# 手性与旋转，所有坐标系

## 一、手性和旋转

### 1、概念

手性：指一个物体或其坐标系无法与其镜像重合的性质，比如无论怎么旋转左手套都无法套进右手上。

- **右手坐标系 (Right-Handed Coordinate System)**：
  - **规则**：伸出你的**右手**，让食指指向X轴正方向，中指指向Y轴正方向，那么**大拇指的方向就是Z轴的正方向**。
  - **常见应用**：OpenGL、ROS、机器人学、物理学、数学、Blender（默认）。
  - **旋转正方向**：通常，**逆时针**旋转为正方向（从旋转轴的正方向看向原点）。
- **左手坐标系 (Left-Handed Coordinate System)**：
  - **规则**：伸出你的**左手**，让食指指向X轴正方向，中指指向Y轴正方向，那么**大拇指的方向就是Z轴的正方向**。
  - **常见应用**：Unity、DirectX、Unreal Engine、计算机视觉中的相机坐标系（注意：这是一个关键区别！）。
  - **旋转正方向**：通常，**顺时针**旋转为正方向。

**核心关系：手性决定了坐标轴的方向和旋转的正方向。**

旋转：旋转描述了物体或坐标系在三维空间中朝向的变化，而保持其原点位置不变。它是一种保距、保向的变换（保持长度和角度不变，且不改变手性）。

**数学表示（与手性紧密相关）：**

1. **旋转矩阵 (Rotation Matrix)**：一个 3x3 的**正交矩阵**（`R^T = R^{-1}`），其**行列式 det(R) = +1**（这保证了它不改变手性）。
2. **旋转向量/轴角 (Rotation Vector / Axis-Angle)**：一个3D向量，其**方向**代表旋转轴，其**长度**代表旋转角度。**旋转方向遵循所在坐标系的手性规则**（右手系逆时针为正）。
3. **欧拉角 (Euler Angles)**：用三个绕特定坐标轴（如X-Y-Z）的旋转角度来表示。**顺序极其重要**（例如，绕X-Y-Z旋转和绕Z-Y-X旋转结果完全不同），并且存在万向节死锁问题。
4. **四元数 (Quaternion)**：用四个数 `(x, y, z, w)` 来表示旋转，能有效避免万向节死锁，是插值和计算的常用形式。

## 2、关系

**关系：旋转的定义和解释完全依赖于其所在坐标系的手性。**

**最大的影响：相机坐标系 vs. 世界坐标系**

这是计算机视觉和三维定位中最容易出错的地方！

1. **相机坐标系 (Camera Coordinate System) - 通常是左手系：**
   - **定义**：X轴向右，Y轴向下，Z轴**向前**（从相机指向场景）。
   - **为什么是左手系？**：因为图像传感器上的像素坐标(u, v)通常定义为：u（列）向右增加（+X），v（行）向下增加（+Y）。根据左手定则，Z轴自然指向场景前方。这使得相机前方的点都有正的Z值，非常直观。
2. **世界坐标系 (World Coordinate System) - 通常是右手系：**
   - **定义**：通常，X轴向前，Y轴向左，Z轴向上（例如在机器人或航空中），或者采用标准的北东地（NED）等。关键是，它遵循右手定则。

### 总结与实践建议

| 特性 | 旋转 (Rotation) | 手性变换 (Handedness Change) |
| --- | --- | --- |
| **本质** | 改变朝向，保持手性 | 产生镜像，改变手性 |
| **矩阵行列式** | +1 | -1 |
| **数学性质** | 正交矩阵（`R^T = R^{-1}`） | 非正交矩阵 |
| **在定位中的角色** | 描述物体或坐标系的方向 | 通常是不同系统（如世界系vs相机系）间转换的**固有属性**，需要确和处理 |

## 3、实践问题

**为什么变换了手性之后轨迹的几何关系被破坏了？**

这是一个非常经典且重要的问题，它触及了三维视觉和SLAM中坐标系变换的核心。简单来说，施加一个行列式为-1的旋转（即“Improper Rotation”或反射旋转）实际上不仅仅是旋转，它还包含了一次镜面反射。这会破坏整个系统的 handedness（手性），从而导致轨迹和几何关系完全混乱。

下面进行详细的解释。

1. 核心概念：旋转矩阵与行列式

- 真正的旋转矩阵 (Proper Rotation Matrix):
  - 属于 SO(3) 群 (Special Orthogonal Group)。
  - 性质： `R^T * R = I` 且 det(R) = +1。
  - 作用：在保持坐标系手性 (Handedness)（即“右手定则”）不变的情况下，对物体进行纯粹的旋转。长度、角度、体积都不变。
- 行列式为-1的矩阵 (Improper Rotation Matrix):
  - 属于 O(3) 群但不属于 SO(3) 群。
  - 性质： `R^T * R = I` 但 det(R) = -1。
  - 作用：它可以分解为一个纯粹的旋转和一个镜面反射 (Reflection) 的复合操作。最常见的例子就是镜像或坐标轴的反转（例如，将z轴从朝前变成朝后）。

2. 手性 (Handedness) 是什么?

手性是坐标系的一个根本属性：

- 右手坐标系 (Right-handed System): 伸出你的右手，让食指指向X轴正方向，中指指向Y轴正方向，那么你的大拇指将指向Z轴正方向。X × Y = Z (叉乘)。绝大多数计算机视觉和机器人系统（如OpenCV、ROS、COLMAP）使用右手坐标系。
- 左手坐标系 (Left-handed System): 伸出你的左手，做同样的动作，你会得到 X × Y = -Z。

一个 `det = -1` 的变换会翻转这种手性。它将一个右手坐标系变成了左手坐标系，反之亦然。

3. 为什么这会导致轨迹混乱?

想象一个典型的SLAM或SfM系统，它的所有计算都基于一个一致的坐标系手性假设（通常是右手坐标系）。

4. 三维点重建：
   三维点 `P_w` 通过相机位姿 `[R|t]` 被投影到像素坐标 `u`。公式为：
   `s * u = K * [R | t] * P_w`
   其中 `K` 是相机内参， `R` 是一个 `det(R)=+1` 的旋转矩阵。

现在，如果你强行将一个 `det(R) = -1` 的矩阵 `R_improper` 应用到某个（或所有）相机位姿上，会发生什么?

- 局部不一致性：对于单个相机， `R_improper` 会将其视角“反射”到镜面世界。它看到的场景几何与其他 `det=+1` 的相机看到的场景几何是镜像关系。

- 全局不一致性：BA (Bundle Adjustment) 优化过程试图最小化重投影误差。优化器会发现，无论怎么调整这个带有 `R_improper` 的相机位姿和它看到的三维点，都无法与其他正常的相机和三维点达成一致的重投影结果。因为一方在真实世界，另一方在镜中世界。优化器为了减小这个无法调和的巨大误差，会开始“拉扯”这些错误的三维点和相机位姿，导致整个轨迹和点云发生扭曲、断裂和混乱。

5. 姿态估计中的旋转增量：
   在视觉里程计中，我们通过相邻帧间的特征点匹配来计算相对位姿变换 `T_{i->i+1} = [R_{rel} | t_{rel}]`。这个 `R_{rel}` 也应该是一个 `det=+1` 的旋转。

如果你错误地引入了一个 `det=-1` 的变换，例如在集成陀螺仪数据或处理旋转参数化（四元数、欧拉角）时出现了符号错误，那么计算出的 `R_rel` 可能包含反射。

- 当你用这个错误的 `R_rel` 去更新当前位姿 `T_i` 以得到 `T_{i+1}` 时，`T_{i+1}` 的手性就被破坏了。
- 这个错误的位姿又会被用来三角化新的三维点，这些新点自然就位于错误的（镜像的）位置。
- 这个错误会随着机器人的移动不断累积和传播，导致轨迹逐渐偏离，最终变得完全混乱且无法理解。

一个生动的比喻：

这就像试图用一张镜像地图在一个正常世界里导航。地图告诉你“向右转”，但在镜像地图里，“右”其实是现实世界的“左”。如果你严格按照这张错误的地图走，你的路径很快就会变得一塌糊涂，与真实环境完全对不上。

**问题总结**

施加 `det = -1` 的“旋转”矩阵导致轨迹混乱的根本原因是：

1. 它不是纯旋转，而是旋转+反射。
2. 它改变了坐标系的 Handedness（手性），将一个一致的、物理上正确的（通常是右手）坐标系系统，变成了一个不一致的、一半正常一半镜像的混合系统。
3. 优化算法（如BA）无法在这样一个存在根本性矛盾的系统中找到一致的解，为了最小化误差，它会产生扭曲的结果。
4. 在时序估计中，错误的手性会随着积分过程不断累积，导致轨迹发散。

# 二、所有坐标系

## 坐标系总览（按领域分类）

| 坐标系名称 | 主要应用领域 | 手性 | 关键特征 |
| --- | --- | --- | --- |
| **1. 世界坐标系 (World CS)** | 所有领域 | 通常为右手系 | 全局的、固定的参考基准 |
| **2. 相机坐标系 (Camera CS)** | CV, CG | **左手系** | Z轴指向成像方向，原点在光心 |
| **3. 图像坐标系 (Image CS)** | CV, CG | 2D | 原点在图像中心，物理单位（毫米） |
| **4. 像素坐标系 (Pixel CS)** | CV, CG | 2D | 原点在左上角，像素单位 |
| **5. 机体坐标系 (Body/Vehicle CS)** | 机器人、自动驾驶 | 通常为右手系 | 固定在机器人本体上的坐标系 |
| **6. 传感器坐标系 (Sensor CS)** | 机器人、多传感器融合 | 依传感器而定 | 描述雷达、IMU等传感器自身的姿态 |
| **7. 关节坐标系 (Joint CS)** | 机器人学 | 依模型而定 | 定义在每个机器人关节上 |
| **8. 物体坐标系 (Object/Local CS)** | CG, 机器人 | 依定义而定 | 固定在某个物体上，随物体移动 |

# 1、世界坐标系

- **介绍**：这是一个**全局的、固定的**参考框架。所有其他坐标系的位置和方向最终都是相对于世界坐标系来定义的。它是我们定义的“宇宙中心”。
- **手性**：**通常为右手系**（如X前，Y左，Z上；或X东，Y北，Z天）。
- **转换关系**：
  - 所有其他坐标系都可以通过一个**刚体变换**（旋转矩阵 `R` 和平移向量 `t`）转换到世界坐标系。
  - `P_world = R * P_local + t`

普通的自我定义的世界坐标系不予讨论，下面讨论一下大地测量坐标系的种类，用于在地球上描述点的位置，作为绝对位置的参考。

**大地坐标系基于一个假定的地球模型——参考椭球体（Reference Ellipsoid），它是一个尽可能接近大地水准面（Mean Sea Level）的规则数学曲面（一个旋转椭球体）**。最常见的椭球体是 WGS84（World Geodetic System 1984），这也是GPS系统所使用的标准。

| 椭球体参数 (WGS84) | 值 |
| --- | --- |
| 长半轴 (Semi-major axis) a | 6,378,137.0 m |
| 短半轴 (Semi-minor axis) b | ≈6,356,752.3142 m |
| 扁率 (Flattening) f | 1/298.257223563 |

## 1. ECEF，地心地固坐标系

- **定义**：
  - **原点** `(0,0,0)`：地球的质心。
  - **Z轴**：指向**国际参考极（IRP）**，即北极方向（与地球自转轴对齐）。
  - **X轴**：指向**本初子午线（0°经度）**与**赤道面**的交点。
  - **Y轴**：在赤道平面内，与X轴垂直，完成右手坐标系（指向90°东经方向）。
- **关键特性**：
  - **“地固”**意味着该坐标系**固定在地球上**，随地球一起自转。一个固定在地球上的点，其ECEF坐标是基本不变的（忽略极小的板块运动等）。
  - 它是一个**三维直角坐标系**，单位是米。
- **如何使用**：
  - 它是所有其他大地坐标系进行转换的**中间枢纽**和**计算基础**。
  - GNSS接收机接收的卫星信号最终会解算出一个ECEF坐标（如 `(X, Y, Z)`），再转换为更易理解的经纬高。
  - 用于需要精确计算两点间**直线距离**的场景（如卫星和地面站的几何关系）。

## 2. LLA，经纬高

- **定义**：
  - **经度 (Longitude, λ)**：本初子午面（通过格林尼治的经线面）与目标点经线面之间的夹角（-180° 到 +180°）。
  - **纬度 (Latitude, φ)**：目标点所在椭球面的法线与赤道平面的夹角（-90° 到 +90°）。
  - **海拔高度 (Altitude, h)**：目标点沿**椭球面法线方向**到**参考椭球面**的距离（单位：米）。注意：这不是到海平面的距离（那是“正高”，需要大地水准面模型）。
- **关键特性**：
  - 这是**人类最熟悉和使用最广泛**的全球定位系统。
  - 它是一个**球面坐标系**（单位：度、米），无法直接进行欧几里得几何运算（如加减、求距离）。
- **如何使用**：
  - **人机交互**：显示给用户看的绝对是LLA坐标。
  - **数据存储与交换**：几乎所有地理信息系统（GIS）和全球数据集（如OpenStreetMap）都使用LLA坐标。
  - **绝对位置参考**：为所有局部坐标系提供全局锚点。

## 3. UTM，通用横轴墨卡托投影坐标系

- **定义**：
  - UTM是**一种地图投影方法**，它将地球表面投影到一个二维平面上，从而获得**平面直角坐标** `(Easting, Northing)`，单位是米。
  - 为了控制投影变形，地球被划分为**60个经度带**（Zone），每个带跨6°经度。
    - **Zone编号**：从180°经线开始，编号1-60。
    - **例如**：北京大约在东经116.4°，其UTM带号为 `floor((116.4 + 180)/6) + 1 = 50`，即 Zone 50N。
  - 每个Zone内都有自己的原点和坐标：
    - **中央经线 (Central Meridian)**：每个Zone中央的经线，其东偏移（Easting）被设为 **500,000米**（“假东”）。
    - **赤道 (Equator)**：其北偏移（Northing）被设为 **0米**（北半球）或**10,000,000米**（南半球，“假北”）。
- **关键特性**：
  - 它将弯曲的地球表面**映射为2D平面**，从而可以在小范围内（一个Zone内）使用**欧几里得几何**进行精确计算（如距离、面积、角度）。
  - 在每个Zone中央经线附近，投影变形极小，精度非常高。
- **如何使用**：
  - **局部区域的高精度操作**：是自动驾驶、机器人、测绘、军事等领域**最常用的全局坐标系**。因为它提供了**米为单位的平面坐标**，可以直接输入给路径规划、控制等算法。
  - **数据融合**：将来自不同传感器的局部地图数据转换到UTM坐标系下，可以进行全局一致的地图构建和定位。


1. **LLA ↔ ECEF**：转换是精确的、可逆的，有严格的数学公式。
   - **正算 (LLA → ECEF)**：有标准公式，根据 `(φ, λ, h)` 和椭球参数 `(a, f)` 计算 `(X, Y, Z)`。
   - **反算 (ECEF → LLA)**：通常采用迭代算法（如Bowring's method）来求解 `(φ, λ, h)`，因为纬度方程是超越方程。
2. **ECEF/LLA ↔ UTM**：转换是**地图投影**过程，涉及复杂的数学变换（横轴墨卡托投影），通常通过**专业库**（如PROJ, GDAL, libproj）来实现。它**不是完全可逆的**，会引入投影变形（但在一个Zone内极小）。

## 2、相机坐标系

- **介绍**：以相机为中心定义的坐标系。**原点位于相机的光心（O）**。
  - **X轴**：向右
  - **Y轴**：向下
  - **Z轴**：**向前**（指向拍摄方向）
- **手性**：**左手系**（这是计算机视觉中的标准约定，因为它与图像像素坐标方向一致）。
- **转换关系（与世界坐标系）**：
  - 通过**外参（Extrinsics）**：旋转矩阵 `R` 和平移向量 `t` 进行转换。
  - `P_camera = R_{world->camera} * (P_world - t_{camera in world})`
  - 更常见的写法：`P_camera = R * P_world + t`，其中 `t` 是世界原点在相机坐标系中的坐标（`cv2.solvePnP` 的输出之一）。

## 3、图像/像素坐标系

这两个坐标系密切相关，通常放在一起讨论。

- **介绍**：
  - **图像坐标系**：是一个2D坐标系，**原点** `(u, v)` **位于图像平面的中心**（光学中心），坐标单位是**物理单位**（如毫米）。
  - **像素坐标系**：也是一个2D坐标系，**原点** `(0, 0)` **位于图像的左上角**，坐标单位是**像素**。
- **手性**：2D坐标系，不涉及手性。
- **转换关系**：
  1. **相机坐标系 -> 图像坐标系**（透视投影）：
     `x_image = (f_x * X_c) / Z_c`
     `y_image = (f_y * Y_c) / Z_c`
     （这里 `(x_image, y_image)` 是物理单位）
  2. **图像坐标系 -> 像素坐标系**：
     `u_pixel = x_image / dx + c_x`
     `v_pixel = y_image / dy + c_y`
     （其中 `dx`, `dy` 是一个像素的宽和高（毫米/像素），`(c_x, c_y)` 是主点（光学中心）的像素坐标。）
  3. **合并转换（相机坐标系 -> 像素坐标系）**：
     以上两步合并，并用**内参矩阵** `K` 表示：
     `[u; v; 1] = K * [X_c/Z_c; Y_c/Z_c; 1]`，其中
     `K = [[f_x, 0, c_x], [0, f_y, c_y], [0, 0, 1]]`
     (`f_x = f/dx`, `f_y = f/dy`, `f` 是焦距)

## 4、机体/传感器/关节/物体坐标系

没啥好说的，都是类似某个物体为中心的坐标系，手性一般看定义，右手系居多

## 5. 机体坐标系 (Body/Vehicle Coordinate System, BCS)

- **介绍**：固定在机器人或车辆本体上的坐标系。
  - **常见定义**：
    - **X轴**：向前（指向车头）
    - **Y轴**：向左（指向驾驶员侧）
    - **Z轴**：向上
  - **手性**：**右手系**。
- **转换关系（与世界坐标系）**：
  - 通过车辆的**全局位姿**（`R_vehicle_to_world`, `t_vehicle_in_world`）进行转换。
  - `P_world = R_vehicle_to_world * P_vehicle + t_vehicle_in_world`

## 6. 传感器坐标系 (Sensor Coordinate System)

- **介绍**：定义在某个传感器（如激光雷达、IMU、另一个相机）上的坐标系。描述该传感器相对于机体坐标系的安装位置和角度。
- **手性**：取决于传感器制造商的规定。**激光雷达通常为右手系**（X前，Y左，Z上）。**IMU** 的坐标系由其数据手册定义。
- **转换关系（与机体坐标系）**：
  - 通过**外参标定**得到的固定变换：旋转矩阵 `R_sensor_to_body` 和平移向量 `t_sensor_in_body`。
  - `P_body = R_sensor_to_body * P_sensor + t_sensor_in_body`
  - 这是多传感器融合（Camera-LiDAR, Camera-IMU）的核心。

## 7. 关节坐标系 (Joint Coordinate System)

- **介绍**：在机器人学中，基于**DH参数（Denavit-Hartenberg Parameters）** 或**标准模型**，在每个关节上定义的坐标系。用于通过正向运动学计算机器人末端执行器（End-Effector）的位置。
- **手性**：依模型而定，但通常有统一约定。
- **转换关系**：通过连续的齐次变换矩阵 `T_i^{i-1}` 从一个关节坐标系转换到下一个。

### 8. 物体坐标系 (Object/Local Coordinate System)

- **介绍**：固定在一个物体（如一个杯子、一辆车）上的坐标系。通常定义在物体的几何中心或底部中心。用于描述物体自身的姿态和其上的点。
- **手性**：依定义而定，通常与世界坐标系一致（右手系）。
- **转换关系（与世界坐标系）**：
  - 通过物体的**位姿**（`R_object_to_world`, `t_object_in_world`）进行转换。
  - `P_world = R_object_to_world * P_object + t_object_in_world`

# 三、坐标系的相同程度

### 🌈 最简洁版总结（你可以记这个）

| 级别 | 名称 | 手性 | 尺度 | 允许变换 | 数学群 |
|---|---|---|---|---|---|
| 0 | 完全相同 | 保持 | 相同 | 无 | Identity |
| 1 | 旋转等价 | 保持 | 相同 | 旋转 | SO(3) |
| 2 | 刚性等价 | 保持 | 相同 | 旋转+平移 | SE(3) |
| 3 | 相似等价 | 保持 | 不同 | 旋转+平移+缩放 | Sim(3) |
| 4 | 正交等价 | **可翻转** | 相同 | 旋转或反射 | O(3) |
| 4A | 反射等价 | **翻转** | 相同 | 反射/反演 | O(3)-SO(3) |
| 5 | 反射刚性等价 | **翻转** | 相同 | 反射+平移 | Euclidean+reflection |
| 6 | 反射相似等价 | **翻转** | 不同 | 反射+平移+缩放 | Sim(3)+reflection |
| 7 | 同类型 | 任意 | 任意 | 任意（结构同类即可） | — |