MindPalace Odin1 Lite 产品使用手册

文档版本：V1.1.1 · 发布日期：2026-11-06

1. 产品概述

1.1 产品简介

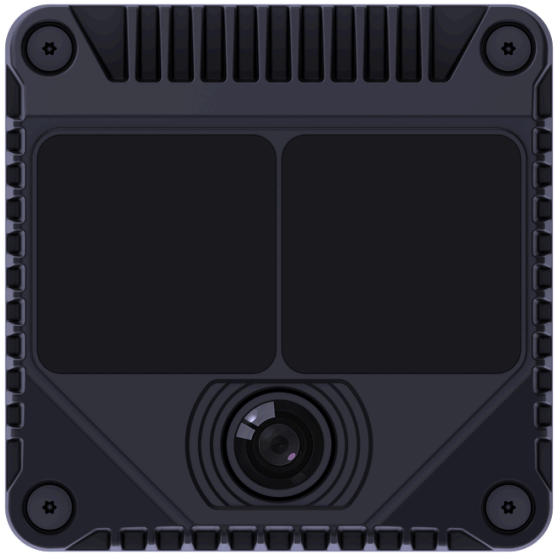
MindPalace Odin1 Lite：紧凑型空间记忆模组

> **为机器人、无人机及各类前研智能设备而生**

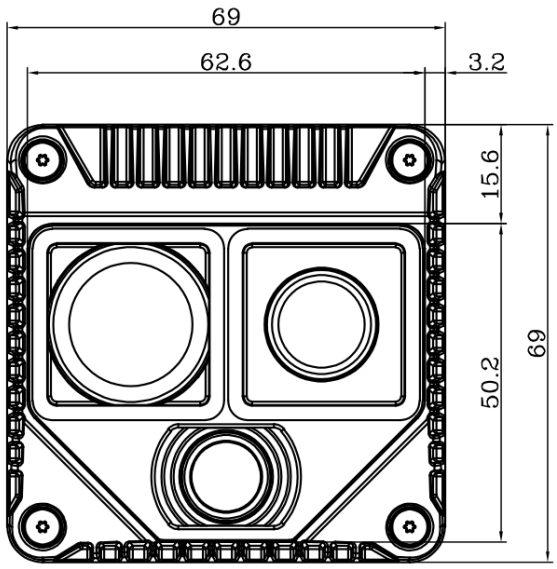
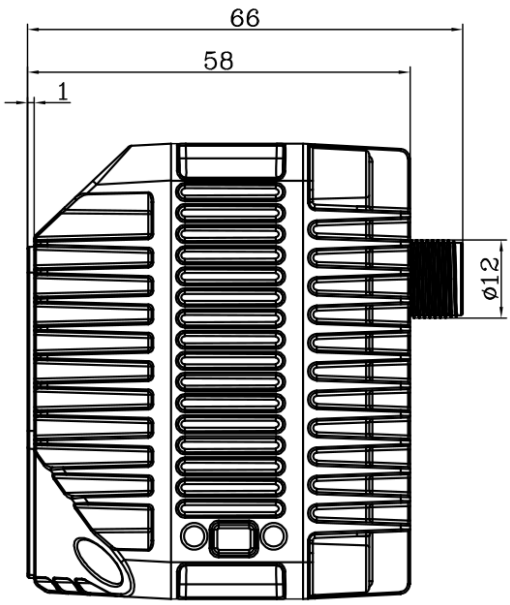
深度融合 LiDAR 与视觉传感器，实现高精度、广视角的实时三维感知，内置 MindSLAM^{TM} 高性能融合 SLAM 算法，实现持久空间记忆。适合对尺寸和成本敏感的机器人、自主导航平台与智能边缘设备的快速集成与部署，可在室内外复杂环境中稳定运行。

1.2 外观尺寸

产品外观



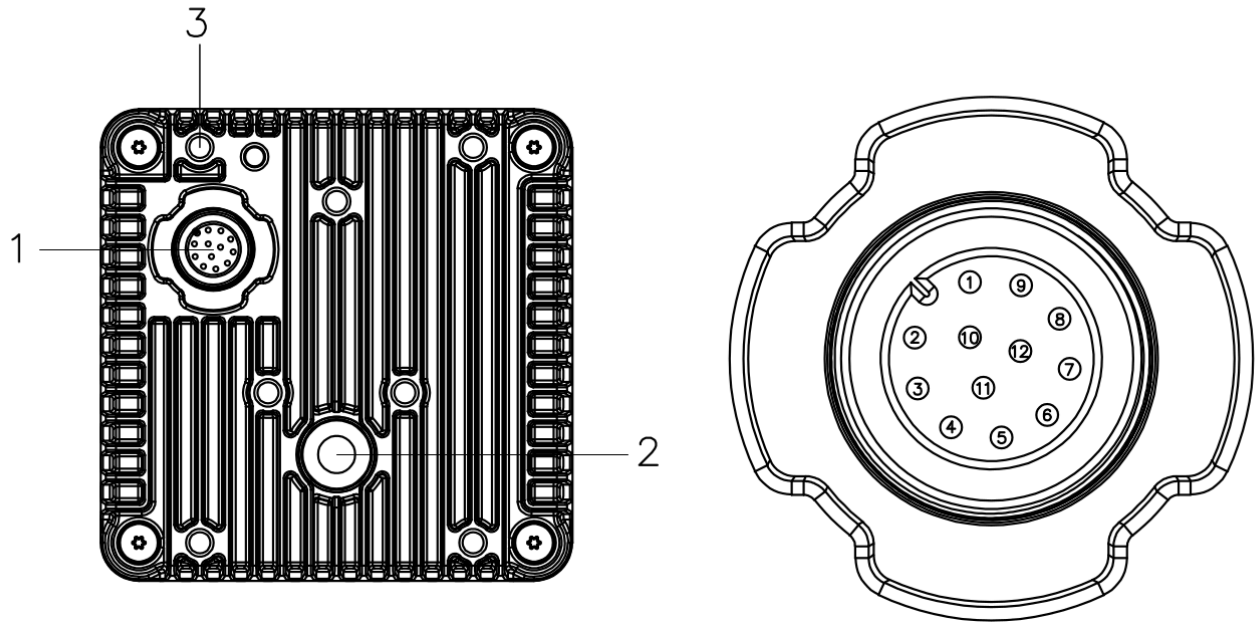
产品尺寸

前视图	右视图
	

← 返回首页 | 下一章：硬件说明 →

2. 硬件说明

2.1 物理接口



1. 12芯航空插座
2. 防水透气膜
3. 电源指示灯

Odin1 Lite 仅有一个航插接口，通过随机配件 **电源数据一分二航插线** 同时实现供电与数据传输：

序号	接口类型	线缆	功能描述
1	航空插头（机端）	电源数据一分二航插线	同时承担供电 + 千兆以太网数据传输
2	DC 头（线端）	5.5×2.1mm DC 公头	接 12V/2A 适配器
3	RJ45（线端）	千兆以太网	接 PC 网口或交换机

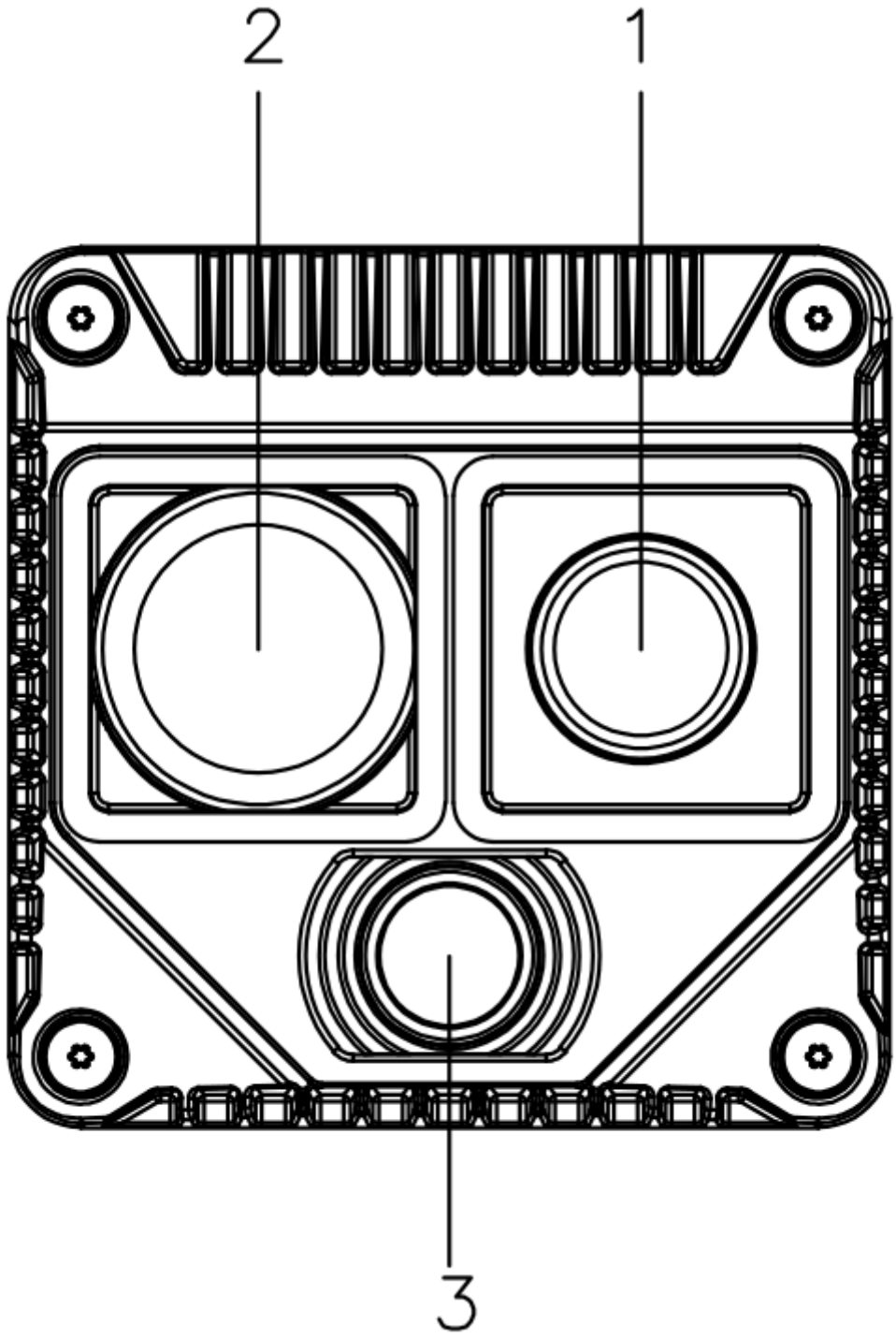
2.2 接口定义

Odin1 Lite 采用 **12 芯航空插座**，同时承担供电、千兆网口与外部同步信号，管脚线序如下：

管脚	信号	属性	功能
1	DC+	数据	千兆网口信号
2	VIN	Power	供电正极
3	DB−	数据	千兆网口信号
4	DB+	数据	千兆网口信号
5	DD+	数据	千兆网口信号
6	DD−	数据	千兆网口信号
7	DA+	数据	千兆网口信号
8	DA−	数据	千兆网口信号
9	DC−	数据	千兆网口信号
10	GPIO_OUT	GPIO	外部同步触发信号输入
11	GND	Power	供电负极
12	FSYNC_OUT	GPIO	同步触发信号输出

> 千兆网口采用 4 对差分线：DA±、DB±、DC±、DD±，对应 1000BASE-T 标准。

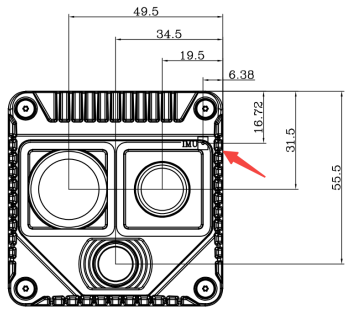
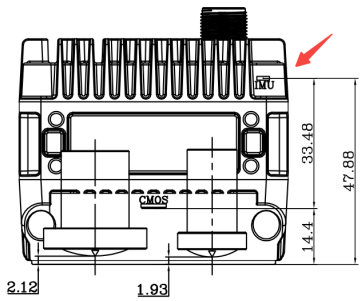
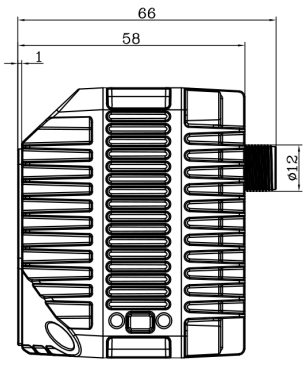
2.3 核心部件



序号	部件	说明
1	激光发射端（TX）	dToF TX
2	激光接收端（RX）	dToF RX
3	RGB 相机	1280×1088

2.4 IMU 位置关系

> **说明**：IMU 与 Body_base_link 为同一物理位置，仅命名不同。以下图纸展示 IMU 在设备外壳内部的安装位置及与各传感器之间的空间关系。

正视图	侧视图	俯视图
		

项目	说明
IMU 位置	IMU 位于设备内部，紧固于主结构件上，与 LiDAR 光学中心及 RGB 相机（CMOS）处于同一刚体结构
Body_base_link 坐标系	IMU 坐标系即 Body_base_link 坐标系，原点位于 IMU 芯片几何中心
与外壳关系	IMU 紧固于设备主结构件上，外壳变动不影响 IMU 与 LiDAR / 相机之间的外参
安装方向	设备支持任意方向安装，IMU 会自动校准到重力系

2.5 网络参数

项	值	备注
设备出厂 IP	`192.168.1.251`	**支持修改**
通信协议	UDP（数据）+ HTTPS / mTLS（文件传输）	端口 60000–60010、59000–59005、8080
HTTP Dashboard	`http://192.168.1.251:8080`	设备状态 / 标定 / OTA / 网络配置

2.6 数模文件

> **TODO：上传 Odin1 Lite 航插头数模文件（stp）并更新链接**

- 下载 Odin1 Lite 航插头数模文件（stp）
- 下载 Odin1 Lite 航插头数模文件弯头（stp）

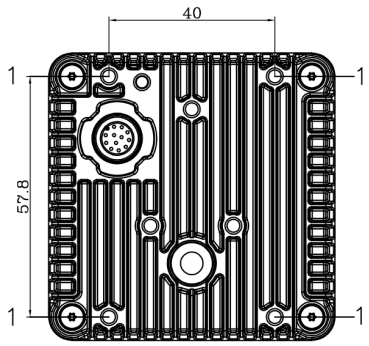
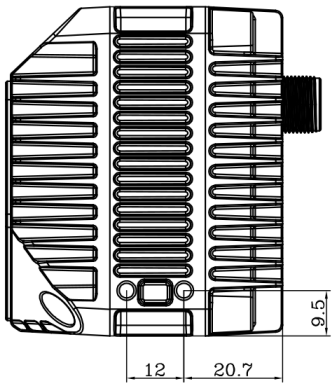
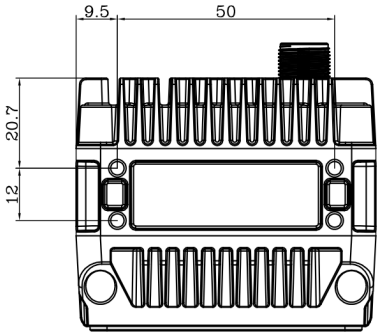
← 上一章：产品概述 | 下一章：安装指南 →

3. 安装指南

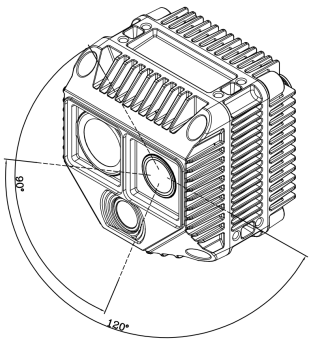
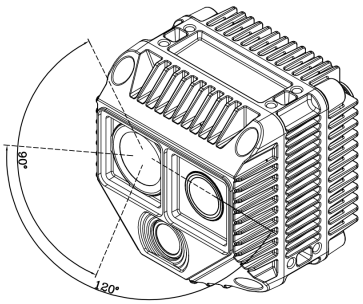
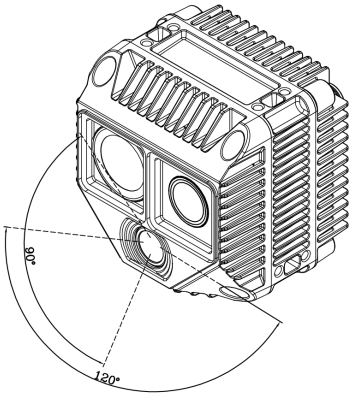
正式安装 Odin1 Lite 前，请阅读以下注意事项：

1. **清洁安装表面与窗口：** - 安装前，务必清除安装位置和激光雷达光学窗口上的灰尘、油污等脏污。建议使用干净的气吹除尘，再用无绒布（如镜头布）沾少许清水轻轻擦拭窗口，确保完全清洁。残留污渍会影响测量精度。请勿使用有刺激性的液体进行清理。
2. ⚠️ **确保视野无遮挡：** - 安装时，传感器前方视场角（FOV）区域不允许有任何遮挡物。特别注意：即使是透明的玻璃、亚克力板等，安装在传感器正前方也会显著干扰激光信号，导致性能下降。
3. **灵活的安装方向（含优化建议）：** - 本产品支持任意方向安装（包括倒装）。 - 设备会自动适应安装姿态（如倒装、倾斜），内部传感器会进行校准到重力系。
4. ⚠️ **禁止施加额外负载：** - 设备的机身（尤其是散热外壳和光学窗口）不能承受额外重量或压力。请仅通过设计好的安装接口（如底部的螺纹孔）进行固定。
5. ⚠️ **保障散热空间：** - 设备四周需留出足够的空间用于散热。要求：设备本体与周围物体的最小净空距离 ⩾ 10mm，确保空气能顺畅流通，防止过热。
6. **稳固安装：** - 强烈建议将设备安装在结实的支架上，传感器下底部距离物体 0.2 米以上，以防遮挡视野。
7. ⚠️ **严禁正负极反接：** - 请仔细阅读使用说明及线序定义。

3.1 机械安装

背部M3安装孔	左右两侧M3安装孔	上下两侧M3安装孔
		

有效视场角（FOV）范围：

TX	RX	RGB
		

3.2 数模文件

- 下载 Odin1 Lite 数模文件（stp）
- 下载 Odin1 Lite 带视场角数模文件（stp）

← 上一章：硬件说明 | 下一章：电源与连接 →

4. 电源与连接

4.1 外部供电系统设计指南

1. **电压范围要求**

Odin1 Lite 标称工作电压为 **DC 9-24V**，建议使用 **12V / 2A** 直流适配器（≥ 24W 功率裕量），否则可能因瞬时电流不足导致设备无法正常启动或工作不稳定。

2. **电源适配注意事项**

- 通过 **电源数据一分二航插线** 的 DC 头接入适配器，DC 头规格为 **5.5×2.1mm**。
- Odin1 Lite **出厂不标配电源适配器**，用户需自行准备符合规格的 12V 适配器。

3. **电压浪涌防护警告**

⚠️ 电源线路上超出额定上限的电压波动可能导致设备永久性损坏，建议在供电回路中增加过压保护电路。

4. **功耗特性说明**

工作状态	功率特性
稳定运行	典型 ≈ **11** W
冷启动峰值	最大 **24** W

5. **电源选型关键参数**

- 持续输出能力 ≥ **2A**
- 推荐功率裕量 ≥ **24W**
- 过压保护阈值建议 ≤ **26V**

4.2 连接步骤

Odin1 Lite 通过以太网与 PC 通信。

1. 将 **电源数据一分二航插线** 的航插头按卡槽方向插入 Odin1 Lite 接口，并锁紧螺纹环。
2. **DC 头** 接入 12V/2A 适配器并通电；设备指示灯常亮代表上电成功。
3. **RJ45 网线** 接入 PC 网口（或交换机）。
4. 将 PC 网卡 IP 配置为 `192.168.1.x/24`（与设备 `192.168.1.251` 同网段）。
5. 通过 `ping 192.168.1.251` 确认通讯正常后即可启动驱动或访问 HTTP Dashboard。

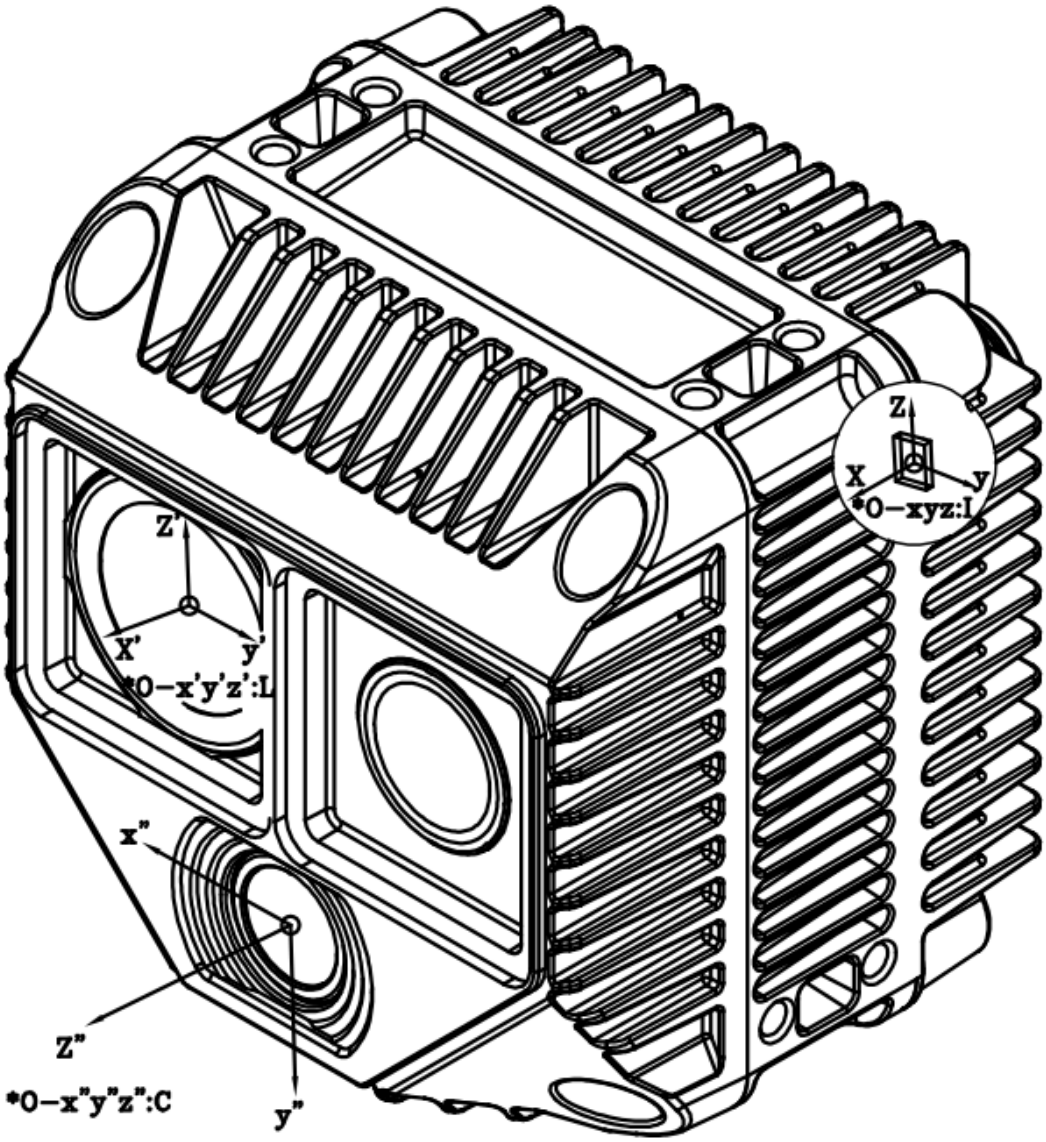
> 🔧 关于网络配置的详细步骤参见 7. 快速启动 §7.2.3 网络配置。

← 上一章：安装指南 | 下一章：数据输出 →

5. 数据输出

5.1 坐标系定义

Odin1 Lite 直角坐标的定义如下图所示，**I** 为 IMU（即 Body，二者为同一物理位置，仅命名不同）坐标系，**L** 为点云坐标系，**C** 为相机坐标系。



5.2 外参描述

外参符号说明

下面使用 $\mathbf{T}_{\mathrm{B}}^{\mathrm{A}}$ 来表示坐标转换，具体含义为：

$$\mathbf{P}_{\mathrm{A}} = \mathbf{T}_{\mathrm{B}}^{\mathrm{A}} \cdot \mathbf{P}_{\mathrm{B}}$$

其中 P_A 和 P_B 分别表示物体在 A 坐标系和 B 坐标系下的坐标。

Camera - LiDAR

外参 $\mathbf{T}_{\text{lidar}}^{\text{camera}}$ 不同机器存在差异，请在驱动程序启动后，在 `config/calib.yaml` 中获取（`Tcl_0`），具体信息请查看驱动。

Body - LiDAR

> **说明**：此处的 **Body** 即原来的 **IMU**，仅命名（表述）不同，实际对应同一个物理位置。

Body（原 IMU）到 LiDAR 的外参 $\mathbf{T}_{\text{lidar}}^{\text{body}}$（标定文件中的 `body_T_lidar`）如下：

$$\mathbf{T}_{\text{lidar}}^{\text{body}} = \begin{bmatrix} 0.9999543679 & -0.0086157257 & 0.0041269153 & -0.0305398721 \\ 0.0086800624 & 0.9998369581 & -0.0158339418 & 0.0234679052 \\ -0.0039898215 & 0.0158690412 & 0.9998661185 & 0.0333162287 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

其他

- 如需 Camera 到 Body 的外参，请根据上述自行计算。

5.3 内参说明

内参主要讨论相机内参，LiDAR 内参不考虑。

5.3.1 Odin1 Lite 相机内参

- 获取内参的方法：
- **方式一（运行时）**：驱动启动后会从设备拉取标定文件并保存到 `~/catkin_ws/src/ros_driver/config/camera_calib_<SN>.yaml`，每台设备各异。
- **方式二（ROS 服务）**：运行 `ros2 service call /manifold/device0/get_calibration odin_ros_driver_rev1/srv/GetCalibration "{}"` 直接获取 YAML 字符串。实际服务前缀请以启动日志或 `ros2 service list` 为准。
- **方式三（HTTP Dashboard）**：浏览器访问 `http://192.168.1.251:8080`，在 Dashboard → 标定文件管理中下载。

```
# O1L-P040100047
cam_num: 1
img_topic_0: /camera/rgb
Tcl_0: [
  0.01814, -0.99980, -0.00813, 0.00579,
  -0.00071, 0.00812, -0.99997, -0.03109,
  0.99984, 0.01815, -0.00057, -0.01629,
  0.00000, 0.00000, 0.00000, 1.00000
]

cam_0_640_544:
        cam_model: FishPoly
        image_width: 640
        image_height: 544
        k2: 3.9309765393149745e-02
        k3: 1.5817608030575755e-01
        k4: 5.8836968291029024e-01
        k5: -8.2827749950756613e-01
        k6: 3.8543134263557260e-01
        k7: -6.3564147672175983e-02
        p1: 0.
        p2: 0.
        A11: 2.3172737533779440e+02
        A12: -4.1424461677121709e-02
        A22: 2.3167070553813502e+02
        u0: 3.1331230979976016e+02
        v0: 2.7959256780073559e+02
        isFast: 0
        numDiff: 3000
        maxIncidentAngle: 120

cam_0_1280_1088:
        cam_model: FishPoly
        image_width: 1280
        image_height: 1088
        k2: 3.9309765393149745e-02
        k3: 1.5817608030575755e-01
        k4: 5.8836968291029024e-01
        k5: -8.2827749950756613e-01
        k6: 3.8543134263557260e-01
        k7: -6.3564147672175983e-02
        p1: 0.
        p2: 0.
        A11: 4.6345475067558880e+02
        A12: -8.2848923354243417e-02
        A22: 4.6334141107627005e+02
        u0: 6.2662461959952032e+02
        v0: 5.5918513560147119e+02
        isFast: 0
```

```
        numDiff: 3000
        maxIncidentAngle: 120
```

5.3.2 相机模型描述

Odin1 Lite RGB 相机模型与内参

本文档描述 Odin1 Lite RGB 相机的投影模型及内参参数，供工程参考。

1 相机投影模型

Odin1 Lite RGB 使用**多项式鱼眼投影模型**描述大视场成像。去畸变后，该模型可简化为标准**针孔模型**。

2 多项式鱼眼模型

2.1 投影公式

给定相机坐标系下的三维点 $\mathbf{P_c} = (X, Y, Z)^T$，计算入射半径：

$$r = \sqrt{X^2 + Y^2}$$

计算入射角：

$$\theta = \arccos\left(\frac{Z}{\sqrt{X^2 + Y^2 + Z^2}}\right)$$

多项式径向映射：

$$\theta_d = \theta + k_2\theta^2 + k_3\theta^3 + k_4\theta^4 + k_5\theta^5 + k_6\theta^6 + k_7\theta^7$$

投影到畸变平面：

$$x_d = \frac{\theta_d}{r} \cdot X, \quad y_d = \frac{\theta_d}{r} \cdot Y$$

2.2 仿射变换

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

参数	说明
k2–k7	多项式畸变系数
A11	X 轴焦距（水平缩放）
A12	非正交补偿（skew）
A22	Y 轴焦距（垂直缩放）
u0, v0	主点坐标

2.3 去畸变后的针孔模型

$$u = f_x \frac{X}{Z} + c_x, \quad v = f_y \frac{Y}{Z} + c_y$$

2.4 坐标系约定

Odin1 Lite RGB 遵循 OpenCV 相机坐标系：X → 右，Y → 下，Z → 前（右手坐标系）。

2.5 工程注意事项

- 去畸变不改变外参。
- 高精度重投影请使用多项式鱼眼模型。
- 检测、SLAM、融合等算法框架请使用去畸变后的针孔模型。
- 当 $\theta \to 0$ 时可能出现数值不稳定（$r \to 0$ 导致除零）。

5.4 数据说明

原始点云（`cloud/raw`，分辨率 256×192）PointCloud2 字段如下：

字段	数据类型	说明
`x`	FLOAT32	X 轴，单位：米
`y`	FLOAT32	Y 轴，单位：米
`z`	FLOAT32	Z 轴，单位：米
`intensity`	UINT8	反射率，范围 0–255
`confidence`	UINT8	点置信度，范围 0–255，越高越可信。建议阈值：室内场景 4，可通过 `confidence_threshold` 参数调整
`offset_time`	FLOAT32	相对于帧基准时间戳的时间偏移，单位：秒

> 注：当点的 `confidence` 低于 `confidence_threshold` 时，驱动会将该点的 x/y/z 置为 0。 `point_step` = 18 字节。

5.5 ROS 话题与服务一览

> 包名： `odin_ros_driver_rev1` ，主节点： `odin_ros_driver_node` ，后处理节点： `post_process_node` 。
>
> 所有话题和服务均带有逐设备前缀 `{prefix} = /{topic_prefix}/device{N}/` （默认 `/manifold/device0/` ）。下表中仅列出后缀部分。
>
> ℹ️ 设备型号不再出现在话题/服务路径中；型号与 SN 由 `/{topic_prefix}/driver/device_online` 的 JSON 字段发布。
>
> ℹ️ 话题 / 字段名中出现的 `imu` 与本文档所称 **Body** 为同一概念（同一物理位置，仅命名不同）。

5.5.1 主驱动节点（odin_ros_driver_node）

话题后缀	类型	说明
`cloud/raw`	`sensor_msgs/PointCloud2`	原始点云（256×192）
`cloud/slam`	`sensor_msgs/PointCloud2`	SLAM 点云
`camera0/compressed`	`sensor_msgs/CompressedImage`	左 RGB 相机 JPEG 压缩图像
`camera1/compressed`	`sensor_msgs/CompressedImage`	右 RGB 相机 JPEG 压缩图像（仅在双目设备有效）
`gray`	`sensor_msgs/Image`	灰度图像（mono8）
`imu`	`sensor_msgs/Imu`	IMU（加速度 + 角速度）
`odometry_lf`	`nav_msgs/Odometry`	低频里程计（含位姿协方差）
`odometry_hf`	`nav_msgs/Odometry`	高频里程计（含位姿协方差）

此外，主驱动还会在驱动级命名空间 `/{topic_prefix}/driver/` 下发布以下话题（不带设备前缀，所有设备共享）：

话题	类型	说明
`/{topic_prefix}/driver/device_online`	`std_msgs/String`	设备上线事件（latched，JSON 含 SN/型号/prefix/IP）
`/{topic_prefix}/driver/device_offline`	`std_msgs/String`	设备下线事件（latched，JSON 含 SN/型号）
`/{topic_prefix}/driver/resolution_change`	`std_msgs/String`	分辨率变更事件（latched）

5.5.2 后处理与功能节点

话题后缀	类型	说明
`camera0/raw`	`sensor_msgs/Image`	左 RGB 相机解码后图像
`camera1/raw`	`sensor_msgs/Image`	右 RGB 相机解码后图像（仅在双目设备有效）
`camera0/undistort`	`sensor_msgs/Image`	左 RGB 相机去畸变图像
`camera1/undistort`	`sensor_msgs/Image`	右 RGB 相机去畸变图像（仅在双目设备有效）
`camera0/rect`	`sensor_msgs/Image`	校正后左图像
`camera1/rect`	`sensor_msgs/Image`	校正后右图像（仅在双目设备有效）
`camera0/camera_info`	`sensor_msgs/CameraInfo`	校正后左相机内参
`camera1/camera_info`	`sensor_msgs/CameraInfo`	校正后右相机内参（仅在双目设备有效）
`cloud/render`	`sensor_msgs/PointCloud2`	着色点云（点云 + 图像融合，Lidar 坐标系）
`reprojection/overlay`	`sensor_msgs/Image`	点云重投影到相机图像的叠加结果
`pointcloud2depth/depth`	`sensor_msgs/Image`（32FC1）	稠密深度图，单位米，`0` 表示无效像素
`pointcloud2depth/depth_color`	`sensor_msgs/Image`（bgr8）	JET 配色深度图
`pointcloud2depth/depth_overlay`	`sensor_msgs/Image`（bgr8）	深度图与相机图像叠加结果

> ℹ️ Odin1 Lite 为单目 RGB。驱动代码内保留的 `camera1/*`、`stereo/*` 等双目 / 视差点云相关话题在 Odin1 Lite 上不会发布数据。

5.5.3 点云压缩传输

`cloud/raw` 与 `cloud/slam` 的基础话题始终为标准 `sensor_msgs/PointCloud2`，可直接在 RViz 中订阅。安装压缩插件并重启驱动后，会额外提供 `/zlib`、`/zstd`、`/draco` 压缩子话题：

```
# ROS2 Humble；请按实际 ROS 发行版替换 humble
sudo apt install ros-humble-point-cloud-transport-plugins
```

> RViz 无法直接显示压缩点云；如需可视化，须使用 `point_cloud_transport republish` 解压为普通 `PointCloud2`。仅使用未压缩点云时无需安装插件。

5.5.4 服务

服务后缀	类型	说明
`get_calibration`	`odin_ros_driver_rev1/srv/GetCalibration`	获取相机标定 YAML 内容
`set_sensor_mode`	`odin_ros_driver_rev1/srv/SetSensorMode`	切换 Sensor Mode（0=HDR / 1=High Peak / 2=Low Peak / 3=Adaptive）
`reset_initial_pose`	`odin_ros_driver_rev1/srv/ResetInitialPose`	重置 SLAM 初始位姿、地图和因子图
`get_adaptive_param` / `set_adaptive_param`	`GetAdaptiveParam` / `SetAdaptiveParam`	查询 / 设置自适应模式参数，仅 Adaptive 模式生效
`get_sync_mode` / `set_sync_mode`	`GetSyncMode` / `SetSyncMode`	查询 / 设置默认时间同步模式；设置后需重启设备
`get_ptp_sync_status`	`odin_ros_driver_rev1/srv/GetPtpSyncStatus`	查询 PTP 同步状态、偏移、链路延迟与频率调整
`restore_default_config`	`odin_ros_driver_rev1/srv/RestoreDefaultConfig`	恢复设备出厂配置；执行后需重启设备
`device_mode`	`odin_ros_driver_rev1/srv/DeviceMode`	设置设备模式（0=Standby / 1=Normal / 2=Upgrade / 3=SLAM / 4=Relocation / 5=ReMap）
`save_map`	`odin_ros_driver_rev1/srv/SaveMap`	保存闭环地图到主机文件
`load_map`	`odin_ros_driver_rev1/srv/LoadMap`	从主机文件加载地图用于重定位或 ReMap
`exploy_map`	`odin_ros_driver_rev1/srv/ExployMap`	将当前闭环地图导出到主机文件（具体格式由插件实现）
`set_diag_monitor`	`odin_ros_driver_rev1/srv/SetDiagMonitor`	启用 / 关闭 1Hz 帧率与延迟诊断，并发布 `/diagnostics`
`set_channel_config` / `get_channel_config`	`SetChannelConfig` / `GetChannelConfig`	批量设置 / 查询主驱动 A 层数据通道开关
`set_post_process_config` / `get_post_process_config`	`SetPostProcessConfig` / `GetPostProcessConfig`	批量设置 / 查询后处理 B 层开关；服务前缀为 `/{topic_prefix}/postprocess/`

5.6 ROS Driver 获取

Odin1 Lite ROS 驱动：https://git.manifoldtech.cn/manifoldtech-sdk/odin_ros_driver2

Odin1 Lite sdk:https://git.manifoldtech.cn/manifoldtech-sdk/odin-sdk2

详细部署与运行步骤请参见 7. Odin1 Lite 快速启动（Ubuntu）。

← 上一章：电源与连接 | 下一章：建图与重定位建议调用流程 →

6. 建图与重定位 API

6.1 文档说明

本文档介绍 Odin1 Lite 建图（SLAM）与重定位（Relocalization）功能的 ROS Service 调用方式及典型工作流程。

> 重定位由与固件版本匹配的 ROS Driver 和 `manifold-lc-sdk` 算法 deb 包共同提供。仅部署 ROS Driver 时可获取传感器数据，无法运行建图和重定位算法；算法 deb 的安装方式参见 9. 重定位功能使用说明。

6.2 概念说明

术语	说明
设备	Odin1 Lite 模组
主机	连接 Odin1 Lite 并运行 ROS 驱动的 PC / 机器人
Device Mode	设备工作模式，通过 `device_mode` 服务切换
Normal（1）	正常里程计模式，设备输出 odom / 点云等数据
SLAM（3）	建图模式，算法边运行边构建回环地图
Relocation（4）	重定位模式，加载已保存地图进行位置重定位
ReMap（5）	重定位 + 增量建图模式，在已载入地图基础上扩展地图
地图文件	SLAM / ReMap 模式下由算法生成的二进制地图文件（ `.bin` ）

6.3 相关 ROS Service

> 所有服务前缀为 `/{topic_prefix}/device{N}/` （默认 `/manifold/device0/` ）。
>
> ℹ️ 设备型号不再出现在话题/服务路径中；型号与 SN 由 `/{topic_prefix}/driver/device_online` 的 JSON 字段发布。

服务	类型	说明
`device_mode`	`DeviceMode`	切换工作模式：0=Standby / 1=Normal / 2=Upgrade / 3=SLAM / 4=Relocation / 5=ReMap
`save_map`	`SaveMap`	保存当前回环地图到指定路径
`load_map`	`LoadMap`	加载已保存的地图文件用于重定位或 ReMap
`exploy_map`	`ExployMap`	将当前闭环地图导出到主机文件（具体格式由插件实现）

服务定义

DeviceMode.srv

```
uint8 mode   # 0=Standby 1=Normal 2=Upgrade 3=SLAM 4=Relocation 5=ReMap
---
bool   success
int32  error_code   # 0=OK, 其它为 SDK OdinResult 枚举
string message
```

SaveMap.srv

```
string path   # 输出地图文件路径（主机本地路径）
---
bool   success
int32  error_code
string message
```

LoadMap.srv

```
string path   # 输入地图文件路径（主机本地路径）
---
bool   success
int32  error_code
string message
```

6.4 典型流程

6.4.1 建图流程

步骤 1：启动驱动

```
# ROS2
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py

# ROS1
roslaunch odin_ros_driver_rev1 odin1Lite.launch
```

步骤 2：切换到 SLAM 模式（开始建图）

> 执行前请先进入 ROS 工作目录并 source 环境：```bash cd ~/catkin_ws source install/setup.bash # ROS2
>
> # source devel/setup.bash # ROS1
>
> ```

```
# ROS2
ros2 service call /manifold/device0/device_mode \
  odin_ros_driver_rev1/srv/DeviceMode "{mode: 3}"

# ROS1
rosservice call /manifold/device0/device_mode "{mode: 3}"
```

此时设备进入 SLAM 建图模式，可移动设备开始建图。

步骤 3：保存地图

建图完成后，调用 `save_map` 将地图保存到主机本地文件：

```
# ROS2
ros2 service call /manifold/device0/save_map \
  odin_ros_driver_rev1/srv/SaveMap "{path: '/tmp/odin_map.bin'}"

# ROS1
rosservice call /manifold/device0/save_map "path: '/tmp/odin_map.bin'"
```

> 保存地图不会停止建图，可继续运行并再次保存。

步骤 4：停止建图（可选）

切回 Normal 模式：

```
# ROS2
ros2 service call /manifold/device0/device_mode \
  odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"

# ROS1
rosservice call /manifold/device0/device_mode "{mode: 1}"
```

6.4.2 重定位流程

前提：已通过上述建图流程保存了地图文件。

步骤 1：切换到 Relocation 模式

> 执行前请先进入 ROS 工作目录并 source 环境：```bash cd ~/catkin_ws source install/setup.bash # ROS2
>
> source devel/setup.bash # ROS1
>
> ```

```
# ROS2
ros2 service call /manifold/device0/device_mode \
  odin_ros_driver_rev1/srv/DeviceMode "{mode: 4}"

# ROS1
rosservice call /manifold/device0/device_mode "{mode: 4}"
```

步骤 2：加载地图

```
# ROS2
ros2 service call /manifold/device0/load_map \
  odin_ros_driver_rev1/srv/LoadMap "{path: '/tmp/odin_map.bin'}"

# ROS1
rosservice call /manifold/device0/load_map "path: '/tmp/odin_map.bin'"
```

加载成功后，设备开始运行重定位算法。请根据应用侧的定位状态与位姿输出确认重定位结果。

步骤 3：停止重定位

```
# ROS2
ros2 service call /manifold/device0/device_mode \
  odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"

# ROS1
rosservice call /manifold/device0/device_mode "{mode: 1}"
```

6.4.3 ReMap 流程（增量建图）

ReMap（`mode: 5`）在已保存地图的基础上进行重定位，同时扩展地图。适用于需要在已有地图中追加新区域的场景。

步骤 1：切换到 ReMap 模式并加载已有地图

```
# ROS2
ros2 service call /manifold/device0/device_mode \
  odin_ros_driver_rev1/srv/DeviceMode "{mode: 5}"

ros2 service call /manifold/device0/load_map \
  odin_ros_driver_rev1/srv/LoadMap "{path: '/tmp/odin_map.bin'}"
```

在 RViz 中将 **Fixed Frame** 设为 `device0/map` 即可查看已载入地图。

步骤 2：移动设备采集新区域

算法会在旧地图基础上进行定位，并增量构建新的地图内容。

步骤 3：保存扩展后的地图

```
ros2 service call /manifold/device0/save_map \
  odin_ros_driver_rev1/srv/SaveMap "{path: '/tmp/odin_map_extended.bin'}"
```

> 增量保存后的地图通常比原始地图更大。

步骤 4：停止 ReMap

```
ros2 service call /manifold/device0/device_mode \
  odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"
```

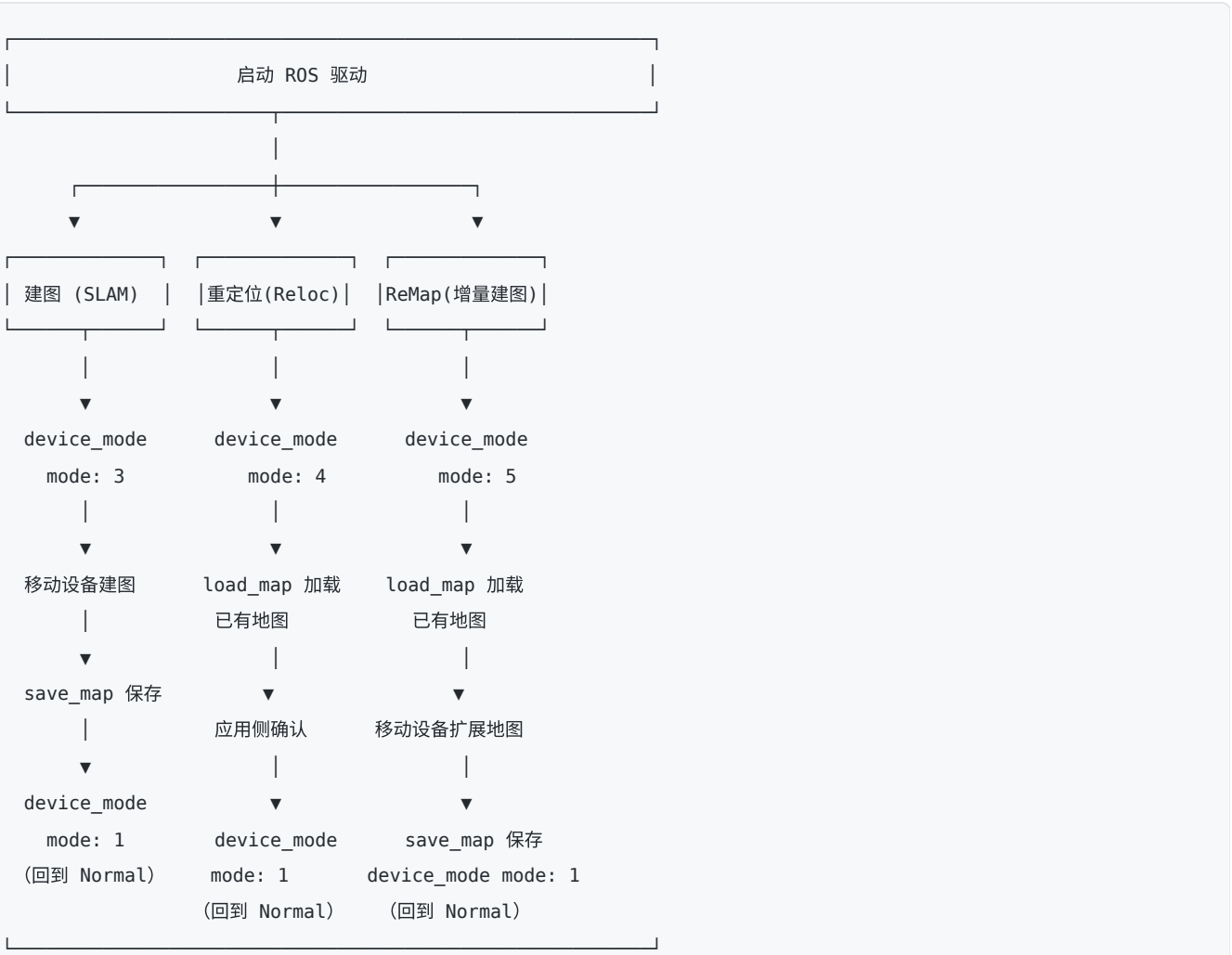
6.4.4 loop-closure 可视化话题

在 SLAM / Relocation / ReMap 模式下，算法会发布以下 loop-closure 可视化话题：

话题后缀	类型	说明
`loop_closure/optimization_busy`	`std_msgs/Bool`	优化器是否正在运行
`loop_closure/optimization_status_marker`	`visualization_msgs/Marker`	文本标记显示 `IDLE` / `OPTIMIZATION`
`loop_closure/optimized_path`	`nav_msgs/Path`	`local_map` 坐标系下优化后的轨迹
`loop_closure/optimized_cloud`	`sensor_msgs/PointCloud2`	`local_map` 坐标系下优化后的全局点云
`loop_closure/map_cloud`	`sensor_msgs/PointCloud2`	`map` 坐标系下加载的地图点云

> 这些话题由 `manifold-lc-sdk` 算法包产生；仅当算法 deb 包装好且设备处于 SLAM / Relocation / ReMap 模式时才会出现。

6.5 流程图总结



6.6 注意事项

1. `save_map` 和 `load_map` 的 `path` 参数为**主机本地路径**，请确保该路径可写 / 可读。
2. 空路径返回 `error_code=2`（`kInvalidArgument`）；设备未连接或算法会话未就绪返回 `error_code=1`（`kNotInitialized`）。
3. `save_map` 应在 SLAM（`mode: 3`）或 ReMap（`mode: 5`）模式下调用；`load_map` 应在 Relocation（`mode: 4`）或 ReMap（`mode: 5`）模式下调用。
4. `mode` 超出 `0~5` 范围时返回 `kInvalidArgument`。
5. 典型流程：- 建图：`device_mode mode: 3` → `save_map` → `device_mode mode: 1` - 重定位：`device_mode mode: 4` → `load_map` → `device_mode mode: 1` - ReMap：`device_mode mode: 5` → `load_map` → `save_map` → `device_mode mode: 1`
6. 服务调用成功后会更新驱动的期望模式，用于设备重连后的状态恢复。

← 上一章：数据输出 | 下一章：快速启动（Ubuntu）→

7. Odin1 Lite 快速启动（Ubuntu）

> 本章用于指导用户在 Ubuntu 系统中成功运行 Odin1 Lite ROS 驱动并取得点云、图像、IMU、里程计等数据。

📄 下载中文 PDF：O1Lite 快速启动手册 v1.2

7.1 固件版本 / 驱动版本对应关系

序号	固件版本	驱动版本
1	v1.3.7	v0.7.0
2	v2.0.x	v0.12.0
3	v3.x	v3.x（当前最新：v3.7.0）

> 请确保固件与驱动版本匹配，版本不匹配可能导致功能异常。 `v3.x` 固件必须搭配 `v3.x` 驱动，不可使用 `v0.12.0` 驱动。本文后续的 Service、参数和话题说明以 `v3.7.0` 为准。

7.2 硬件组成与安装

#	部件	说明
1	Odin1 Lite 主机	一体化 LiDAR + 单目 RGB + IMU 模组
2	电源数据一分二航插线	一端航插接主机，另一端分出 DC 头与 RJ45 网口
3	12V/2A DC 电源适配器	用户自备（建议 ≥ 24W 电源裕量）
4	网线（直连或交换机）	连接 Odin1 Lite 与 PC

安装步骤：

1. 将航插头按卡槽方向对准 Odin1 Lite 接口插入并锁紧螺纹环。
2. 航插另一端：DC 头接 12V/2A 适配器，RJ45 水晶头接 PC 网口（或交换机）。
3. 上电后设备指示灯应常亮，进入正常工作状态。

> ⚠️ 锁紧航插前请确认卡槽方向，避免插针弯折。

7.3 PC 端准备

7.3.1 系统要求

Ubuntu 版本	支持的 ROS 版本
20.04	ROS1 Noetic
22.04	ROS2 Humble
24.04	ROS2 Jazzy

> 请在运行驱动前确认电脑已正确安装并 `source` 对应版本的 ROS 环境。

7.3.2 网络配置

Odin1 Lite 通过 **以太网** 与 PC 通信，默认 IP 为 `192.168.1.251`（模组 IP 可根据客户需求更改）。请将 PC 网卡配置到同一网段（例如 `192.168.1.100/24`）。

```
# 1. 查看网卡 IP 是否生效
ifconfig

# 2. 测试设备连通性
ping 192.168.1.251
```

> 如果 ping 不通，请重新启用网卡或检查 IP / 子网掩码 / 防火墙是否阻断 UDP 广播（端口 60000-60010）。

7.3.3 网卡缓冲区与 ROS_DOMAIN_ID

在运行驱动前，建议放大内核网络接收缓冲区，避免高码率传感器数据导致 UDP 丢包；同一局域网内有多台 PC 运行 ROS2 时，还应设置独立的 `ROS_DOMAIN_ID`。

```
# 放大接收缓冲区（建图/重定位稳定运行建议）
sudo sysctl -w net.core.rmem_max=16777216
sudo sysctl -w net.core.rmem_default=8388608
sudo sysctl -w net.core.netdev_max_backlog=50000

# 设置独立的 ROS2 域（1–99；0 表示共享）
export ROS_DOMAIN_ID=1
```

> `export ROS_DOMAIN_ID=...` 仅对当前终端生效，写入 `~/.bashrc` 可永久生效。ROS1 的多机隔离参见 7.14 多机隔离。

7.4 获取驱动

获取步骤：

1. 联系留形售后技术支持人员
2. 提供公司信息、用户信息、邮箱
3. 技术支持将提供 GitLab 账号
4. 使用提供的账号登录 GitLab 获取驱动（登录后请尽快修改密码）

驱动仓库地址： https://git.manifoldtech.cn/manifoldtech-sdk/odin_ros_driver2

> 当前最新驱动为 `v3.7.0`，对应 `v3.x` 固件。若设备使用 `v1.3.7` 或 `v2.0.x` 固件，请按上表获取匹配的历史驱动版本。

驱动压缩包命名形如：

```
ros_driver_v<版本号>_<发布类型>_<日期>.tar.xz
```

例：`ros_driver_v3.7.0_release_20261024.tar.xz`

7.5 部署驱动

以下示例假定驱动包已下载到 `~/Downloads`，工作空间名为 `catkin_ws`。

```
# 1. 创建 ROS 工作空间
mkdir -p ~/catkin_ws/src

# 2. 解压驱动并搬运到 src 下
cp ~/Downloads/ros_driver_v3.7.0_release_20261024.tar.xz ~/catkin_ws/src
cd ~/catkin_ws/src
tar -xvf ros_driver_v3.7.0_release_20261024.tar.xz
mv ./ros_driver_v3.7.0_release_20261024/ros_driver ./
rm -rf ros_driver_v3.7.0_release_20261024 ros_driver_v3.7.0_release_20261024.tar.xz
```

完成后目录结构应为：

```
catkin_ws/
└── src/
    └── ros_driver/
        ├── script/build_ros1.sh
        ├── script/build_ros2.sh
        ├── launch/
        ├── config/
        └── ...
```

7.6 编译驱动

> SDK（`module/sdk_api`）会被编译脚本自动构建，无需单独安装；该说明仅适用于获取传感器数据。使用重定位功能时，还需按 7.10 建图与重定位操作 安装算法 deb 包。

ROS1（Noetic）

```
source /opt/ros/noetic/setup.bash
cd ~/catkin_ws
./src/ros_driver/script/build_ros1.sh
```

ROS2（Humble / Jazzy）

```
source /opt/ros/humble/setup.bash       # 或 jazzy
cd ~/catkin_ws
./src/ros_driver/script/build_ros2.sh
```

7.7 运行驱动

ROS1

```
cd ~/catkin_ws
source devel/setup.bash

# 默认启动（自动发现设备 + RViz）
roslaunch odin_ros_driver_rev1 odin1Lite.launch

# 指定图像分辨率与帧率
roslaunch odin_ros_driver_rev1 odin1Lite.launch \
    image_width:=640 image_height:=544 image_fps:=30

# 不启动 RViz
roslaunch odin_ros_driver_rev1 odin1Lite.launch start_rviz:=false
```

ROS2

```
cd ~/catkin_ws
source install/setup.bash

# 默认启动
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py

# 指定图像分辨率与帧率
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py \
    image_width:=640 image_height:=544 image_fps:=30

# 不启动 RViz
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py start_rviz:=false
```

成功运行后，RViz 中将自动显示点云与图像。

常用启动参数

以下参数在节点启动时读取；修改后需重新启动驱动。完整参数列表可通过 `ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py --show-args` 查看。

参数	默认值	说明
`start_rviz`	`true`	是否启动 RViz
`topic_prefix`	`manifold`	话题前缀根；最终前缀为 `/{topic_prefix}/device{N}/`
`device_mode`	`normal`	启动时设备模式：`normal` 或 `standby`
`discovery_timeout_ms`	`2000`	设备发现超时，单位 ms
`image_width` / `image_height` / `image_fps`	`0`	图像宽、高、帧率；`0` 表示使用设备默认能力
`image_format`	`mjpeg`	图像格式：`mjpeg` 或 `h264`
`filter_confidence_threshold`	`-1`	设备端置信度滤波阈值；`-1` 表示保持设备默认值
`filter_adaptive_cdf_percentile`	`-1`	自适应滤波 CDF 百分位（`1..10000`）；`-1` 表示设备默认值
`filter_adaptive_outdoor_thr`	`-1`	自适应滤波室外距离阈值（mm）；`-1` 表示设备默认值
`filter_range_min_depth`	`-1`	设备端距离滤波最小有效深度（mm）；`-1` 表示设备默认值
`raw_frame_id` / `gray_image_frame_id`	`lidar`	原始点云、灰度图像的 `frame_id`
`slam_frame_id` / `odom_frame_id`	`odom`	SLAM 点云、里程计父坐标系的 `frame_id`
`image_frame_id` / `image2_frame_id`	`camera0` / `camera1`	相机图像的 `frame_id`
`imu_frame_id`	`imu`	IMU 的 `frame_id`
`tf_publish_rate`	`0`	TF 定时重发频率（Hz）；`0` 禁用定时重发
`enable_row_filter`	`false`	是否启用行滤波
`min_row`	`0`	行滤波最小行号，范围 `0..191`；低于该行的点置零
`confidence_threshold`	`7`	初始点云置信度阈值（`0..255`）
`sensor_mode`	`-1`	初始 dToF 传感器模式；`-1` 保持设备当前模式
`log_level`	`1`	SDK / ROS 日志级别：`0`=ERROR、`1`=INFO、`2`=WARN、`3`=DEBUG

7.8 参数设置说明

驱动参数分为两类：**启动期参数**只在节点启动时读取；**运行期参数**支持在节点运行中修改并立即生效。除下表及 7.11 数据流开关 中标注为可动态修改的参数外，其他参数即使通过 `ros2 param set` 写入参数服务器，也不会应用到节点逻辑，必须通过 launch 参数设置后重启驱动。

运行期参数

参数	范围 / 默认值	说明
`confidence_threshold`	`0..255` / `7`	点云置信度阈值，低于该值的点会被置零；在保证无噪点的前提下尽可能小，室内场景建议从 `4` 开始调整
`sensor_mode`	`-1..3` / `-1`	dToF 传感器模式；`-1` 为启动时不下发、保持设备当前模式，`0..3` 的模式含义见 7.9 长短帧切换
`log_level`	`0..3` / `1`	SDK / ROS 日志级别：`0`=ERROR、`1`=INFO、`2`=WARN、`3`=DEBUG

设置方式

```
# ROS2：查看、读取和修改运行期参数
ros2 param list /odin_ros_driver_node
ros2 param get /odin_ros_driver_node sensor_mode
ros2 param set /odin_ros_driver_node sensor_mode 3
ros2 param set /odin_ros_driver_node confidence_threshold 15
ros2 param set /odin_ros_driver_node log_level 1

# ROS2：通过 launch 配置启动期参数，然后重启驱动
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py \
    image_width:=640 image_height:=544 image_fps:=30 \
    raw_frame_id:=lidar imu_frame_id:=imu_link

# ROS1：使用 rqt_reconfigure，或通过命令行修改运行期参数
rosrun rqt_reconfigure rqt_reconfigure
rosrun dynamic_reconfigure dynparam set /odin_ros_driver_node sensor_mode 3
```

7.9 长短帧（Sensor Mode）切换

Odin1 Lite 支持四种工作模式：

枚举	模式	说明
0	HDR	高动态范围，自动平衡远近
1	High Peak（长帧）	优化强光下接收质量，**适合户外**，近距离精度略低
2	Low Peak（短帧）	优化近距离与小空间噪点，**适合室内**，受强光影响大
3	Adaptive	自适应模式

方式一：rqt 图形界面

```
rqt
# 在 Dynamic Reconfigure 面板中找到 odin_ros_driver_node 节点，
# 调整 sensor_mode 参数即可。
```

方式二：命令行参数设置（ROS2）

```
source ~/catkin_ws/install/setup.bash

# 切换为 High Peak
ros2 param set /odin_ros_driver_node sensor_mode 1

# 切换为 Low Peak
ros2 param set /odin_ros_driver_node sensor_mode 2
```

7.10 建图与重定位操作

> 详细 API 说明请参见 6. 建图与重定位 API。
>
> 重定位由 **ROS Driver** 和 **算法 deb 包（`manifold-lc-sdk`）** 共同提供：仅部署 ROS Driver 时，设备可正常输出点云、图像、IMU、里程计等数据，但无法运行建图和重定位算法。
>
> 请确保固件已升级到最新版本。

7.10.1 安装重定位算法 deb 包

1. 下载 download_lc_sdk.sh，在运行 ROS Driver 的主机上执行：

`bash chmod +x download_lc_sdk.sh ./download_lc_sdk.sh`

2. 脚本会检测主机的 CPU 架构及 GPU / NPU、CUDA、TensorRT、OpenVINO、RKNN 等运行环境，并在 `[Download links]` 中输出适用的 `manifold-lc-sdk` 下载链接；优先下载标记为 `RECOMMENDED` 的 deb 包。
3. 下载完成后，在 deb 文件所在目录安装。将以下文件名替换为实际下载的文件名：

`bash sudo dpkg -i manifold-lc-sdk_<version>_<architecture>.deb`

> ⚠️ 仅下载或编译 ROS Driver 不会安装重定位算法。完成 deb 安装后，再启动 ROS Driver 并执行以下建图或重定位流程。

7.10.2 建图（SLAM）

```
# 1. 确保驱动已启动且设备已连接

# 2. 切换到 SLAM 模式（开始建图）
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 3}"

# 3. 移动设备进行建图...

# 4. 保存地图
ros2 service call /manifold/device0/save_map \
    odin_ros_driver_rev1/srv/SaveMap "{path: '/tmp/odin_map.bin'}"

# 5. 建图完成，切回 Normal
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"
```

7.10.3 重定位（Relocalization）

```
# 1. 切换到 Relocation 模式
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 4}"

# 2. 加载之前保存的地图
ros2 service call /manifold/device0/load_map \
    odin_ros_driver_rev1/srv/LoadMap "{path: '/tmp/odin_map.bin'}"

# 3. 移动设备，等待重定位算法完成匹配

# 4. 停止重定位，切回 Normal
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"
```

7.10.4 ReMap（增量建图）

ReMap（`mode: 5`）在已保存地图的基础上进行重定位，同时扩展地图。适用于需要在已有地图中追加新区域的场景。

```
# 1. 切换到 ReMap 模式并加载已有地图
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 5}"

ros2 service call /manifold/device0/load_map \
    odin_ros_driver_rev1/srv/LoadMap "{path: '/tmp/odin_map.bin'}"

# 2. 在 RViz 中将 Fixed Frame 设为 device0/map，移动设备采集新区域

# 3. 保存扩展后的地图
ros2 service call /manifold/device0/save_map \
    odin_ros_driver_rev1/srv/SaveMap "{path: '/tmp/odin_map_extended.bin'}"

# 4. 停止 ReMap，切回 Normal
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"
```

> 增量保存后的地图通常比原始地图更大。

7.10.5 loop-closure 可视化话题

在 SLAM / Relocation / ReMap 模式下，当 `manifold-lc-sdk` 算法 deb 包装好时，会发布以下 loop-closure 可视化话题：

话题后缀	类型	说明
`loop_closure/optimization_busy`	`std_msgs/Bool`	优化器是否正在运行
`loop_closure/optimization_status_marker`	`visualization_msgs/Marker`	文本标记显示 `IDLE` / `OPTIMIZATION`
`loop_closure/optimized_path`	`nav_msgs/Path`	`local_map` 坐标系下优化后的轨迹
`loop_closure/optimized_cloud`	`sensor_msgs/PointCloud2`	`local_map` 坐标系下优化后的全局点云
`loop_closure/map_cloud`	`sensor_msgs/PointCloud2`	`map` 坐标系下加载的地图点云

7.10.6 确认重定位结果

`load_map` 服务成功返回仅表示地图已被算法会话加载。加载地图后移动设备一段距离，并根据应用侧的定位状态与位姿输出确认重定位是否收敛。

7.11 数据流开关

所有数据流开关已从 `config/control_command.yaml` 迁移为 launch 参数。A/B 层参数支持运行期热切换（`ros2 param set` / `rqt_reconfigure` / Service）；C/D/E 层参数仅在启动期生效，修改后需重启驱动。

A 层：主驱动数据通道（`odin_ros_driver_node`）

参数	默认值	说明
`enable_raw_point`	`true`	原始点云通道
`enable_slam_point`	`true`	SLAM 点云通道
`enable_image0`	`true`	左 RGB 相机图像通道
`enable_image1`	`false`	右 RGB 相机图像通道（Odin1 Lite 上无效）
`enable_imu`	`true`	IMU 数据通道
`enable_odom_lf`	`true`	低频里程计通道
`enable_odom_hf`	`true`	高频里程计通道
`enable_tf`	`true`	TF 变换通道

B 层：后处理输出（`post_process_node`）

参数	默认值	说明
`post_process`	`true`	后处理总开关，仅启动期生效
`send_image_raw`	`true`	发布解码后的左 RGB 图像
`send_image2_raw`	`false`	发布解码后的右 RGB 图像（Odin1 Lite 上无效）
`send_cloud_render`	`false`	发布着色点云
`send_undistort`	`false`	发布去畸变图像
`send_rectify`	`false`	发布校正图像及 `camera_info`
`send_opencv_undistort`	`false`	启用 OpenCV 去畸变节点

C/D/E 层：启动期功能开关

参数	节点	默认值	说明
`enable_reprojection`	`pointcloud_reprojector_node`	`false`	启用点云重投影叠加
`enable_pointcloud_to_depth`	`pointcloud_to_depth_node`	`false`	启用点云 → 稠密深度图
`enable_depth_to_pointcloud`	`depth_to_pointcloud_node`	`false`	启用深度图 → 稠密彩色点云

```
# 单通道热切换
ros2 param set /odin_ros_driver_node enable_raw_point false
ros2 param set /postprocess send_undistort true

# 批量切换主驱动通道：0=关，1=开，-1=保持当前；未列出字段默认 0
ros2 service call /manifold/device0/set_channel_config \
    odin_ros_driver_rev1/srv/SetChannelConfig \
    "{enable_raw_point: -1, enable_slam_point: 0, enable_image0: -1, enable_image1: -1, enable_imu: -1,
enable_odom_lf: -1, enable_odom_hf: -1, enable_tf: -1}"

# 批量切换后处理输出
ros2 service call /manifold/postprocess/set_post_process_config \
    odin_ros_driver_rev1/srv/SetPostProcessConfig \
    "{send_image_raw: -1, send_image2_raw: -1, send_cloud_render: -1, send_undistort: 1, send_rectify: -1,
send_opencv_undistort: -1}"
```

> 调用 `set_channel_config` / `set_post_process_config` 时，所有不想修改的字段必须显式设为 `-1`，否则未列出的字段会被默认置 `0` 而意外关闭对应数据流。
>
> ℹ️ Odin1 Lite 为单目 RGB。`enable_image1`、`send_image2_raw` 及双目 / 视差点云相关参数在 Odin1 Lite 上不会发布数据。

录制 ROS2 rosbag

录包前请确认已配置 `ROS_DOMAIN_ID` 并放大网卡缓冲区（参见 7.3.3）。ROS2 建议使用驱动提供的 QoS 覆盖文件，避免高频话题丢包：

```
ros2 bag record -a \
  --qos-profile-overrides-path ~/catkin_ws/src/ros_driver/config/rosbag2_qos.yaml
```

ROS1 使用 TCPROS，无需 QoS 覆盖文件；`rosbag record -a` 即可。

7.12 Odin HTTP Dashboard

设备内置 Web 控制台，无需额外软件即可访问。

- 访问地址：`http://192.168.1.251:8080`（保持 PC 与设备同网段）
- 推荐浏览器：Chrome / Edge

功能模块	页面	说明
设备信息	Dashboard	产品名称、固件版本、序列号
系统资源监控	Dashboard	CPU、内存、系统负载（1/5/15min）
温度实时图表	Dashboard	SoC 温区与传感器温度（1Hz 折线图）
帧率实时图表	Dashboard	各传感器实时 FPS
标定文件管理	Dashboard	标定文件上传 / 下载
地图文件下载	Dashboard	重定位地图文件下载
系统日志下载	Dashboard	设备日志打包下载（.tar.gz）
遥测数据录制	Dashboard	实时监控数据录制为 CSV
OTA 固件升级	Update	拖放固件包 → 自动安装 → 重启验证
网络配置	Setting	IP / 网关 / 子网掩码 / DHCP
设备重启	Setting	远程重启设备
主题切换	全局	亮色 / 暗色模式
管理员登录	全局	Admin 权限解锁高级操作

7.13 升级固件

1. 联系留形技术支持获取固件包
2. 固件获取后解压，确保可以看到 `.bin` 文件
3. 升级步骤： - 确保 PC 和 Odin1 Lite 处于同一网段，且可以 `ping` 通 - 浏览器打开 `http://192.168.1.251:8080` - 左侧点击 **Update** - 点击 **Browse Files** 选择解压后的 `.bin` 文件 - 等待升级结束 - 点击左侧 **Dashboard**，查看固件版本确认升级成功

7.14 多机隔离（避免话题串扰）

多台 PC 在同一局域网启动驱动时，相同话题会互相干扰。

ROS2： 设置独立 `ROS_DOMAIN_ID`

```
echo $ROS_DOMAIN_ID            # 查看，空白或 0 表示共享
export ROS_DOMAIN_ID=1         # 1–99；0 为共享
```

ROS1： 限制仅本机或更换端口

```
# 推荐
export ROS_LOCALHOST_ONLY=1

# 或：使用其他端口
roscore -p 11312
export ROS_MASTER_URI=http://localhost:11312
```

> 上述命令只对当前终端生效，永久生效请追加到 `~/.bashrc` 。

← 上一章：建图与重定位建议调用流程 | 下一章：固件升级 →

8. 固件升级

Odin1 Lite 提供两种升级途径：**HTTP Dashboard（推荐）** 与 **SDK 命令行工具**。

8.1 固件获取

Odin1 Lite 固件请通过下方下载页面获取最新固件包及对应的版本说明。

项	信息
固件下载页	https://version.manifoldtech.cn/download/o1l
文件格式	`.bin`（HTTPS OTA / SDK CLI 通用）

8.2 固件更新内容

固件版本更新说明请咨询留形技术支持获取。

8.3 配套 ROS Driver 更新内容（v3.7.0 / 20261016）

1. 新增 loop-closure 可视化话题，发布载入地图、关键帧矫正结果、优化结果及里程计轨迹。
2. 新增 ReMap 模式（`device_mode=5`，`OdinDeviceMode` 新增 `kReMap`）：支持基于已载入地图扩展建图；主机端 loop_closure 切换到 ReMap，固件数据源处理与 SLAM 一致；`SaveMap` / `LoadMap` 说明更新为支持 ReMap；新增 corrected loop-closure 点云发布。
3. 恢复 `map_cloud` 回调注册；RViz 以三层色阶区分载入、矫正和优化点云，并显示连续里程计轨迹。
4. 修复 `map_cloud` 坐标系和 RViz 轨迹残留时间，恢复 SDK 算法控制与回调互斥锁分离。
5. ROS service 返回消息和日志改用 `OdinResultStr` 枚举名，替代裸错误码。
6. 新增 `ExployMap` 闭环地图导出接口：loop-closure 插件 ABI、SDK `ExployMap` API 及 ROS1/ROS2 `exploy_map` service 完成透传；具体导出格式由插件实现。
7. 增加 openssl 和 crypto 的静态库。

v3.2.0 以来的重要变更

- **v3.6.0 / 2026-09-30**：
- 增加 FFmpeg H.264 解码器运行时硬件加速自动检测与动态解码器发现（NVDEC/CUVID/VAAPI/VDPAU），硬件解码失败时自动回退到软件解码。
- 增加 VAAPI/VDPAU 硬件解码支持（Intel/AMD GPU），并支持通过环境变量 `ODIN_CUDA_DEVICE` / `ODIN_VA_DEVICE` / `ODIN_VDPAU_DEVICE` 指定设备节点。
- 增加 libyuv 可选集成，加速 YUV→BGR 转换（H.264/JPEG 解码及 NV12 合成路径，较 libswscale/OpenCV 约快 7 倍），运行时可用 `ODIN_DISABLE_LIBYUV=1` 禁用。
- 增加设备专属启动文件 `odin1Lite.launch(.py)`（单目 RGB）和 `odin2Special.launch(.py)`（双目 RGB）。
- **不兼容变更**：话题/服务命名移除设备型号段，逐设备前缀由 `/{topic_prefix}/{model}/device{N}/` 改为 `/{topic_prefix}/device{N}/`（例如 `/manifold/ODIN2/device0/` → `/manifold/device0/`）。
- `device_online` 消息 JSON 增加设备型号字段；`device_offline` 由纯 SN 字符串改为 JSON。
- **v3.2.0 / 2026-08-22**：
- 移除 `control_command.yaml`；数据流开关迁移为 launch 参数。

- 增加 `set_channel_config` / `get_channel_config` 和 `set_post_process_config` / `get_post_process_config` 服务，支持批量热切换。
- 增加 `opencv_undistort_node`。
- 增加 `cloud/raw` 行滤波参数 `enable_row_filter` / `min_row`。
- 修复点云发布插件 `point_cloud_transport`。
- **v3.1.0 / 2026-08-13**：
- **不兼容变更**：launch 参数 `operating_mode` 重命名为 `device_mode`。
- **不兼容变更**：`image_format` 可选值调整，移除 `yuyv` / `nv21` / `rgb24`，新增 `h264`；现支持 `mjpeg|h264`。
- **不兼容变更**：`sensor_mode` 默认值从 `1`（High Peak）改为 `-1`（不下发，保持设备当前模式）。
- **不兼容变更**：各传感器 `frame_id` 默认值改为语义化命名（`lidar`、`camera0`、`camera1`、`imu` 等）。
- 新增 ROS 服务：`set_sync_mode` / `get_sync_mode`、`set_adaptive_param` / `get_adaptive_param`、`reset_initial_pose`、`restore_default_config`、`get_ptp_sync_status`、`set_diag_monitor`。
- 新增设备端滤波器 launch 参数和 `tf_publish_rate`。
- **v3.0.x 及更早**：初始 v3.x 固件支持、稠密深度/彩色点云节点、点云重投影、Jetson/Rockchip MJPEG 硬件解码、ROS2 Jazzy 支持。

8.4 升级方式一：HTTP Dashboard（推荐）

> ⚠️ 升级期间请勿运行 ROS 驱动，请勿断电。建议使用稳定的有线网络。

1. 浏览器访问 `http://192.168.1.251:8080`（保持 PC 与设备同网段）。
2. 打开 **Update** 页面，将固件包 `.bin` **拖入上传区域**（或点击选择）。
3. 系统将自动校验签名并安装，过程中页面会展示进度条。
4. 安装完成后设备 **自动重启**；待 Dashboard 重新可访问后，进入 Dashboard 页面校对： - **固件版本号** 是否更新到目标版本 - **设备序列号** 是否一致 - **温度 / 帧率** 是否回到正常区间

8.5 升级方式二：SDK 命令行工具

适用于服务器 / 自动化部署场景。前置条件：已编译 SDK（`module/sdk_api/build/odin_upgrade`）。

```
# 自动发现设备并升级
./build/odin_upgrade /path/to/firmware_v<version>.bin

# 指定设备 IP
./build/odin_upgrade /path/to/firmware.bin --device 192.168.1.251

# 指定发现超时
./build/odin_upgrade /path/to/firmware.bin --timeout 5000
```

升级链路基于 **mTLS 双向认证 + Ed25519 签名**，全程加密；签名校验失败会立即中止安装，不会损坏设备。

8.6 标定文件管理（同样基于 HTTPS）

- HTTP Dashboard → **Dashboard → 标定文件管理**：上传 / 下载 `camera_calib_<SN>.yaml`
- 命令行工具：

`bash # 获取标定 ./build/odin_get_calibration --device 192.168.1.251 --output ./camera_calib.yaml # 烧写标定 ./build/odin_send_calibration ./camera_calib.yaml --device 192.168.1.251`

← 上一章：快速启动（Ubuntu） | 下一章：重定位功能使用说明 →

9. 重定位功能使用说明

9.1 说明

Odin1 Lite 支持以下工作模式，通过 ROS 服务 `device_mode` 进行切换：

mode	模式	说明
0	Standby	待机
1	Normal	正常运行（默认）
2	Upgrade	升级模式
3	SLAM	建图模式，驱动同时切换算法到 Build 模式
4	Relocation	重定位模式，驱动同时切换算法到 Reloc 模式
5	ReMap	重定位 + 增量建图模式，在已载入地图基础上扩展地图

> 重定位 / ReMap 由 **ROS Driver** 和 **算法 deb 包（`manifold-lc-sdk`）** 共同提供。仅部署 ROS Driver 时，设备可正常输出点云、图像、IMU、里程计等数据，但无法运行建图、重定位和增量建图算法。
> 请确保固件已升级到最新版本。

9.2 使用教程

9.2.1 前置条件

1. 已获取、编译并启动与设备固件版本匹配的 ROS Driver。
2. 已安装与当前主机平台匹配的 `manifold-lc-sdk` 算法 deb 包：

```bash
# 在运行 ROS Driver 的主机上执行检测脚本
chmod +x download_lc_sdk.sh
./download_lc_sdk.sh

# 从脚本输出的 [Download links] 下载 RECOMMENDED 的 deb 包后安装
sudo dpkg -i manifold-lc-sdk__.deb
```

检测脚本位于 download_lc_sdk.sh。该脚本仅输出适用的下载链接，不会自动下载或安装 deb 包。

3. 已通过 SLAM 模式保存可用地图文件，并确认其主机本地路径。
4. 在启动 ROS Driver 的终端或新终端中加载 ROS 环境：

```bash
cd ~/catkin_ws
source install/setup.bash
```

9.2.2 建图（SLAM）

```
# Step 1: 切换到 SLAM 建图模式
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 3}"

# Step 2: 移动设备采集地图...

# Step 3: 采集结束后保存地图
ros2 service call /manifold/device0/save_map \
    odin_ros_driver_rev1/srv/SaveMap "{path: '/tmp/odin_map.bin'}"

# Step 4: 切回 Normal 模式
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"
```

> ❗ 建图过程中请勿直接 `Ctrl+C` 退出驱动，必须先调用 `save_map` 保存地图。

9.2.3 重定位（Relocalization）

```
# Step 1: 切换到重定位模式
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 4}"

# Step 2: 加载之前保存的地图
ros2 service call /manifold/device0/load_map \
    odin_ros_driver_rev1/srv/LoadMap "{path: '/tmp/odin_map.bin'}"

# Step 3: 移动设备，等待重定位成功...

# Step 4: 重定位完成后切回 Normal
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"
```

9.2.4 确认重定位结果

`load_map` 服务成功返回仅表示地图已被算法会话加载。加载后请移动设备一段距离，并根据应用侧的定位状态与位姿输出确认重定位是否收敛。

> ⚠️ Odin1 Lite 无法在静止状态下完成重定位；请勿在设备静止时判断重定位失败。

9.3 注意事项

- 地图保存依赖算法回环会话已就绪（设备已连接且标定可用），会话未就绪时服务返回 `error_code` 非 0
- `mode` 超出 0–5 范围时服务返回 `kInvalidArgument`
- 设备未连接时服务返回 `kNotInitialized`
- 请用 `ros2 service list` 确认实际服务名；v3.x 设备型号不再出现在话题/服务路径中

← 上一章：固件升级 | 下一章：重定位地图获取手册 →

10. 重定位地图获取手册

10.1 说明

Odin1 Lite 提供重定位功能，支持以下方案获取重定位地图（`.bin` 文件）：

方案	说明
1	使用 Odin1 Lite SLAM 模式直接生成 bin 文件（推荐）
2	使用留形手持扫描仪采集 lx 数据，通过 Mindcloud 导出 bin 文件

10.2 方案 1：SLAM 模式直接建图（推荐）

这是最简单的方式，通过 ROS 服务即可完成：

```
cd ~/catkin_ws
source install/setup.bash

# 1. 切换到 SLAM 建图模式
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 3}"

# 2. 移动 Odin1 Lite 采集环境数据...

# 3. 采集结束后保存地图
ros2 service call /manifold/device0/save_map \
    odin_ros_driver_rev1/srv/SaveMap "{path: '/tmp/odin_map.bin'}"

# 4. 切回 Normal
ros2 service call /manifold/device0/device_mode \
    odin_ros_driver_rev1/srv/DeviceMode "{mode: 1}"
```

地图文件将保存到指定路径（如 `/tmp/odin_map.bin`）。

> 也可在 SLAM 建图完成后，通过 ReMap（`device_mode=5`）在已有地图基础上继续增量建图并保存扩展后的地图。增量保存后的地图通常比原始地图更大。
>
> ❗ 建图期间请勿 `Ctrl+C` 退出驱动，必须先调用 `save_map` 保存地图。

10.3 方案 2：留形手持扫描仪采集 lx + Mindcloud 导出

如无留形手持设备，请选择其他方案。

1. Q9000 连接手机，使用 Mindcloud Go 进行数据扫描
2. 扫描结束后将工程文件导出到本地
3. 使用 Mindcloud 处理 lx 文件，在后处理中点击“导出地图”按钮，获取重定位 bin 文件

10.4 地图下载（HTTP Dashboard）

也可以通过设备内置 HTTP Dashboard 下载重定位地图：

1. 浏览器访问 `http://192.168.1.251:8080`
2. 在 Dashboard 页面找到“地图文件下载”功能
3. 下载 bin 文件到本地

← 上一章：重定位功能使用说明 | 下一章：维护与故障排除 →

11. 维护与故障排除

11.1 日常维护

1. **清洁规范：** - 使用气吹清除镜头灰尘 - 使用干净的软布擦拭设备外壳 - ⚠️ 禁止使用腐蚀性溶剂擦拭窗口玻璃，腐蚀性溶剂会破坏玻璃表面镀膜；当腐蚀性溶剂从窗口玻璃边缘缝隙进入，会腐蚀玻璃的防水背胶，导致防水失效。禁用的腐蚀性溶剂包含但不限于：酒精、异丙醇、白电油、松节油等。
2. **存储条件：** - 温度：-20°C ~ 60°C - 湿度：< 85% RH

11.2 常见故障排除

> 更多问题参见 13. FAQ。

故障现象	可能原因	解决方法
上电后无指示灯	供电异常	检查 12V 适配器输出、DC 头是否插紧、航插是否锁紧
`ping 192.168.1.251` 不通	PC 与设备不同网段 / 网线 / 网口问题	设置 PC 网口为 `192.168.1.x/24`；重启网口；更换网线
HTTP Dashboard 无法访问	端口 8080 被防火墙拦截 / 设备未完全启动	关闭防火墙或放行 8080；等待 ~30s 重试
驱动运行后无数据输出	UDP 广播被防火墙拦截 / 网络未同网段	放行 UDP 60000–60010；确认 PC 与设备在同一网段。设备发现、网卡选择和 UDP 端口由 SDK 自动处理，无需配置 `device_ip`
RViz 提示 TF 变换失败	设备原始 TF 频率低，时间戳无法插值	启动时设置 `tf_publish_rate:=50`；推荐范围 50–200 Hz，`0` 为禁用
`post_process_node` 发布首帧图像时崩溃	系统 OpenCV 与 ROS 的 `cv_bridge` 链接版本不一致	在当前工作空间重新编译与系统 OpenCV 匹配的 `vision_opencv` / `cv_bridge`，详见 13. FAQ
重定位一直失败	设备静止 / 地图文件路径错误 / 算法会话未就绪	移动设备；检查 bin 文件路径；确认已安装 `manifold-lc-sdk` 且在对应模式下调用地图服务
建图/重定位漂移或丢包严重	网卡接收缓冲区不足 / UDP 丢包	启动驱动前放大缓冲区：`sudo sysctl -w net.core.rmem_max=16777216 net.core.rmem_default=8388608 net.core.netdev_max_backlog=50000`
`ros2 service call` 无响应	服务名称错误（大小写不匹配）	用 `ros2 service list` 确认实际服务名；v3.x 设备型号不再出现在话题/服务路径中

← 上一章：重定位地图获取手册 | 下一章：技术参数 →

12. 技术参数

分类	参数名称	规格/数值
dToF 深度模组	深度分辨率	240 × 180
	测距范围（Max）	25m @ 10%反射率（<50Klux）／50m @ 90%反射率（<500lux）
	测距范围（Min）	0.15m
	点频	最高 430,000 pts/s
	测距精度	±3 cm @ 1σ [1]
	视场角（FOV，水平×垂直）	120° × 90°
	角分辨率	0.5° × 0.5°
	激光波长	940 nm
	激光安全等级	Class 1 人眼安全
	帧率	最高 10 FPS
RGB 相机模组	分辨率	1280 × 1088
	帧率	30 FPS
	曝光方式	全局曝光（Global Shutter）
	FOV（H × V × D）	121° H × 106° V × 151° D
整机参数	内置 SLAM 定位精度	±5 cm + 1% [2]
	重量	274 g
	体积（L×W×D）	约 69 mm × 69 mm × 58 mm（主体）/ 66 mm（含航插）
	功耗	额定 11W，峰值 24W
	防护等级	IP66
	抗冲击	50G × 5ms
	工作电压	9–24 V
	工作温度	-20℃ ~ 45℃
	存储温度	-20°C ~ 60°C
数据输出	点云	真彩 SLAM / 原始点云
	照片	原始 RGB 图像
	IMU	高频原始数据
	位姿	高频旋转、位置数据

[1] 室内测试环境，在 80% Lambertian 反射板上连续 20 帧的数据计算 Std 精度值。5m 条件下 ±3cm @ 1σ，5–30m 1σ 精度值 < 1%。

[2] 实验室条件下。

← 上一章：维护与故障排除 | 下一章：FAQ →

13. 常见问题 FAQ

❓ Odin1 Lite 支持哪些系统?

回答:

Ubuntu 版本	支持的 ROS 版本
20.04	ROS1 Noetic
22.04	ROS2 Humble
24.04	ROS2 Jazzy

支持 x86 与 ARM（Jetson）架构。

❓ 设备上电后无法发现或无法 ping 通?

回答:

1. 确认设备指示灯是否常亮（供电正常）。
2. 确认 PC 网卡 IP 与设备处于 **同一网段**： - 设备默认 IP：`192.168.1.251` - PC 建议设置：`192.168.1.x/24`（避免与设备 `.251` 冲突）
3. 通过 `ifconfig` 检查 IP 是否真正生效，必要时**禁用再启用网卡**。
4. 检查防火墙是否阻断 UDP 广播（端口 60000–60010）。
5. 设备发现、网卡选择和 UDP 端口均由 SDK 自动处理，无需也无法通过 launch 配置 `device_ip`。

❓ 相机没有固定 IP，如何通过 DHCP 自动分配?

回答： 使用仓库提供的 `script/start_dhcp.sh` 脚本启动 udhcpd 服务器即可：

1. 安装 udhcpd：`sudo apt install udhcpd`
2. 按 `script/my_dhcp.conf` 修改监听网卡、IP 池和网关。
3. 启动 DHCP：`sudo ./script/start_dhcp.sh /path/to/my_dhcp.conf`
4. 查看租约文件 `/var/lib/misc/udhcpd.leases` 确认相机已拿到 IP。

❓ 多台 PC 同时连接同一局域网，话题数据相互干扰?

回答:

- **ROS2**：为每台 PC 设置不同 `ROS_DOMAIN_ID`

`bash echo $ROS_DOMAIN_ID # 查看当前值（空白或 0 表示共享） export ROS_DOMAIN_ID=1 # 1–99; 0 为共享`

- **ROS1**：限制 master 仅本机可见，或更换端口

```bash # 推荐 export ROS_LOCALHOST_ONLY=1

# 或：使用其他端口 roscore -p 11312 export ROS_MASTER_URI=http://localhost:11312 ```
```

> 上述命令仅对当前终端有效，如需永久生效请加入 `~/.bashrc` 。

❓ 建图 / 重定位漂移或丢包严重?

回答:

高码率传感器数据可能撑爆内核网络接收缓冲区，导致 UDP 丢包，进而影响 SLAM 和重定位效果。运行驱动前请放大缓冲区并设置独立的 `ROS_DOMAIN_ID`：

```
# 放大内核接收缓冲区
sudo sysctl -w net.core.rmem_max=16777216
sudo sysctl -w net.core.rmem_default=8388608
sudo sysctl -w net.core.netdev_max_backlog=50000

# 将当前 PC 与同一局域网内的其他 ROS2 主机隔离
export ROS_DOMAIN_ID=1   # 1–99；不同 PC 使用不同 ID
```

如需永久生效，将 `sysctl` 设置写入 `/etc/sysctl.conf` （或 `/etc/sysctl.d/` 下的文件），并将 `export ROS_DOMAIN_ID=1` 加入 `~/.bashrc` 。

❓ ROS1 下设置 `image_width` / `image_height` 不生效?

回答： 该问题已在 v0.9.0 及之后版本修复，所有 v3.x 驱动均包含此修复。

❓ Sensor Mode 应该选择 High Peak 还是 Low Peak?

回答:

模式	推荐场景	说明
0 / HDR	通用	自动平衡远近
1 / High Peak（长帧）	**户外**	抗强光好，近距精度略低
2 / Low Peak（短帧）	**室内**	优化近距与小空间噪点，受强光影响大
3 / Adaptive	通用	自适应切换

切换方式参见 7. 快速启动 § 7.9。

❓ 编译失败提示 `ODIN SDK not found`?

回答： 通常是首次编译没有进入 `module/sdk_api` 子模块。请按下列任一方式处理：

1. 使用驱动包内提供的 `script/build_ros1.sh` / `script/build_ros2.sh` （脚本会自动编译并安装 SDK）。
2. 手动初始化 SDK 子模块： `git submodule update --init module/sdk_api` 。
3. 如果 SDK 已经预装到自定义路径，设置环境变量： `export ODIN_SDK_PREFIX=/path/to/install` 。

❓ 编译时整机卡顿、内存占满甚至 GUI 无响应?

回答： 驱动中部分节点（如 `post_process_node`）单编译单元需要 1.0–1.6 GB 内存，默认并发可能撑爆小内存平台。`build_ros1.sh` / `build_ros2.sh` 已按内存自动限制并发（约每任务 2 GB），也可手动指定：

```
# 限制并发数
ODIN_BUILD_JOBS=2 ./src/ros_driver/script/build_ros2.sh

# 或用 MAKEFLAGS 完全控制
MAKEFLAGS="-j2 -l3" ./src/ros_driver/script/build_ros2.sh
```

8 GB 及以下设备建议 `-j1` 或 `-j2`，必要时用 cgroup 保护 GUI：

```
systemd-run --user --scope -p MemoryHigh=5G -p CPUWeight=20 \
  ./src/ros_driver/script/build_ros2.sh
```

❓ RViz 提示 TF 变换失败，点云无法显示?

回答： 设备原始 TF 发布频率较低时，点云时间戳可能落在最新 TF 之后，导致 tf2 无法插值。启动驱动时启用 TF 定时重发：

```
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py tf_publish_rate:=50
```

推荐设置为 50–200 Hz；设为 `0` 时禁用定时重发，保持设备原始频率。

❓ ROS2 录制 rosbag 时高频话题丢包?

回答： 使用驱动提供的 QoS 覆盖配置，以可靠模式录制：

```
ros2 bag record -a \
  --qos-profile-overrides-path ~/catkin_ws/src/ros_driver/config/rosbag2_qos.yaml
```

ROS1 使用 TCPROS，直接使用 `rosbag record -a` 即可；磁盘 I/O 较慢时可增加 `--buffsize`。

❓ 用 `ros2 topic hz` 看 `cloud_raw` 帧率很低，是驱动发布频率不够吗?

回答： 不一定是驱动发布频率低。有客户反馈用 `ros2 topic hz` 查看 `cloud_raw` 时，打印出来的频率明显偏低；这通常是 `ros2 topic hz` 的测量方式导致，尤其是在重定位（Relocation）开始后，算法会向上发送较大的点云数据，`ros2 topic hz` 基于 Python 订阅端计数，对 `PointCloud2` 这类大消息容易受反序列化与传输开销影响，导致显示的帧率低于实际发布帧率。

要验证真实帧率，建议改用**录包后统计**：

```
ros2 bag record /manifold/device0/cloud/raw \
  --qos-profile-overrides-path ~/catkin_ws/src/odin_ros_driver/config/rosbag2_qos.yaml
```

录制结束后用 `ros2 bag info <bag>` 查看 `Message count` 和 `Duration`，按 `message_count / duration` 计算出的帧率更准确。

> 如需实时观测，可启动驱动自带的 `diag_subscriber_node`（C++ 订阅端），它对 `PointCloud2` 等大消息的帧率统计比 `ros2 topic hz` 更准确。

❓ `post_process_node` 发布第一帧图像时崩溃?

回答： 这通常是系统安装了多个 OpenCV，导致驱动链接的 OpenCV 与 ROS apt 版 `cv_bridge` 不一致。请在当前工作空间重新编译 `vision_opencv` 中的 `cv_bridge`，使其与系统 OpenCV 使用相同版本，然后再编译驱动：

```
cd ~/catkin_ws/src
git clone -b $ROS_DISTRO https://github.com/ros-perception/vision_opencv.git
cd ~/catkin_ws
colcon build --packages-select cv_bridge
source install/setup.bash
./src/ros_driver/script/build_ros2.sh
```

ROS1 请使用 `catkin_make -DCATKIN_WHITELIST_PACKAGES="cv_bridge"` 编译 `cv_bridge`。可通过 `ldd <post_process_node路径> | grep -iE 'cv_bridge|opencv_core'` 确认只加载同一版本的 OpenCV。

❓ 节点崩溃后如何用 GDB 调试?

回答： 先用 Debug 模式编译，再用 `launch-prefix` 启动 GDB：

```
# Debug 编译
./src/ros_driver/script/build_ros2.sh --cmake-args -DCMAKE_BUILD_TYPE=Debug

# 交互式 GDB
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py --launch-prefix "gdb -ex run --args"

# 非交互式（适合 SSH / 嵌入式）
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py \
  --launch-prefix "gdb -batch -ex run -ex 'bt full' --args"

# 仅调试单个节点
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py \
  --launch-prefix "gdb -ex run --args" --launch-prefix-filter post_process_node
```

崩溃后在 GDB 中执行 `bt full`、`thread apply all bt` 查看调用栈。若调用栈全是 `??`，说明未带调试符号，需重新 Debug 编译。

❓ 如何验证硬件 JPEG 解码已实际生效?

回答： 驱动启动日志仅说明编译期解码路径，不能作为硬解已生效的依据。建议同时验证硬件中断持续增长、`post_process_node` 持有硬解设备节点、且 CPU 占用较低：

```
# RK3588
watch -n 1 'grep -E "jpegd|vdpu|rkvdec" /proc/interrupts'
sudo lsof /dev/mpp_service | grep post_process

# Jetson
sudo tegrastats --interval 500 | grep -E "NVJPG"
sudo lsof /dev/nvhost-nvjpg | grep post_process
```

❓ Jetson 上硬件 JPEG 解码 fallback 到软解时崩溃?

回答： Jetson R36/JetPack 6 上，当硬件解码失败并 fallback 到软件解码时，可能因 `libnvjpeg.so` 与 OpenCV 使用的 `libjpeg-turbo` 符号冲突导致 `free()` 野指针崩溃。若需稳定使用软件解码，请重新编译：

```
# 仅 ROS2 示例；ROS1 用 build_ros1.sh 并加 -DJPEG_PREFER_SOFTWARE_DECODER=ON
colcon build --cmake-args -DJPEG_PREFER_SOFTWARE_DECODER=ON
```

默认 `JPEG_PREFER_SOFTWARE_DECODER=OFF` 使用硬件解码；仅在需要纯软解或遇到 fallback 崩溃时开启。

❓ 如何远程升级固件?

回答： 浏览器访问 `http://192.168.1.251:8080` → **Update** 页面拖入固件包，系统将自动安装并重启。详见 8. 固件升级。

❓ 标定文件 / 重定位地图 / 设备日志怎么导出?

回答： 全部可在 HTTP Dashboard 完成：

- 标定文件管理：上传 / 下载 `camera_calib_<SN>.yaml`
- 地图文件下载：导出重定位地图
- 系统日志下载：打包成 `.tar.gz`
- 遥测数据录制：导出 CSV

← 上一章：技术参数 | 返回首页 →

14. PTP 时间同步

Odin1 Lite 支持基于 **PTP（Precision Time Protocol，IEEE 1588）** 的高精度时间同步，可将设备系统时钟与上位机（Host）时钟同步到**毫秒级以内**，使 LiDAR、相机、IMU 等传感器数据的硬件时间戳与主机时钟处于同一时间基准，满足多传感器融合、多机协同等对时间一致性要求较高的应用场景。

> Odin1 Lite 默认使用系统时钟（SysTick）作为传感器数据时间戳来源。未部署 PTP 环境时无需任何配置。

14.1 PTP 协议原理

14.1.1 协议概述

PTP（IEEE 1588）是一种面向局域网的高精度时钟同步协议，与 NTP（毫秒~十毫秒级）相比，PTP 通过**硬件时间戳**（Hardware Timestamping）将同步精度提升至**亚微秒级**，广泛应用于工业自动化、电信、车载与机器人领域。

PTP 采用**主从（Master-Slave）架构**：

- **主时钟（Grandmaster / GM）**：网络中的时间源，通常为上位机或专用时钟服务器；
- **从时钟（Slave）**：与主时钟对齐的设备，Odin1 Lite 在同步网络中作为从时钟；
- **边界时钟（Boundary Clock, BC）** / **透明时钟（Transparent Clock, TC）**：中间网络设备（如 PTP 交换机），逐跳修正链路时延，避免多级转发引入的累积误差。

14.1.2 同步机制（Delay Request/Response 机制）

主从时钟之间通过四类报文完成时钟偏移与链路时延的测量：

```
主时钟（Master）                          从时钟（Slave）
      |                                        |
      | —— Sync（t1 硬件打戳） ——————————→ | t2
      | —— Follow_Up（携带 t1） ——————————→ |
      |                                        |
      | ←————————— Delay_Req（t3 打戳）—— | t3
      | ←———————— Delay_Resp（携带 t4）———— | t4
      |                                        |
  偏移：offset = [(t2-t1) - (t4-t3)] / 2
  链路时延：delay = [(t2-t1) + (t4-t3)] / 2
```

1. 主时钟周期性广播 `Sync` 报文，并在报文经过网卡时由 **PHY/MAC 硬件打上精确时间戳**（t1）；从时钟记录接收时间戳（t2）；
2. `Follow_Up` 报文将 t1 传递给从时钟；
3. 从时钟发送 `Delay_Req`（打戳 t3），主时钟回复 `Delay_Resp`（携带接收时间戳 t4）；
4. 从时钟根据上述公式解算时钟偏移（offset）与链路时延（delay），并逐步**驯服（discipline）**本地时钟。

关键精度保障在于 **t1–t4 均由硬件产生**，消除了协议栈软件处理带来的不确定时延（抖动可达数十微秒），这也是 PTP 相对软件 NTP 方案的核心优势。

14.1.3 传输协议：L2（IEEE 802.3）

PTP 报文可承载于 UDP/IPv4（L3）或直接承载于以太网帧（L2，EtherType `0x88F7`）。Odin1 Lite 采用 **L2（IEEE 802.3）**传输：

- L2 报文不经过 IP 协议栈，时延更小、路径确定性更强；
- 需要网络中所有 PTP 端口配置为同一传输协议（本方案交换机端口协议须设置为 IEEE 802.3）。

14.1.4 主从选举（BMCA）

网络中的时钟通过**最佳主时钟算法（Best Master Clock Algorithm, BMCA）**依据时钟等级（clockClass）、精度（clockAccuracy）、优先级（Priority1/Priority2）等参数自动选举主时钟。在 Odin1 Lite 的典型部署中，上位机被显式配置为 `masterOnly`，交换机作为边界时钟向上游（Host）同步、向下游（设备）提供时间源。

14.2 系统架构

Odin1 Lite 的 PTP 同步体系由三层组成：



组件	角色	说明
Host `ptp4l`	主时钟端口	`-H`（硬件时间戳）`-2`（L2 传输）`--masterOnly=1`
Host `phc2sys`	时钟驯服	将 Host 系统时钟（CLOCK_REALTIME）同步到网卡 PHC
PTP 交换机	边界时钟（BC）	上游跟随 Host，下游为最多三台设备提供时间；SYNC 指示灯同步状态可视化
设备 `ptp4l` / `phc2sys`	从时钟	将网卡 PHC 与主时钟对齐，并把 PHC 同步到设备系统时钟
设备 `odin_ptp_daemon`	常驻守护进程	管理设备端 PTP 会话，配合驱动时间同步模式

14.3 网络与交换机配置

Odin1 Lite 通过 **HiSync PTP 交换机（R01 型，北京飞灵科技）** 组建同步网络：

项	说明
端口定义	`P0`：连接管理电脑（WebUI 配置，**不参与 PTP**）；`PTP1 ~ PTP4`：一主三从 PTP 端口
WebUI 地址	`http://192.168.1.82`（需与 Odin1 Lite / Host 同网段）
登录账户	用户名 `admin` / 密码 `admin`
指示灯	`SYNC`：绿灯慢闪 = PHC 未同步；绿灯常亮 = 已与上游 PHC 同步

配置步骤：

1. 管理电脑接入 `P0`，将其配置为 `192.168.1.x` 网段，浏览器访问 `http://192.168.1.82` 登录 WebUI；
2. 左侧列表进入「时间同步 → PTP」，点击「端口配置」；

3. 若各 PTP 端口 `Protocol` 列显示的不是 **L2**，点击「编辑」，将 Protocol 更改为 **IEEE 802.3**（其余配置保持默认），保存；
4. 将**所有** PTP 端口协议改为 L2 后，点击「保存并应用」，等待交换机重启。

> 主时钟 Host 与从设备均须接入 `PTP1 ~ PTP4`，不要接入 `P0`。

14.4 主机端（Host）配置

前提：确认 Host 网卡支持硬件时间戳（`ethtool -T <网卡>` 输出含 `hardware-raw-clock` / `SOF_TIMESTAMPING_RAW_HARDWARE`），并确认接入交换机的网卡名称（下例为 `enp1s0f0`，请替换为实际网卡）。

```
# 启动 ptp4l：L2 传输 + 硬件时间戳 + 仅主模式
sudo ptp4l -i enp1s0f0 -m -H -2 --masterOnly=1

# 启动 phc2sys：将系统时钟同步到网卡 PHC（另一终端）
sudo phc2sys -s CLOCK_REALTIME -c enp1s0f0 -m -O 0 -S 0.001
```

Host 启动后，无论从设备是否接入，交换机都应先与 Host 完成同步——可通过交换机 `SYNC` 指示灯（绿灯常亮）或 WebUI「同步状态」页面（时间同步状态、PTP 时钟时间、偏移量、祖父时钟信息等）确认。

14.5 硬件支持性检查与软同步方案

14.5.1 如何判断 Host 网卡是否支持 PTP 硬件时间戳

PTP 硬件时间戳依赖网卡的 PHC（PTP Hardware Clock）。在 Host 上依次执行以下检查：

① 查看网卡时间戳能力

```
ethtool --json -T enp1s0f0     # 或旧版：ethtool -T enp1s0f0
```

输出关键字	含义	结论
`hardware-raw-clock: true` / `SOF_TIMESTAMPING_RAW_HARDWARE` 且 `ptp-hardware-clock` 指向 `/dev/ptpN`	网卡具备独立 PHC	✅ 支持硬件同步，按 14.4 正常部署
仅有 `software-raw-clock: true` / `SOF_TIMESTAMPING_SOFTWARE`	网卡无 PHC	❌ 不支持硬件同步，转 14.5.2 软同步

常见支持硬件时间戳的网卡：Intel i210/i211/i225/i226、82574、82599/X520、X550 系列；大多数消费级板载网卡（如 Realtek RTL8111/8125、部分 Intel 1225-V 系列）**不支持**。

② 确认 PHC 设备节点

```
ls /dev/ptp*
# 支持硬件时间戳的网卡会枚举出 /dev/ptp0 等节点；
# 可用 ethtool --json -T enp1s0f0 输出中的 ptp-hardware-clock 字段确认对应关系
```

③ 验证 PHC 可读

```
sudo phc_ctl /dev/ptp0 get
```

若返回当前 PHC 时间则说明硬件时钟工作正常。

> 设备端（Odin1 Lite）已内置 PHC 硬件时钟，上述检查仅针对 Host。若 Host 不支持硬同步，整条链路将退化为软同步精度。

14.5.2 软件时间戳方案（无 PHC 网卡）

若 Host 网卡不支持硬件时间戳，仍可使用 linuxptp 的**软时间戳（software timestamping）**模式完成 PTP 同步：报文打戳点从 PHY/MAC 退化为内核协议栈，消除用户态处理时延，但无法消除内核收发路径抖动，典型同步精度从亚微秒级退化为**百微秒级（约 10 µs ~ 1 ms）**。

Host 侧只需一处变化（对比 14.4）：`ptp4l` 的 `-H`（硬件时间戳）改为 `-S`（软件时间戳）：

```
# Host: 软时间戳主时钟（-S = software timestamping）
sudo ptp4l -i enp1s0f0 -m -S -2 --masterOnly=1

# phc2sys 不再适用（Host 无 PHC）：
# -S 模式下 ptp4l 的时间戳直接来源于系统时钟 CLOCK_REALTIME，
# Host 系统时钟本身就是对外发布的时间基准，无需额外驯服

# 设备端：保持硬件时间戳从模式不变（odin_ptp_daemon / ptp4l -H），无需任何改动
```

说明：设备侧 PHC 始终可用，建议保持硬件时间戳从模式；链路整体精度瓶颈在 Host 软戳一侧。若客户对精度要求较高且 Host 不带 PHC，也可建议客户更换一颗支持硬件时间戳的网卡（如 Intel i226，成本低且主流发行版免驱）。

> **精度预期与选型建议**
>
> | 方案 | Host 硬件要求 | 典型同步精度 | 适用场景 |
> | --- | --- | --- | --- |
> | 硬件 PTP（推荐） | 网卡带 PHC | 亚微秒级 | 多传感器紧耦合融合、运动重建 |
> | 软件 PTP（-S） | 普通网卡 | 10 µs ~ 1 ms | 时间粗对齐、日志对时 |

14.5.3 软同步下的替代工具

软同步对 Host 网卡无硬件要求，实现工具也不限于 linuxptp。除本方案默认的 **linuxptp（ptp4l）** 外，主流可选实现如下：

工具	协议/Profile	平台	特点	适用性
linuxptp（ptp4l + phc2sys）	IEEE 1588-2019 全配置（含 L2/UDP、802.1AS）	Linux	官方内核配套实现，功能最全，软/硬时间戳均支持	✅ 推荐，本文档默认方案
ptpd2（PTPd）	IEEE 1588-2008	Linux/BSD	老牌实现，配置简单，社区活跃度已下降	⚠️ 可用，无新特性，不建议新项目采用
OpenAvnu gPTP	IEEE 802.1AS（gPTP）	Linux/Windows	Intel 主导，面向 AVB/TSN 音视频场景，Windows 侧可用	⚠️ 仅推荐 gPTP/TSN 场景

说明：

1. **802.1AS（gPTP）注意**：若客户环境是车载/AVB/TSN 网络并使用 gPTP profile，需确认 PTP 交换机同样支持 802.1AS，且 Odin1 Lite 侧 daemon 是否兼容对应 profile，请先联系留形技术支持确认；
2. **ptpd2 与 ptp4l 不能同时运行**在同一主机，端口会冲突；
3. 无论选择哪种工具，软同步的精度上限由**打戳点位置**（内核协议栈）决定，而非工具本身，各工具在相同软戳模式下精度基本相当。

14.6 设备端配置

设备端 PTP 会话由固件内置的 `odin_ptp_daemon` 守护进程自动管理（与 `ptp4l` / `phc2sys` 配套），**无需用户登录设备做任何手工操作**。

用户的全部操作都在 Host 侧通过 ROS 服务完成：

驱动提供时间同步模式服务（详见 5. 数据输出）：

```
# 查询当前同步模式（0=SysTick 系统时钟，1=PTP）
ros2 service call /manifold/device0/get_sync_mode \
  odin_ros_driver_rev1/srv/GetSyncMode "{}"

# 切换到 PTP 硬件时间戳模式（需重启设备后生效）
ros2 service call /manifold/device0/set_sync_mode \
  odin_ros_driver_rev1/srv/SetSyncMode "{sync_mode: 1}"
```

PTP 生效后，可实时查询同步质量：

```
ros2 service call /manifold/device0/get_ptp_sync_status \
  odin_ros_driver_rev1/srv/GetPtpSyncStatus "{}"
```

返回的快照包含同步状态、时钟偏移（offset from master）、链路时延（mean path delay）与频率调整量等，可用于量化评估同步精度。

> 若 `get_ptp_sync_status` 长时间显示未同步，请按顺序排查：交换机端口协议是否为 L2（14.3）→ Host 主时钟是否运行（14.4）→ 设备时间同步模式是否为 PTP（14.6）；仍无法解决请联系留形技术支持。

14.7 同步验证

1. **交换机同步状态**：`SYNC` 指示灯绿灯常亮，WebUI「同步状态」显示已同步且偏移量在毫秒级以内；
2. **PTP 同步状态查询**：通过 `get_ptp_sync_status` 确认设备侧同步状态与偏移量（见 14.6）；
3. **数据时间戳验证**：Host 运行驱动后，检查传感器话题时间戳已切换到同步后的时钟：

```
ros2 launch odin_ros_driver_rev1 odin1Lite.launch.py
ros2 topic echo /manifold/device0/imu --field header.stamp
```

14.8 注意事项

1. 全网络 PTP 端口必须统一使用 **L2（IEEE 802.3）** 协议，混用 UDPv4 会导致无法同步；软同步模式下报文同样走 L2，交换机端口配置要求不变；
2. `set_sync_mode` 修改后**必须重启设备**才能生效；
3. 设备端 `odin_ptp_daemon` 与系统默认 `ptp4l` / `phc2sys` 互斥，同时运行会导致同步异常；
4. Host 网卡最好支持硬件时间戳；不支持时可退化为软时间戳（`ptp4l -S`），但同步精度将显著下降（详见 14.5）；
5. PTP 交换机与 Host、设备须处于同一网段，Host 接入交换机 `PTP1 ~ PTP4`，管理电脑接入 `P0`；
6. 建议通过 `get_ptp_sync_status` 定期监控偏移量，偏移持续增大时优先排查链路、交换机端口配置与 Host 主时钟状态。

← 上一章：FAQ | 返回首页 →