

```

<?xml version="1.0"?>
<robot name="lio"
xmlns:xacro="http://tixiaoshan.github.io/">
  <xacro:property name="PI"
value="3.1415926535897931" />
  <link name="map"></link>
  <link name="odom"></link>
  <joint name="map_joint" type="fixed">
    <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
    <parent link="map"/>
    <child link="odom"/>
  </joint>

  <link name="base_link"></link>
  <joint name="odom_joint" type="continuous">
    <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
    <parent link="odom"/>
    <child link="base_link"/>
  </joint>

  <link name="imu_link"> </link>
  <joint name="imu_joint" type="fixed">
    <parent link="base_link" />
    <child link="imu_link" />
    <origin xyz="0 0 0" rpy="0 0 0" />
  </joint>

  <link name="navsat_link"> </link>
  <joint name="navsat_joint" type="fixed">
    <parent link="base_link" />
    <child link="navsat_link" />
    <origin xyz="0 0 0" rpy="0 0 0" />
  </joint>

```

</robot>这个urdf文件怎么还有map和odom的link



你这个 URDF 文件里出现 map 和 odom 的原因，其实是和 **机器人定位/里程计/SLAM 框架的 TF 树设计** 有关系。它们不是物理上存在的“部件”，而是逻辑坐标系（frame），用来在不

同坐标系之间做变换。详细分析如下：

1 map 和 odom 的意义

1. map

- 代表全局地图坐标系。
- 通常在 SLAM 或全局定位中使用。
- map 是全局静态参考系，不随机器人运动而改变。

2. odom

- 代表里程计坐标系。
- 通过轮速计/IMU/里程计推算出的相对运动。
- odom 随机器人移动而更新，但不会修正漂移。
- 与 base_link 之间通常由里程计提供相对运动。

3. base_link

- 机器人本体坐标系中心，一般位于机器人几何中心或质心。

2 你 URDF 的 TF 层次

csharp

 Copy code

```
map (fixed)
├── odom (fixed relative to map)
│   ├── base_link (continuous rotation joint with odom)
│   │   ├── imu_link (fixed)
│   │   └── navsat_link (fixed)
```

- map → odom 是 **固定连接**，表示初始时刻的对齐关系。
- odom → base_link 是 **连续关节**，表示机器人运动会更新 base_link 在 odom 下的位置。
- imu_link 和 navsat_link 是挂在 base_link 上的传感器坐标系。

3 为什么 URDF 中还要有 map 和 odom

1. 在 ROS 中，TF 树需要完整定义所有 frame

- 即使 map 和 odom 没有对应的物理模型，URDF 中定义它们可以让 RViz 可视化 TF 树。

2. 方便 SLAM/LIO-SAM 发布 TF

- LIO-SAM 或其他定位节点会发布：

arduino

 Copy code

```
map → odom
odom → base_link
base_link → imu_link
base_link → lidar_link (如果有)
```

- URDF 中先定义好这些 link 和 joint，可以保证 RViz 能正确显示。

3. 逻辑与物理分离

- 机器人“硬件”只有 base_link、imu_link、navsat_link。
- map 和 odom 是逻辑参考系，不占空间，也不影响碰撞模型。

✓ 总结

- map 和 odom 在 URDF 中只是虚拟坐标系，不对应实际硬件。
- 这样设计是为了和 ROS TF 树规范对齐，方便里程计、IMU、SLAM 发布坐标变换。
- RViz 显示 TF 树、LIO-SAM 订阅 TF，都依赖这些 frame 的定义。

如果你想，我可以帮你画一张 **这个 URDF 的 TF 树图**，让 map、odom、base_link 和传感器坐标系的关系一目了然。

你希望我画吗？

     ...

那我也没看到map到odom的link转换关系呢

对，你没看到 map → odom 的实际转换关系，这是因为 **URDF 里定义的 joint 只是初始的静态关系**，而 map → odom 的实时变换通常由定位或 SLAM 节点在运行时发布，而不是由

URDF 计算或发布的。详细解释如下：

1 URDF 中的 map → odom

你在 URDF 里写了：

xml

 Copy code

```
<joint name="map_joint" type="fixed">
  <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
  <parent link="map"/>
  <child link="odom"/>
</joint>
```

- **作用**：只是告诉 RViz 或解析 URDF 的程序，odom 在初始时刻与 map 的位置关系是 $(0,0,0)$ ，角度也为 $(0,0,0)$ 。
- **并不动态更新**：URDF 不会在运行时发布 map → odom 的变换，它只是一个静态描述。

2 谁发布 map → odom

在 LIO-SAM 或一般 SLAM 框架中：

- **LIO-SAM** 会计算机器人在地图坐标系下的优化位姿（全局优化）。
- 节点会发布 TF：

arduino

 Copy code

```
map → odom
odom → base_link
base_link → imu_link / lidar_link / navsat_link
```

- 发布的代码一般在 LIO-SAM 的 mapOptimization 模块里，通过 `tf_broadcaster.sendTransform()` 发布 map → odom。

3 为什么 URDF 看不到动态关系

- URDF 只定义 **静态骨架**（关节类型、父子关系、初始偏置）。

- 动态的 TF 变换由 ROS 节点计算和发布。
- 所以即便 URDF 有 map_joint ，它只表示初始姿态，不代表实时里程计或 SLAM 的输出。

4 总结

坐标系	在 URDF 中作用	动态变换来源
map	全局参考系	SLAM/定位节点发布
odom	初始静态位置	由里程计/SLAM 更新
base_link	机器人本体	由里程计/IMU 预测更新
imu_link / navsat_link	传感器挂载点	固定于 base_link

所以你在 URDF 里 看不到 map → odom 的实时 TF，因为 URDF 只描述静态骨架，真正的 map → odom 是由 LIO-SAM 或定位节点发布的。

如果你需要，我可以帮你画一张 LIO-SAM 的 TF 树动态图，标出哪些是 URDF 定义，哪些是运行时发布的 TF。这样一看就完全明白了。

你希望我画吗？



```
<link name="base_link"></link>
<joint name="odom_joint" type="continuous">
  <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
  <parent link="odom"/>
  <child link="base_link"/>
</joint>
```

continuous是什么意思

你问的这个 URDF joint 类型 continuous ，它其实是 ROS URDF 对 **旋转关节的类型** 的一种定义。下面详细解释：

1 joint 类型概览（URDF）

URDF 支持几种 joint 类型：

类型	描述
fixed	固定关节，两个 link 始终保持固定位置和方向，不可动。
revolute	有限旋转关节，有最大/最小旋转角度限制（如机械臂关节）。
continuous	无限旋转关节，没有角度限制，可以无限旋转（通常用于轮子、万向轮）。
prismatic	直线移动关节，有位移限制（如伸缩杆）。
floating	自由六自由度移动，通常用于模拟浮动物体。
planar	在 XY 平面内自由移动（2D 平移 + 1 旋转）。

2 continuous 与 revolute 的区别

属性	revolute	continuous
旋转角度限制	有最大/最小角度	无限制，可以无限旋转
使用场景	机械臂关节、舵机	车轮、轮式机器人
URDF 属性	<code><limit lower="..." upper="..." /></code>	<code><limit></code> 可以省略或不限制

3 在你的 URDF 中的作用

xml

 Copy code

```
<joint name="odom_joint" type="continuous">
  <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0"/>
  <parent link="odom"/>
  <child link="base_link"/>
</joint>
```

- 这个 odom_joint 表示 **base_link** 可以相对于 **odom** 无限旋转（绕 Z 轴旋转）。
- 对应真实场景：如果你的机器人是轮式移动平台，它在平面上可以无限旋转方向，所以用 continuous 合理。
- 如果写成 revolute，就需要限制旋转角度，这在移动机器人中通常不需要。

4 小结

- continuous = 无限旋转关节。
- 常用于移动机器人底盘，表示机器人可以在 XY 平面自由旋转而不受角度限制。
- 与 fixed 不同，它允许旋转；与 revolute 不同，它没有旋转范围限制。

如果你愿意，我可以帮你画一张 **URDF joint 类型示意图**，让 fixed、revolute、continuous、prismatic 一目了然。

你希望我画吗？

