UC San Diego

L1: Introduction

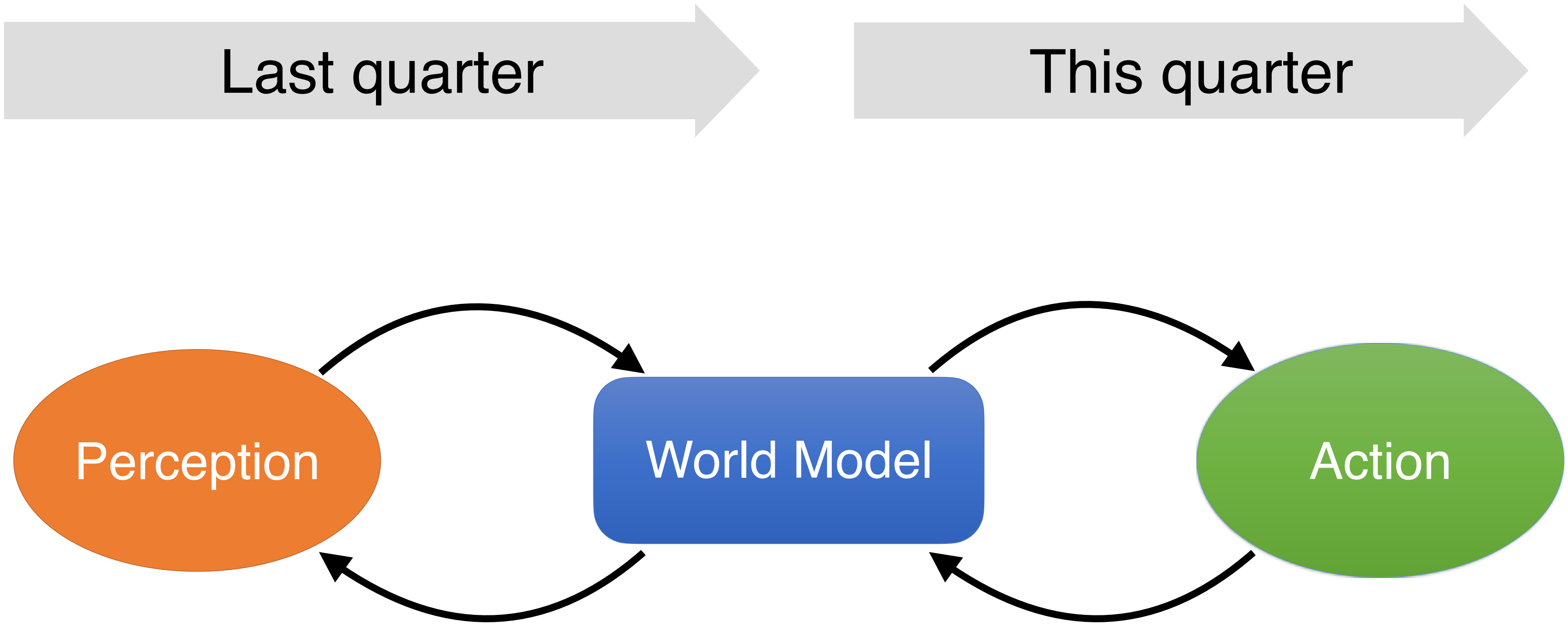
Hao Su

- https://www.youtube.com/watch?v=fn3KWM1kuAw

Agenda

- Syllabus
- Logistics
- $\mathbb{SO}(3)$

Syllabus


Last quarter
This quarter
Perception
World Model
Action

Vision → Robotics

Passive AI

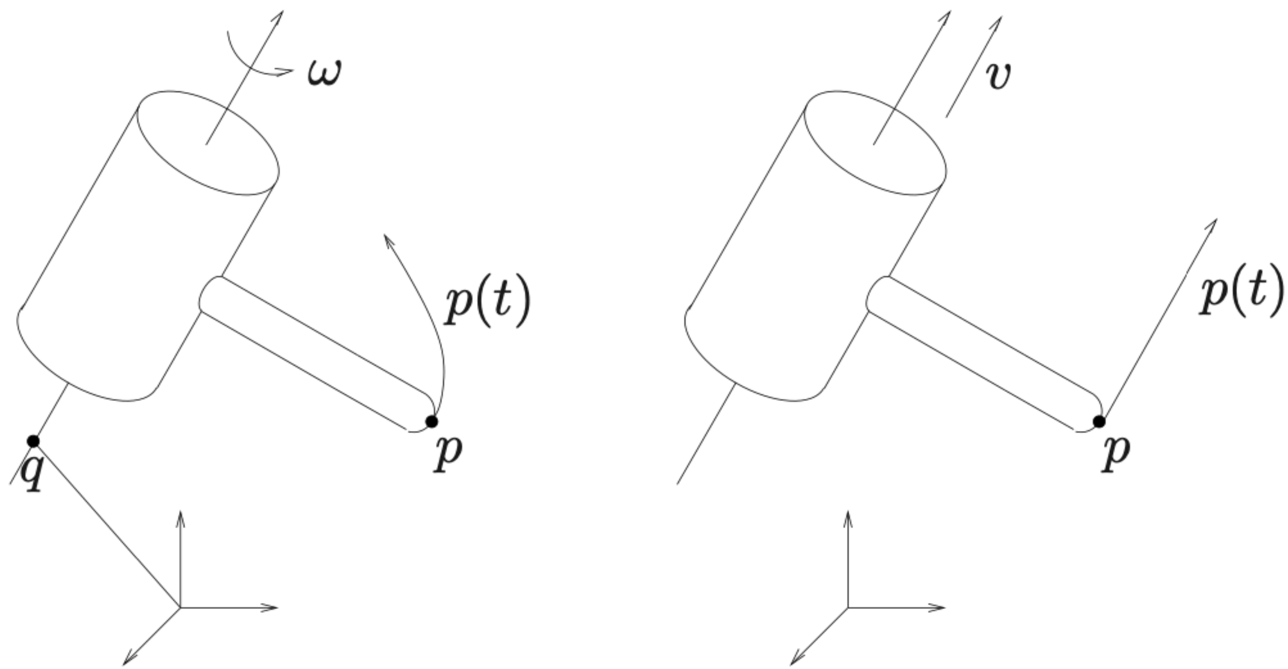
- We know how to fit data well (by "deep learning")
  - e.g., computer vision, natural language processing

Vision → Robotics

- We aspire that autonomous agents can perform tasks and “grow” through interaction experiences
  - Need the ability to **interact**

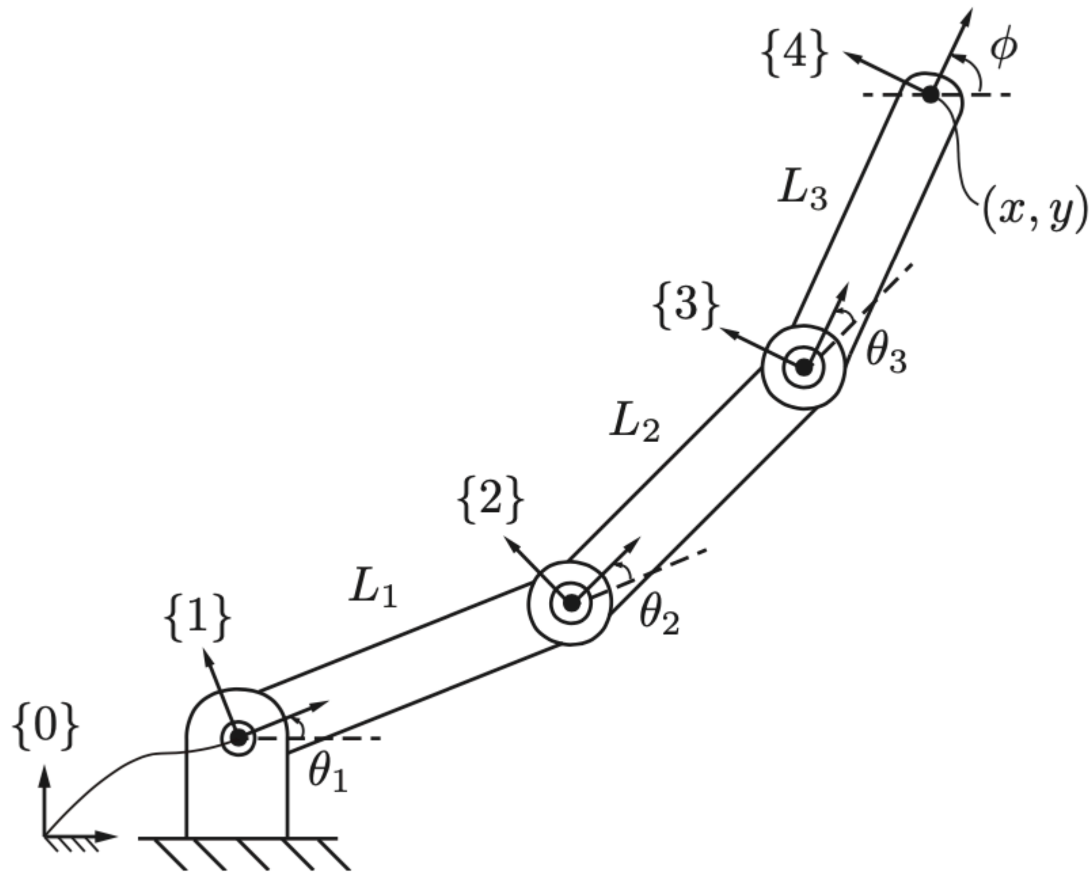
Topics Covered in This Course

- Modeling Robots by Rigid-Body Geometry



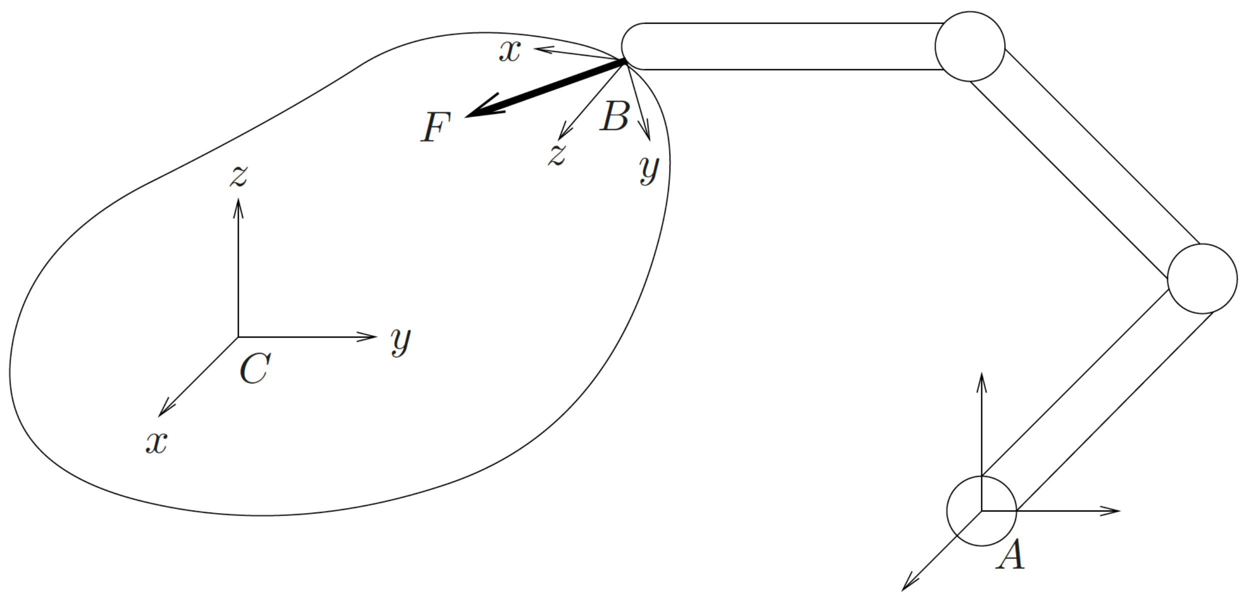
Topics Covered in This Course

- Forward and Inverse Kinematics of Robots



Topics Covered in This Course

- Generalized Force and Inertia



Topics Covered in This Course

- Friction, Contact Model, and Grasp

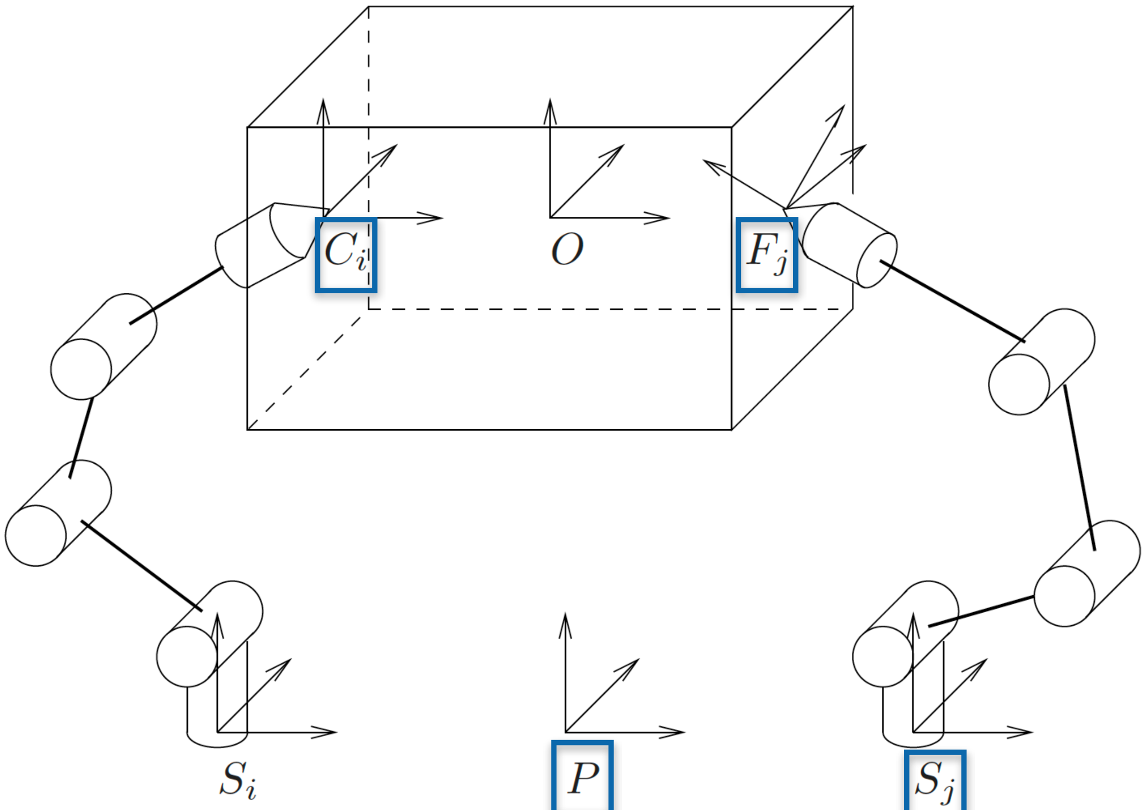
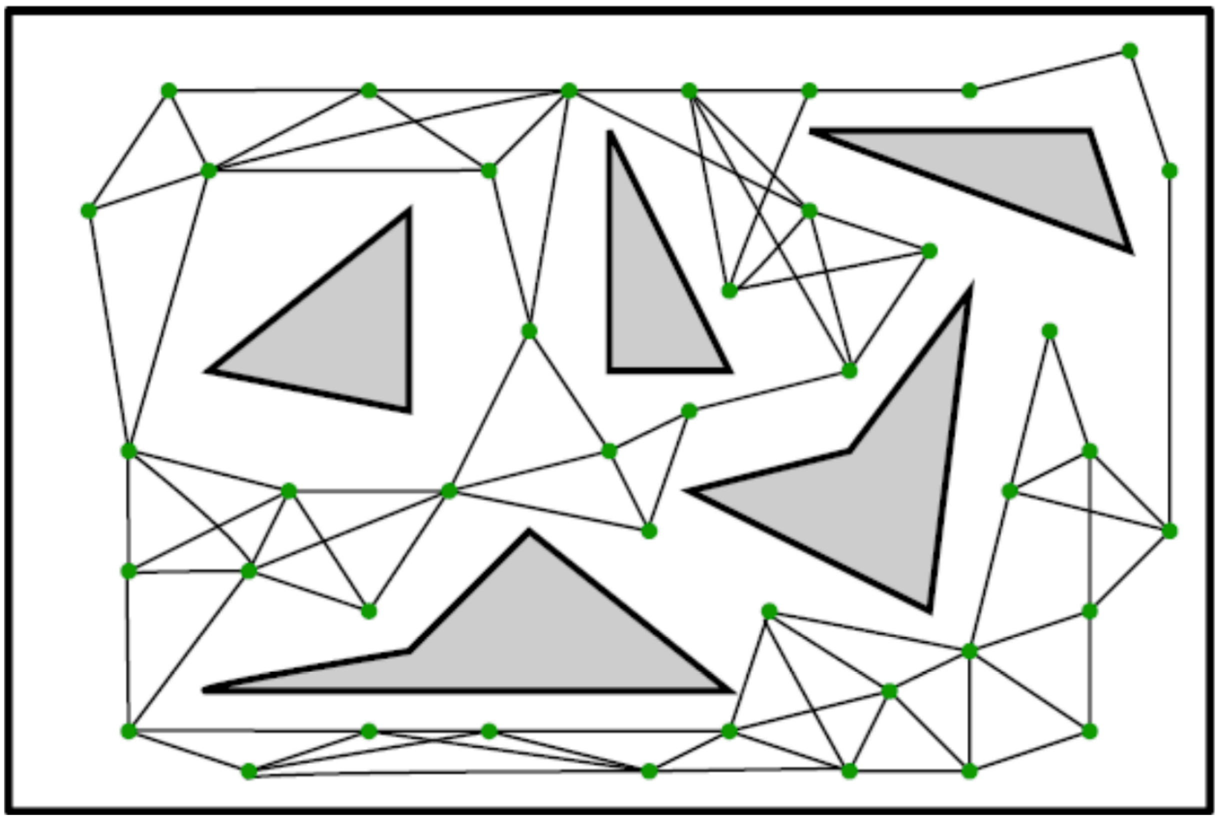


Figure 5.14: Grasp coordinate frames.

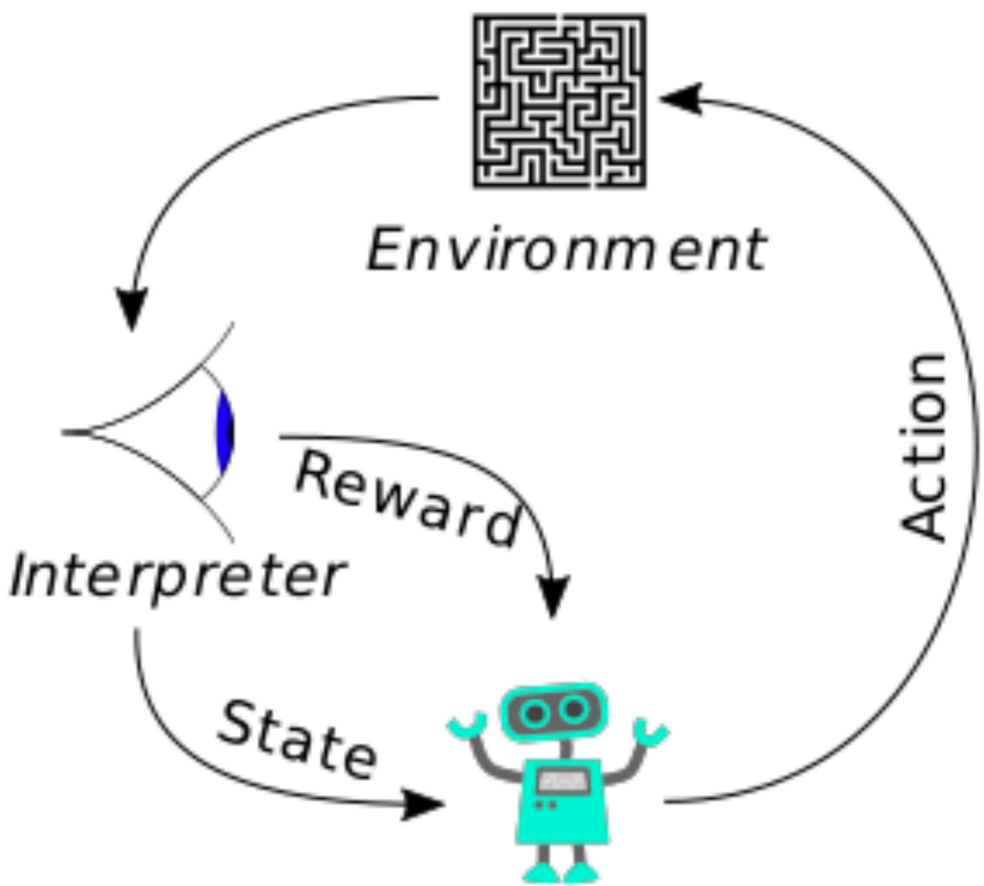
Topics Covered in This Course

- Classical Planning and Control



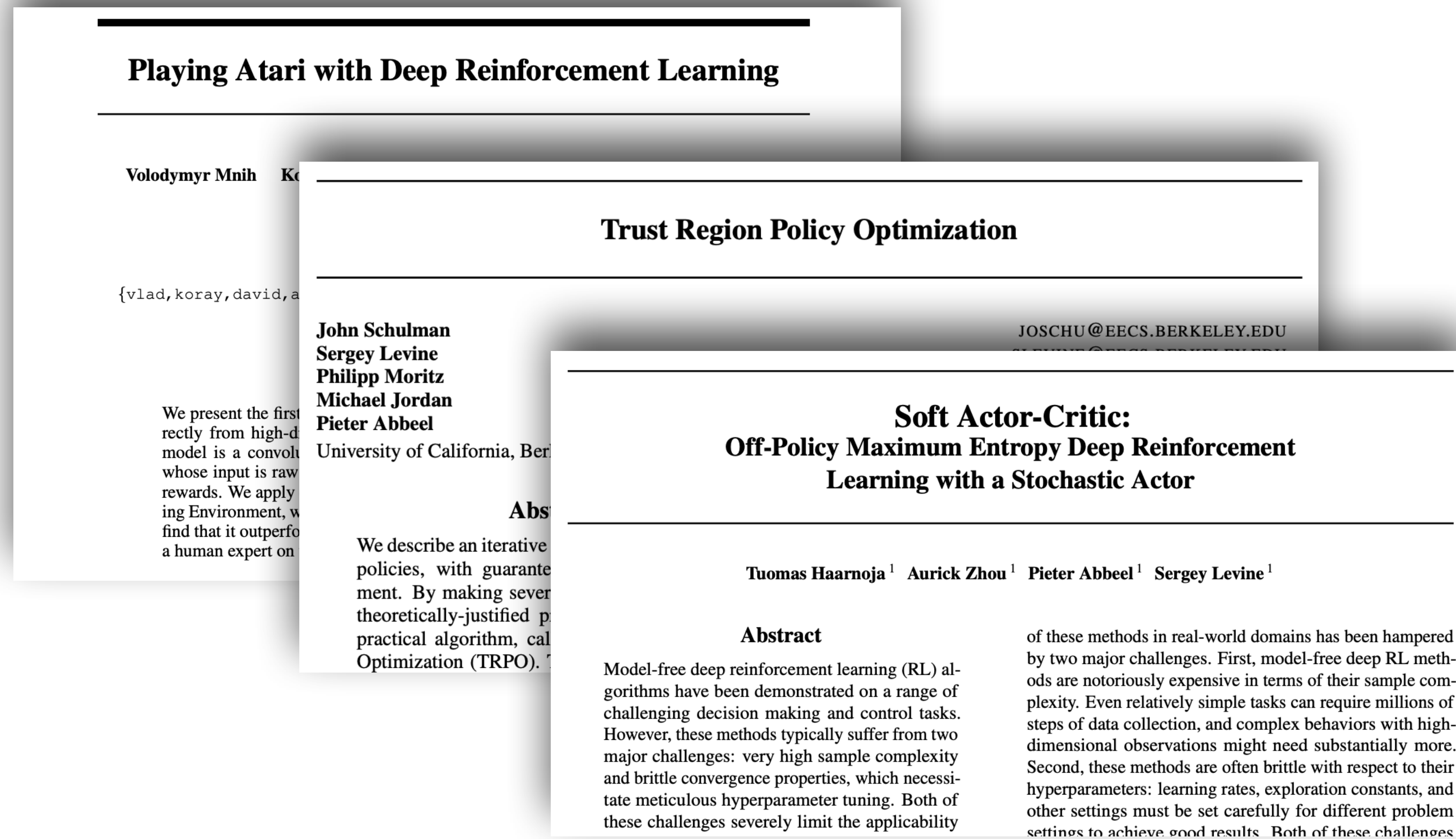
Topics Covered in This Course

- Concepts of Reinforcement Learning



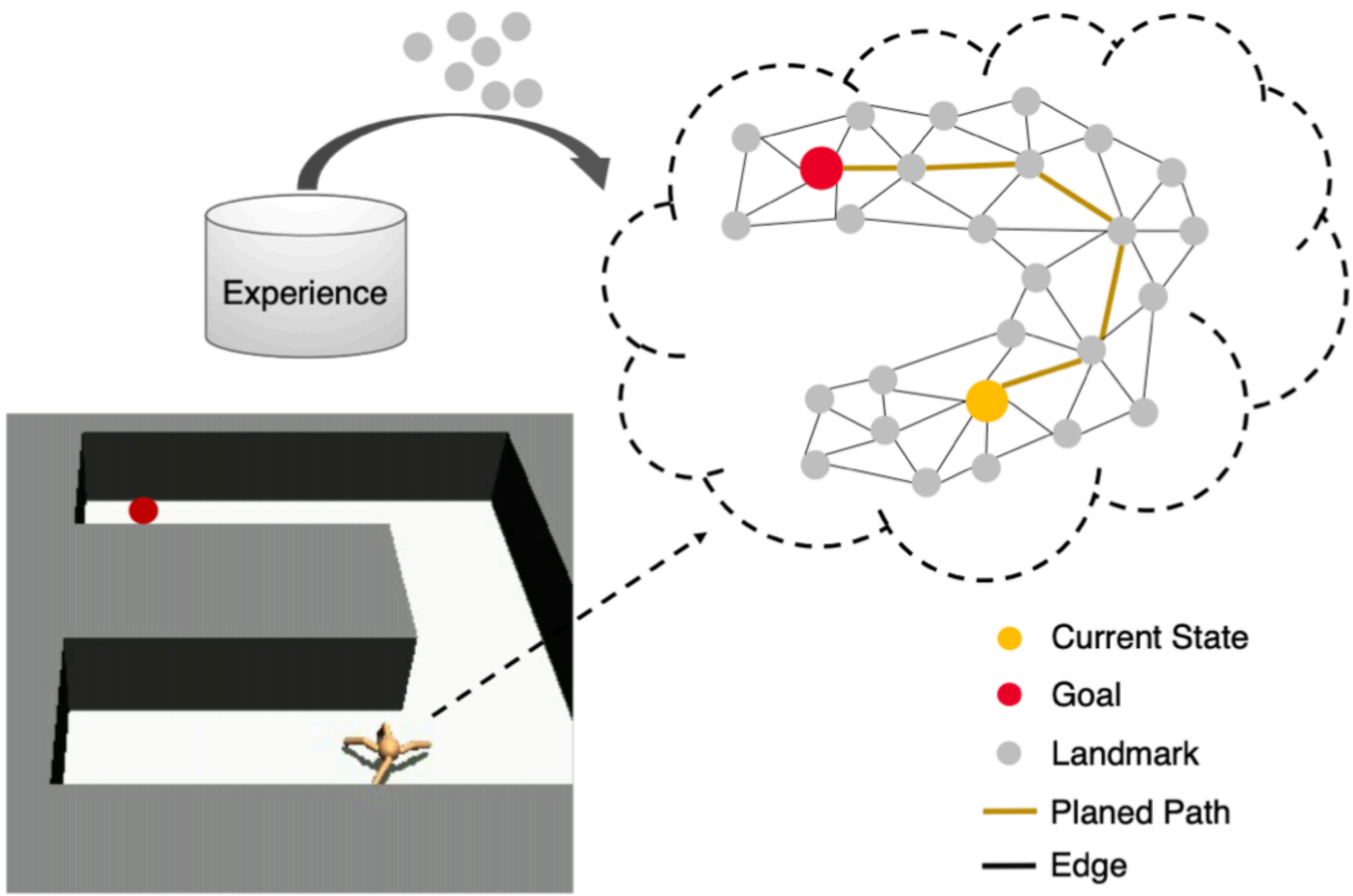
Topics Covered in This Course

- Deep RL Frameworks



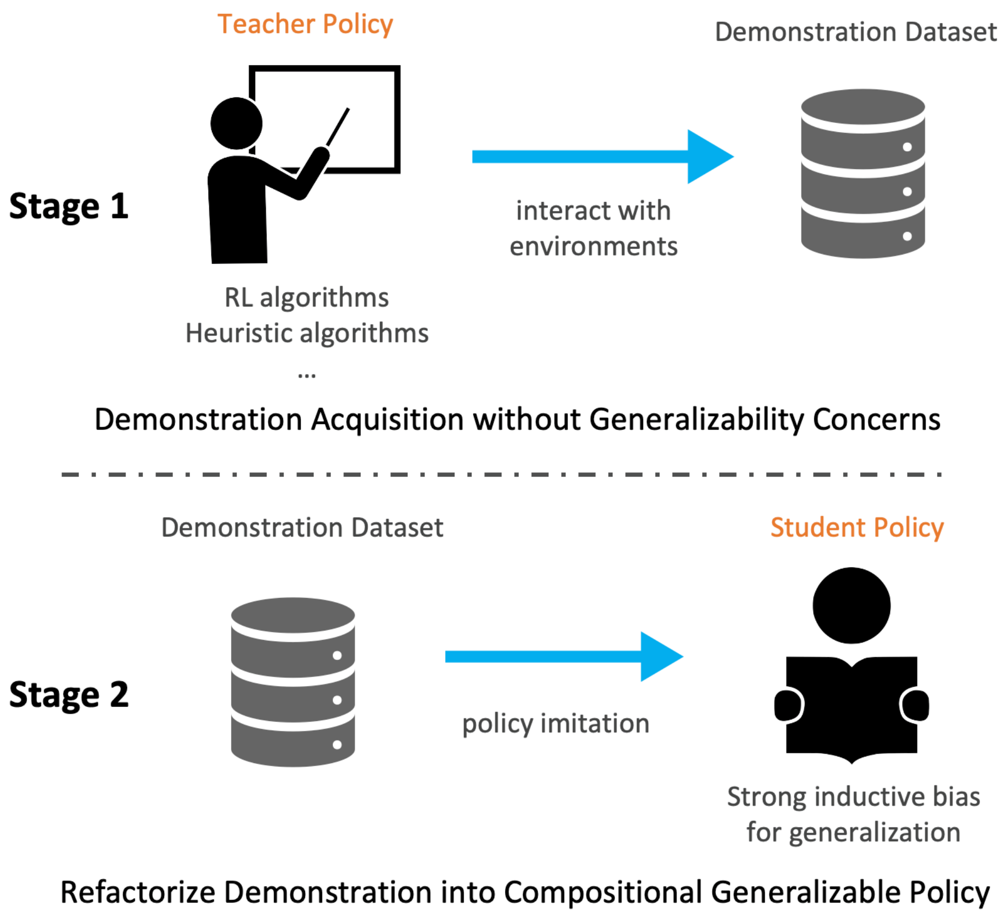
Topics Covered in This Course

- Hierarchical RL



Topics Covered in This Course

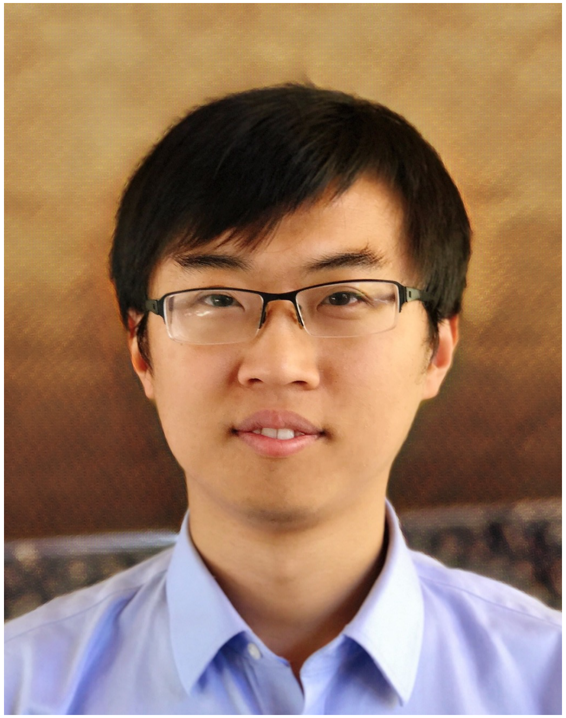
- Generalizability of RL



Course Logistic

Instructors

Instructor: Hao Su



TA: Minghua Liu

Teaching Goal

- Foundational
  - Programming problems ask you to **implement low-level modules from scratch**
- Hands-on
  - **Heavy** programming assignments to exercise what are taught in class

Pre-requisite: Technique

- **Skilled** in Linear Algebra, Multi-variable Calculus, and Deep Learning
- **Familiar** with Probability and Numerical Methods
- **Strong** programming skills
  - Familiar with Linux Toolchain
  - Familiar with python, numpy, and pytorch
- Course/project experiences in deep learning

Background Check

- On Piazza now (HW0)
  - Visible to enrolled and waitlist students
- 5 points in your final grade
- **Mandatory!** We will not grade your subsequent homeworks without seeing your HW0.
- If you are in the waitlist and intend to enroll, you need to submit HW0 by this deadline
- Due: 04/06/2021

Pre-requisite: Resources

- This course requires deep learning resources (to run reinforcement learning challenges)

- Unfortunately, we do not have computational resources to support ~50 students

- Please find the server with the following configuration:
  >= 50G disk space
  >= 1 GPU for deep learning

Assignments

- 4 assignments and 1 final project
  - HW0: due week 2 (5 points)
  - HW1: due week 4 (20 points)
  - HW2: due week 6 (20 points)
  - HW3: due week 8 (20 points)
  - Final project: final week (35 points)
  - No mid-term/final exams
- Extra credit for participation 5% (ask/answer questions in class, attend office hours)
- Late policy: 15% grade reduction for each 12 hours late. No acceptance 72 hours after the due time.

Assignments

- HW1-HW3: practice basic concepts and algorithms; build individual modules
- Final project: integrate modules and test new ideas. Score by performance ranking. Online evaluation system will be set up.
- We estimate **>=15 hrs per week** (out of class) solid time commitment
- We allow you to see homework (through Piazza) and attend the competition *even if you **audit** the course*

Course Resources

- Course website: https://haosulab.github.io/ml-for-robotics/SP21/index.html (Google “Hao Su” → Prof. Homepage → Teaching → this link)
  - Collaboration policy
  - Lecture slides
  - Office hour and location

- Piazza
  - Homework/Solution release
  - Discussions

Office Hour

- Time to be determined.

- Please fill in our Piazza survey for information about time zone and etc.

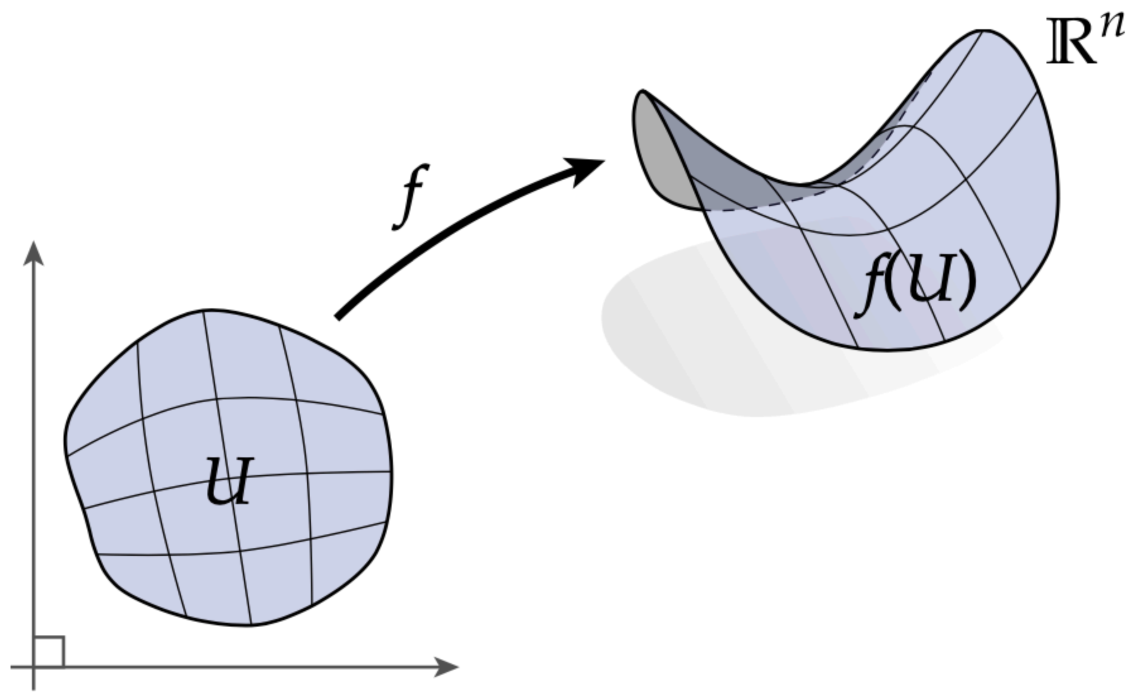
Questions?

Concepts of Differential Geometry

Parameterized Surfaces

A **parameterization** is a map from the domain U into $\mathbb{R}^n$

$$f : U \to \mathbb{R}^n$$



The set of points $f(U)$ is called the **image** of the parameterization.

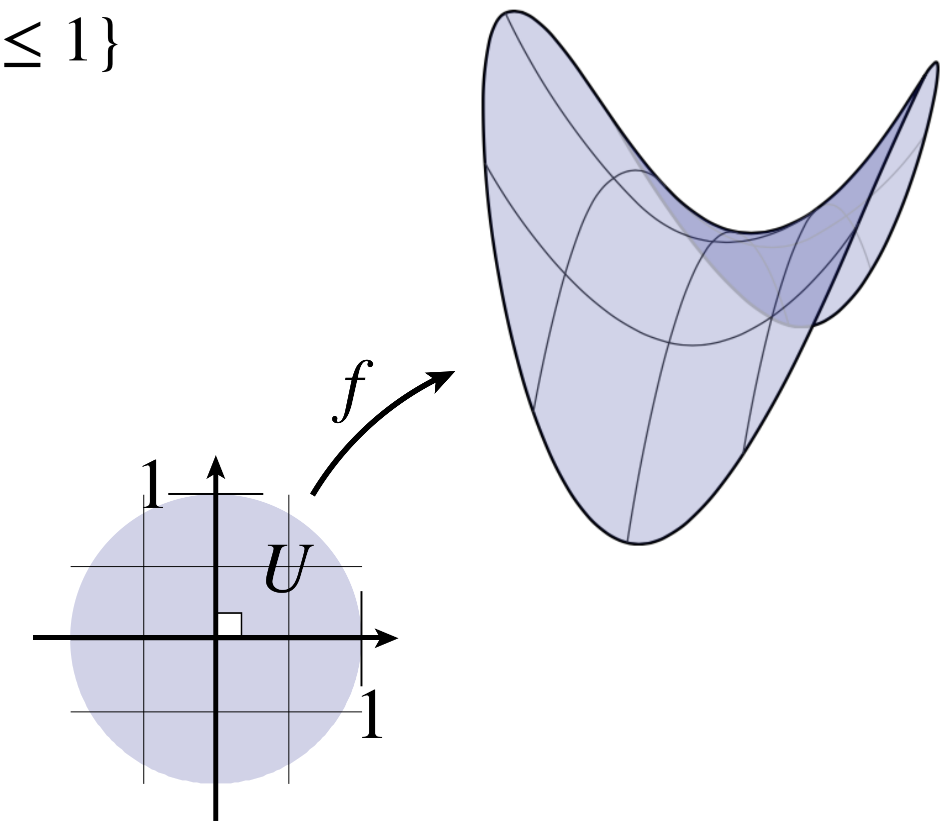
Image from Wikipedia

Example

- Example: We can express a *saddle as a parameterized* surface*:*

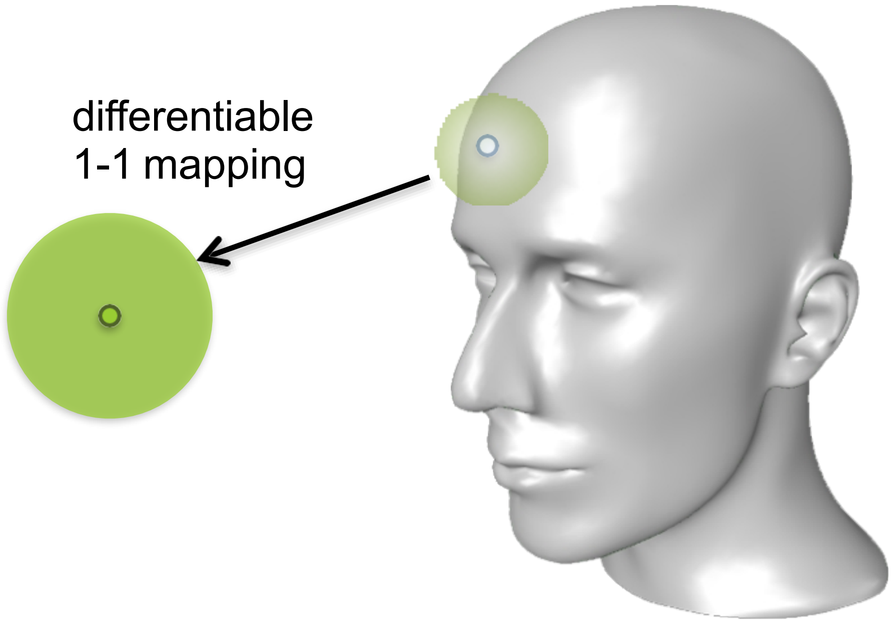
$$U := \{(u,v) \in \mathbb{R}^2 : u^2 + v^2 \leq 1\}$$

$$f(u,v) = [u, v, u^2 - v^2]^T$$



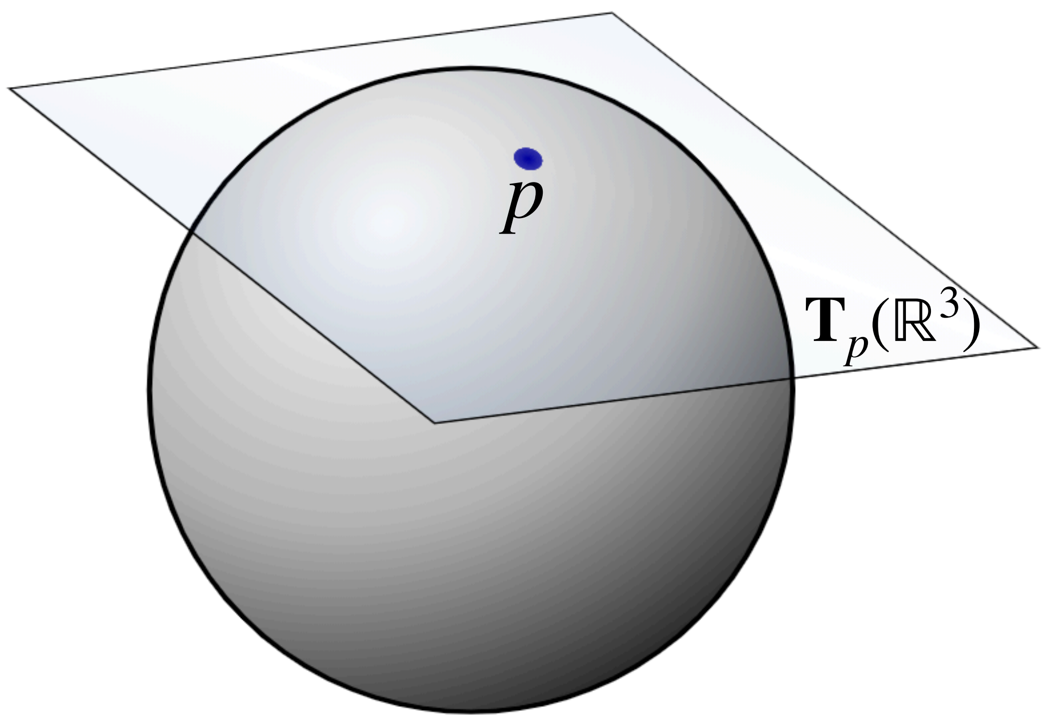
Manifold

- Things that can be discovered by local observation: point + neighborhood



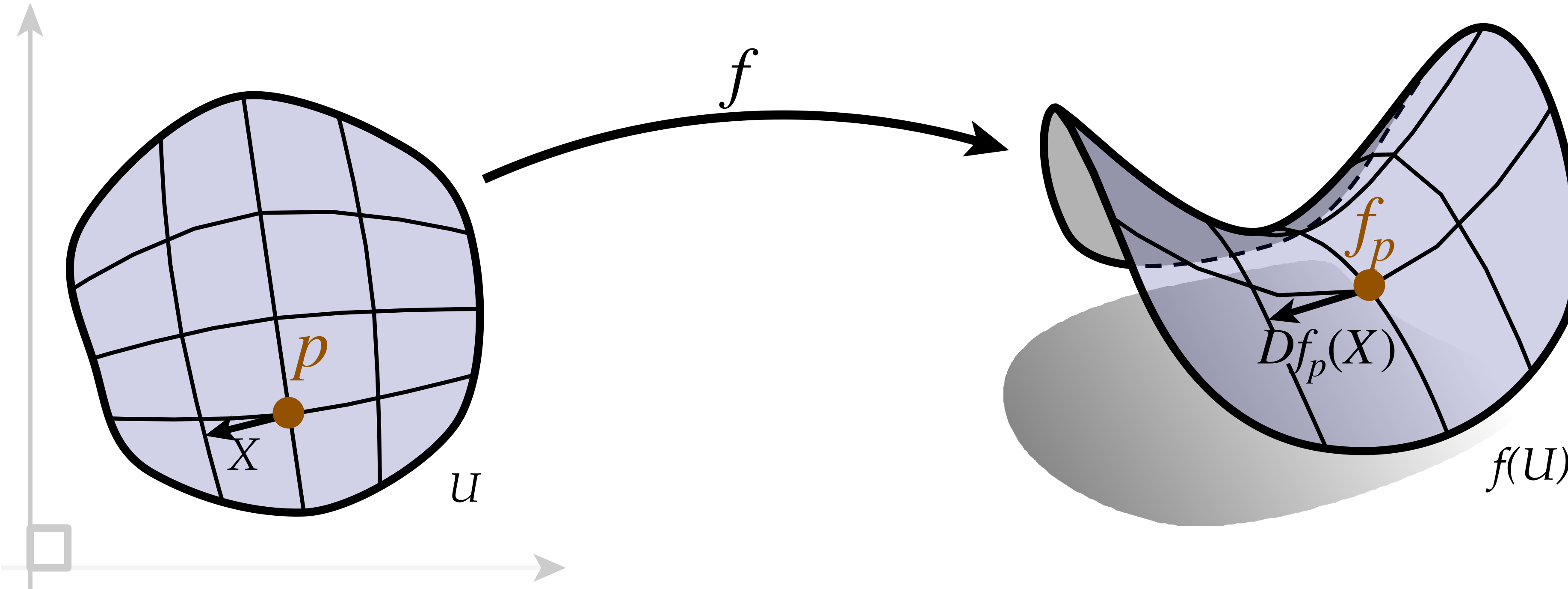
Tangent Plane

- One can attach to every point p a tangent plane $\mathbf{T}_p$
- Intuitively, it contains the possible directions in which one can tangentially pass through p.

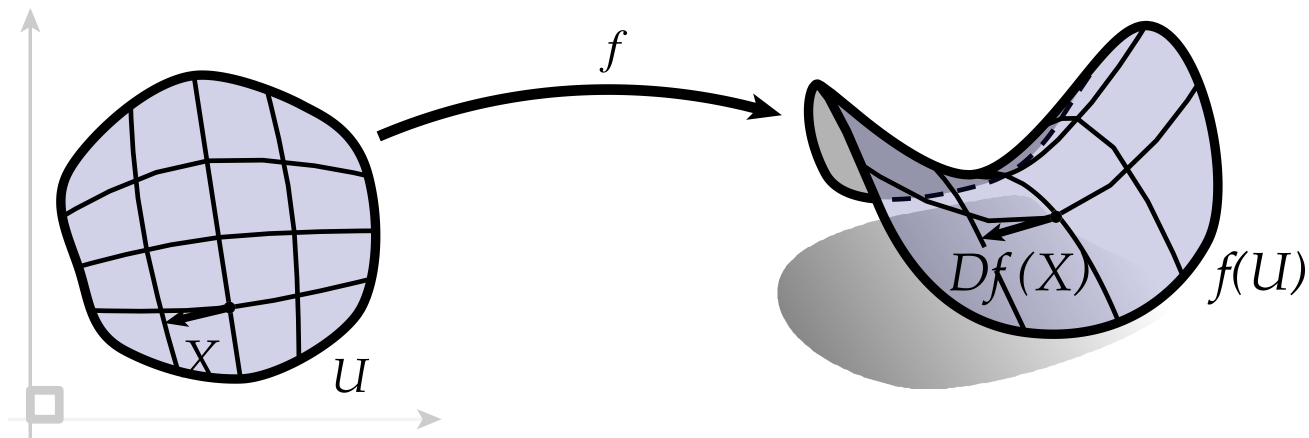


Differential of a Surface

- Relate the movement of point in the domain and on the image



Differential of a Surface

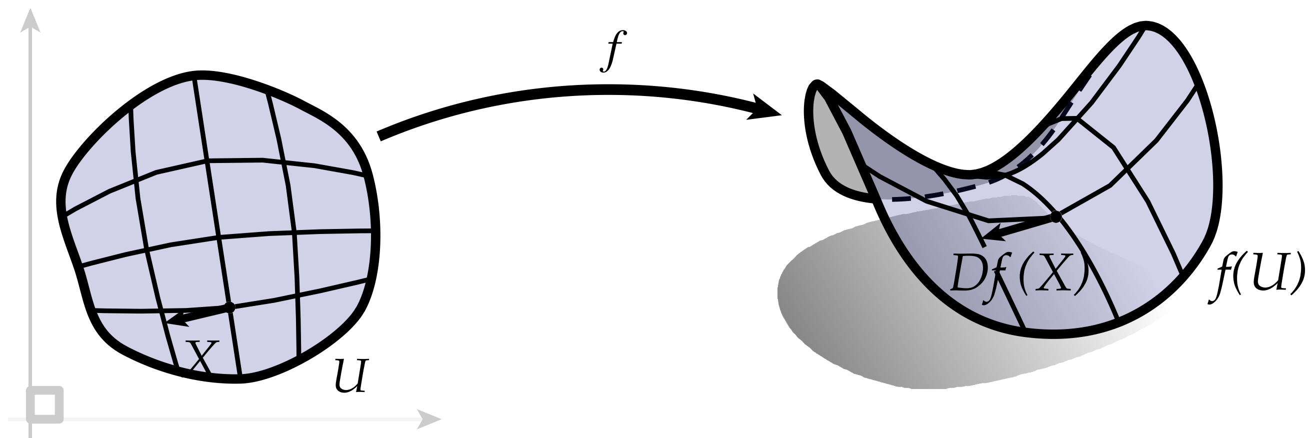


Total differential: $df = \frac{\partial f}{\partial u} du + \frac{\partial f}{\partial v} dv \Longrightarrow \Delta f \approx \frac{\partial f}{\partial u} \Delta u + \frac{\partial f}{\partial v} \Delta v$

If point $p \in \mathbb{R}^2$ moves along vector $X = [u, v]^T$ by ϵ, the movement of f_p is:

$$\Delta f_p \approx \frac{\partial f}{\partial u}(\epsilon u) + \frac{\partial f}{\partial v}(\epsilon v) = \epsilon \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v}\right] \begin{bmatrix} u \\ v \end{bmatrix}$$

Differential of a Surface



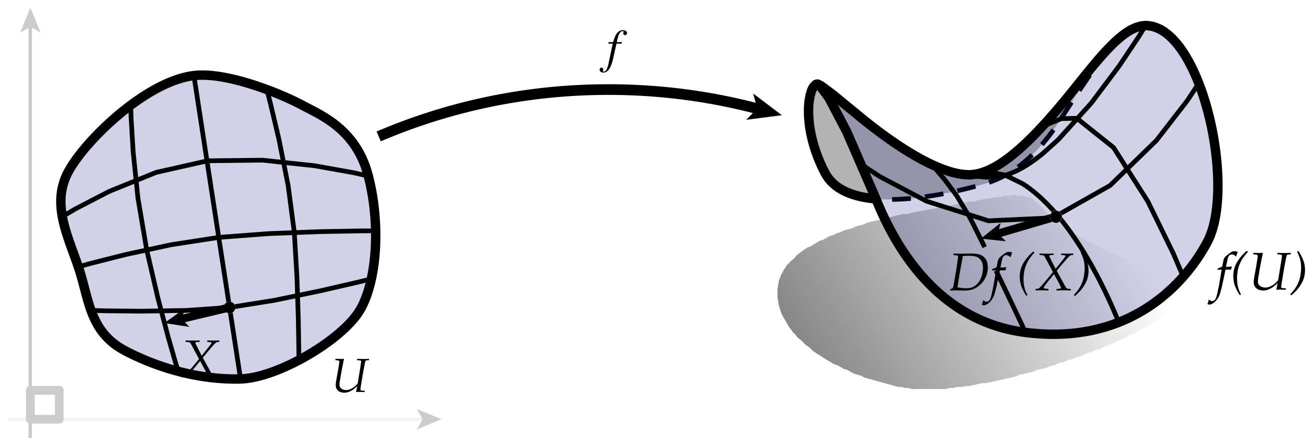
Total differential: $df = \frac{\partial f}{\partial u} du + \frac{\partial f}{\partial v} dv$

If point $p \in \mathbb{R}^2$ moves with velocity $X = [u, v]^T$ by ϵ, the movement of f_p is:

$$\Delta f_p \approx \frac{\partial f}{\partial u}(\epsilon u) + \frac{\partial f}{\partial v}(\epsilon v) = \epsilon \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v} \right] \begin{bmatrix} u \\ v \end{bmatrix}$$

$$Df_p := \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v} \right] \in \mathbb{R}^{3 \times 2}$$

Differential of a Surface



Total differential: $df = \frac{\partial f}{\partial u} du + \frac{\partial f}{\partial v} dv$

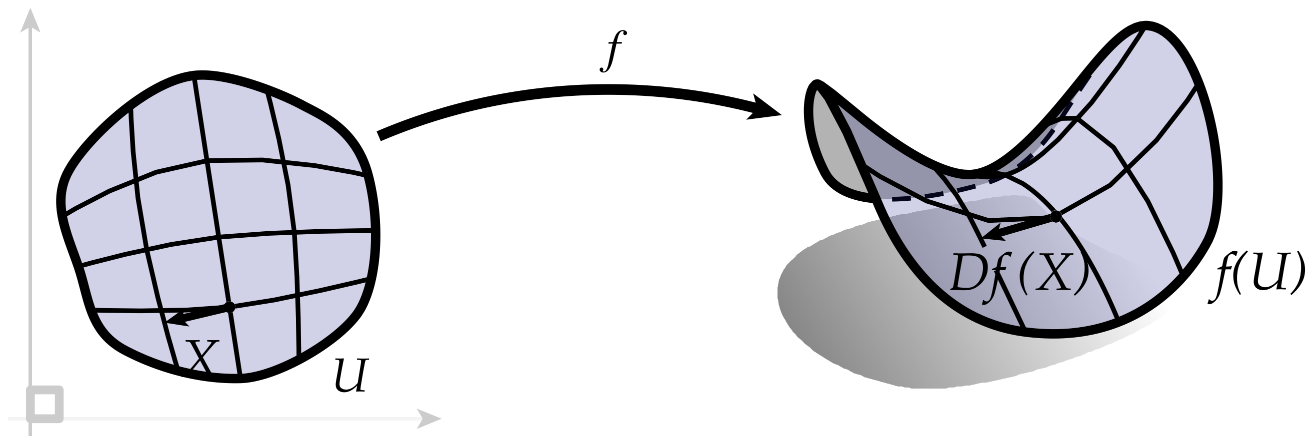
If point $p \in \mathbb{R}^2$ moves with velocity $X = [u, v]^T$ by ϵ, the movement of f_p is:

$$\Delta f_p \approx \frac{\partial f}{\partial u}(\epsilon u) + \frac{\partial f}{\partial v}(\epsilon v) = \epsilon \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v} \right] \begin{bmatrix} u \\ v \end{bmatrix} = \epsilon [Df_p] X$$

$$Df_p := \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v} \right] \in \mathbb{R}^{3\times 2}$$

Df_p: differential (Jacobian), a linear map.

Differential of a Surface



Total differential: $df = \frac{\partial f}{\partial u} du + \frac{\partial f}{\partial v} dv$

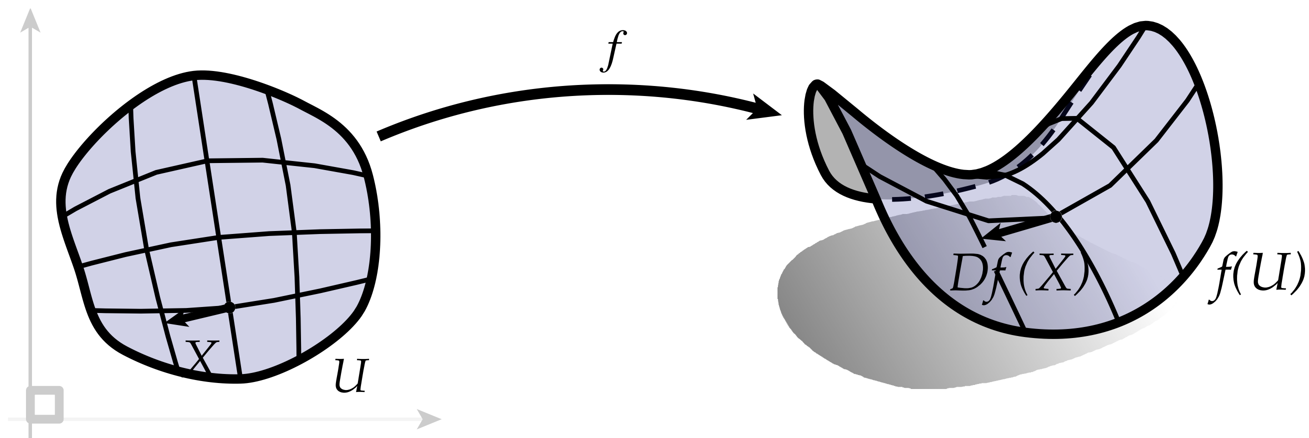
If point $p \in \mathbb{R}^2$ moves with velocity $X = [u, v]^T$ by ϵ, the movement of f_p is:

$$\Delta f_p \approx \frac{\partial f}{\partial u}(\epsilon u) + \frac{\partial f}{\partial v}(\epsilon v) = \epsilon \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v} \right] \begin{bmatrix} u \\ v \end{bmatrix} = \epsilon [Df_p] \boxed{X}$$

$$Df_p := \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v} \right] \in \mathbb{R}^{3 \times 2}$$

velocity in the 2D domain

Differential of a Surface



Total differential: $df = \frac{\partial f}{\partial u}du + \frac{\partial f}{\partial v}dv$

If point $p \in \mathbb{R}^2$ moves with velocity $X = [u, v]^T$ by ϵ, the movement of f_p is:

velocity in 3D space

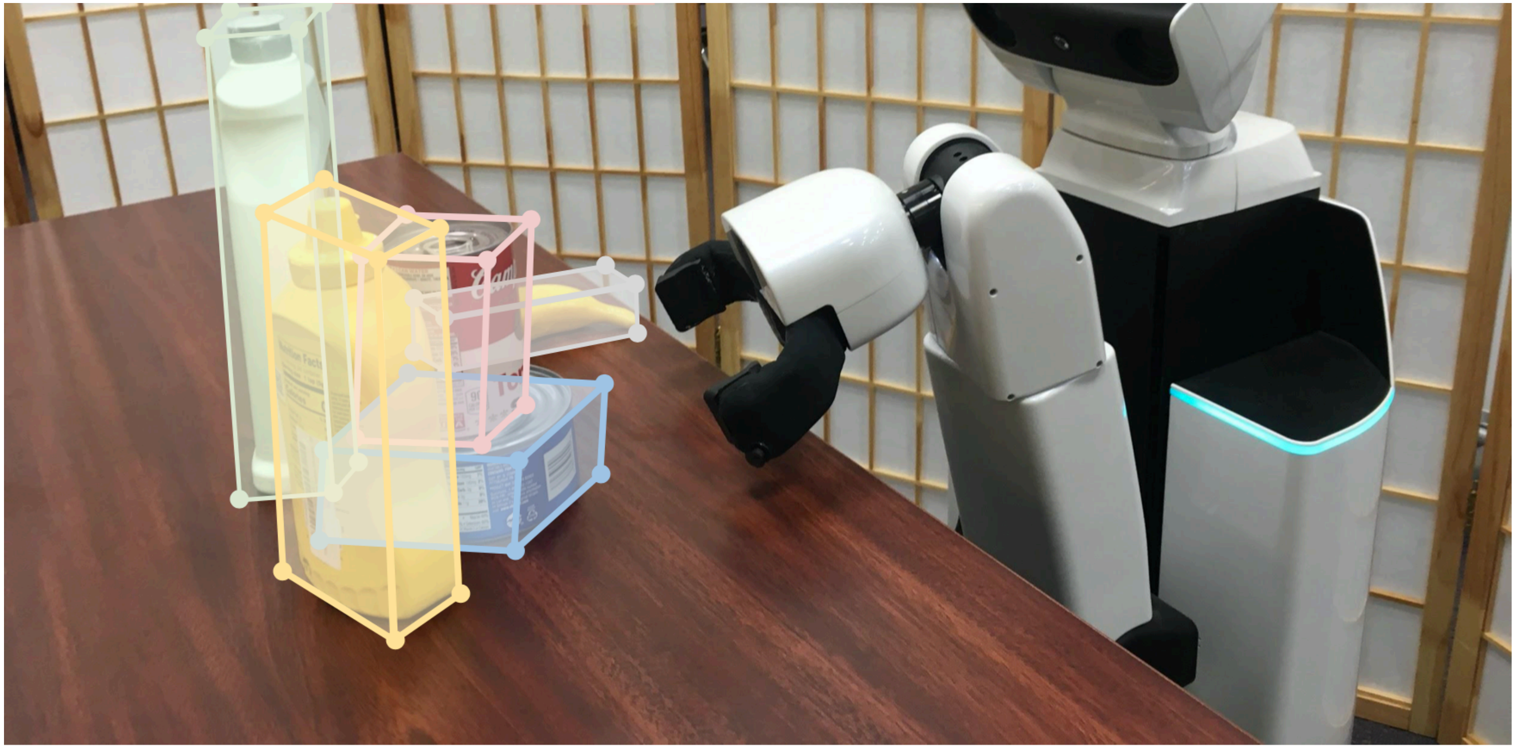
$$\Delta f_p \approx \frac{\partial f}{\partial u}(\epsilon u) + \frac{\partial f}{\partial v}(\epsilon v) = \epsilon \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v}\right] \begin{bmatrix} u \\ v \end{bmatrix} = \epsilon \boxed{[Df_p]\boxed{X}}$$

$$Df_p := \left[\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v}\right] \in \mathbb{R}^{3\times 2}$$

velocity in 2D domain

Rigid Transformation

Describe the Pose of Agent and Objects

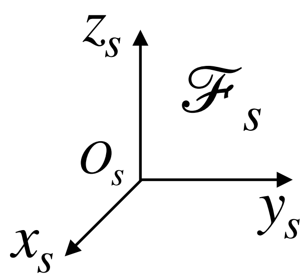


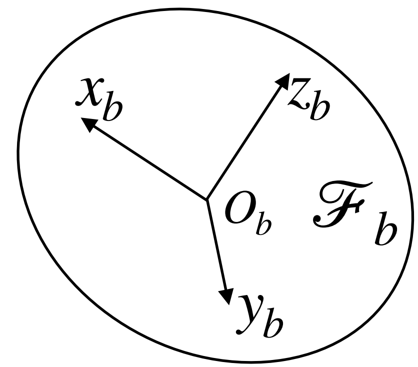
Wang et al., "**DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion**", *CVPR 2019*

Describe the Pose of Agent and Objects

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

Rigid Transformation

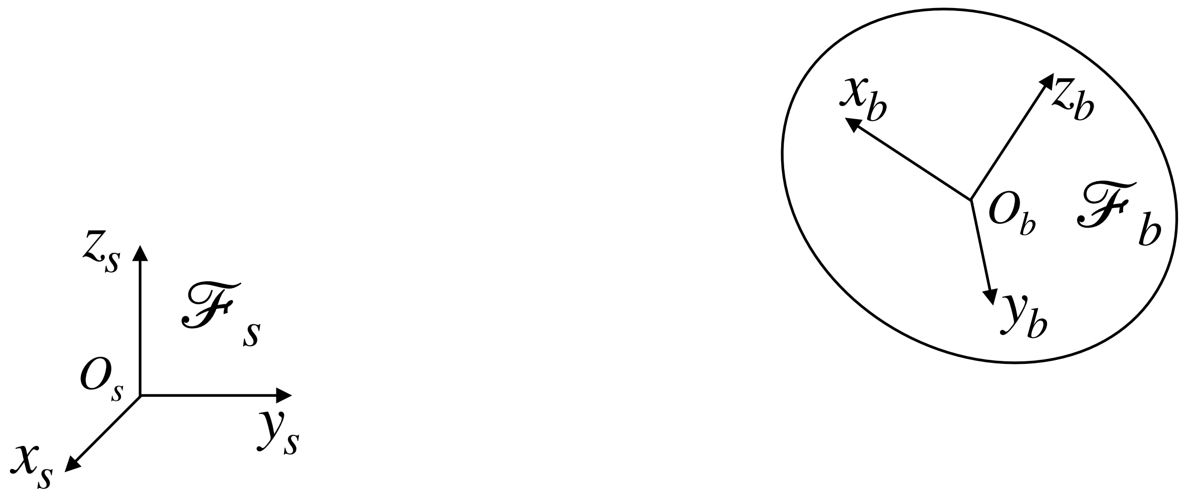




- An observer **records** the position of any point in the space **using a frame** $\mathscr{F}_s$ (space frame)

- There is a rigid object, to which we bind a frame $\mathscr{F}_b$ (body frame) tightly, so that $\mathscr{F}_b$ moves along with the object

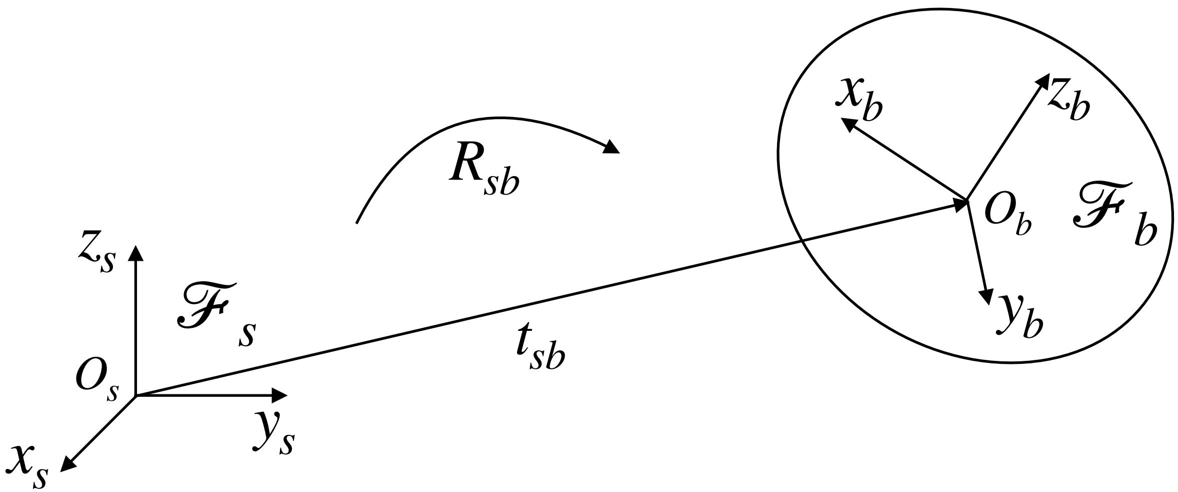
Rigid Transformation



- When talking about the pose of the *rigid* object, we ask:

How to **transform** $\mathscr{F}_s$ so that it overlaps with $\mathscr{F}_b$?

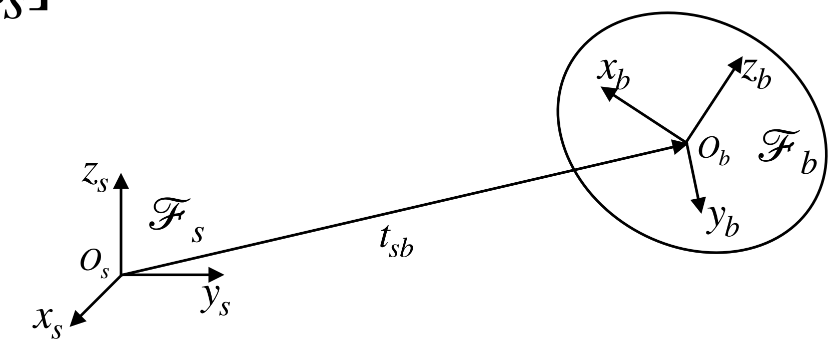
Rigid Transformation



- We first translate $\mathscr{F}_s$ by t_{sb} to align O_s and O_b
- And then rotate by R_{sb} to align $\{x_i, y_i, z_i\}$

Rigid Transformation

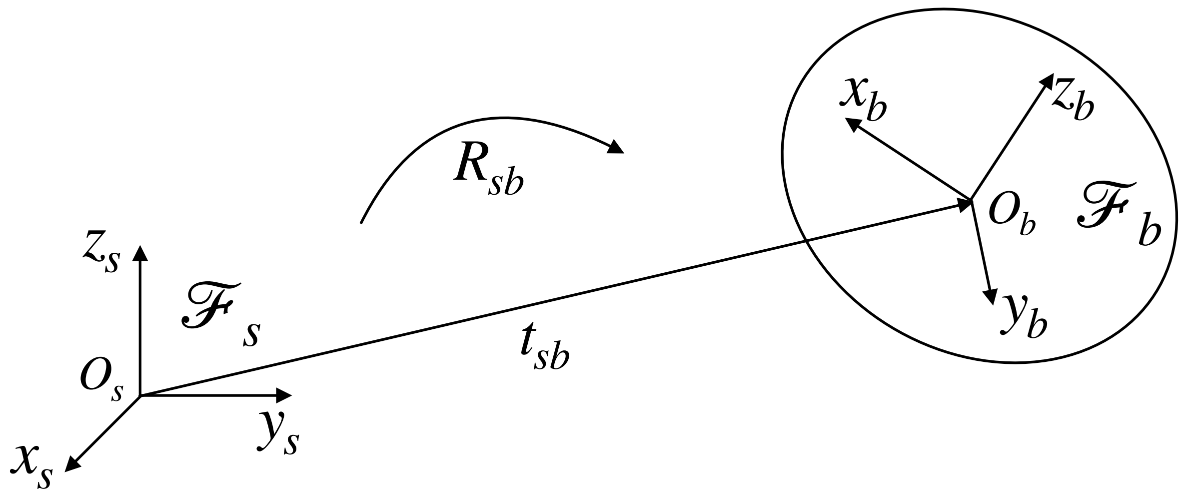
- Formally,
  - $O_b = O_s + t_{sb}$
  - $[x_b, y_b, z_b] = R_{sb}[x_s, y_s, z_s]$



- Since the observer records everything using $\mathscr{F}_s$,
  - $O_s = 0$
  - $[x_s, y_s, z_s] = I_{3\times 3}$

- Therefore,
  - $t_{sb} = O_b$
  - $R_{sb} = [x_b, y_b, z_b] \in \mathbb{R}^{3\times 3}$

Effect of (R_{sb}, t_{sb}) on Frames



- First of all, (R_{sb}, t_{sb}) transforms any point in the *whole space*:

$$x' = R_{sb}x + t_{sb}$$

Effect of (R_{sb}, t_{sb}) on Frames

- Suppose $\mathscr{F}_p = \{O_p, (x_p, y_p, z_p)\}$ is a frame at an arbitrary point O_p
- **Then, the new origin is:** $O_p' = ?$

Effect of (R_{sb}, t_{sb}) on Frames

- Suppose $\mathscr{F}_p = \{O_p, (x_p, y_p, z_p)\}$ is a frame at an arbitrary point O_p
- **Then, the new origin is:** $O_p' = R_{sb}O_p + t_{sb}$

Effect of (R_{sb}, t_{sb}) on Frames

- Suppose $\mathcal{F}_p = \{O_p, (x_p, y_p, z_p)\}$ is a frame at an arbitrary point O_p
- **Then, the new origin is:** $O_p' = R_{sb}O_p + t_{sb}$
- How about the frame?
  - Assume three curves, $\gamma_x, \gamma_y, \gamma_z$, passing O_p at $t = 0$ with tangents x_p, y_p, z_p

Effect of (R_{sb}, t_{sb}) on Frames

- Suppose $\mathscr{F}_p = \{O_p, (x_p, y_p, z_p)\}$ is a frame at an arbitrary point O_p
- **Then, the new origin is:** $O'_p = R_{sb}O_p + t_{sb}$
- How about the frame?
  - Assume three curves, $\gamma_x, \gamma_y, \gamma_z$, passing O_p at $t = 0$ with tangents x_p, y_p, z_p
  - Then, the new tangents after transformation are:

$$\frac{d}{dt}R_{sb}\gamma_x(0),\ \frac{d}{dt}R_{sb}\gamma_y(0),\ \frac{d}{dt}R_{sb}\gamma_z(0)$$

Effect of (R_{sb}, t_{sb}) on Frames

- Suppose $\mathscr{F}_p = \{O_p, (x_p, y_p, z_p)\}$ is a frame at an arbitrary point O_p
- **Then, the new origin is:** $O_p' = R_{sb}O_p + t_{sb}$
- How about the frame?
  - Assume three curves, $\gamma_x, \gamma_y, \gamma_z$, passing O_p at $t = 0$ with tangents x_p, y_p, z_p
  - Then, the new tangents after transformation are:

$$\frac{d}{dt}R_{sb}\gamma_x(0) = R_{sb}x_p,\ \frac{d}{dt}R_{sb}\gamma_y(0) = R_{sb}y_p,\ \frac{d}{dt}R_{sb}\gamma_z(0) = R_{sb}z_p$$

- **So the new frame is:** $\mathscr{F}_{p'} = \{O_p', R_{sb}[x_p, y_p, z_p]\}$

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

Rotation in $\mathbb{R}^3$

3 Degree of Freedoms

$\mathbb{SO}(3)$: 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

Homogenous Coordinates

- Homogeneous coordinate for 3D Space:

$$\tilde{x} := \begin{bmatrix} x \\ 1 \end{bmatrix} \in \mathbb{R}^4$$

- Homogeneous transformation matrix:

$$T_{sb} = \begin{bmatrix} R_{sb} & t_{sb} \\ 0 & 1 \end{bmatrix}$$

- Rigid transformation under linear form:

$$\tilde{x}' = T_{sb}\tilde{x}$$

- Composition: $T_{ac} = T_{ab}T_{bc}$

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

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

- "Group": roughly, closed under matrix multiplication
- "Euclidean": R and t
- "Special": $\det(R) = 1$

- 6 DoF

Multi-Link Rigid-Body Geometry

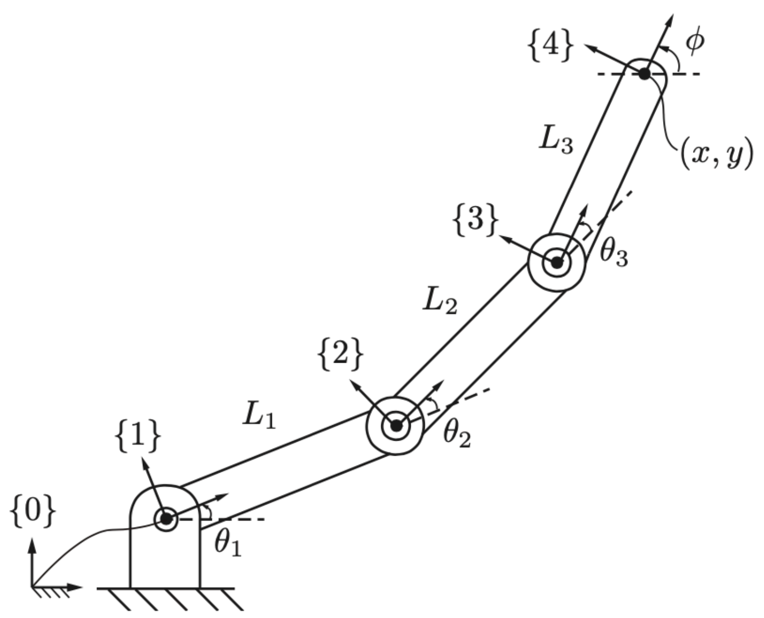
Link and Joint

Link:

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

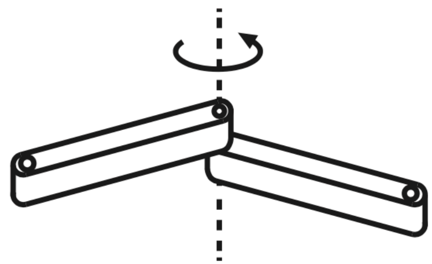
Joint:

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



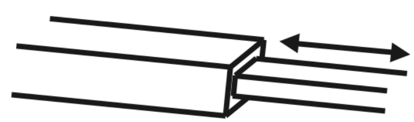
Two Common Joint Types

- 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:** Each coordinate is a vector of joint poses (**angles** around **joint axis**)
  - **Cartesian space:** Each coordinate the rigid transformation of the end-effector (by (R, t))

Kinematics Equations

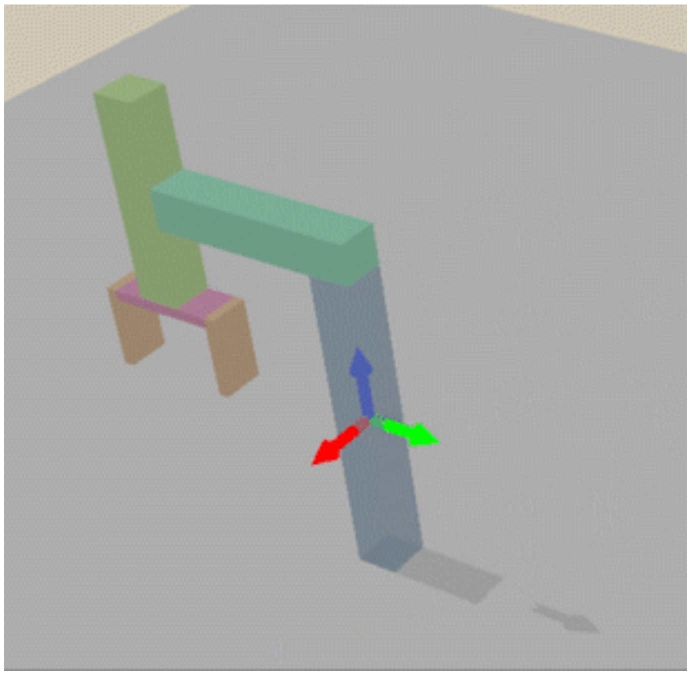
- Map the joint space coordinate $\theta \in \mathbb{R}^n$ to the Cartesian space coordinate $x \in \mathbb{R}^m$:

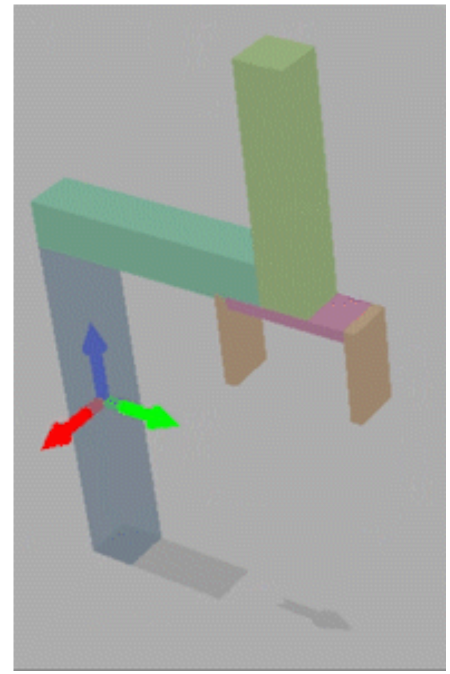
$$x = f(\theta)$$

- Calculated by composing transformations along the kinematic chain

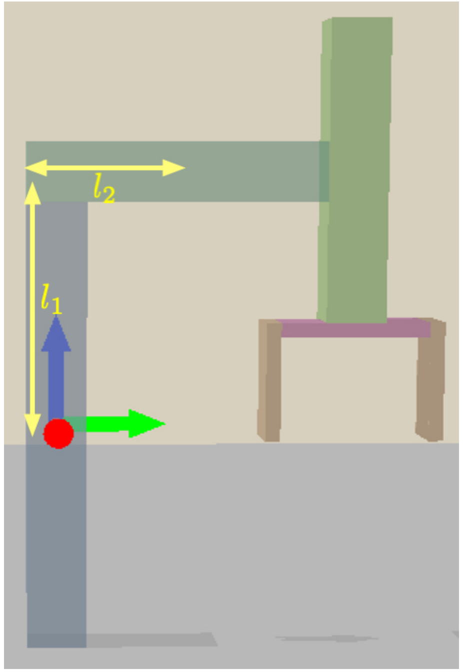
Example

A simple 2 DoF robot arm



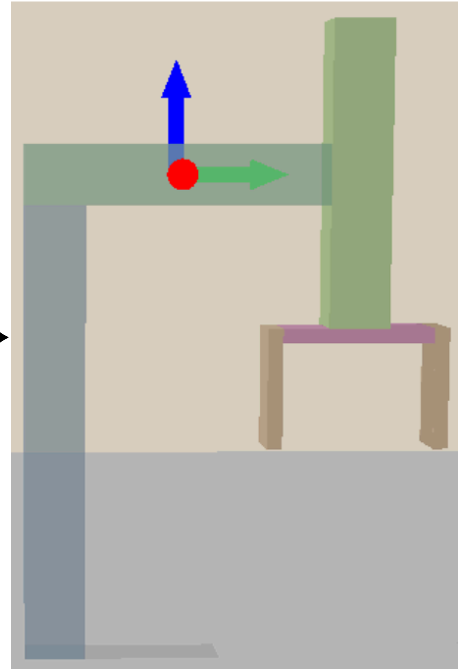


revolute(θ_1) prismatic(θ_2)

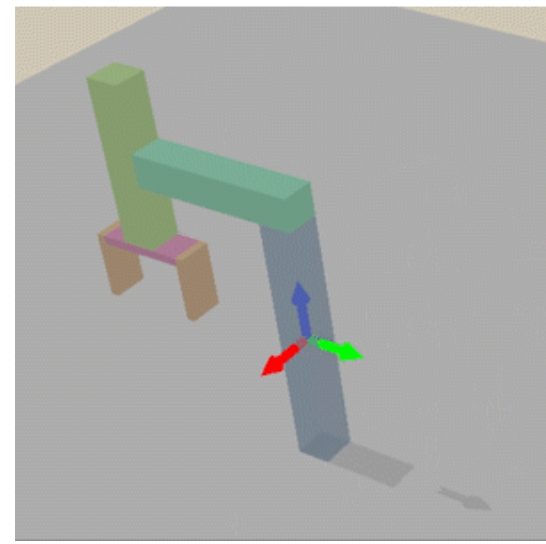


base

$\rightarrow$

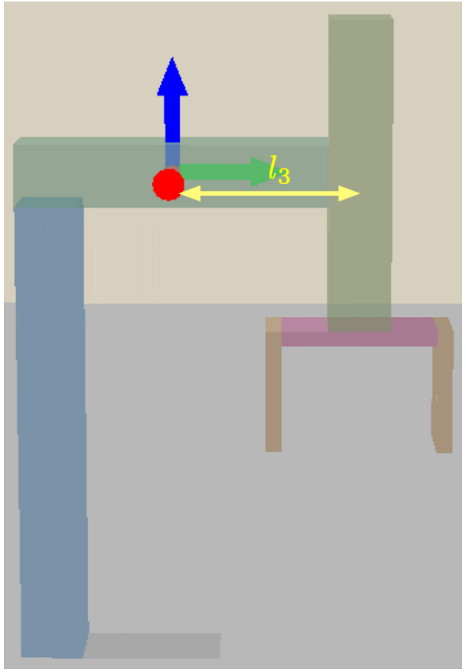


link1

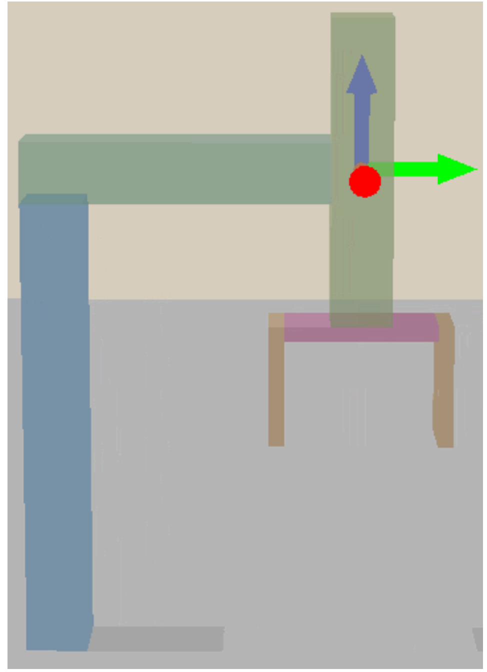


revolute(θ_1)

$$g_{01} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & -l_2\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & l_2\cos\theta_1 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

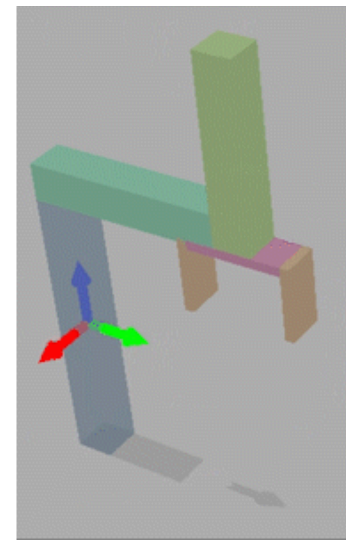


→



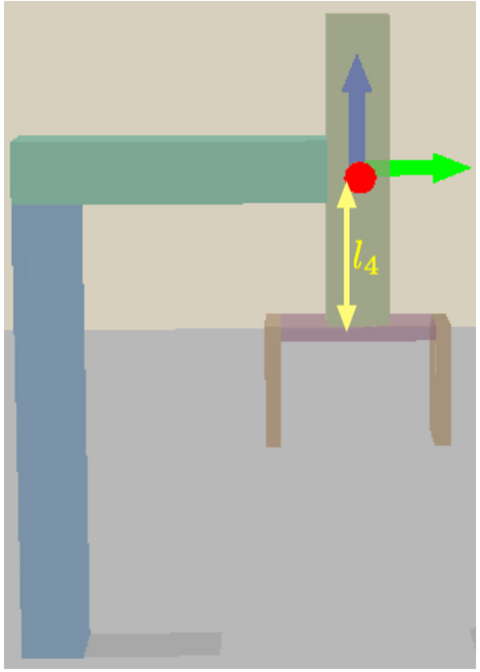
link1

link2



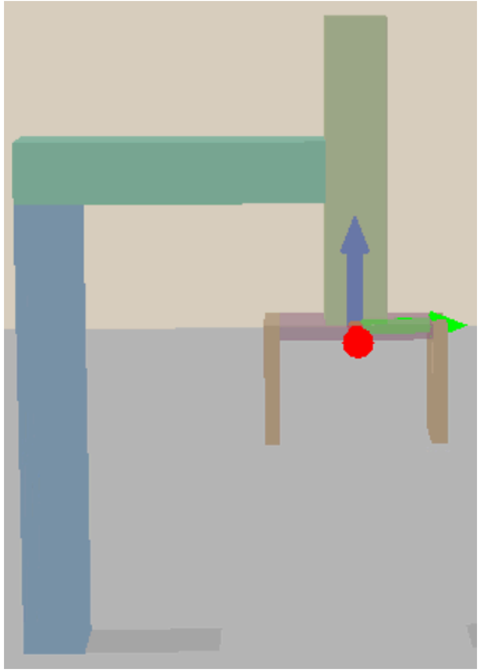
prismatic(θ_2)

$$g_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_3 \\ 0 & 0 & 1 & \theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



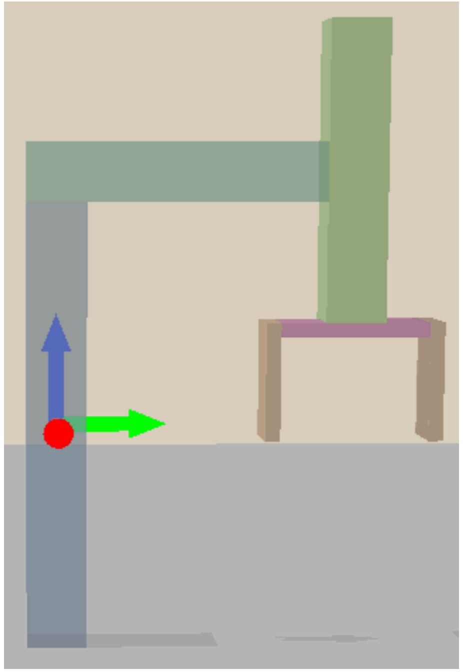
link2

$\rightarrow$



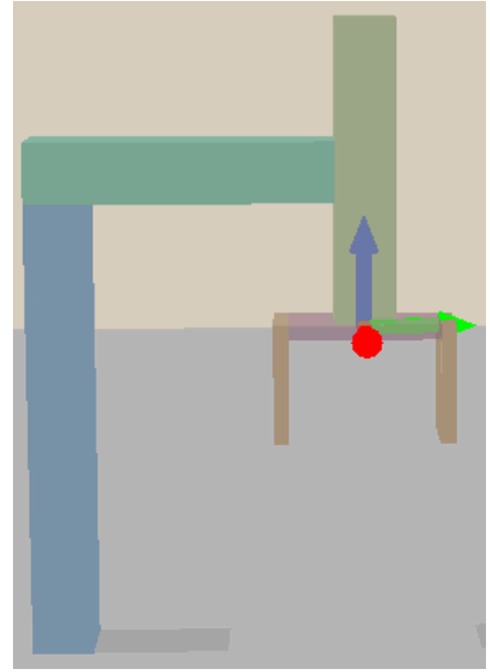
end_effector

$$g_{23} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



base

→



end_effector

$$g_{03} = g_{01}g_{12}g_{23} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & -\sin\theta_1(l_2 + l_3) \\ \sin\theta_1 & \cos\theta_1 & 0 & \cos\theta_1(l_2 + l_3) \\ 0 & 0 & 1 & l_1 - l_4 + \theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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