Motion Model

Graduate Course INTR-6000P

Week 2 - Lecture 3

Changhao Chen
Assistant Professor
HKUST (GZ)

Recap: Course Roadmap

What you will learn:

- Background knowledge about Intelligent Vehicles and Visual Navigation
- 3D vision foundations
- Visual Localization and Mapping
- Sensor fusion and Vehicle Control
- AI for vehicle navigation

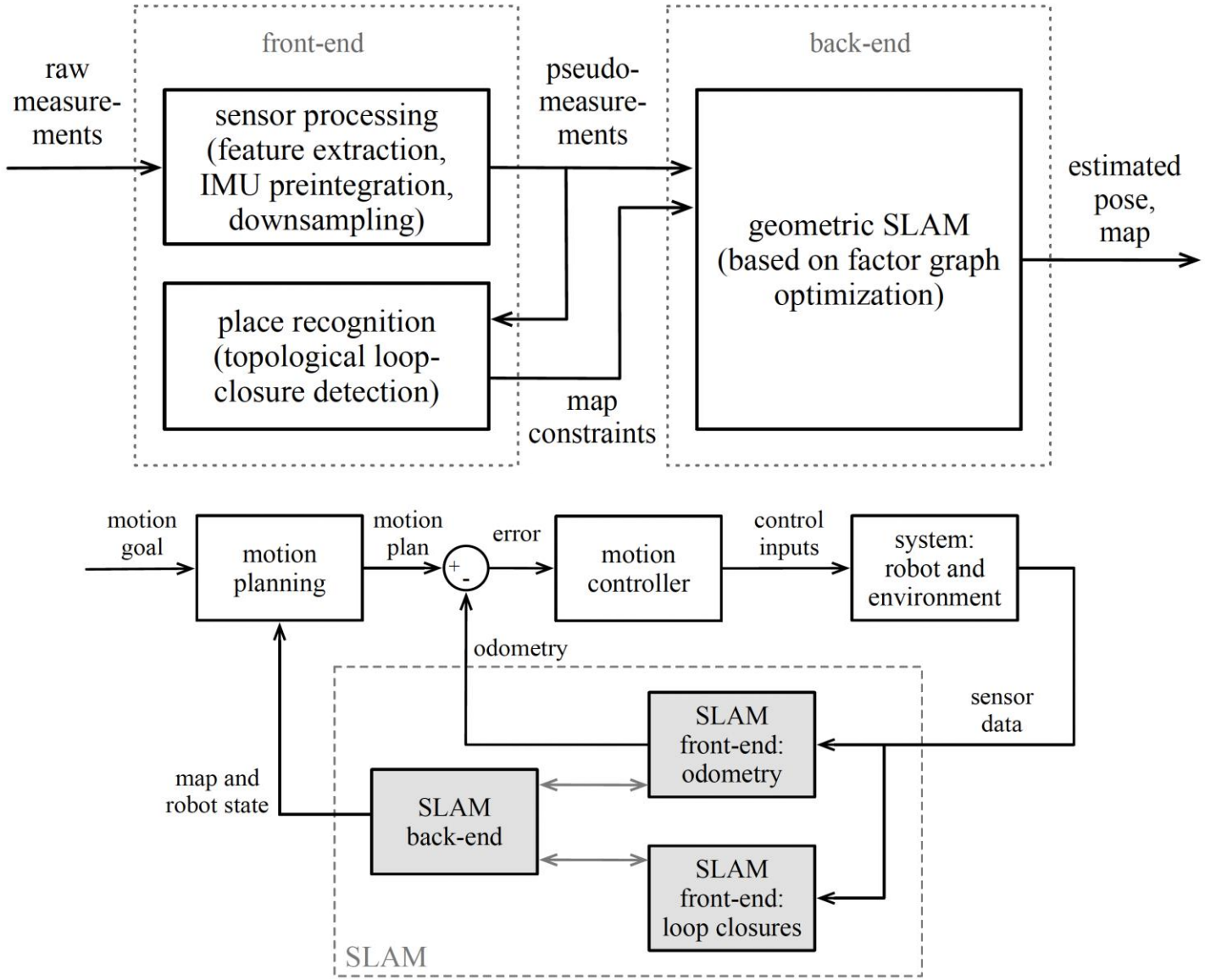
Expectations: hands-on projects, case studies, discussions

Research frontiers & open questions

INTR 6000P - Visual Navigation for Intelligent Vehicles

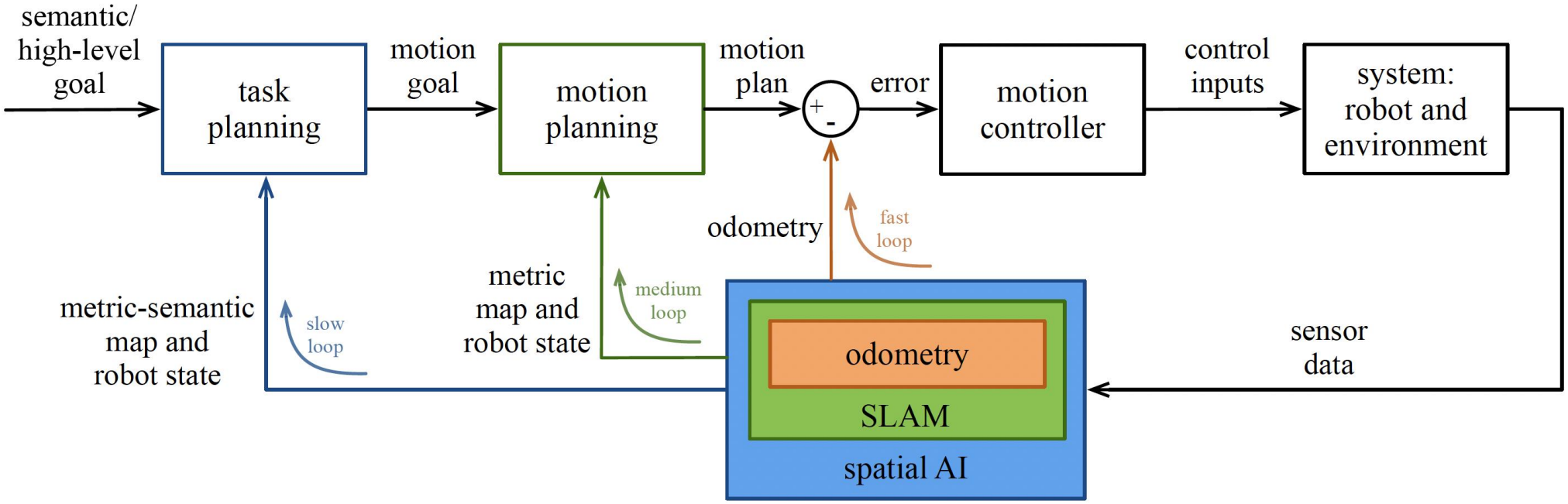
Week	Lecture	Date	Topics	
1	1	Tuesday, Sep 2	Introduction	Introduction
1	2	Tuesday, Sep 2	Visual Navigation System Overview	
2	3	Tuesday, Sep 9	3D Motion Model	3D Vision Foundations
2	4	Tuesday, Sep 9	Lie Groups	
3	5	Tuesday, Sep 16	Camera Model	
3	6	Tuesday, Sep 16	Feature Detection & Description	
4	7	Tuesday, Sep 23	Visual Pose Estimation	Visual Localization and Mapping
4	8	Tuesday, Sep 23	Feature-Based Visual Odometry	
5	9	Sunday, Sep 28	Direct Visual Odometry	
5	10	Sunday, Sep 28	Place Recognition	
6	11	Tuesday, Sep 30	Filtering and State Estimation	
6	12	Tuesday, Sep 30	Non-Linear Optimization	
7	13	Tuesday, Oct 14	Scene Reprensentation	
7	14	Tuesday, Oct 14	Mapping	
8	15	Tuesday, Oct 21	Inertial Integration	Multisensor Fusion & Vehicle Control
8	16	Tuesday, Oct 21	Visual-Inertial Navigation	
9	17	Tuesday, Oct 28	Point Cloud Processing	
9	18	Tuesday, Oct 28	LiDAR-Visual Navigation	
10	19	Tuesday, Nov 4	Trajectory Planning	
10	20	Tuesday, Nov 4	Vehicle Motion Control	
11	21	Tuesday, Nov 11	Enhancing Visual SLAM with Deep Learning	AI for Vehicle Navigation
11	22	Tuesday, Nov 11		
12	23	Tuesday, Nov 18	Embodied Navigation	
12	24	Tuesday, Nov 18	End-to-End Self-Driving	
13	25	Tuesday, Nov 25	Project Presentation	
13	26	Tuesday, Nov 25		

Recap: Visual Navigation



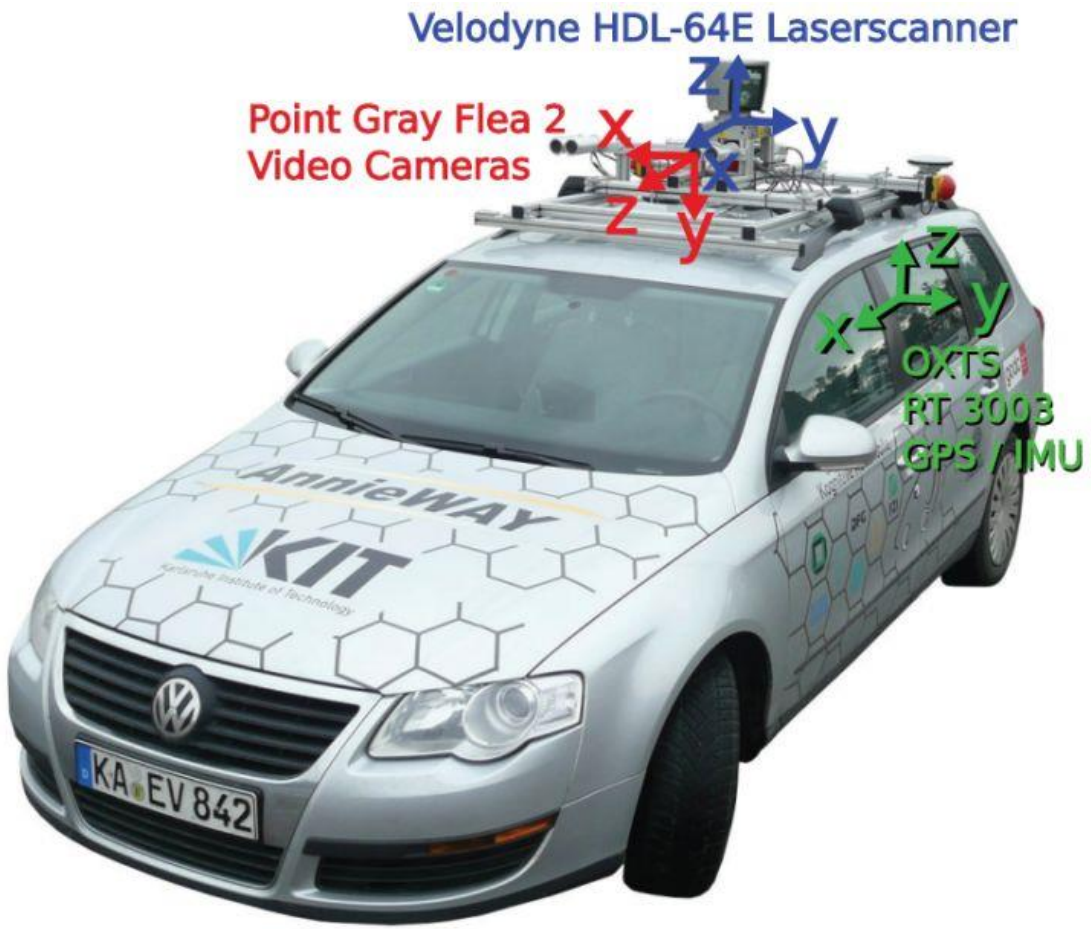
Source: SLAM Handbook

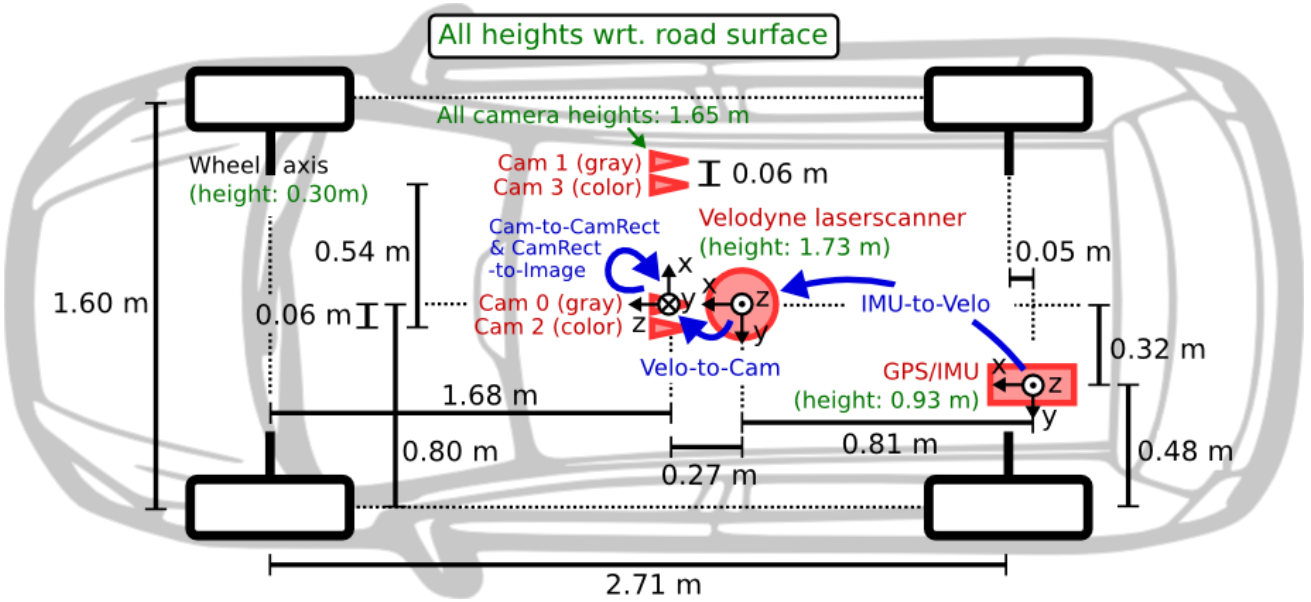
Recap: Visual Navigation



Source: SLAM Handbook

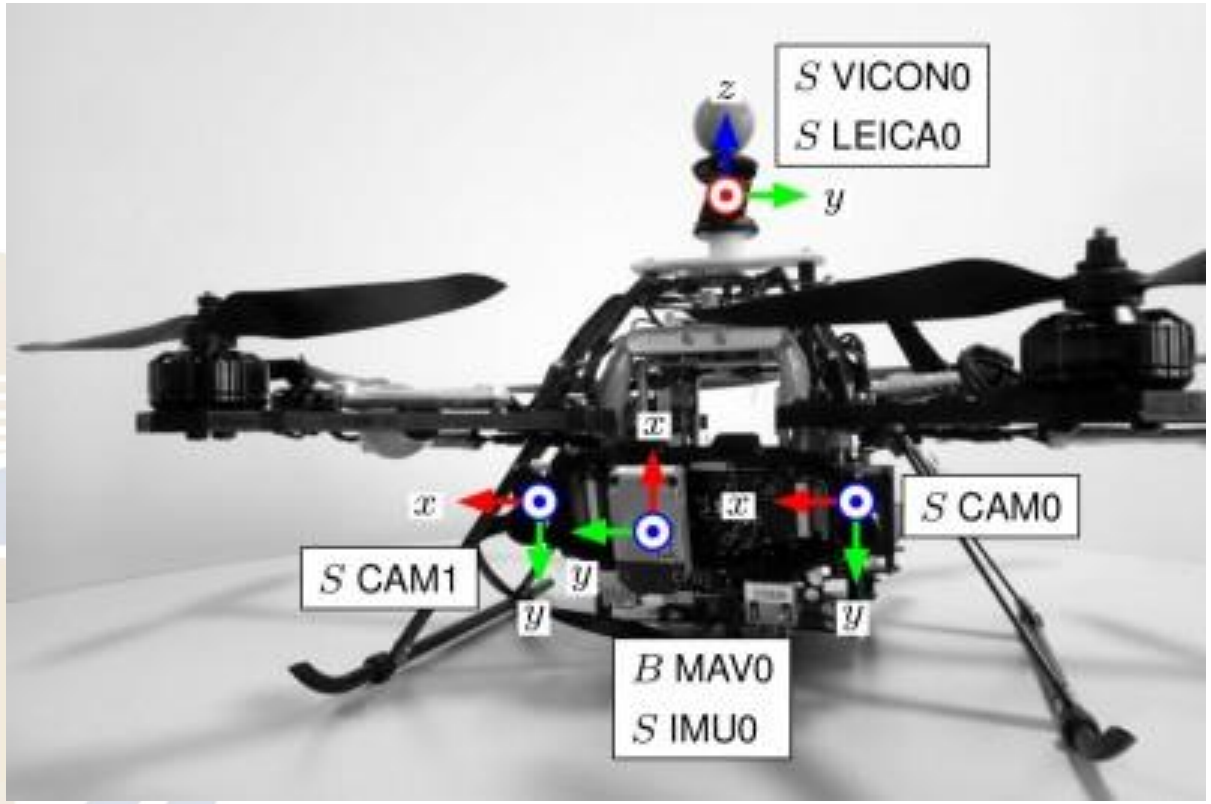
Typical Visual Navigation Platform: KITTI Dataset

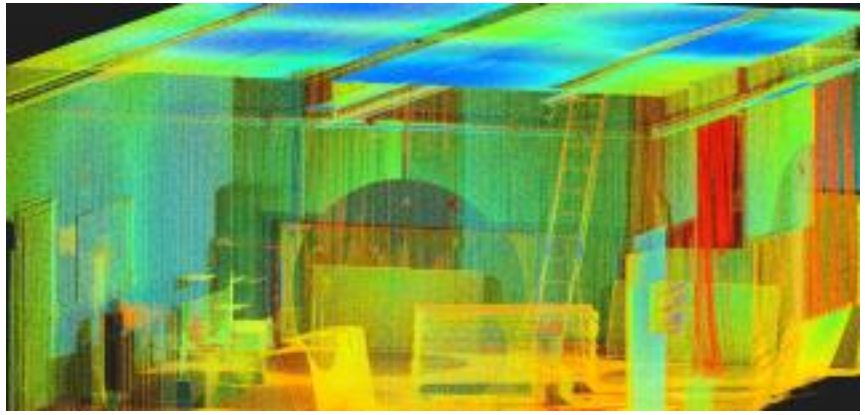




- 1 Inertial Navigation System (GPS/IMU): OXTS RT 3003
- 1 Laserscanner: Velodyne HDL-64E
- 2 Grayscale cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3M-C)
- 2 Color cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3C-C)
- 4 Varifocal lenses, 4-8 mm: Edmund Optics NT59-917

Typical Visual Navigation Platform: EuRoC Dataset





- Visual-Inertial Sensor Unit
  - **Stereo Images** (Aptina MT9V034 global shutter, WVGA monochrome, 2×20 FPS)
  - **MEMS IMU** (ADIS16448, angular rate and acceleration, 200 Hz)
  - Shutter-centric temporal alignment
- Ground-Truth
  - **Vicon** motion capture system (6D pose)
  - **Leica MS50** laser tracker (3D position)
  - Leica MS50 3D structure scan
- Calibration
  - Camera intrinsics
  - Camera-IMU extrinsics
  - Spatio-temporally aligned ground-truth

Coordinate Frames

A coordinate frame (or reference frame) is defined by a set of orthogonal axes fixed to a body. It is used to describe the position and orientation of points relative to that body.

In visual navigation systems, we typically define three primary frames:

World Frame (w): A fixed, global reference frame attached to the environment.

Body Frame (b): A frame attached to the moving platform (e.g., a vehicle or robot).

Camera Frame (c): A frame attached to the camera(s) on the platform.

IMU Frame (i)

Right-handed Rule

In robotics and visual navigation, we standardize on right-handed coordinate frames. The orientation of the axes is defined by the right-hand rule.

Curl (X → Y → Z): Point your right thumb along the X-axis. Curl your fingers toward the Y-axis. Your fingers now point in the direction of the Z-axis.

Points and Positions

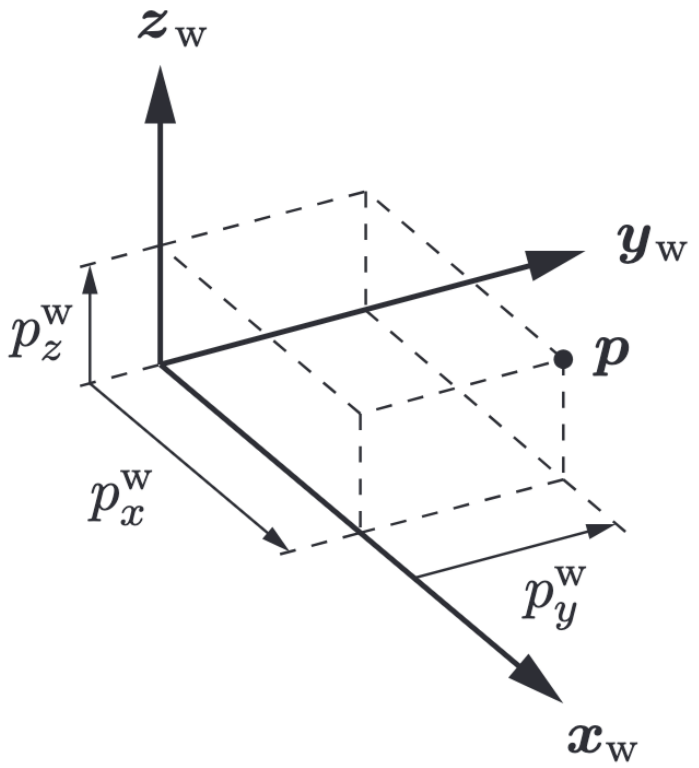
Defining a coordinate frame allows us to describe the position of points using the tools of linear algebra.

For example, the position of a 3D point p, with respect to the world frame {w}, is represented by a 3D vector

$$\boldsymbol{p}^{\mathrm{w}} = \begin{bmatrix} p_x^{\mathrm{w}} \\ p_y^{\mathrm{w}} \\ p_z^{\mathrm{w}} \end{bmatrix}$$

The coordinates of p in the coordinate frame w

Geometrically, these coordinates are the projections of the vector from the origin of {w} to point p onto the axes x_w, y_w, and z_w.



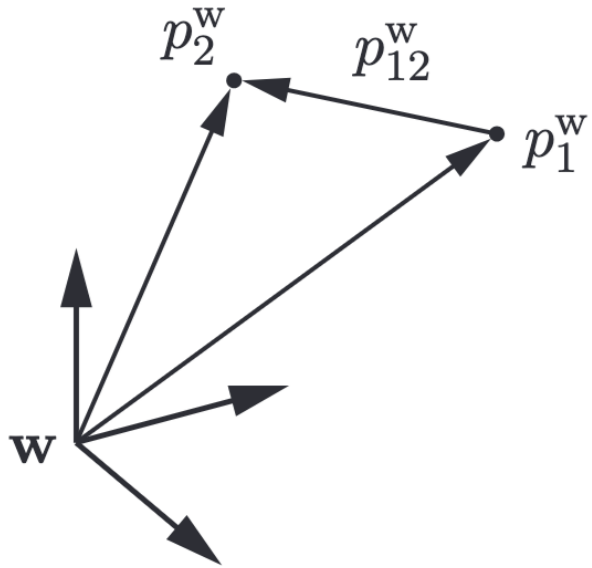
Translation

The translation between point 1 and point 2 is expressed as a vector in the world frame {w}.

$$\boldsymbol{p}_{12}^{\mathrm{w}} = \boldsymbol{p}_{2}^{\mathrm{w}} - \boldsymbol{p}_{1}^{\mathrm{w}}$$

We use a single subscript to denote the reference frame:

$\boldsymbol{p}_{1}^{\mathrm{w}}$ is the position of point 1 with respect to the origin of frame {w}.



Attitude and rotations

From Axes to Attitude: the Rotation Matrix or Direction Cosine Matrix (DCM)

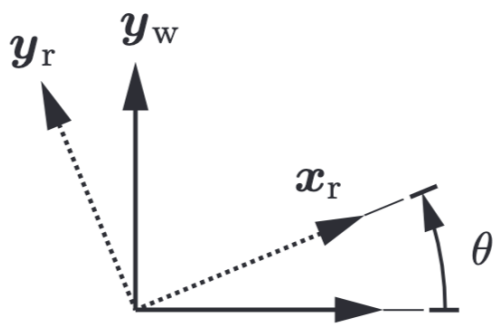
This intuitive method for representing the orientation of frame {r} relative to {w} :

Step 1: Consider the tips of the orthogonal unit vectors defining frame {r}: x_r, y_r, and z_r.

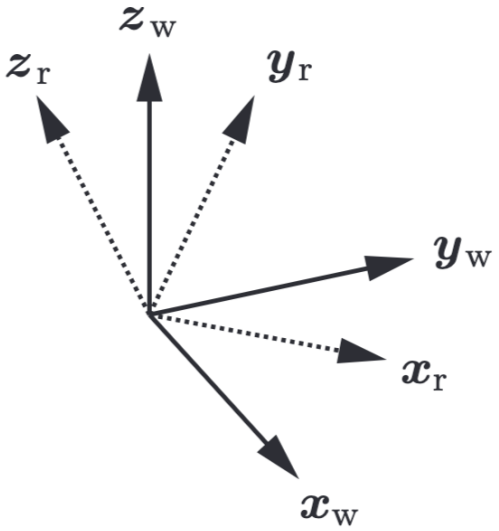
Step 2: Determine the coordinates of each of these vector tips with respect to the world frame {w}. This yields three vectors: x_r_w, y_r_w, z_r_w.

Step 3: Form a 3×3 matrix R_wr by using these vectors as its columns:

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \begin{bmatrix} \boldsymbol{x}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{y}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{z}_{\mathrm{r}}^{\mathrm{w}} \end{bmatrix}$$



$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin\theta & \cos(\theta) \end{bmatrix}$$



Questions

1. Point Transformation
Given: p_r and R_wr
Find: p_w
How do we express a point from frame {r} in frame {w}?

2. Rotation Composition
Given: R_wr and R_rc
Find: R_wc
How do we combine two sequential rotations into a single one?
Analogy: Similar to composing translations (e.g., p_w = p_r + t).

3. Rotation Inverse
Given: R_wr
Find: R_rw
How do we reverse a rotation?
Analogy: Similar to inverting a translation vector (e.g., t_rw = -t_wr).

Operations involving Rotations

Transforming Points Between Rotated Frames:

When two coordinate frames share an origin but are rotated relative to each other, we can transform a point from one frame to the other using its attitude.

Definitions:

- p_r: Coordinates of point p in frame {r}
- R_wr: Rotation matrix defining the attitude of {r} with respect to {w}
- p_w: Coordinates of point p in frame {w} (the unknown)

Transformation Equation:

The coordinates in the world frame {w} are computed by multiplying the rotation matrix by the coordinates in the rotated frame {r}:

$$\boldsymbol{p}^{\mathrm{w}} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}}$$

This equation linearly maps the representation of the point from frame {r} to frame {w}.

Operations involving Rotations

Rotation Composition

Objective: Determine the attitude of frame {c} relative to frame {w}, given the rotations linking them through an intermediate frame {r}.

The Composition Rule:
The total rotation is found by successively applying the relative rotations. In matrix form, this is achieved by multiplication:

$$\boldsymbol{R}_{\mathrm{c}}^{\mathrm{w}} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\ \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}}$$

This equation indicates that the rotation from {c} to {w} is equivalent to first rotating from {c} to {r}, then from {r} to {w}.

Operations involving Rotations

The Inverse Rotation

Objective: Find the rotation matrix R_rw that describes the orientation of frame {w} relative to frame {r}, given R_wr.

Key Property: Rotation matrices are orthogonal, meaning their inverse is equal to their transpose.

Calculation:
The inverse rotation is therefore computed as:

$$\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} = (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{-1} = (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}}$$

$$(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \mathbf{I}_d \qquad (\mathbf{orthogonality})$$

Euler Angles

Euler's Rotation Theorem (1775)

Theorem: Every rotation in three-dimensional space can be represented by a composition of three elementary rotations.

Constraints:

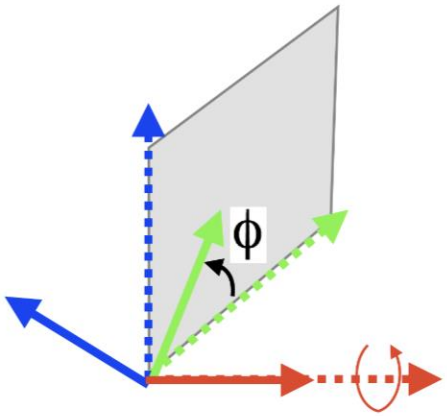
No two consecutive rotations are about the same axis.

The rotations are performed in a specific sequence.

Definition:
An elementary rotation is a rotation about one of the three principal Cartesian axes (X, Y, or Z).

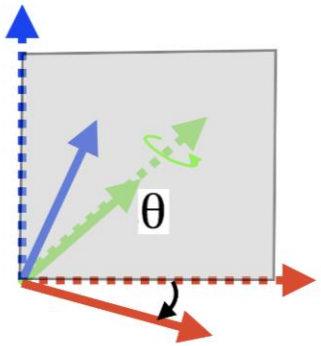
Euler Angles

Rotation around the **x** axis



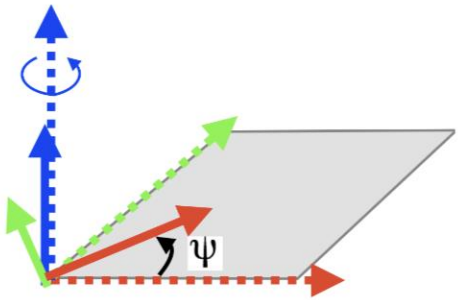
$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

Rotation around the **y** axis



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

Rotation around the **z** axis



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

Euler's Theorem implies that only three parameters (Euler angles) are needed to represent a 3D rotation.

However, the theorem does not specify a unique sequence of axes. The same rotation R_wr can be parameterized by different orders of elementary rotations:

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \boldsymbol{R}_x(\phi_1)\,\boldsymbol{R}_y(\theta_1)\,\boldsymbol{R}_x(\psi_1)$$

$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \boldsymbol{R}_x(\phi_2)\,\boldsymbol{R}_y(\theta_2)\,\boldsymbol{R}_z(\psi_2)$$

Euler Angles

Roll-Pitch-Yaw (RPY) Convention

A popular Euler angle convention uses the Z-Y-X rotation order:

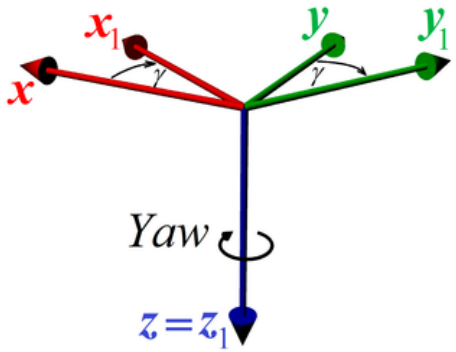
$$\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} = \boldsymbol{R}_z(\gamma)\,\boldsymbol{R}_y(\beta)\,\boldsymbol{R}_x(\alpha)$$

where the angles are defined as:

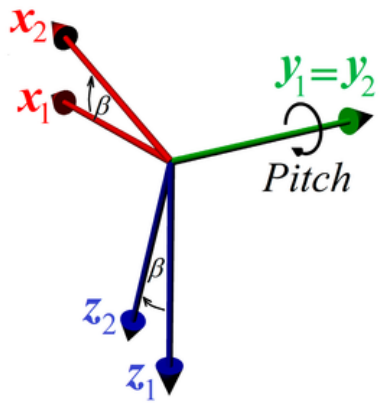
γ (Yaw): Rotation around the Z-axis

β (Pitch): Rotation around the Y-axis

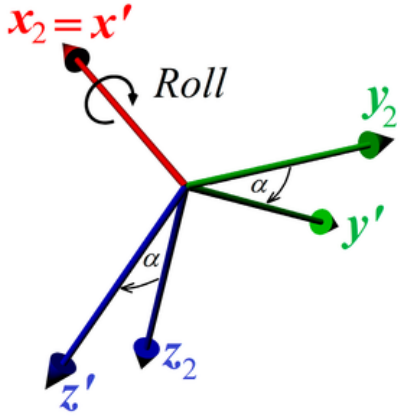
α (Roll): Rotation around the X-axis



1



2



3

Axis-angle representation

Euler's Rotation Theorem, 1775

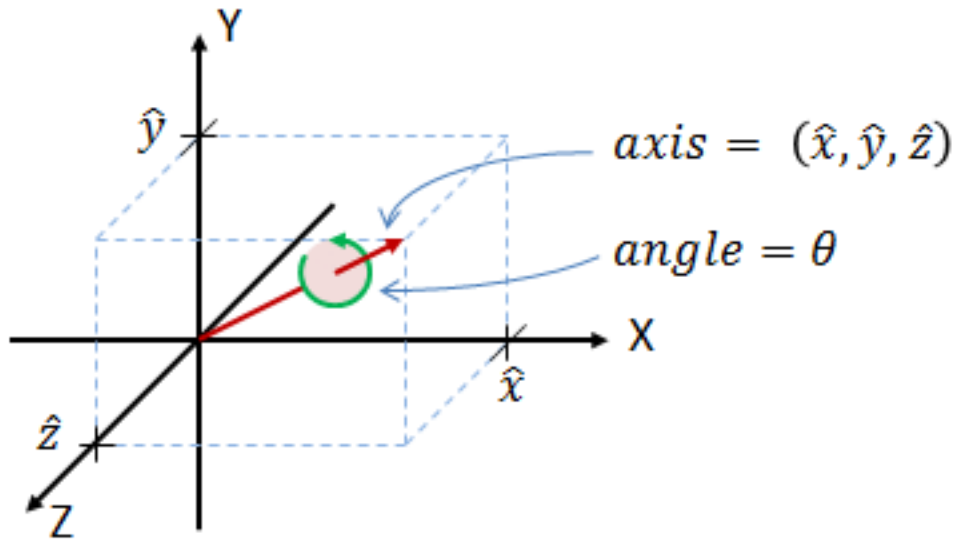
Every 3D rotation can be described as a single rotation by an angle θ about a fixed axis defined by a unit vector u (where ||u|| = 1).

This leads directly to the axis-angle representation, which encodes a rotation using the pair (u, θ).

Advantages of this representation:

Intuitive: Easy to visualize and understand geometrically.

Compact: Requires only 4 parameters (3 for the axis u, 1 for the angle θ), leveraging the unit norm constraint ||u|| = 1.



Quaternion representation

We have a trade-off:

3-Parameter Reps (Euler Angles): Storage-efficient, but suffer from singularities and require trigonometry.

Rotation Matrices (9 parameters): Singularity-free, but over-parameterized.

This leads to a fundamental question:
Is there a representation that is both singularity-free and minimally parameterized?

Quaternion representation

Quaternion plaque on Brougham (Broom) Bridge, Dublin, which says:
Here as he walked by on the 16th of October 1843 Sir William Rowan Hamilton in a flash of genius discovered the fundamental formula for quaternion multiplication

$$i^2 = j^2 = k^2 = ijk = -1$$

& cut it on a stone of this bridge.

Quaternion representation

A quaternion is denoted as a column vector

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

A quaternion can also be represented as a hypercomplex number:

$$q = q_0 + iq_1 + jq_2 + kq_3$$

where q_0 is the scalar part, and the fundamental quaternion units i, j, k satisfy:

$$i^2 = j^2 = k^2 = ijk = -1$$

$$ij = -ji = k \qquad jk = -kj = i \qquad ki = -ik = j$$

A quaternion can be decomposed as:

$$q = [v, s]^T$$

with the vector part $v \in R^3$ and the scalar part $s \in R$.

Quaternion representation

Unit quaternions (quaternions with norm 1) provide a compact, singularity-free representation for 3D rotations.

The quaternion elegantly encodes the axis-angle parameters (u, θ) as follows:

$$\boldsymbol{q} = \begin{bmatrix} \sin(\theta/2)\boldsymbol{u} \\ \cos(\theta/2) \end{bmatrix}$$

Quaternion composition

Given two quaternions:
q_a = [q_a1, q_a2, q_a3, q_a4]^T
q_b = [q_b1, q_b2, q_b3, q_b4]^T

Their product $\boldsymbol{q}_c = \boldsymbol{q}_a \otimes \boldsymbol{q}_b$ is defined as:

$$\boldsymbol{q}_c = \begin{bmatrix} q_{a,4} & -q_{a,3} & q_{a,2} & q_{a,1} \\ q_{a,3} & q_{a,4} & -q_{a,1} & q_{a,2} \\ -q_{a,2} & q_{a,1} & q_{a,4} & q_{a,3} \\ -q_{a,1} & -q_{a,2} & -q_{a,3} & q_{a,4} \end{bmatrix} \begin{bmatrix} q_{b,1} \\ q_{b,2} \\ q_{b,3} \\ q_{b,4} \end{bmatrix} = \Upsilon(\boldsymbol{q}_a)\, \boldsymbol{q}_b$$

$$\boldsymbol{q}_{\mathrm{c}}^{\mathrm{w}} = \boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}} \otimes \boldsymbol{q}_{\mathrm{c}}^{\mathrm{r}}$$

Inverse of a quaternion

Given: The rotation from {r} to {w} as a unit quaternion q_wr.

Find: The inverse rotation from {w} to {r}, denoted q_rw.

Solution:
For a unit quaternion, the inverse is computed by negating the vector part:

$$\boldsymbol{q}_{\mathrm{w}}^{\mathrm{r}} = \begin{bmatrix} -\boldsymbol{v}_{\mathrm{r}}^{\mathrm{w}} \\ s_{\mathrm{r}}^{\mathrm{w}} \end{bmatrix} \doteq (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}})^{-1}$$

Transforming Points using Quaternions

Given:
A point's coordinates in frame {r}: $\mathbf{p}^{r}$
The rotation from {r} to {w} as a unit quaternion: $\mathbf{q}_{r}^{w}$
Frames {w} and {r} share an origin.

Objective: Compute the point's coordinates in frame {w}: $\mathbf{p}^{w}$

Method:
We can represent the point as a "pure quaternion" (zero scalar part) and apply a similarity transform:

$$\boldsymbol{p}^{\mathrm{w}} = (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}) \otimes \begin{bmatrix} \boldsymbol{p}^{\mathrm{r}} \\ 1 \end{bmatrix} \otimes (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}})^{-1} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}} \\ 1 \end{bmatrix}$$

$$\boldsymbol{p}^{\mathrm{w}} = (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}}) \otimes \begin{bmatrix} \boldsymbol{p}^{\mathrm{r}} \\ 0 \end{bmatrix} \otimes (\boldsymbol{q}_{\mathrm{r}}^{\mathrm{w}})^{-1} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{p}^{\mathrm{r}} \\ 0 \end{bmatrix}$$

Pose

Defining Pose: Position and Orientation

The pose of a body frame {r} relative to a world frame {w} is completely described by:

Position: $\mathbf{t}_r^w$ - The translation from the origin of {w} to the origin of {r}.

Orientation: $\mathbf{R}_r^w$ - The rotation from {r} to {w}.

The pair $(\boldsymbol{R}_r^w, \boldsymbol{t}_r^w)$ is the complete geometric representation of frame {r} in {w}.

A pose has 6 degrees of freedom (3 for translation, 3 for rotation). We can combine these into a single, convenient transformation matrix.

$$\boldsymbol{T}_r^w = \begin{bmatrix} \boldsymbol{R}_r^w & \boldsymbol{t}_r^w \\ \mathbf{0}_d^\intercal & 1 \end{bmatrix}$$

Rigid-body Transformations

We now generalize from pure rotation to full pose (rotation and translation).

Given:

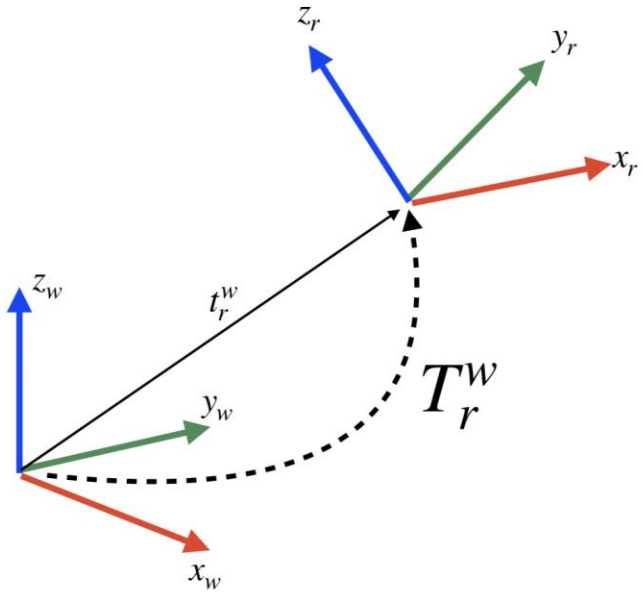
A point's coordinates in frame {r}: $\mathbf{p}^{r}$

The pose of {r} relative to {w}, represented by the transformation matrix: $\mathbf{T}_{r}^{w}$

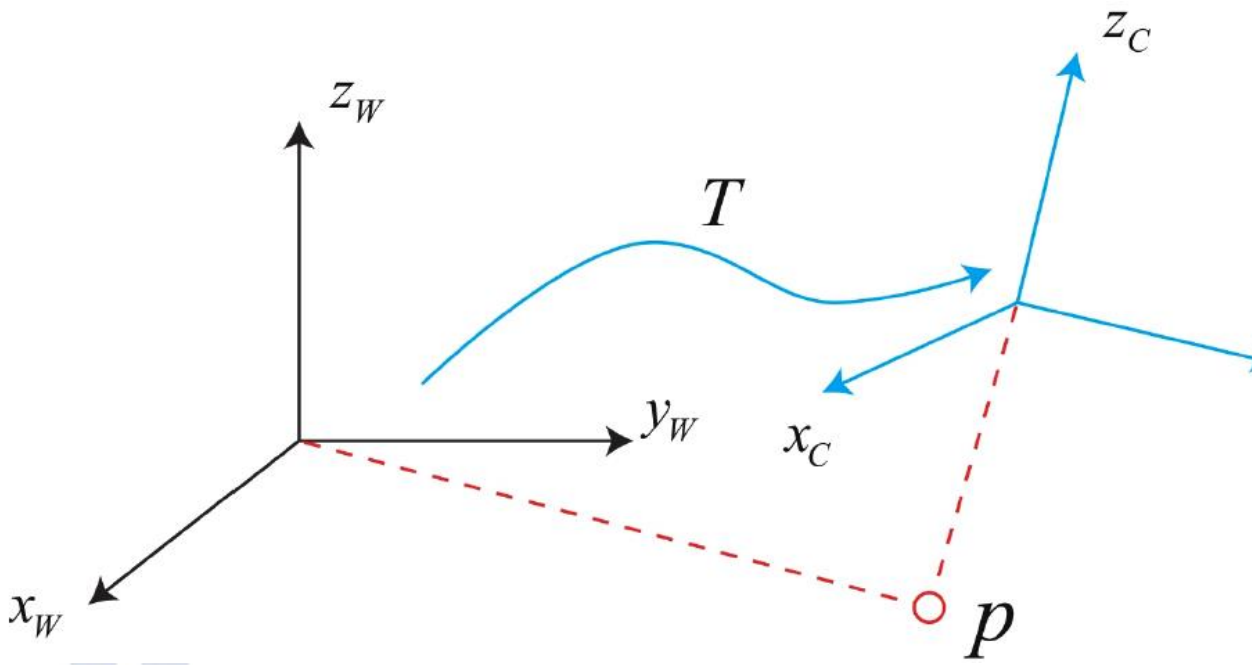
Compute: The point's coordinates in frame {w}: $\mathbf{p}^{w}$

The transformation is given by:

$$\boldsymbol{p}^{\mathrm{w}} = \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}}\boldsymbol{p}^{\mathrm{r}} + \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}}$$



Motion Definition



$$\begin{bmatrix} \boldsymbol{a}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ 1 \end{bmatrix} \triangleq \boldsymbol{T} \begin{bmatrix} \boldsymbol{a} \\ 1 \end{bmatrix}.$$

$$\mathrm{SE}(3) = \left\{ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \boldsymbol{R} \in \mathrm{SO}(3), \boldsymbol{t} \in \mathbb{R}^3 \right\}$$

Pose Composition

$$\tilde{\boldsymbol{p}}^{\mathrm{w}} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \\ \mathbf{0}_d^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}^{\mathrm{r}} \\ 1 \end{bmatrix} = \boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}} \tilde{\boldsymbol{p}}^{\mathrm{r}}$$

Given:
Pose of {r} relative to {w}: $\mathbf{T}_{\mathrm{r}}^{\mathrm{w}}$
Pose of {c} relative to {r}: $\mathbf{T}_{\mathrm{c}}^{\mathrm{r}}$

Compute:
Pose of {c} relative to {w}: $\mathrm{T}_{\mathrm{c}}^{\mathrm{w}}$

Solution:
Similar to rotation composition, poses are composed by matrix multiplication:

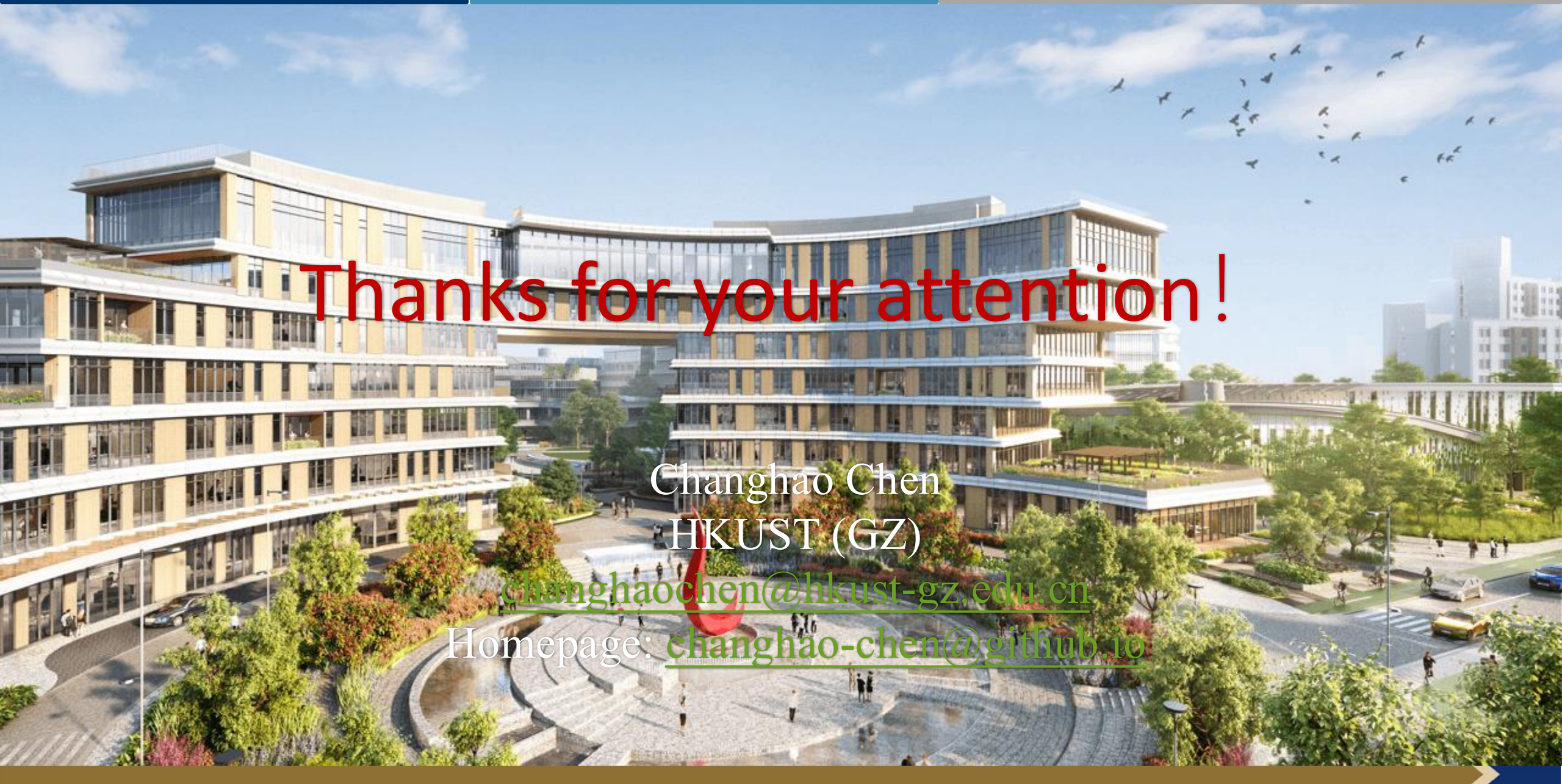
$$\boldsymbol{T}_{\mathrm{c}}^{\mathrm{w}} = \boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}}\, \boldsymbol{T}_{\mathrm{c}}^{\mathrm{r}}$$

$$\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}}\, \boldsymbol{T}_{\mathrm{c}}^{\mathrm{r}} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} & \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \\ \mathbf{0}_d^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}} & \boldsymbol{t}_{\mathrm{c}}^{\mathrm{r}} \\ \mathbf{0}_d^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{R}_{\mathrm{c}}^{\mathrm{r}} & \boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}} \boldsymbol{t}_{\mathrm{c}}^{\mathrm{r}} \\ \mathbf{0}_d^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{c}}^{\mathrm{w}} & \boldsymbol{t}_{\mathrm{c}}^{\mathrm{w}} \\ \mathbf{0}_d^{\mathsf{T}} & 1 \end{bmatrix} = \boldsymbol{T}_{\mathrm{c}}^{\mathrm{w}}$$

Inverse of a Pose

$$\boldsymbol{T}_{\mathrm{w}}^{\mathrm{r}} = (\boldsymbol{T}_{\mathrm{r}}^{\mathrm{w}})^{-1} = \begin{bmatrix} (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} & -(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} \boldsymbol{t}_{\mathrm{r}}^{\mathrm{w}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix}$$

$$\begin{bmatrix} (\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} & -(\boldsymbol{R}_{\mathrm{r}}^{\mathrm{w}})^{\mathsf{T}} \boldsymbol{t}_{\mathrm{wr}}^{\mathrm{w}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} & -\boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} \boldsymbol{t}_{\mathrm{wr}}^{\mathrm{w}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} & -\boldsymbol{t}_{\mathrm{wr}}^{\mathrm{r}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{w}}^{\mathrm{r}} & \boldsymbol{t}_{\mathrm{rw}}^{\mathrm{r}} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix}$$



Thanks for your attention!

Changhao Chen
HKUST (GZ)
changhaochen@hkust-gz.edu.cn
Homepage: changhao-chen@github.io