Basic

SLAM Tutorials for Everyone Lec. #8. Three Practical Tips for Transformations in Code

Hyungtae Lim, Ph.D.

<shapelim [at] mit [dot] edu> / <fudxo5143+slam [at] gmail [dot] com>

Recap: Rigid Body Motion Group SE(3)

- A full 3D pose (position + orientation) is represented as:

Definition

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

6 degrees of freedom: 3 for rotation + 3 for translation.

Rigid body transformation of a point $\boldsymbol{p} \in \mathbb{R}^3$:

$$\boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p} + \boldsymbol{t} \qquad \Longleftrightarrow \qquad \begin{bmatrix} \boldsymbol{p}' \\ 1 \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix}$$

Composition of transforms (e.g., sensor → body → world):

$$^{\text{world}}\boldsymbol{T}_{\text{sensor}} = {}^{\text{world}}\boldsymbol{T}_{\text{body}} \cdot {}^{\text{body}}\boldsymbol{T}_{\text{sensor}}$$

Inverse:

$$\boldsymbol{T}^{-1} = \begin{bmatrix} \boldsymbol{R}^\top & -\boldsymbol{R}^\top \boldsymbol{t} \\ \boldsymbol{0}^\top & 1 \end{bmatrix}$$

Three Practical Tips for Transformations

- Three categories of silent bugs recur across real SLAM systems.

Tip 1	Tip 2	Tip 3
Make the transform meaning explicit Always specify direction explicitly (e.g., `T_cam_lidar`)	Verify quaternion coefficient order Do not mix WXYZ (Eigen style) and XYZW (ROS tf style)	Document all pose frames explicitly Same keyword may mean opposite across systems

Tip 1: Make the Transform Meaning Explicit

`T_LiDAR_to_CAM` is ambiguous — same name, two opposite interpretations:

Case A — Transform a point: left-multiply

```
// Points from a LiDAR sensor is transformed
// to the camera frame
// i.e., ^C_p = T * ^L_p
pcl::transformPointCloud(
cloud_lidar, cloud_cam,
T_lidar_wrt_cam); // "A wrt B"
```

Case B — Compose poses: right-multiply

```
// T_W->C = T_W->L * T_L->C
pose_cam = pose_lidar * T_lidar_to_cam;
// "A to B"
```

Recommended conventions:

- `A_wrt_B` — point transform ${}^{B}\boldsymbol{p} = \boldsymbol{T}_{A \text{ wrt } B} \cdot {}^{A}\boldsymbol{p}$
- `A_to_B` — pose composition ${}^{W}\boldsymbol{T}_{B} = {}^{W}\boldsymbol{T}_{A} \cdot \boldsymbol{T}_{A \to B}$
- GTSAM style: `bTa` for ${}^{B}\boldsymbol{T}_{A}$

Rule of thumb

Agree on one convention before coding begins. One inconsistency causes silent, hard-to-trace errors.

https://gtsam.org/gtsam.org/2020/06/28/gtsam-conventions.html

Tip 2: Verify Quaternion Coefficient Order

- Quaternion $\boldsymbol{q} = (q_w, q_x, q_y, q_z)$ has 4 components.
- Do *not* mix WXYZ (Eigen constructor) and XYZW (`.coeffs()`, ROS tf).

Eigen — constructor: W, X, Y, Z:
(Hamilton convention)

```
// Constructor: WXYZ order
Eigen::Quaterniond q(w, x, y, z);
// TRAP: .coeffs() returns XYZW!
auto c = q.coeffs();
// c[0]=x, c[1]=y, c[2]=z, c[3]=w
```

ROS tf — always X, Y, Z, W:
(JPL convention)

```
tf::Quaternion tq(x, y, z, w);
```

FAST-LIO's fix — explicit comments:

```
// w=coeffs[3], x=coeffs[0],
// y=coeffs[1], z=coeffs[2]
// euler[0]=roll, [1]=pitch,
// [2]=yaw
```

Rules

- Prefer native types:
  `Eigen::Quaterniond`,
  `geometry_msgs::Quaternion`.
- Avoid raw `vector<double>`
  without an order comment.
- Document Euler order too:
  RPY $\neq$ YPR.

Tip 3: Document All Pose Frames Explicitly

- LIO-SAM and FAST-LIO both use *extrinsic*
- But they mean *opposite* reference frames. Never assume conventions across systems.

LIO-SAM `config/params.yaml`:

```
# Extrinsics: T_lb (lidar -> imu)
extrinsicTrans: [0.0, 0.0, 0.0]
extrinsicRot: [-1, 0, 0, ...]
```

Pose in **LiDAR frame** w.r.t. world.
Extrinsic $= T_{\text{lidar}\to\text{imu}}$ (i.e., ${}^{\text{lidar}}T_{\text{imu}}$).

FAST-LIO `config/avia.yaml`:

```
mapping:
extrinsic_T: [0.04165,...]
extrinsic_R: [1, 0, 0,...]
```

Pose in **IMU frame** w.r.t. world.
Extrinsic $=$ `T_Lidar_wrt_IMU` (i.e., ${}^{\text{imu}}T_{\text{lidar}}$).

Same keyword, opposite frame convention

Users must read source code to configure correctly. One wrong frame $\Rightarrow$ silent drift or divergence.

Fix: Add one comment line per entry to document the transformation direction explicitly.

Conclusion

Tip 1	Tip 2	Tip 3
Make the transform meaning explicit Always specify direction explicitly (e.g., `T_cam_lidar`)	Verify quaternion coefficient order Do not mix WXYZ (Eigen style) and XYZW (ROS tf style)	Document all pose frames explicitly Same keyword may mean opposite across systems

∴ *Never rely on implicit conventions.*
Make everything explicit