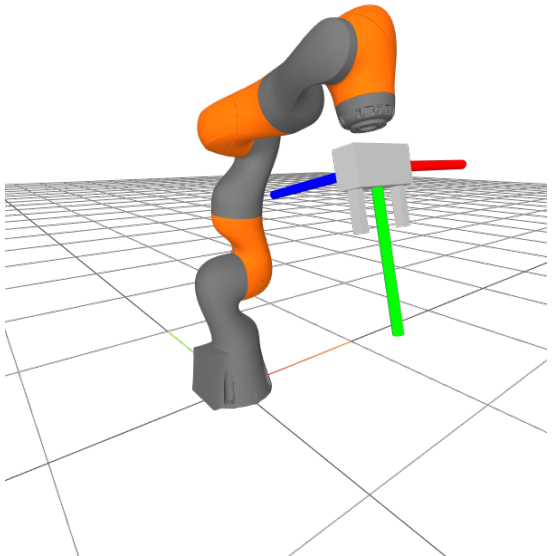
Unitree A1

Forward kinematics

- For a robot, a very basic and important question is, where is its **end-effector**?

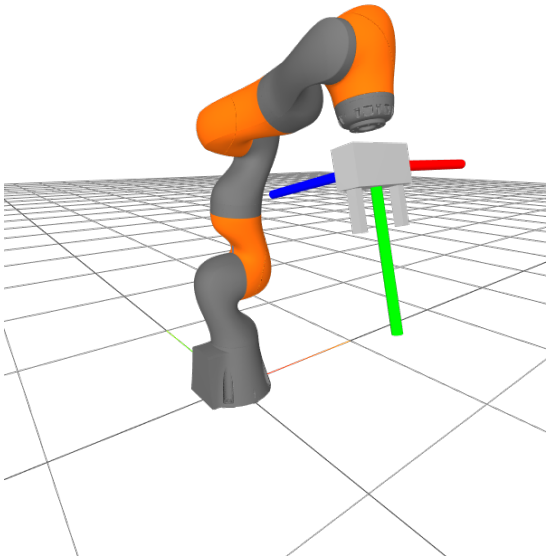
- **Forward kinematics**:
  - Calculate robot's end effector pose from its joint coordinates.

$$T = f(\theta)$$

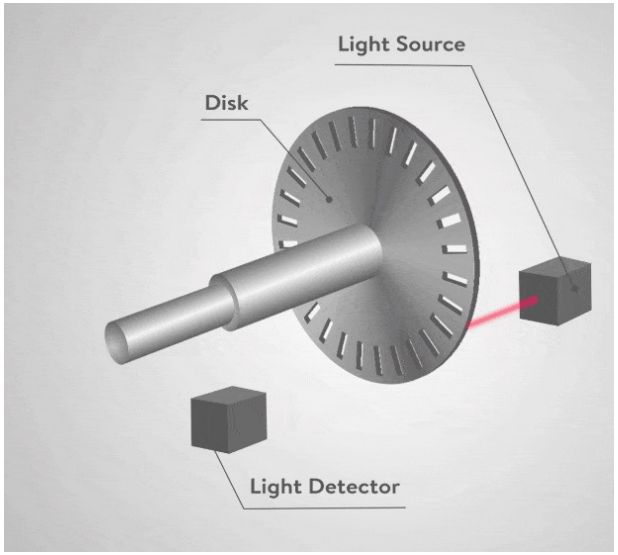


Why study forward kinematic?

- Difficult and expensive to detect the end-effector pose precisely using direct sensors
- Easy and cheap to detect the joint angles directly with encoders



End-effector pose



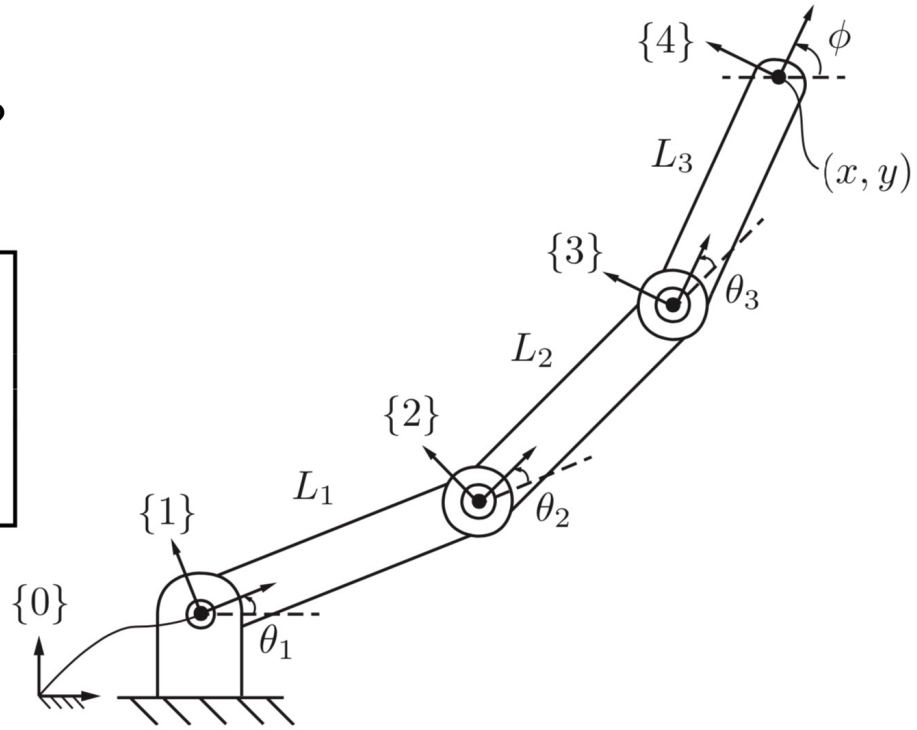
Joint encoder

Calculating forward kinematics

- Use **screw-axis representation**
  - Start from zero-position (all $\theta = 0$)
  - Displace the end-effector from link to link
  - Each displacement represented by a screw motion

What's the pose of {4} at zero position?

$$M = \begin{bmatrix} 1 & 0 & 0 & L_1 + L_2 + L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Calculating forward kinematics

- The displacement of rotating J3 for θ_3 when $\theta_1 = \theta_2 = 0$?

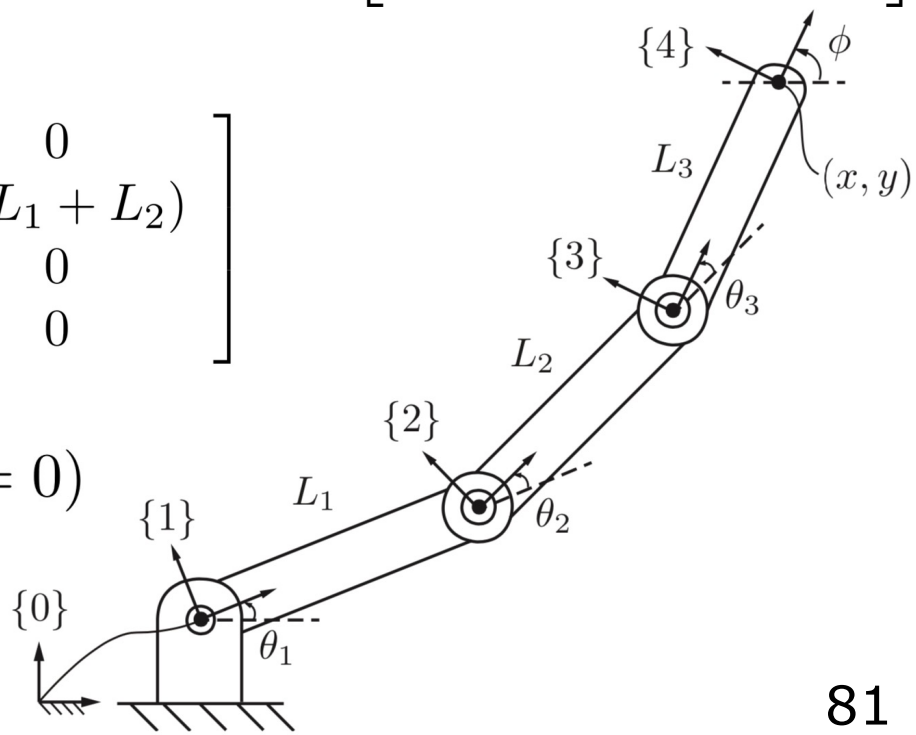
$$T(\mathcal{S}, \theta) = e^{[\mathcal{S}]\theta}$$

$$\mathcal{S}_3 = \begin{bmatrix} \omega_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ -(L_1 + L_2) \\ 0 \end{bmatrix}$$

$$[\mathcal{S}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & v \\ \omega_3 & 0 & -\omega_1 & \\ -\omega_2 & \omega_1 & 0 & \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[\mathcal{S}_3] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -(L_1 + L_2) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T_{04} = e^{[\mathcal{S}_3]\theta_3} M \qquad (\text{for } \theta_1 = \theta_2 = 0)$$



Calculating forward kinematics

- The displacement of rotating J2 for θ_2 when $\theta_1 = 0$?

$$[\mathcal{S}_2] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -L_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

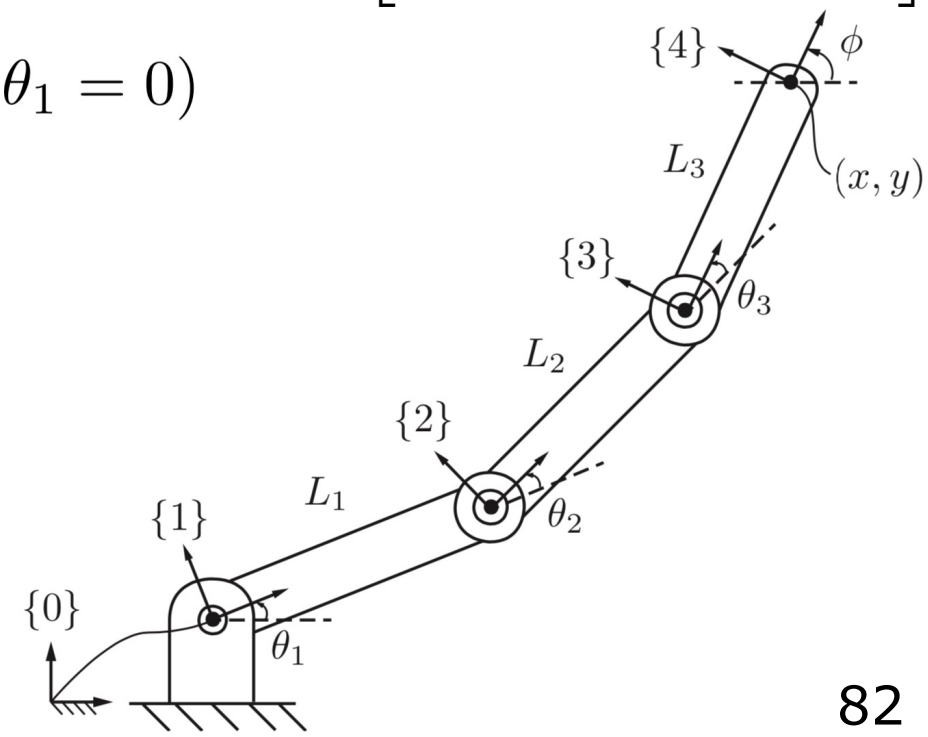
$$T(\mathcal{S}, \theta) = e^{[\mathcal{S}]\theta}$$

$$[\mathcal{S}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} & v \\ 0 & 0 \end{bmatrix}$$

$$T_{04} = e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} M \qquad (\text{for } \theta_1 = 0)$$

$$[\mathcal{S}_1] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T_{04} = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} M$$



Calculating forward kinematics

- End-effector zero-position M and all screws $\mathcal{S}_i$ are **predefined by the robot structure**

$$T_{04} = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} M$$

$$[\mathcal{S}_2] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -L_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & 0 & L_1 + L_2 + L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[\mathcal{S}_1] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad [\mathcal{S}_3] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -(L_1 + L_2) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Forward kinematics formula

- Now the steps to compute robot forward kinematics using product of exponentials is clear:
  - Obtain the end-effector configuration $M \in SE(3)$ at zero-position.
  - Obtain screw axes $\mathcal{S}_1, \ldots, \mathcal{S}_n$ expressed in space frame when the robot is at zero-position.
  - Determine the joint angles $\theta_1, \ldots, \theta_n$ that the robot should be positioned.
  - Apply **the product of exponential formula**:

$$T(\theta) = e^{[\mathcal{S}_1]\theta_1} \cdots e^{[\mathcal{S}_{n-1}]\theta_{n-1}} e^{[\mathcal{S}_n]\theta_n} M$$

Velocity kinematics and Jacobian

- In **forward kinematics**, we calculate the end-effector's **pose** given a set of **joint positions**.
- Now let's see how we can calculate a robot's end-effector's **velocity** given given a set of **joint positions** and **velocities**.
- Take the linear velocity as an example

$$x(t) = f(\theta(t))$$

$$\begin{aligned} \dot{x} &= \frac{\partial f(\theta)}{\partial \theta} \frac{d\theta(t)}{dt} = \frac{\partial f(\theta)}{\partial \theta} \dot{\theta} \\ &= J(\theta)\dot{\theta}, \end{aligned}$$

- This is the **velocity kinematics**, and $J(\theta)$ is the robot's **Jacobian**.

Jacobian

- For spatial velocity

$$\begin{aligned} \mathcal{V}_s &= \begin{bmatrix} J_{s1} & J_{s2}(\theta) & \cdots & J_{sn}(\theta) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} \\ &= J_s(\theta)\dot{\theta}. \end{aligned}$$

- Jacobian
  - The robot forward kinematics is $T = e^{[\mathcal{S}_1]\theta_1} \cdots e^{[\mathcal{S}_n]\theta_n} M$
  - The **Jacobian** $J_s(\theta) \in \mathbb{R}^{6\times n}$ relates the **joint velocity** $\dot{\theta}$ to the **spatial twist** $\mathcal{V}_s = J_s(\theta)\dot{\theta}$
  - The ith column of $J_s(\theta)$ is

$$J_{si}(\theta) = \mathrm{Ad}_{e^{[\mathcal{S}_1]\theta_1} \cdots e^{[\mathcal{S}_{i-1}]\theta_{i-1}}}(\mathcal{S}_i) \qquad J_{s1} = \mathcal{S}_1$$

  - Only the **screws** $\mathcal{S}_i$ and **joint position** θ are needed.

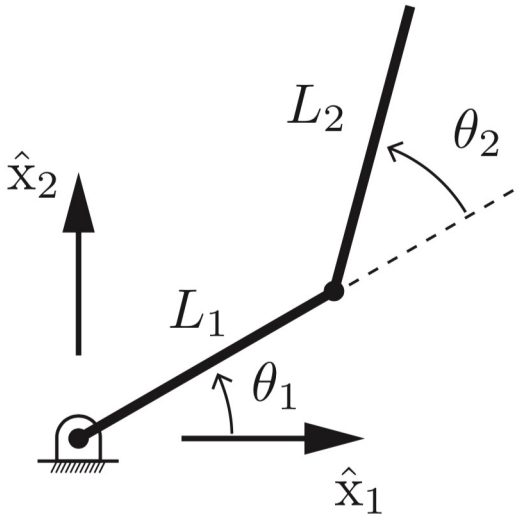
Singularity

- Think about when stretching your arm straight

Singularity: A planar robot example

- Let's look at a planar 2R robot example
- What is the forward kinematics?
  - Graphically, only consider tip position

$$\begin{aligned} x_1 &= L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ x_2 &= L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2). \end{aligned}$$



- What is the velocity kinematics?
  - Direct differentiation, only consider tip position

$$\begin{aligned} \dot{x}_1 &= -L_1\dot{\theta}_1 \sin\theta_1 - L_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2) \\ \dot{x}_2 &= L_1\dot{\theta}_1 \cos\theta_1 + L_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2), \end{aligned}$$

Singularity: A planar robot example

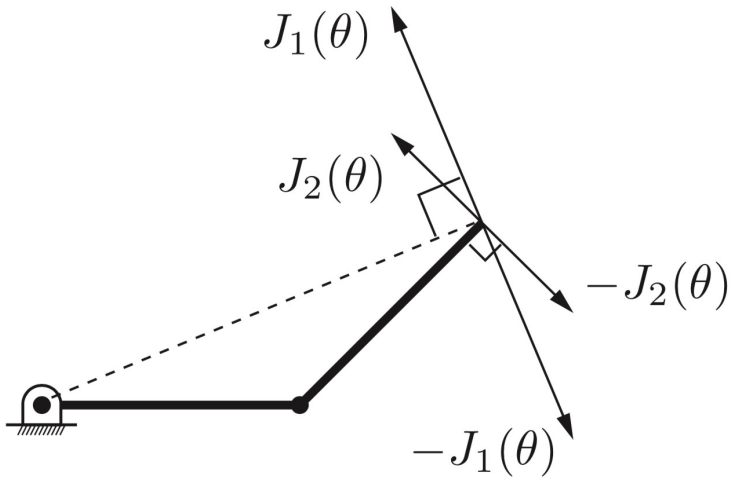
- Writing in the form of $\dot{x} = J(\theta)\dot{\theta}$:

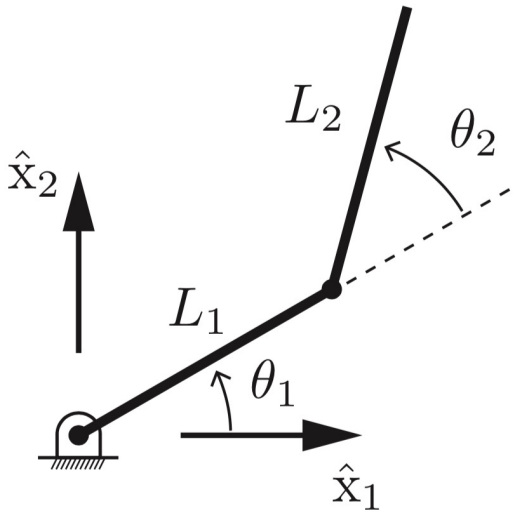
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1+\theta_2) & -L_2 \sin(\theta_1+\theta_2) \\ L_1 \cos\theta_1 + L_2 \cos(\theta_1+\theta_2) & L_2 \cos(\theta_1+\theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

- The tip velocity is a linear combination of $J_1(\theta)$ and $J_2(\theta)$

$$v_{\text{tip}} = J_1(\theta)\dot{\theta}_1 + J_2(\theta)\dot{\theta}_2$$

- Draw the directions of $J_1(\theta)$ and $J_2(\theta)$





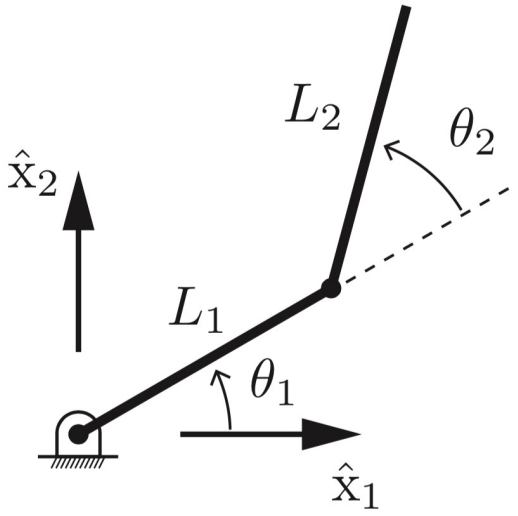
Singularity: A planar robot example

- When will $J_1(\theta)$ and $J_2(\theta)$ be **collinear**?
- When $\theta_2 = 0$:

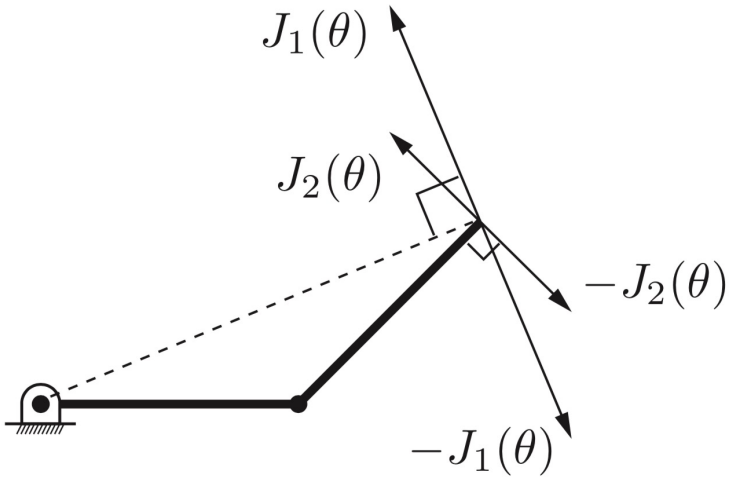
$$J_1(\theta) = \frac{L_1 + L_2}{L_2} J_2(\theta)$$

- When $\theta_2 = \pi$:

$$J_1(\theta) = \frac{L_2 - L_1}{L_2} J_2(\theta)$$



- $J(\theta)$ becomes a **singular** matrix and the robot tip **cannot generate velocities in a certain direction**.



Singularities

- Recall the Jacobian

$$\mathcal{V}_b = \begin{bmatrix} J_{b1}(\theta) & \cdots & J_{bn-1}(\theta) & J_{bn} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_{n-1} \\ \dot{\theta}_n \end{bmatrix}$$

$$= J_{b1}(\theta)\dot{\theta}_1 + \cdots + J_{bn-1}(\theta)\dot{\theta}_{n-1} + J_{bn}\dot{\theta}_n.$$

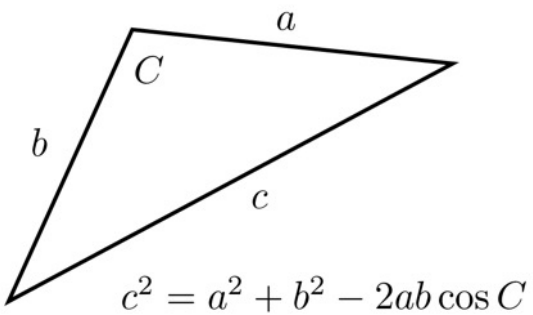
- The end-effector can achieve twists that are **linear combinations** of J_{bi} .

Singularities

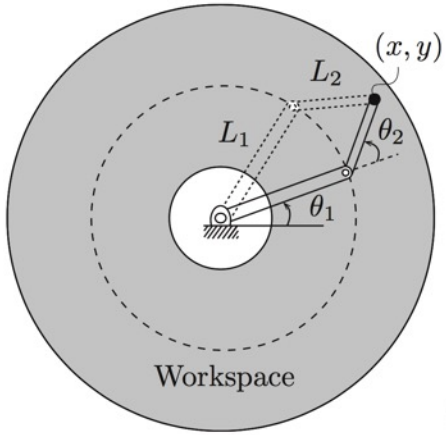
- When $n \geq 6$, the maximum rank $J_b(\theta)$ can attain is 6.
- A **singularity** corresponds to posture such that the rank of $J_b(\theta)$ drops below its maximum possible value.
- At a singularity, the end-effector will lose the ability to generate instantaneous spatial velocities in one or more dimensions.

Inverse kinematics

- The **inverse kinematics** problem is to calculate the joint angles θ_d that satisfy $T_d = f(\theta_d)$ for a given desired end-effector pose $T_d \in SE(3)$
- Unlike FK, IK for serial chains could have zero, one, or multiple solutions
- **Analytical IK**



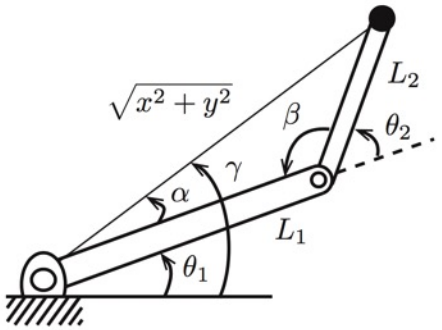
law of cosines



IK for a 2R robot

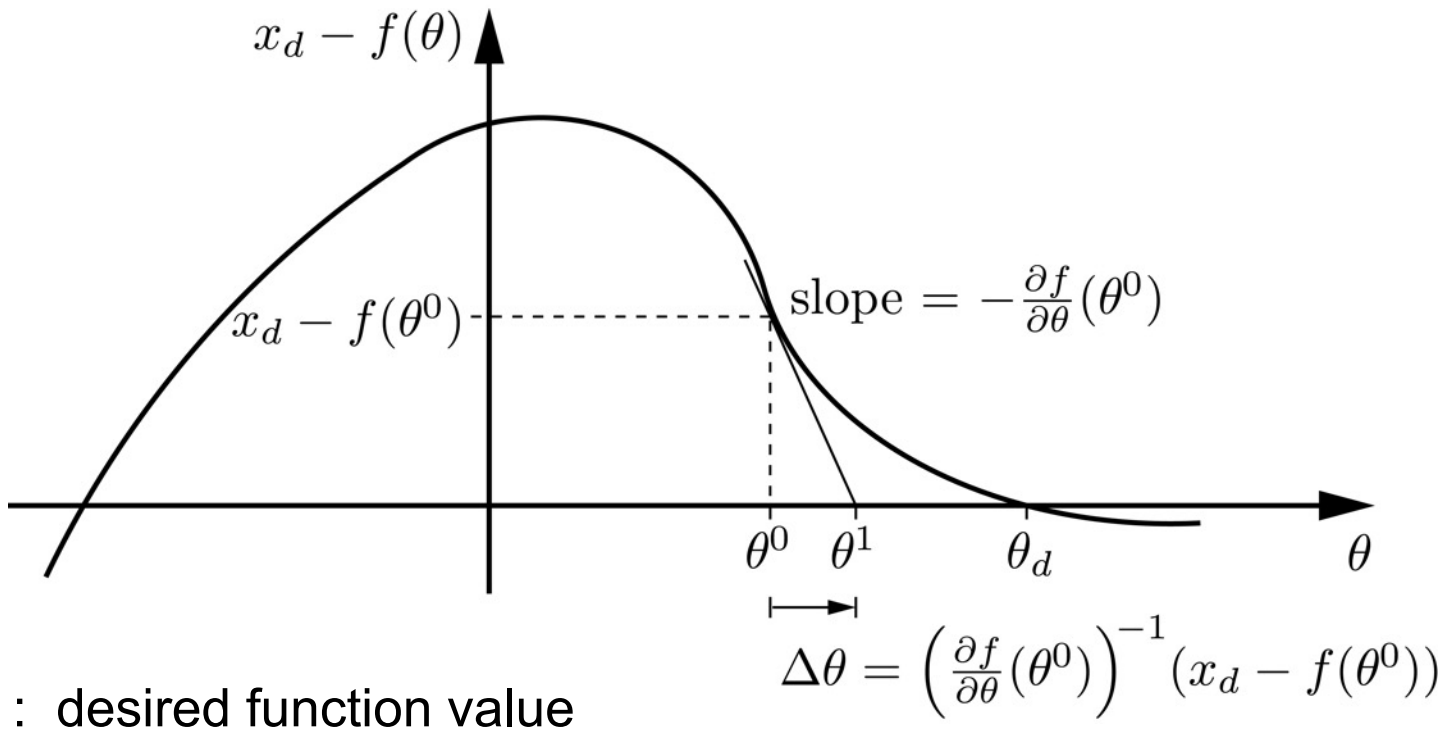
γ from atan2

α, β from law of cosines



Inverse kinematics

- **Numerical IK**: iteratively refine an initial guess θ to find θ^k such that $T(\theta^k) \approx T_d$.
- This is similar to **Newton–Raphson root finding**



x_d : desired function value
$f(\theta)$: actual function value at θ

Inverse kinematics

- Sudo-code for inverse kinematics
  - Initialization: Given T_{sd} and initial guess θ_0 , set i=0
  - Calculate $[\mathcal{V}_s] = \log(T_{sd}T_{sb}^{-1}(\theta^i))$
    - Where

$$[\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \quad \mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix}$$

  - While $||\omega_s|| > \epsilon_\omega$ or $||v_s|| > \epsilon_v$:
    - $\theta^{i+1} = \theta^i + J_s^\dagger(\theta^i)\mathcal{V}_s$
    - Increment i

Rigid body spatial velocity

Sudo-inverse of Jacobian

Application for inverse kinematics

- Tele-operate a robot from human motion

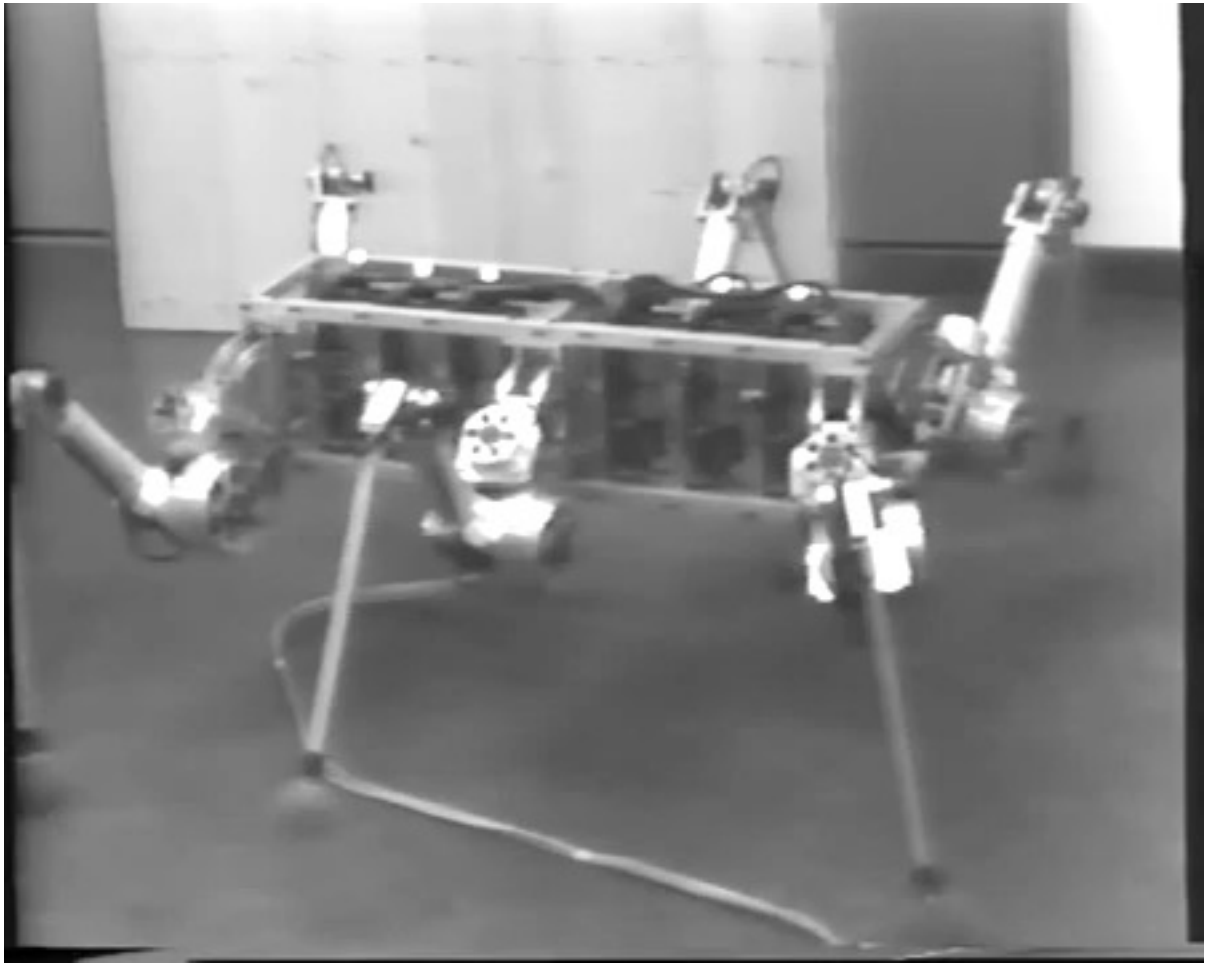
Application for inverse kinematics

- Control the motion of the robot autonomously

Contents

Why care about dynamics?

- Robot with purely kinematics planning

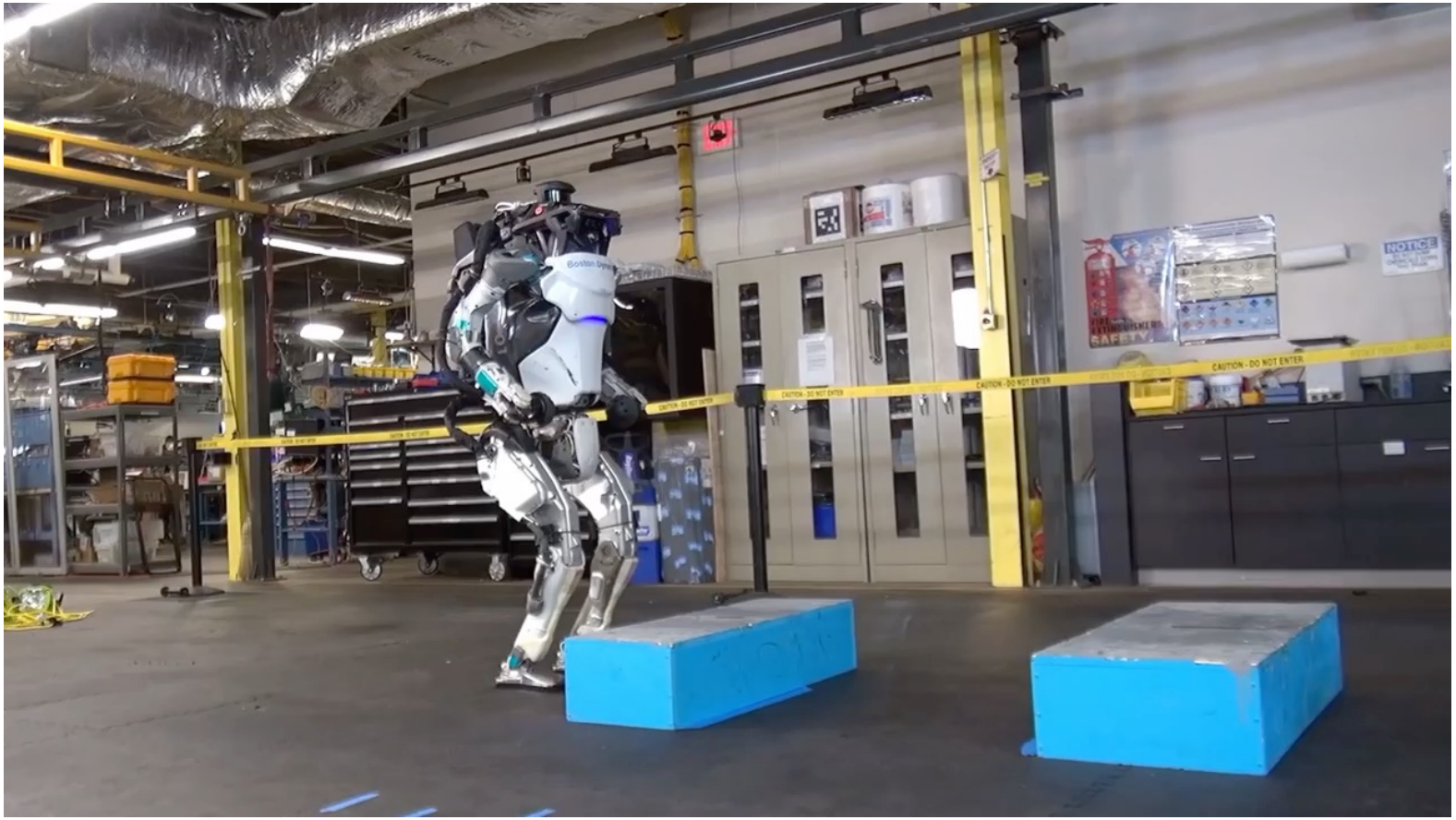


Why care about dynamics?

- When adding some control to the dynamical robots:

Why care about dynamics?

- The Boston Dynamics Atlas robot in 2017

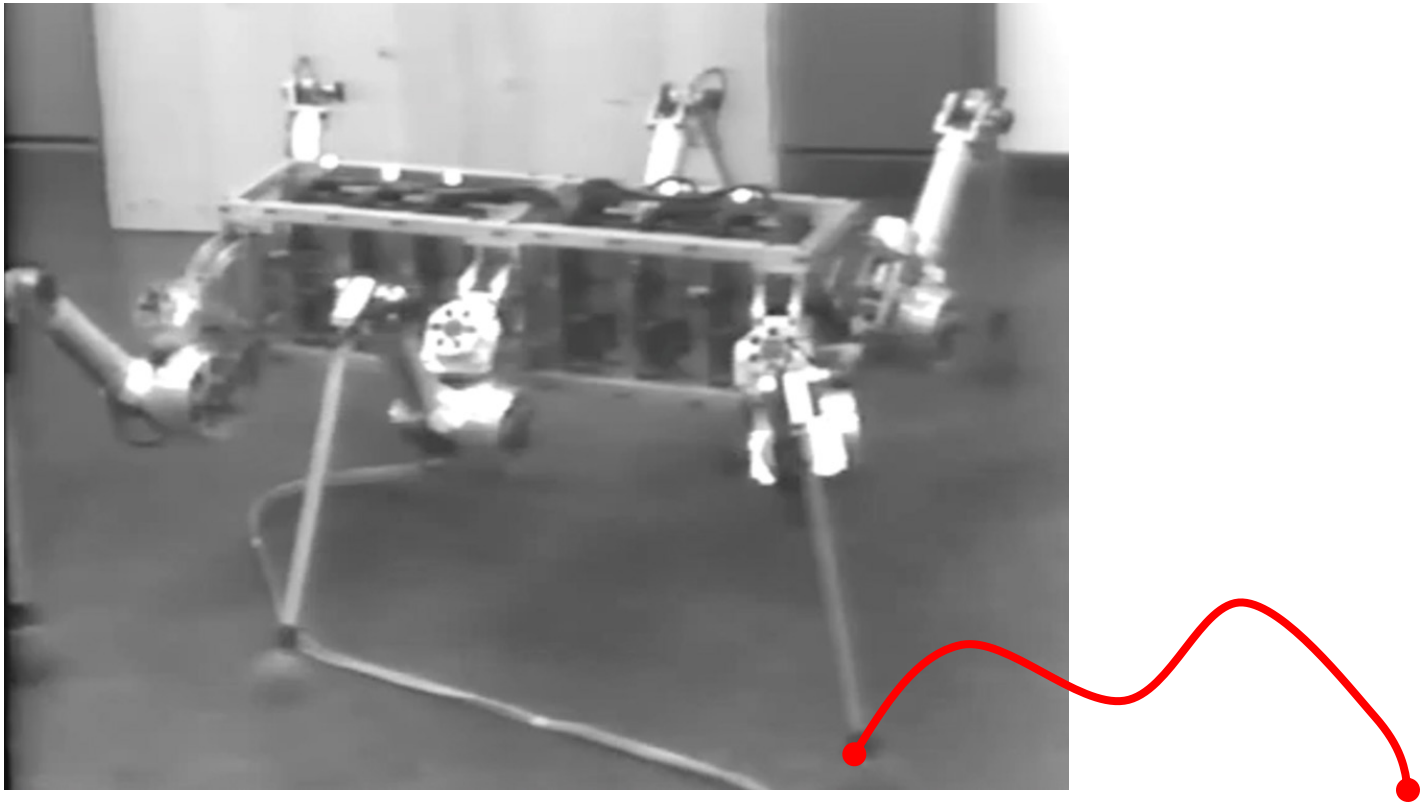


Why care about dynamics?

- The following is a fully passive “robot” with **no power and no actuators**, but makes use of the dynamics

Why care about dynamics?

- Kinematics only talk about how to describe a motion, but does not talk about **how the motion is generated**.
- Nothing to talk about masses, gravity…

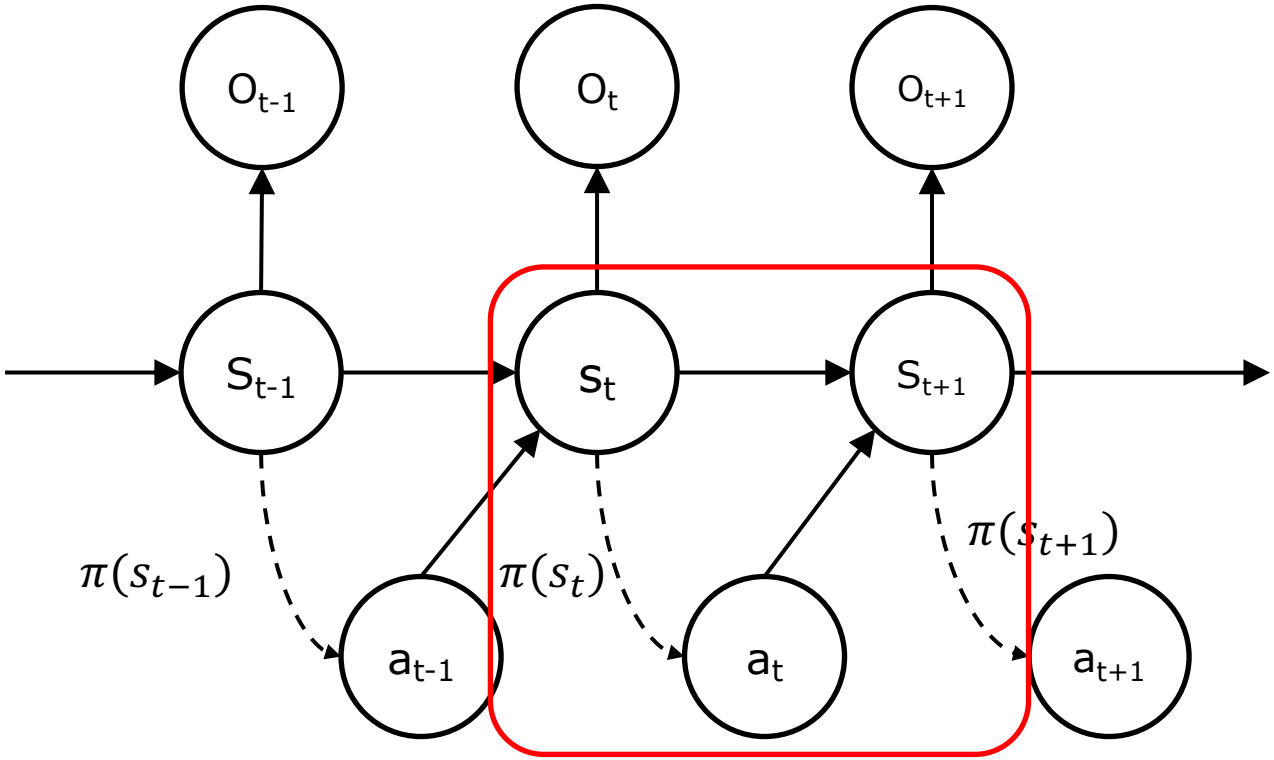


Why care about dynamics?

- Dynamics talks about how the motion is generated, according to the **physical laws constraints**
- There are a lot of different physics laws, all can be described in dynamics

Robot dynamics

- Some dynamics can be modeled, some are hard.
- We concentrate on robot dynamics, which is easier to model precisely



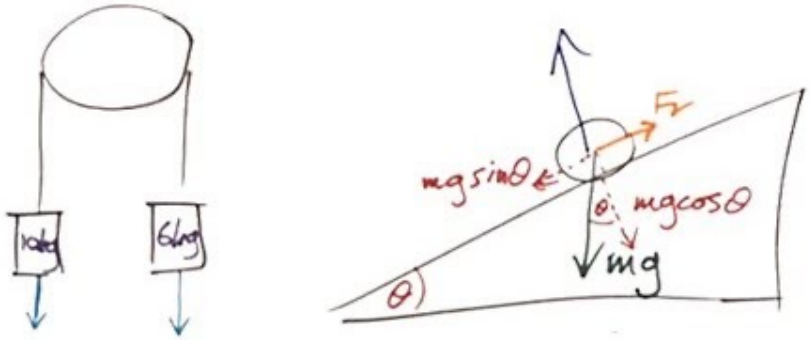
s: state
a: action
o: observation
π : policy

Robot dynamics model inside

Robot dynamics

- So far, we have been concentrating on the robots'
  - The configuration, forward/inverse kinematics
  - Twists, velocity kinematics
- But we haven't talked about:
  - **Accelerations** of the motion
  - **Forces/torques** that generate the motion

$$F = m a$$



Robot dynamics

- What's the "F=ma" equation for robots?
- We will study the equations of motion for robots, or robot dynamics
  - **Inverse** dynamics

$$\theta,\ \dot{\theta},\ \ddot{\theta} \quad \Rightarrow \quad \tau$$

  - **Forward** dynamics

$$\theta,\ \dot{\theta},\ \tau \quad \Rightarrow \quad \ddot{\theta}$$

Basics of Lagrangian dynamics

- Basic concept
  - **Generalized coordinates** q
  - **Generalized forces** f
  - For robots, the above are its dof pos and τ
- Lagrangian dynamics
  - Lagrangian function

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{P}(q)$$

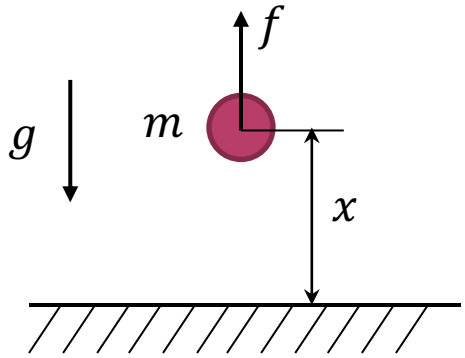
Kinetic energy Potential energy

  - Equations of motion

$$f = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q}$$

A point mass example

- Consider a particle with mass m and height x



- The generalized coordinate is x, generalized force is f
- The kinetic energy is:

$$\mathbf{m}\dot{x}^2/2$$

- The potential energy is:

$$\mathbf{m}gx$$

A point mass example

- The corresponding Lagrangian is

$$\mathcal{L}(x, \dot{x}) = \mathcal{K}(x, \dot{x}) - \mathcal{P}(x) = \frac{1}{2}\mathrm{m}\dot{x}^2 - \mathrm{m}gx$$

- And the Lagrangian-based equation of motion is

$$f = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{x}} - \frac{\partial \mathcal{L}}{\partial x} = \mathrm{m}\ddot{x} + \mathrm{m}g$$

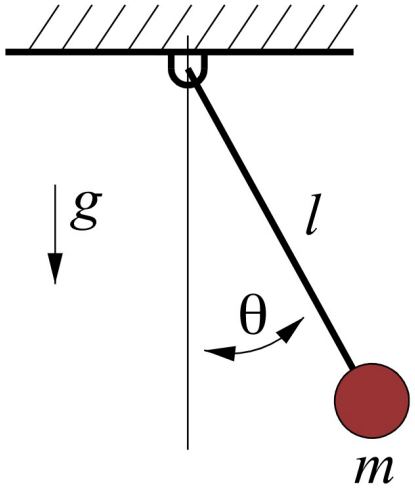
- This resembles the Newton's law

$$f - \mathrm{m}g = \mathrm{m}\ddot{x}$$

A pendulum example

- What is the dynamic equation for a damped pendulum?

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i$$



$$T = \frac{1}{2}m(\dot{\theta}l)^2 \qquad U = -mgl\cos\theta \qquad L = \frac{1}{2}m(\dot{\theta}l)^2 + mgl\cos\theta$$

$$\frac{\partial L}{\partial \dot{\theta}} = ml^2\dot{\theta} \qquad \frac{\partial L}{\partial \theta} = -mgl\sin\theta \qquad Q = u - b\dot{\theta}$$

$$u = ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta$$

Dynamic equations for robots

- The dynamic equations for robots has a generic form

$$u = ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta$$

$$\tau = M(\theta)\ddot{\theta} + c(\theta,\dot{\theta}) + g(\theta)$$

Force

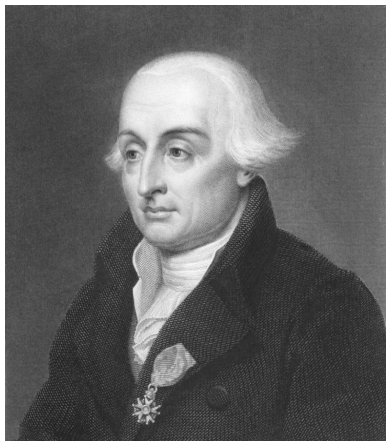
Mass matrix, symmetric and positive definite

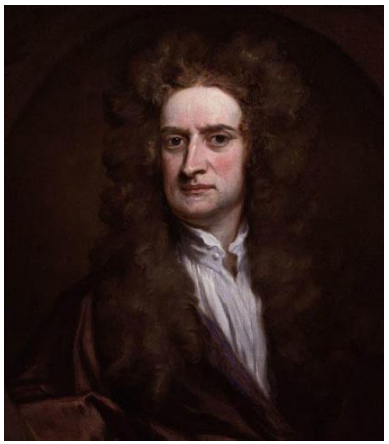
Coriolis & Centripetal

Gravity

Lagrangian vs Newton-Euler dynamics

- **Lagrangian dynamics**
  - Elegant and effective for low dof robots (e.g, 2 or 3)
  - Calculation becomes hard for high dof robots
- **Newton-Euler dynamics**
  - Efficient recursive algorithm for high dof robots
  - No need for differentiation
- The resulted equations are equal for both formulations





Newton-Euler inverse dynamics algorithm

- **Newton-Euler inverse dynamics** algorithm

Forward iterations Given $\theta, \dot{\theta}, \ddot{\theta}$, for $i = 1$ to n do

$$\begin{aligned} T_{i,i-1} &= e^{-[\mathcal{A}_i]\theta_i} M_{i,i-1}, \\ \mathcal{V}_i &= \mathrm{Ad}_{T_{i,i-1}}(\mathcal{V}_{i-1}) + \mathcal{A}_i \dot{\theta}_i, \\ \dot{\mathcal{V}}_i &= \mathrm{Ad}_{T_{i,i-1}}(\dot{\mathcal{V}}_{i-1}) + \mathrm{ad}_{\mathcal{V}_i}(\mathcal{A}_i)\dot{\theta}_i + \mathcal{A}_i \ddot{\theta}_i. \end{aligned}$$

Backward iterations For $i = n$ to 1 do

$$\begin{aligned} \mathcal{F}_i &= \mathrm{Ad}^{\mathrm{T}}_{T_{i+1,i}}(\mathcal{F}_{i+1}) + \mathcal{G}_i \dot{\mathcal{V}}_i - \mathrm{ad}^{\mathrm{T}}_{\mathcal{V}_i}(\mathcal{G}_i \mathcal{V}_i), \\ \tau_i &= \mathcal{F}_i^{\mathrm{T}} \mathcal{A}_i. \end{aligned}$$

- The solution can be written in the same closed form as from Lagrangian dynamics:

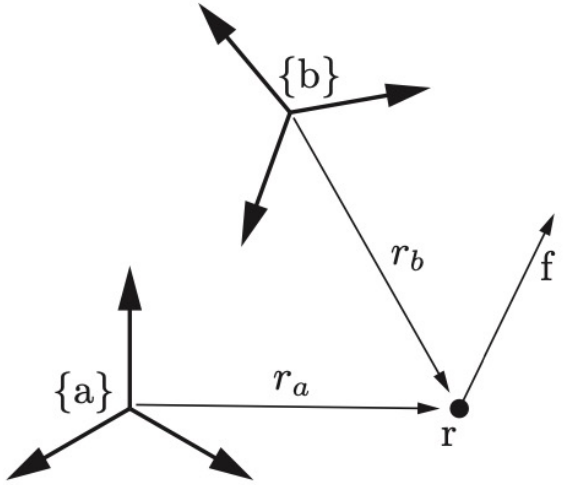
$$\tau = M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) + J^{\mathrm{T}}(\theta)\mathcal{F}_{\mathrm{tip}}$$

End-effector force

Wrench

- Consider a force f acting on a rigid body at point r
- Expressing in frame {a}, this force f_a creates a **torque** or **moment** m_a:

$$m_a = r_a \times f_a$$



- Similar to the twist, we can merge the moment and force as:

$$\mathcal{F}_a = \begin{bmatrix} m_a \\ f_a \end{bmatrix} \in \mathbb{R}^6$$

- This is called **wrench**, the **spatial force**.

Forward dynamics

- The forward dynamics involves solving for $\ddot{\theta}$

$$M(\theta)\ddot{\theta} = \tau(t) - h(\theta, \dot{\theta}) - J^{\mathrm{T}}(\theta)\mathcal{F}_{\mathrm{tip}}$$

$$M\ddot{\theta} = b$$

- **Forward dynamics** function

$$\ddot{\theta} = ForwardDynamics(\theta, \dot{\theta}, \tau, \mathcal{F}_{\mathrm{tip}})$$

- Set $q_1 = \theta,\ q_2 = \dot{\theta}$

Forward dynamics

- **Forward simulation**

$$\begin{aligned} \dot{q}_1 &= q_2, \\ \dot{q}_2 &= ForwardDynamics(q_1, q_2, \tau, \mathcal{F}_{\text{tip}}). \end{aligned}$$

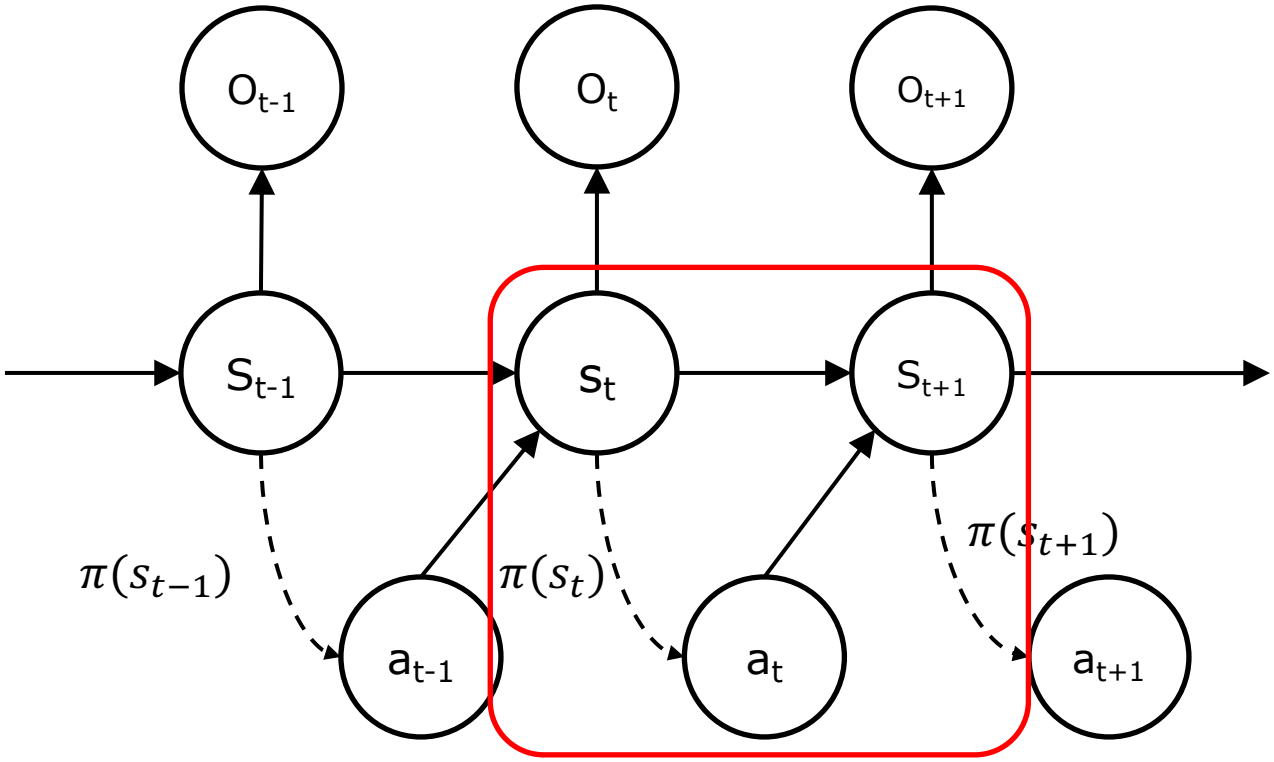
- **Euler integration**

$$\begin{aligned} q_1(t+\delta t) &= q_1(t) + q_2(t)\delta t, \\ q_2(t+\delta t) &= q_2(t) + ForwardDynamics(q_1, q_2, \tau, \mathcal{F}_{\text{tip}})\delta t. \end{aligned}$$

- Higher order: **Runge-Kutta**

Robot modeling

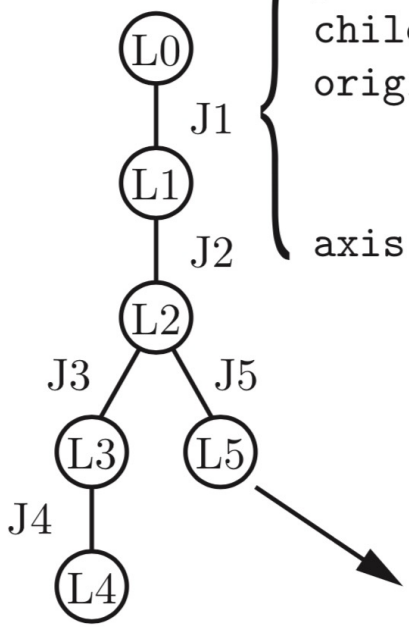
- Robot modeling consists part of the world model



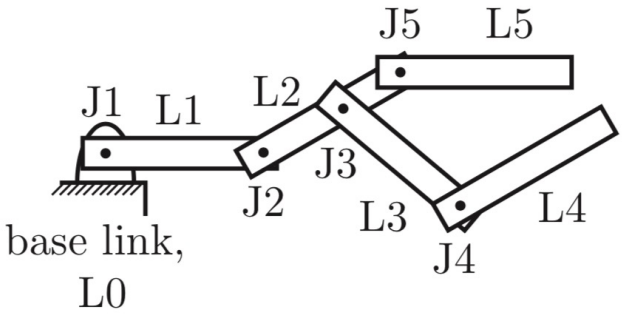
s: state
a: action
o: observation
π : policy

The Universal Robot Description Format

- The **Universal Robot Description Format (URDF)** is a format file describing the kinematics and dynamics of a tree-structured robot



`parent:` J1's parent link, L0
`child:` J1's child link, L1
`origin:` the x–y–z and roll–pitch–yaw coords of the L1 frame relative to the L0 frame when J1 is zero
`axis:` the x–y–z unit vector along the rotation axis in the L1 frame



`mass:` L5's mass
`origin:` the x–y–z and roll–pitch–yaw coords of a frame at the center of mass of L5, relative to the L5 frame
`inertia:` six unique entries of inertia matrix in the `origin` frame

Thank you!

清華大學 交叉信息研究院

Institute for Interdisciplinary Information Sciences, Tsinghua University

ISR Lab

Intelligent Systems and Robotics Lab