Robotic Systems I

Lecture 1: Introduction, Rigid Body Motion, Jacobians, Forward and Inverse Kinematics

Konstantinos Chatzilygeroudis - costashatz@upatras.gr

Department of Electrical and Computer Engineering
University of Patras

Template made by Panagiotis Papagiannopoulos

Logistics

Laboratory of Automation & Robotics

- **Lectures**:
  - 10 + 1 Recitation Lecture
  - ⩾8 Lab Exercises
- **Examination**:
  - Lab code **(20% of total grade)**
  - Intermediate exams **(30% of total grade)**
  - For **50% of total grade** you select **one of the two**:
    - Team Project (max 2 per team) — only for the June exams
    - Final exam
  - **IMPORTANT:** All exams are hybrid: multiple choice + code
- **Office Hours**:
  - **Tue-Wed** *(10:00-12:00)*
  - 24/7 by email (`costashatz@upatras.gr`, subject: *ECE_RSI_AM*)
- **Material and Announcements**

What is this course about?

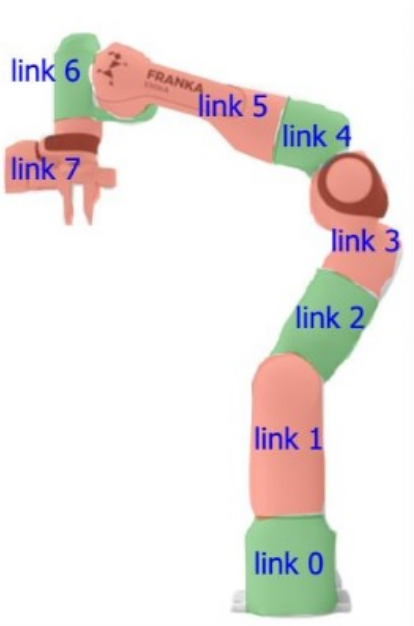
- 3D Rigid Body Motions
- Dynamics and Control of Manipulators
- Mobile Robots, Simulators
- Orientation Representations
- Lie Algebra
- Splines
- Optimization
- Localization, SLAM
- Optimization-based Control of Complex Robots/Whole Body Controllers
- **We focus on implementation/robotics applications. You are required to write code!** Not a theory course!

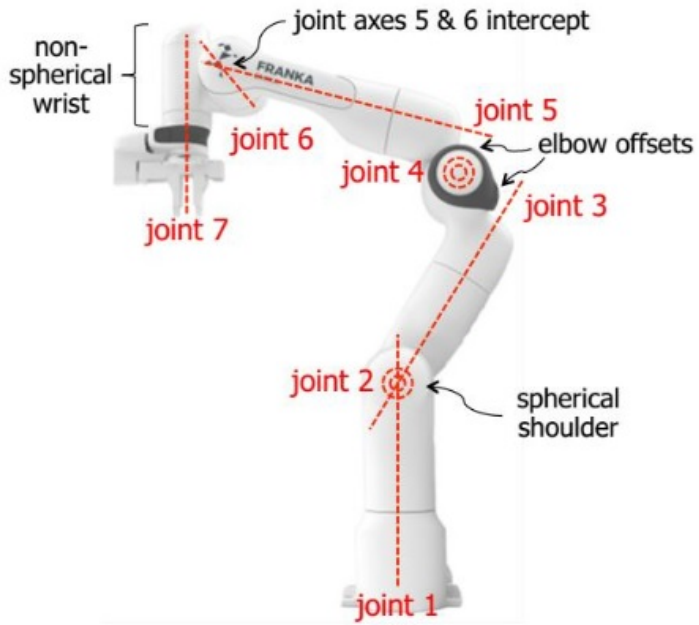
Lectures/Lab Format

- **3-hour lectures:**
  - A lot of theory! Hints for personal reading!
  - Live code examples + analyses!
  - Ask questions please!
- **2-hour lab exercises:**
  - **20% of total grade**
  - Small code examples that you need to fill
  - You need to deliver code
  - Ask questions, experiment!

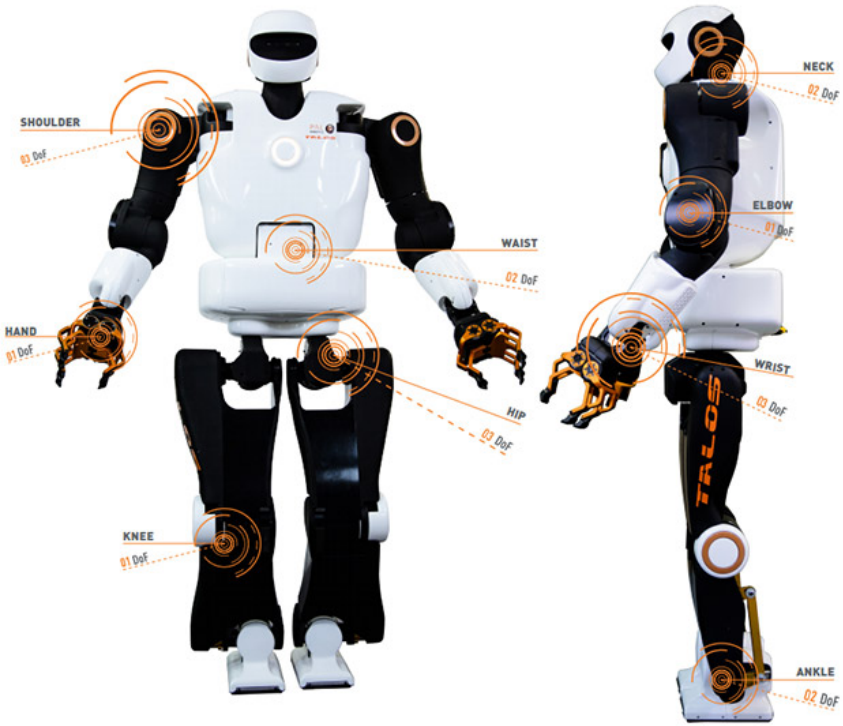
What is a robot?

A robot is a mechanical system that consists of bodies (**links**) which are connected with various types of **joints**. **Motors** or actuators exert **torque** or **force** to the bodies, and thus we get body movement. Usually one motor corresponds to one joint.





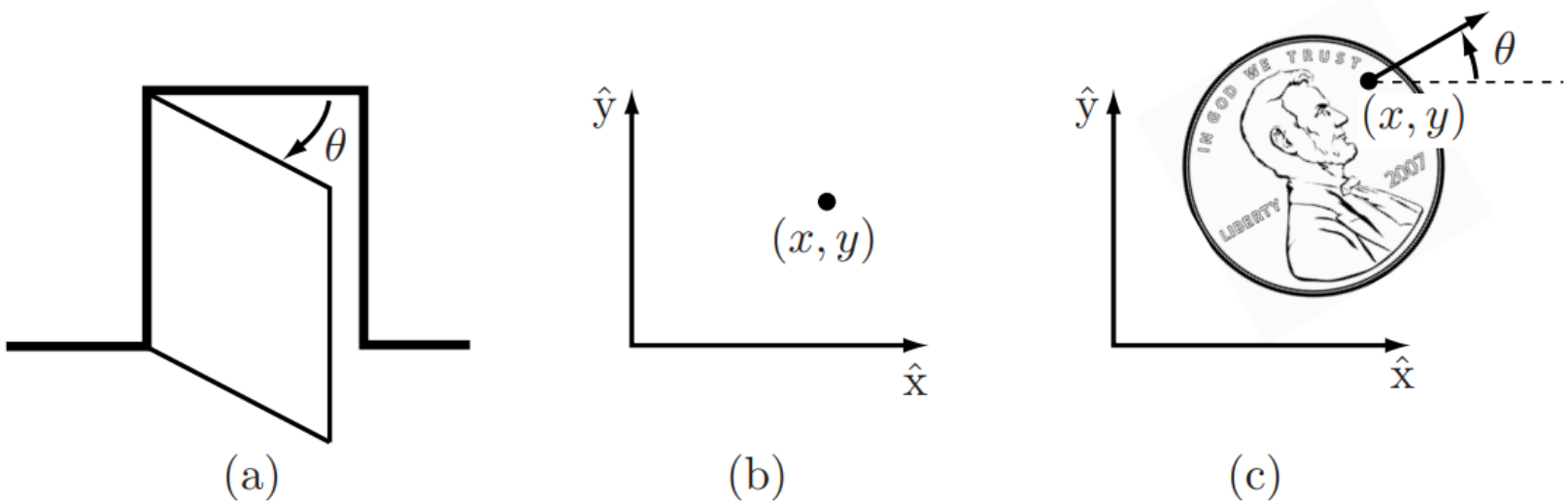
Robot Examples (1)



Robot Examples (2)

Robot Configuration

Configuration of a robot is a description that fully defines the positions (in 3D space) of all points of the robot.



Source: Modern Robotics: Mechanics, Planning, and Control, *Kevin M. Lynch and Frank C. Park*, 2017, Cambridge University Press.

Degrees of freedom

Degrees of freedom (dofs) of a robot is the minimum number of parameters that we need to represent its configuration.

For example:

- The door has 1 dof
- A 2D point has 2 dofs
- A coin moving on a 2D plane, has 3 dofs
- ...

Degrees of freedom of Rigid Bodies

In this course, we will handle only rigid bodies and robots that consist of rigid bodies only.

A rigid body:

- Moving on a 2D plane, has 3 dofs: (x, y, θ)
- Moving in 3D space, has 6 dofs:

$$(x, y, z, \theta_x, \theta_y, \theta_z)$$

- This holds only when there are no constraints in the movement! If we have constraints, then we can find the dofs as follows:

DoFs = (sum of all dofs of all bodies) - (number of independent constraints)

DoFs of a robot

- A robot consists of multiple bodies that are connected via joints
- Every joint **imposes one or more constraints**
- We can also say that each joint **allows for one or more degrees of freedom**
- We assume that **each joint connects exactly two bodies**

Typical Joints (1)

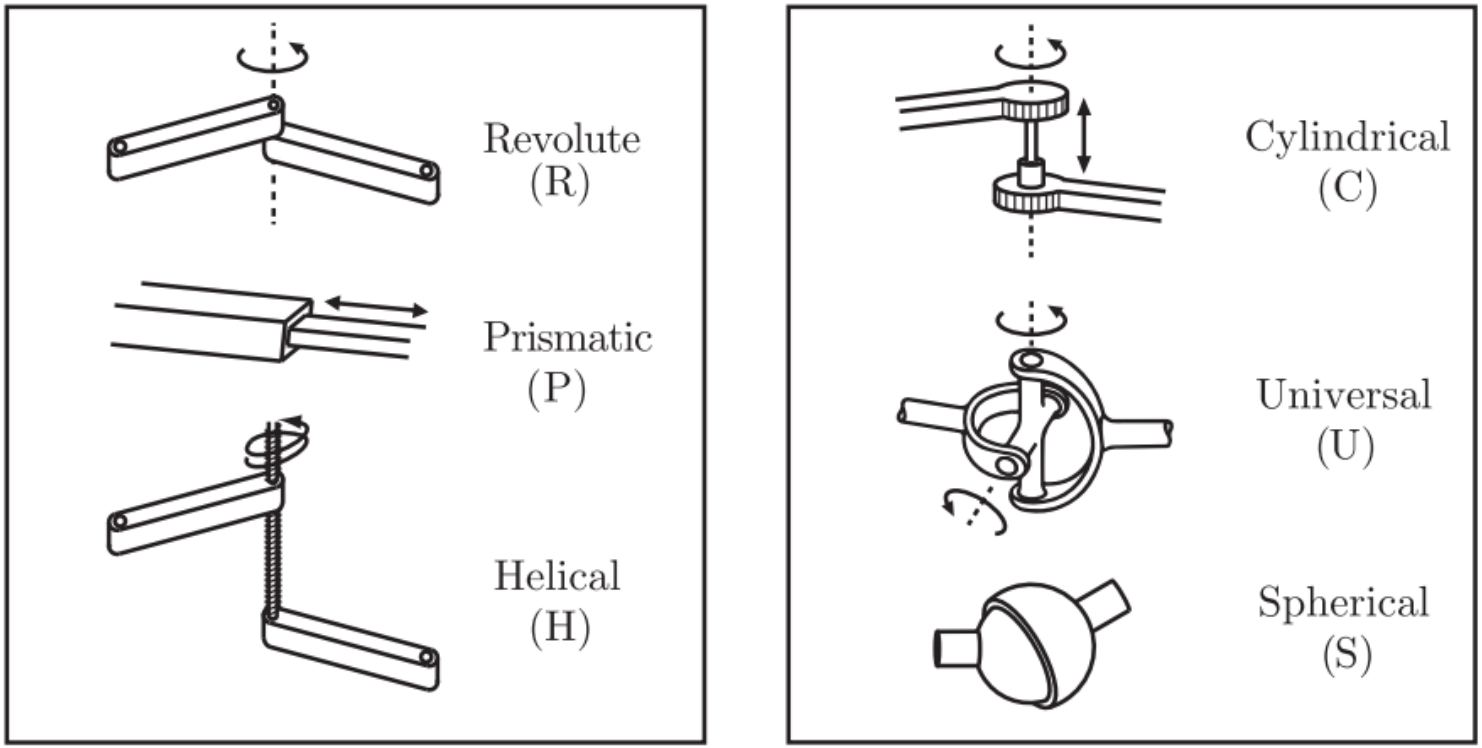


Figure: Typical joint types

Source: Modern Robotics: Mechanics, Planning, and Control, *Kevin M. Lynch and Frank C. Park*, 2017, Cambridge University Press.

Typical Joints (2)

- **Joints of 1 dof**:
  - **Revolute/Hinge**: allows rotational movement around an axis
  - **Prismatic**: allows translational movement along an axis
  - **Helical/Screw**: allows both rotational and translational movement around a helical axis (like a screw)
- **Joints of multiple dofs**:
  - **Cylindrical**: allows for independent rotational and translational movement around an axis (2 dofs)
  - **Universal**: basically 2 revolute joints with perpendicular axes (2 dofs)
  - **Spherical/Ball-and-Socket**: it has 3 dofs and behaves like our shoulder joint

Transformations

- A rigid body in 3D space needs at least six values to completely determine its pose[1]
- But where is the body's pose? With respect to what?

[1] We need at least 3 for bodies in 2D space.

Transformations

- A rigid body in 3D space needs at least six values to completely determine its pose[1]
- But where is the body's pose? With respect to what? We need a *reference frame*
- We always assume a static reference frame that does not change or move at all: we usually refer to it as *"world frame"*
- Each body has its own local static reference frame (It can have more than one!): we usually refer to it as *"body frame"*

[1] We need at least 3 for bodies in 2D space.

What is orientation?

The transformation that transforms a vector/point, $\boldsymbol{v}_b$, expressed in "body frame" to the same vector/point, but expressed in the "fixed/world frame" orientation, $\boldsymbol{v}_w$.

$$\boldsymbol{v}_w = \boldsymbol{R}_{wb}\boldsymbol{v}_b$$

where $\boldsymbol{R_{wb}} \in \mathcal{SO}(3)$ is a **rotation matrix** that rotates the "fixed/world frame" such that it matches the "body frame". We use this rotation (transformation) matrix to rotate a vector from "body frame" to the "world frame".

Frames and Rotation Matrices

A frame is basically a set of 3 unit-norm vectors, that are orthogonal to each other:

$$\boldsymbol{w}_x \cdot \boldsymbol{w}_y = 0$$
$$\boldsymbol{w}_x \cdot \boldsymbol{w}_z = 0$$
$$\boldsymbol{w}_y \cdot \boldsymbol{w}_z = 0$$

where $\mathcal{W} = (\boldsymbol{w}_x, \boldsymbol{w}_y, \boldsymbol{w}_z)$ is the world frame. If we also define a body frame as $\mathcal{B}$, then we have:

$$\boldsymbol{R}_{wb} = \begin{bmatrix} (\boldsymbol{w}_x \cdot \boldsymbol{b}_x) & (\boldsymbol{w}_x \cdot \boldsymbol{b}_y) & (\boldsymbol{w}_x \cdot \boldsymbol{b}_z) \\ (\boldsymbol{w}_y \cdot \boldsymbol{b}_x) & (\boldsymbol{w}_y \cdot \boldsymbol{b}_y) & (\boldsymbol{w}_y \cdot \boldsymbol{b}_z) \\ (\boldsymbol{w}_z \cdot \boldsymbol{b}_x) & (\boldsymbol{w}_z \cdot \boldsymbol{b}_y) & (\boldsymbol{w}_z \cdot \boldsymbol{b}_z) \end{bmatrix}$$

Frames and Rotation Matrices

A frame is basically a set of 3 unit-norm vectors, that are orthogonal to each other:

$$\boldsymbol{w}_x \cdot \boldsymbol{w}_y = 0$$
$$\boldsymbol{w}_x \cdot \boldsymbol{w}_z = 0$$
$$\boldsymbol{w}_y \cdot \boldsymbol{w}_z = 0$$

where $\mathcal{W} = (\boldsymbol{w}_x, \boldsymbol{w}_y, \boldsymbol{w}_z)$ is the world frame. If we also define a body frame as $\mathcal{B}$, then we have:

$$\boldsymbol{R}_{wb} = \begin{bmatrix} (\boldsymbol{w}_x \cdot \boldsymbol{b}_x) & (\boldsymbol{w}_x \cdot \boldsymbol{b}_y) & (\boldsymbol{w}_x \cdot \boldsymbol{b}_z) \\ (\boldsymbol{w}_y \cdot \boldsymbol{b}_x) & (\boldsymbol{w}_y \cdot \boldsymbol{b}_y) & (\boldsymbol{w}_y \cdot \boldsymbol{b}_z) \\ (\boldsymbol{w}_z \cdot \boldsymbol{b}_x) & (\boldsymbol{w}_z \cdot \boldsymbol{b}_y) & (\boldsymbol{w}_z \cdot \boldsymbol{b}_z) \end{bmatrix}$$

We refer to this version of derivation of the rotation matrix as the **Direction Cosine Method/Matrix** (DCM).

Rotation Matrices

- If $\boldsymbol{R}_1, \boldsymbol{R}_2$ are rotation matrices, then $\boldsymbol{R}_1\boldsymbol{R}_2$ is also a rotation matrix
- $(\boldsymbol{R}_1\boldsymbol{R}_2)\boldsymbol{R}_3 = \boldsymbol{R}_1(\boldsymbol{R}_2\boldsymbol{R}_3)$
- $\boldsymbol{R}\boldsymbol{I} = \boldsymbol{I}\boldsymbol{R} = \boldsymbol{R}$ ($\boldsymbol{I}$ is the identity matrix)
- $\boldsymbol{R}^{-1}\boldsymbol{R} = \boldsymbol{I}$
- $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$, aka $\boldsymbol{R}^T_{wb} = \boldsymbol{R}_{bw}$
- $\det(\boldsymbol{R}) = 1$
- $\boldsymbol{R}_{az} = \boldsymbol{R}_{ab}\boldsymbol{R}_{bc}\boldsymbol{R}_{cd}\ldots\boldsymbol{R}_{yz}$

Rotations around Principal Axes

Rotation around x-axis:

$$\boldsymbol{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

Rotation around y-axis:

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

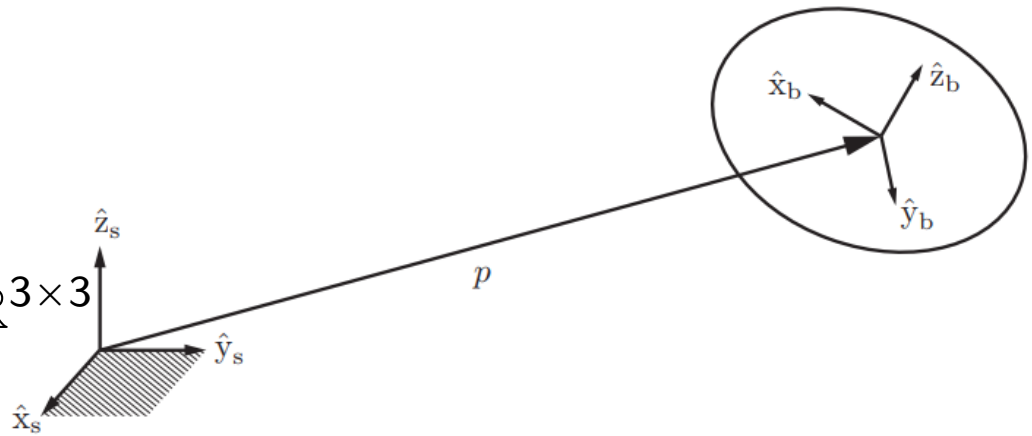
Rotation around z-axis:

$$\boldsymbol{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Full Pose Transformations

We have:

- $\boldsymbol{p}_s = \begin{bmatrix} p_x^s \\ p_y^s \\ p_z^s \end{bmatrix} \in \mathbb{R}^3$
- $\boldsymbol{R}_{sb} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in \mathbb{R}^{3\times 3}$



- $(\boldsymbol{p}_s, \boldsymbol{R}_{sb})$ represent the transformation from the *b frame* to the *s frame*
- We can get the coordinates of a point $\boldsymbol{a}_b = [a_x^b, a_y^b, a_z^b]^T$ from the *b frame* to the *s frame*, using the following:
  $\boldsymbol{a}_s = \boldsymbol{p}_s + \boldsymbol{R}_{sb}\boldsymbol{a}_b$

Figure Source: Modern Robotics: Mechanics, Planning, and Control, *Kevin M. Lynch and Frank C. Park*, 2017, Cambridge University Press.

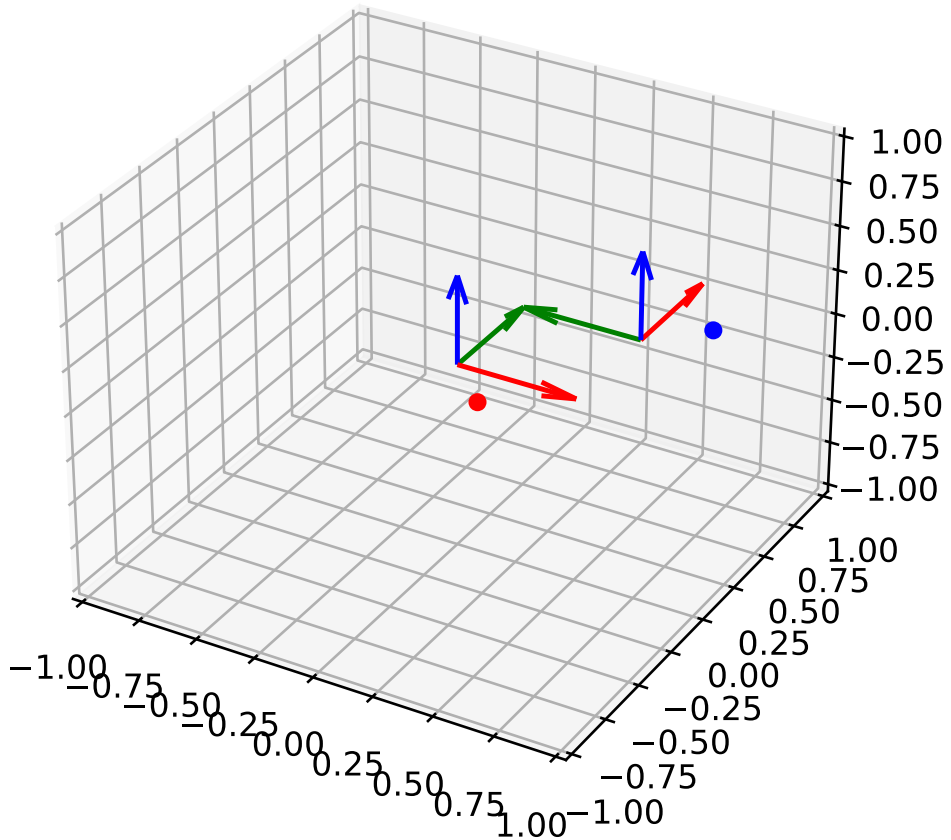
Homogeneous Transformations

- We assume that every point in 3D space can be defined as: $\boldsymbol{p} = [x, y, z, 1]$
- Whereas every vector as: $\boldsymbol{v} = [x, y, z, 0]$
- We define the Homogeneous Transformation Matrix $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}$, where $\boldsymbol{R}$ is a rotation matrix and $\boldsymbol{t}$ a translation vector
- T transforms point $\boldsymbol{p}$ as follows: $\boldsymbol{p}' = \boldsymbol{T}\boldsymbol{p} = \boldsymbol{R}\boldsymbol{p} + \boldsymbol{t}$
- $\boldsymbol{v}' = \boldsymbol{T}\boldsymbol{v} = \boldsymbol{R}\boldsymbol{v}$: vectors can only be rotated!

Homogeneous Transformation Matrices

- If $\boldsymbol{T}_1, \boldsymbol{T}_2$ are transformation matrices, then $\boldsymbol{T}_1\boldsymbol{T}_2$ is also a transformation matrix
- $(\boldsymbol{T}_1\boldsymbol{T}_2)\boldsymbol{T}_3 = \boldsymbol{T}_1(\boldsymbol{T}_2\boldsymbol{T}_3)$
- $\boldsymbol{T}\boldsymbol{I} = \boldsymbol{I}\boldsymbol{T} = \boldsymbol{T}$ (there exist the identity matrix)
- $\boldsymbol{T}^{-1}\boldsymbol{T} = \boldsymbol{I}$
- $\boldsymbol{T}^{-1} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ 0 & 1 \end{bmatrix}$

Transformations Code Example



Skew Symmetric Matrices

The **skew symmetric matrix** representation of a 3-D vector $\boldsymbol{v} = [v_1 \ v_2 \ v_3]^T$ is:

$$\boldsymbol{v}^\wedge = [\boldsymbol{v}] = \boldsymbol{v}_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

Many names in the literature:

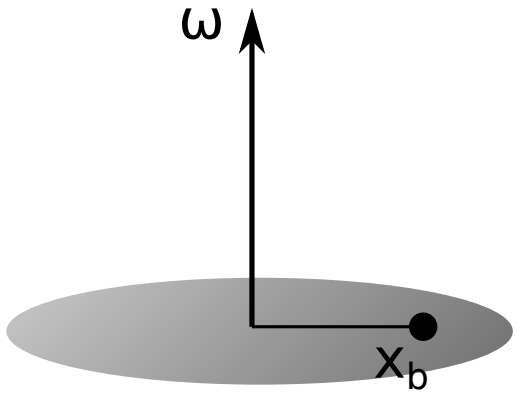
- hat map or matrix, $\boldsymbol{v}^\wedge$ or $\hat{\boldsymbol{v}}$
- skew symmetric matrix, $[\boldsymbol{v}]$
- cross product matrix, $\boldsymbol{v}_\times$ or $[\boldsymbol{v}]_\times$

We define $[\boldsymbol{v}]^\vee = [v_1 \ v_2 \ v_3]^T$ as the inverse operator.

Rotation Kinematics (aka velocities)

$$\boldsymbol{x}_w = \boldsymbol{R}_{wb}\boldsymbol{x}_b$$

$\boldsymbol{x}_b$ is static. **How can we compute** $\dot{\boldsymbol{x}}_w$**?**

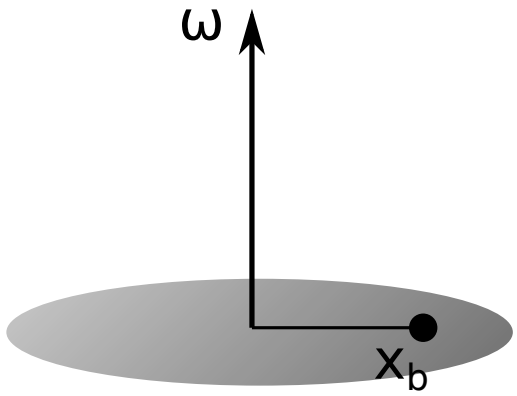


Rotation Kinematics (aka velocities)

$$\boldsymbol{x}_w = \boldsymbol{R}_{wb}\boldsymbol{x}_b$$

$\boldsymbol{x}_b$ is static. **How can we compute $\dot{\boldsymbol{x}}_w$?**

$$\begin{aligned}\dot{\boldsymbol{x}}_w &= \boldsymbol{\omega}_w \times \boldsymbol{x}_w \\ &= \boldsymbol{R}_{wb}(\boldsymbol{\omega}_b \times \boldsymbol{x}_b)\end{aligned}$$



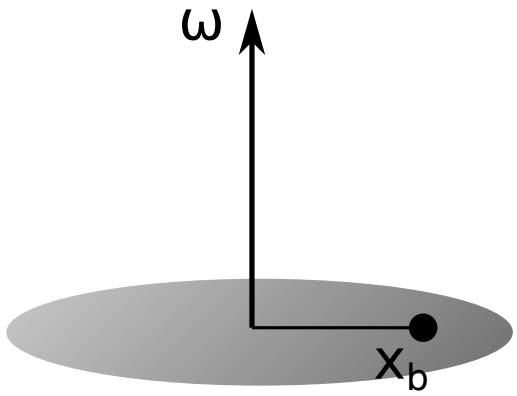
How else?

Rotation Kinematics (aka velocities)

$$\boldsymbol{x}_w = \boldsymbol{R}_{wb}\boldsymbol{x}_b$$

$\boldsymbol{x}_b$ is static. **How can we compute $\dot{\boldsymbol{x}}_w$?**

$$\begin{aligned}\dot{\boldsymbol{x}}_w &= \boldsymbol{\omega}_w \times \boldsymbol{x}_w \\ &= \boldsymbol{R}_{wb}(\boldsymbol{\omega}_b \times \boldsymbol{x}_b)\end{aligned}$$



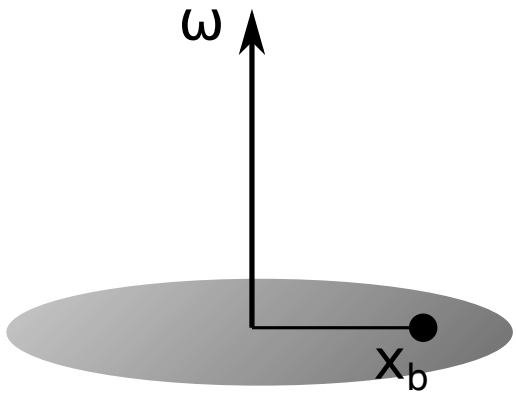
How else?

$$\begin{aligned}\boldsymbol{x}_w &= \boldsymbol{R}_{wb}\boldsymbol{x}_b \Rightarrow \\ \dot{\boldsymbol{x}}_w &= \dot{\boldsymbol{R}}_{wb}\boldsymbol{x}_b + \boldsymbol{R}_{wb}\dot{\boldsymbol{x}}_b \\ \dot{\boldsymbol{x}}_w &= \dot{\boldsymbol{R}}_{wb}\boldsymbol{x}_b, \text{ since } \dot{\boldsymbol{x}}_b = 0\end{aligned}$$

So, what we do have?

Rotation Kinematics (aka velocities)

$$\boldsymbol{x}_w = \boldsymbol{R}_{wb}\boldsymbol{x}_b$$



$\boldsymbol{x}_b$ is static. **How can we compute $\dot{\boldsymbol{x}}_w$?**

$$\begin{aligned}\dot{\boldsymbol{x}}_w &= \boldsymbol{\omega}_w \times \boldsymbol{x}_w \\ &= \boldsymbol{R}_{wb}(\boldsymbol{\omega}_b \times \boldsymbol{x}_b)\end{aligned}$$

How else?

$$\begin{aligned}\boldsymbol{x}_w &= \boldsymbol{R}_{wb}\boldsymbol{x}_b \Rightarrow \\ \dot{\boldsymbol{x}}_w &= \dot{\boldsymbol{R}}_{wb}\boldsymbol{x}_b + \boldsymbol{R}_{wb}\dot{\boldsymbol{x}}_b \\ \dot{\boldsymbol{x}}_w &= \dot{\boldsymbol{R}}_{wb}\boldsymbol{x}_b, \text{ since } \dot{\boldsymbol{x}}_b = 0\end{aligned}$$

So, what we do have?

$$\begin{aligned}\dot{\boldsymbol{R}}_{wb}\boldsymbol{x}_b = \boldsymbol{R}_{wb}(\boldsymbol{\omega}_b \times \boldsymbol{x}_b) = \boldsymbol{R}_{wb}\hat{\boldsymbol{\omega}_b}\boldsymbol{x}_b \Rightarrow \\ \dot{\boldsymbol{R}}_{wb} = \boldsymbol{R}_{wb}\hat{\boldsymbol{\omega}_b}\end{aligned}$$

Rotation Kinematics (2)

Let's appreciate the previous result:

$$\dot{\boldsymbol{R}}_{wb} = \boldsymbol{R}_{wb}\boldsymbol{\omega}_b^{\wedge}$$

What does this mean? What can I do with it?

Rotation Kinematics (2)

Let's appreciate the previous result:

$$\dot{\boldsymbol{R}}_{wb} = \boldsymbol{R}_{wb}\boldsymbol{\omega}_b^{\wedge}$$

What does this mean? What can I do with it?

It means that if I have a measurement of the angular velocity ω_b, I can use it to integrate the full rotation matrix!

In other words:

- I can do forward simulation
- I can use it in combination with gyros/IMUs for state estimation/odometry

With some math manipulation, we can also have ("world space angular velocity"):

$$\dot{\boldsymbol{R}}_{wb} = \boldsymbol{\omega}_w^{\wedge}\boldsymbol{R}_{wb}$$

Angular Velocities

Let's summarize:

$$\hat{\omega}_w = \dot{R}_{wb} R_{wb}^T$$
$$\hat{\omega}_b = R_{wb}^T \dot{R}_{wb}$$

- $\omega_w \in \mathbb{R}^3$ is the angular velocity of the body with respect to (wrt) the world frame
- $\omega_b \in \mathbb{R}^3$ is the angular velocity of the body wrt the body frame
- ω_b **IS NOT** the angular velocity relative to the moving body frame, but relative to the fixed world frame
- ω_w **DOES NOT** depend on the choice of the body frame
- ω_b **DOES NOT** depend on the choice of the world frame
- $\dot{R}_{wb} R_{wb}^T$ is independent from the body frame
- $R_{wb}^T \dot{R}_{wb}$ is independent from the world frame
- Let's assume an angular velocity expressed in frame $\{d\}$, ω_d, we can express it in the frame $\{c\}$: $\omega_c = R_{cd} \omega_d$

Spatial Velocity (1)

Let's try to do the same for homogeneous transformation matrices:

Spatial Velocity (1)

Let's try to do the same for homogeneous transformation matrices:

$$\begin{aligned} \boldsymbol{T}_{wb}^{-1}\dot{\boldsymbol{T}}_{wb} &= \begin{bmatrix} \boldsymbol{R}_{wb}^T & -\boldsymbol{R}_{wb}^T\boldsymbol{p}_w \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{R}}_{wb} & \dot{\boldsymbol{p}}_w \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{R}_{wb}^T\dot{\boldsymbol{R}}_{wb} & \boldsymbol{R}_{wb}^T\dot{\boldsymbol{p}}_w \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \hat{\boldsymbol{\omega}_b} & \boldsymbol{v}_b \\ 0 & 0 \end{bmatrix} \end{aligned}$$

$\dot{\boldsymbol{p}}_w$ is the linear velocity of the body expressed in the world frame, and thus $\boldsymbol{R}_{wb}^T\dot{\boldsymbol{p}}_w$ is the linear velocity of the body expressed in body frame: $\boldsymbol{v}_b$.

Spatial Velocity (2)

Let's put $\boldsymbol{\omega}_b$ and $\boldsymbol{v}_b$ in one vector and call it **Body Spatial Velocity** (or **body twist**): $\mathcal{V}_b = \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix} \in \mathbb{R}^6$. We can now write:

$$\boldsymbol{T}_{wb}^{-1}\dot{\boldsymbol{T}}_{wb} = [\mathcal{V}_b] = \begin{bmatrix} \hat{\boldsymbol{\omega}_b} & \boldsymbol{v}_b \\ 0 & 0 \end{bmatrix}$$

We also have the **World Spatial Velocity** (or **spatial twist**): $\mathcal{V}_w = \begin{bmatrix} \boldsymbol{\omega}_w \\ \boldsymbol{v}_w \end{bmatrix} \in \mathbb{R}^6$. We write as well:

$$\dot{\boldsymbol{T}}_{wb}\boldsymbol{T}_{wb}^{-1} = [\mathcal{V}_w] = \begin{bmatrix} \dot{\boldsymbol{R}}_{wb}\boldsymbol{R}_{wb}^T & \dot{\boldsymbol{p}}_w - \dot{\boldsymbol{R}}_{wb}\boldsymbol{R}_{wb}^T\boldsymbol{p}_w \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{\omega}_w} & \boldsymbol{v}_w \\ 0 & 0 \end{bmatrix}$$

Spatial Velocity (3)

A few remarks:

- $\boldsymbol{v}_w = \dot{\boldsymbol{p}}_w - \dot{\boldsymbol{R}}_{wb}\boldsymbol{R}_{wb}^T\boldsymbol{p}_w$ — **IS NOT** the linear velocity of the body expressed in the world frame

Spatial Velocity (3)

A few remarks:

- $\boldsymbol{v}_w = \dot{\boldsymbol{p}}_w - \dot{\boldsymbol{R}}_{wb}\boldsymbol{R}_{wb}^T\boldsymbol{p}_w$ — **IS NOT** the linear velocity of the body expressed in the world frame (this would just be $\dot{\boldsymbol{p}}_w$)
- $\boldsymbol{v}_w = \dot{\boldsymbol{p}}_w + \boldsymbol{\omega}_w \times (-\boldsymbol{p}_w)$ — now $\boldsymbol{v}_w$ is the velocity of the center of the world frame expressed in the world frame if the body was big enough to include this point
- $\boldsymbol{\omega}_b$ is the angular velocity of the body expressed in the body frame and $\boldsymbol{\omega}_w$ is the angular velocity of the body expressed in the world frame
- $\boldsymbol{v}_b$ is the linear velocity of the center of the body frame
- $\boldsymbol{v}_w$ is the linear velocity of the center of the world frame

Spatial Velocity (4)

We can go from $\mathcal{V}_b$ to $\mathcal{V}_w$:

$$[\mathcal{V}_b] = \boldsymbol{T}_{wb}^{-1}\dot{\boldsymbol{T}}_{wb} = \boldsymbol{T}_{wb}^{-1}[\mathcal{V}_w]\boldsymbol{T}_{wb}$$

And backwards:

$$[\mathcal{V}_w] = \dot{\boldsymbol{T}}_{wb}\boldsymbol{T}_{wb}^{-1} = \boldsymbol{T}_{wb}[\mathcal{V}_b]\boldsymbol{T}_{wb}^{-1}$$

Let's do some calculations:

$$[\mathcal{V}_w] = \begin{bmatrix} \boldsymbol{R}_{wb}\hat{\boldsymbol{\omega}}_b\boldsymbol{R}_{wb}^T & -\boldsymbol{R}_{wb}\hat{\boldsymbol{\omega}}_b\boldsymbol{R}_{wb}^T\boldsymbol{p}_w + \boldsymbol{R}_{wb}\boldsymbol{v}_b \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\omega}_w \\ \boldsymbol{v}_w \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{wb} & 0 \\ \hat{\boldsymbol{p}}_{wb}\boldsymbol{R}_{wb} & \boldsymbol{R}_{wb} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix}$$

Adjoint Representation

Let's assume a transformation $\boldsymbol{T} = (\boldsymbol{R}, \boldsymbol{p})$. We say that its adjoint form, $[Ad_{\boldsymbol{T}}] \in \mathbb{R}^{6\times 6}$, is:

$$[Ad_{\boldsymbol{T}}] = \begin{bmatrix} \boldsymbol{R} & 0 \\ \boldsymbol{p}^{\wedge}\boldsymbol{R} & \boldsymbol{R} \end{bmatrix}$$

In other words:

- $\mathcal{V}_w = [Ad_{\boldsymbol{T}_{wb}}]\mathcal{V}_b$
- $\mathcal{V}_b = [Ad_{\boldsymbol{T}_{bw}}]\mathcal{V}_w$

A few properties:

- $[Ad_{\boldsymbol{T}}]^{-1} = [Ad_{\boldsymbol{T}^{-1}}]$
- $[Ad_{\boldsymbol{T}_1}][Ad_{\boldsymbol{T}_2}]\mathcal{V} = [Ad_{\boldsymbol{T}_1\boldsymbol{T}_2}]\mathcal{V}$

Velocities Code Example

```
    omega_w = T1[:3, :3] @ omega_b # rotate local omega to world frame
    Vw = np.vstack([omega_w, np.zeros((3, 1))])
    T1d = spatial_velocity_to_deriv_homogeneous(Vw, T1)
    T2d = spatial_velocity_local_to_deriv_homogeneous(Vb, T2)

    T1 = T1 + T1d * dt # Euler integration!
    T2 = T2 + T2d * dt # Euler integration!

# These two should be very similar
print(T1)
print("========================")
print(T2)
print("========================")
print(np.linalg.norm(T1-T2))
```

```
[[ 1.       0.       0.       0.     ]
 [ 0.      -0.49001 -0.9005   0.     ]
 [ 0.       0.9005  -0.49001  0.     ]
 [ 0.       0.       0.       1.     ]]
========================
[[ 1.       0.       0.       0.     ]
 [ 0.      -0.49001 -0.9005   0.     ]
 [ 0.       0.9005  -0.49001  0.     ]
 [ 0.       0.       0.       1.     ]]
========================
0.0
```

Spatial Forces

Analogously to the velocities we can study the forces, $\boldsymbol{f}$, and moments, $\boldsymbol{m}$, that are exterted on a rigid body:

$$\mathcal{F} = \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{f} \end{bmatrix} \in \mathbb{R}^6$$

We call $\mathcal{F}$ **spatial force** (or **wrench**). If we have a *spatial force* $\mathcal{F}_b$ expressed in a frame $\{b\}$, we can express it in frame $\{w\}$:

$$\mathcal{F}_w = [Ad_{\boldsymbol{T}_{bw}}]^T \mathcal{F}_b$$

and backwards:

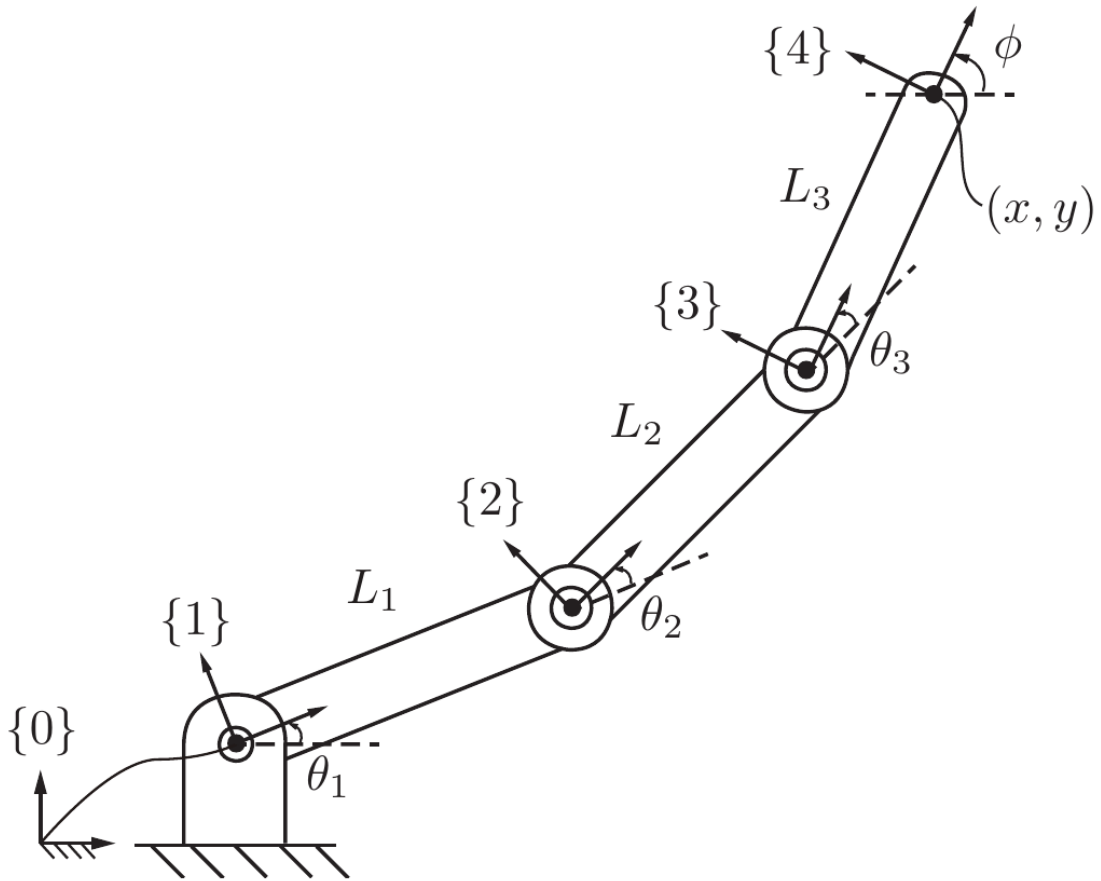
$$\mathcal{F}_b = [Ad_{\boldsymbol{T}_{wb}}]^T \mathcal{F}_w$$

End-effectors

End-effectors (2)

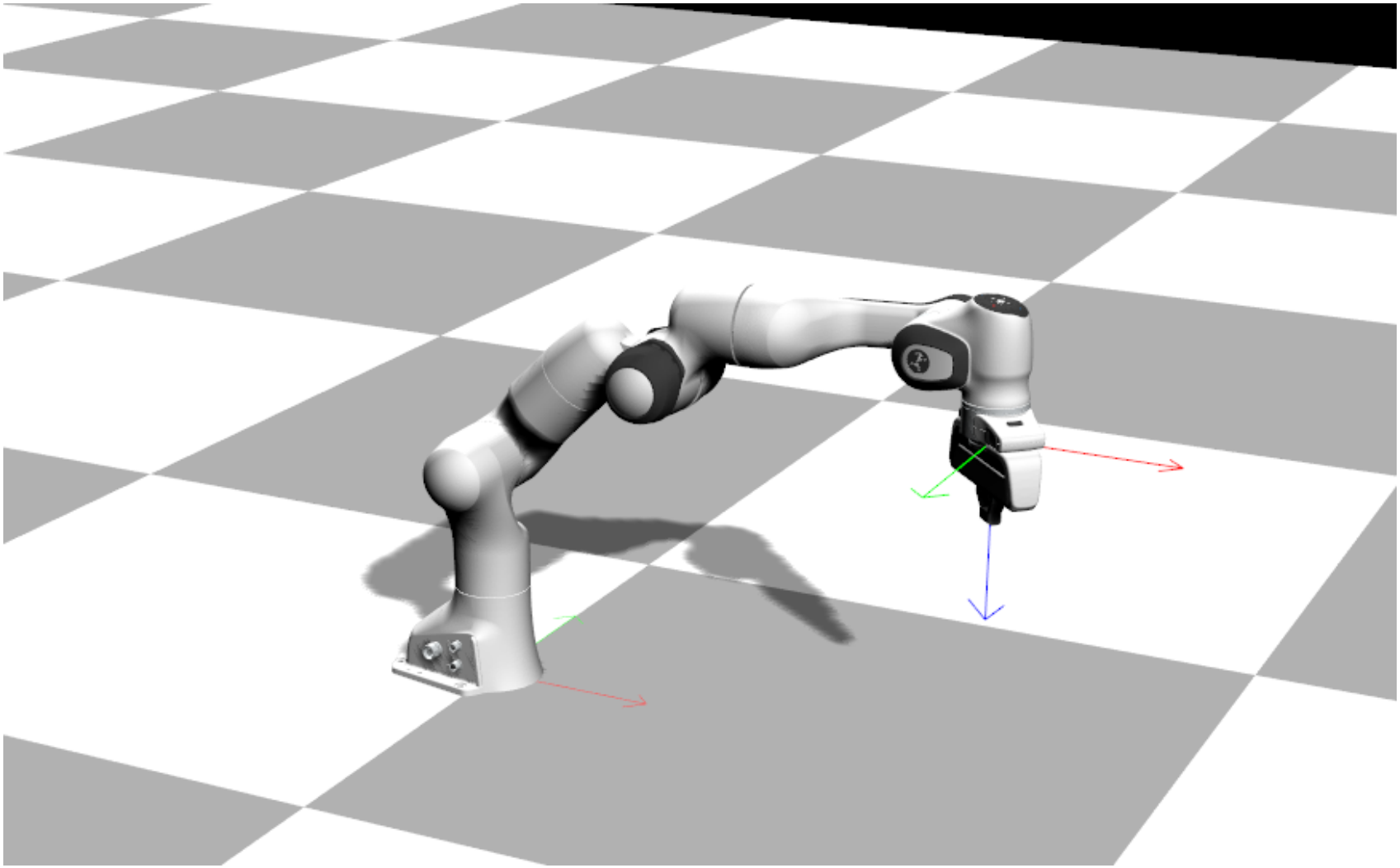
Forward Kinematics (1)

End-effector



Source: Modern Robotics: Mechanics, Planning, and Control, *Kevin M. Lynch and Frank C. Park*, 2017, Cambridge University Press.

Forward Kinematics (2)

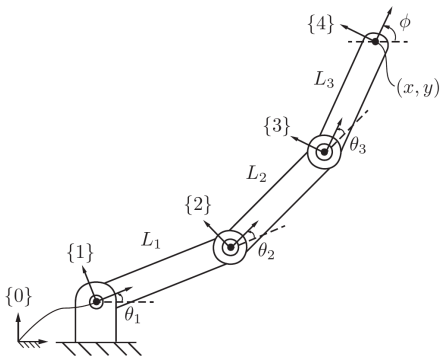


Forward Kinematics (3)

We can use our transformation matrices:

End-effector

$$(x, y, \phi) \equiv \boldsymbol{T}_{04} = \boldsymbol{T}_{01}\boldsymbol{T}_{12}\boldsymbol{T}_{23}\boldsymbol{T}_{34}$$



Source: Modern Robotics: Mechanics, Planning, and Control, *Kevin M. Lynch and Frank C. Park*, 2017, Cambridge University Press.

where

$$\boldsymbol{T}_{01} = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0 \\ sin\theta_1 & cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{T}_{12} = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & L_1 \\ sin\theta_2 & cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{T}_{23} = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & L_2 \\ sin\theta_3 & cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{T}_{34} = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Jacobians (1)

Let's assume that the *end-effector* of our robot is moving with velocity $\dot{\boldsymbol{x}}$[1]. Let's also write the forward kinematics problem as a function of time:

$$\boldsymbol{x}(t) = f_{fk}(\boldsymbol{q}(t))$$

where f_{fk} is the function that gives us the forward kinematics, $\boldsymbol{x} \in \mathbb{R}^m$ the pose of the end-effector, and $\boldsymbol{q} \in \mathbb{R}^n$ the joint values of the robot. If we take the derivative over time:

$$\begin{aligned}\dot{\boldsymbol{x}} &= \frac{\partial f_{fk}(\boldsymbol{q})}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}(t)}{\partial t} \\ &= \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}\end{aligned}$$

where $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{m \times n}$ is the Jacobian matrix.

[1] This is an abstract velocity here.

Jacobians (2)

- Twist $\mathcal{V}_w$ of the end-effector expressed in world frame
- We have: $\mathcal{V}_w = \boldsymbol{J}_w(\boldsymbol{q})\dot{\boldsymbol{q}}$, where $\boldsymbol{J}_w(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$
- We have: $\mathcal{V}_b = \boldsymbol{J}_b(\boldsymbol{q})\dot{\boldsymbol{q}}$, where $\boldsymbol{J}_b(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$ is the Jacobian expressed in body frame
- We also have: $\boldsymbol{J}_b = [Ad_{\boldsymbol{T}_{bw}}]\boldsymbol{J}_w$ and $\boldsymbol{J}_w = [Ad_{\boldsymbol{T}_{wb}}]\boldsymbol{J}_b$

From the principle of energy conservation, we can also derive an equation for the *wrenches*:

$$\boldsymbol{\tau} = \boldsymbol{J}_w(\boldsymbol{q})^T \mathcal{F}_w$$
$$\boldsymbol{\tau} = \boldsymbol{J}_b(\boldsymbol{q})^T \mathcal{F}_b$$

where $\boldsymbol{\tau} \in \mathbb{R}^n$ are the joint torques/forces.

Inverse Kinematics

- This is the "opposite" problem of forward kinematics
- We can find closed-form solutions for many systems
- We can develop iterative algorithms based on the Jacobians
- We can view the problem as an optimization problem
  - We can still use the Jacobians
  - Or we can go black-box!
  - We can take advantage of numerical optimization methods (*more on this later in the course!*)

Bibliography

Chapters 1-4, 6 from **Modern Robotics: Mechanics, Planning, and Control**, *Kevin M. Lynch and Frank C. Park*, 2017, Cambridge University Press. ebook

Thank you

- **Any Questions?**
- **Office Hours**:
  - **Tue-Wed** *(10:00-12:00)*
  - 24/7 by email (`costashatz@upatras.gr`, subject: *ECE_RSI_AM*)
- **Material and Announcements**

Laboratory of Automation & Robotics