依赖声明

本项目基于 **RoboSense LiDAR SDK——rs_driver**（版权所有 © 2020 RoboSense）开发，遵循其3-clause BSD License协议。
使用前请确保已获取RoboSense SDK的合法授权（商业使用需联系RoboSense获取授权）。
rs_driver_update基于rs_driver进行了小部分修改

编译安装

源码文件夹

1. **rs_driver_update**
2. **rs_lidar_pyhton**

先决条件

1. 设置ip

-查看是否可以ping通
bash : ping 192.168.1.102

2. C++编译器支持：g++ >= 7.0(C++17)
3. CMake >= 3.10
4. Python编译器支持：3.6<=Python<=3.14
5. **rs_driver_update**依赖：

```
sudo apt-get install libpcap-dev
libeigen3-dev libboost-dev libpcl-dev
```

6. **rs_lidar_pyhton**依赖：

- 在*RS_LIDAR_Python*目录下执行

```
pip install -r ./rs_lidar_pyhton/requirements.txt
```

7. 确认rs_lidar_python中rs_driver安装路径(**默认已经设置好了,一般不需要修改**)
   setup.py

```
def get_extensions():
    # 头文件路径（适配RoboSense SDK安装路径）
    rs_driver_include = os.getenv("RS_DRIVER_INCLUDE",
    "/usr/local/rs_driver/include/")  # RoboSense rs_driver头文件路径
```

rs_driver_update编译安装

- 在*RS_LIDAR_Python*目录下执行

```
cd rs_driver_update
mkdir build && cd build
cmake .. && make -j4
```

- 若编译失败可以尝试修改CMakeLists.txt中

```
option(COMPILE_TOOL_VIEWER "Build point cloud visualization tool" ON)
option(COMPILE_TOOL_PCDSAVER "Build point cloud pcd saver tool" ON)
option(COMPILE_TESTS "Build rs_driver unit tests" ON)
```

ON->OFF不影响python包的使用
4. sudo make install

rs_lidar_pyhton编译安装

- *RS_LIDAR_Python*目录下执行

```
cd rs_lidar_pyhton/src
python setup.py bdist_wheel
pip install dist/*.whl
```

pip show rs_lidar

- **pip show**

```
pip show rs_lidar
```

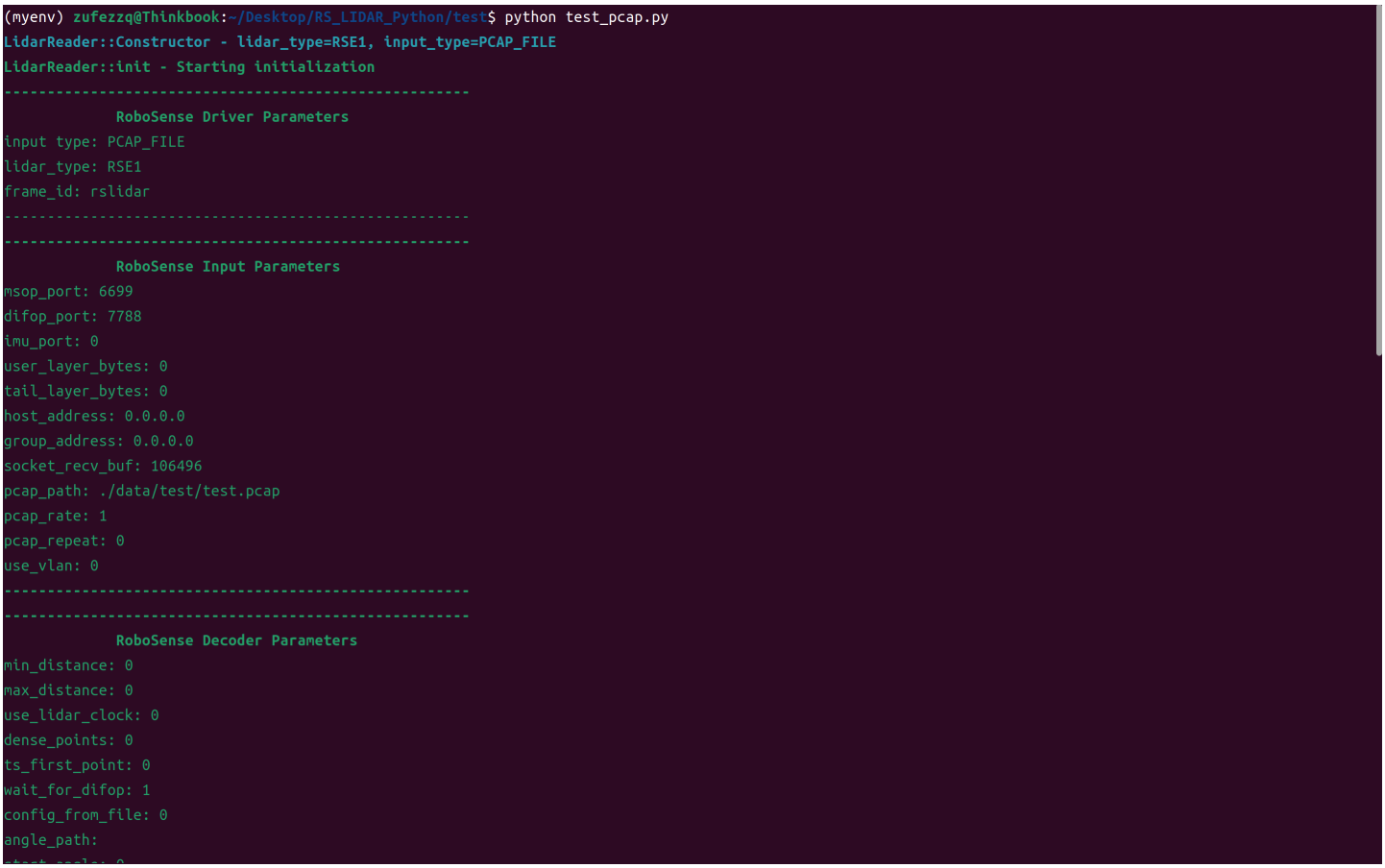
- **输出**

```
Name: rs_lidar
Version: 0.1.0
Summary: 基于RoboSense LiDAR SDK的Python/封装库，支持在线/PCAP点云读取与Numpy转换
Home-page: https://github.com/GoldenPigeon123
Author: Zhejiang University of Finance and Economics - Point Cloud Team:
Zhang Zhongqain , Li Huaiyuan , Cao Yiyun
浙江财经大学信息技术与人工智能学院点云组:张忠谦,李怀苑,曹宜云
Author-email: 2789632062@qq.com
License: BSD-3-Clause
Location: /home/zufezzq/.local/lib/python3.10/site-packages
Requires: numpy, pybind11
Required-by:
```

快速测试

在*RS_LIDAR_Python*目录下执行

```
cd test
python test_pcap.py
```



若无效请尝试

```
# 关闭防火墙
sudo ufw disable || true
```

使用说明

1. 导入rs_lidar 库

```
import rs_lidar
```

2. 创建Lidar对象

请选择好你的Lidar类型，以及工作模式，确认msop_port与difop_port端口，然后初始化并启动Lidar

ONLINE_LIDAR：在线雷达模式

```
reader = rs_lidar.LidarReader("RSE1", "ONLINE_LIDAR", 6699, 7788)
reader.init()
reader.start()
```

PCAP_FILE：PCAP文件模式

```
reader = rs_lidar.LidarReader("RSE1", "PCAP_FILE", 6699, 7788)
reader.set_pcap_path("your.pcap")
```

3.初始化并启动Lidar

```
reader.init()
reader.start()
```

4. 获取点云数据并转换为Numpy数组

```
cloud=reader.getPointCloud()
points_np = cloud.to_numpy()
```

5. 释放资源

```
reader.freePointCloud(cloud)
```

6. 关闭Lidar

```
reader.close()
```

详细API

核心类与结构概览

一、核心类概览

类名	功能描述
`LidarReader`	激光雷达核心读取器，负责驱动初始化、启动 / 停止、点云获取 / 释放
`PointCloudMsg`	点云消息对象，存储点云数据（`PointXYZI` 列表）及元信息（时间戳、帧序号等）
`PointXYZI`	点结构体，描述单个点的三维坐标（X/Y/Z）及激光反射强度（Intensity）

详细API说明

核心类LidarReader

- **构造方法**

```
rs_lidar.LidarReader(lidar_type_str="RSE1", input_type_str="ONLINE_LIDAR",
  msop_port=6699, difop_port=7788)
```

- **参数说明**

参数名	类型	默认值	说明
`lidar_type_str`	str	"RSE1"	激光雷达型号（需与硬件匹配，如 "RSE1"、"RS128" 等）
`input_type_str`	str	"ONLINE_LIDAR"	输入模式： - "ONLINE_LIDAR"：在线模式（实时读取雷达数据） - "PCAP_FILE"：离线模式（解析 PCAP 文件）
`msop_port`	int	6699	MSOP 数据端口（在线模式必填，需与雷达配置一致）
`difop_port`	int	7788	DIFOP 数据端口（在线模式必填，需与雷达配置一致）

- **方法说明**

方法名	功能描述	参数说明	返回值	注意事项
`set_pcap_path`	设置 PCAP 文件路径（仅 `input_type_str="PCAP_FILE"` 时有效）	`file_path` (str)：PCAP 文件的绝对路径或相对路径	无返回值	路径不存在时会打印警告，但不抛出异常；建议先通过 `os.path.exists` 验证路径
`init`	初始化激光雷达驱动（必须在 `start` 前调用）	无参数	`bool`： - `True`：初始化成功 - `False`：初始化失败（如 PCAP 路径无效、端口被占用）	初始化失败需检查参数或硬件连接
`start`	启动数据读取（驱动内部启动线程，开始接收 / 解析数据）	无参数	`bool`： - `True`：启动成功 - `False`：启动失败（如未初始化）	重复调用会打印警告并返回 `True`
`stop`	停止数据读取（释放线程资源）	无参数	无返回值	析构函数会自动调用，建议显式调用避免资源泄漏
`getPointCloud`	获取点云数据（阻塞至超时）	`usec` (int)：超时时间（微秒，默认 1000000 即 1 秒）	`PointCloudMsg` 或 `None`： - 成功：点云消息对象 - 超时：`None`	需在 `start` 后调用；返回的点云需通过 `freePointCloud` 释放
`freePointCloud`	释放点云缓冲区（放回内部空闲队列，复用内存）	`cloud` (PointCloudMsg)：需释放的点云消息对象	无返回值	**必须调用**，否则会导致内存泄漏
`isDriverRunning`	检查驱动是否正在运行	无参数	`bool`： - `True`：运行中 - `False`：已停止	用于判断是否需要继续获取点云
`getTemperature`	获取雷达设备温度	无参数	`float`：温度值（摄氏度）	温度获取失败会抛出 `RuntimeError`
`printDriverParam`	打印驱动配置参数（调试用，如雷达型号、端口、PCAP 路径等）	无参数	无返回值	输出至控制台，包含详细配置信息
`printDeviceInfo`	打印设备信息（调试用，如固件版本、设备 ID 等）	无参数	无返回值	需在 `init` 成功后调用
`printDeviceStatus`	打印设备运行状态（调试用，如是否连接、错误码等）	无参数	无返回值	需在 `start` 成功后调用

点云消息类：PointCloudMsg

- **属性说明**

属性名	类型	说明
`timestamp`	float	点云时间戳（秒，精确到微秒，如 1620000000.123456）
`seq`	int	帧序号（自增，从 0 开始）
`frame_id`	str	坐标系 ID（默认 "rslidar"，可用于 ROS 坐标转换）
`points`	list[PointXYZI]	点数据列表，每个元素为 `PointXYZI` 对象

- **方法说明**

方法名	功能描述	参数	返回值	示例
`to_numpy`	将点云转换为 N×4 的 Numpy 数组（每行：X, Y, Z, Intensity），float32 类型	无	`numpy.ndarray`：形状为 (N, 4)	`points_np = cloud.to_numpy()`
`__repr__`	调试用字符串表示（打印时自动调用，显示关键信息）	无	str：点云简洁描述	`print(cloud)` → 输出 "PointCloudMsg (frame_id='rslidar', seq=5, timestamp=1620000000.123456, points_count=18048)"

点结构体类：PointXYZI

- **属性说明**

属性名	类型	说明
`x`	float	X 轴坐标（单位：米，如 1.234）
`y`	float	Y 轴坐标（单位：米，如 5.678）
`z`	float	Z 轴坐标（单位：米，如 0.912）
`intensity`	int	激光反射强度（范围：0~255，值越大反射越强）

- **方法说明**

方法名	功能描述	示例
`__repr__`	调试用字符串表示（打印时自动调用，显示点坐标和强度）	`print(cloud.points[0])` → 输出 "PointXYZI (x=1.234, y=5.678, z=0.912, intensity=255)"

异常捕获

异常类型	触发场景	处理建议
`RuntimeError`	1. `getTemperature` 获取温度失败 2. 驱动内部错误（如端口占用）	检查硬件连接或端口占用情况，重启驱动
`FileNotFoundError`	`set_pcap_path` 时路径不存在	验证 PCAP 文件路径，使用 `os.path.exists` 提前检查
`ValueError`	1. 构造函数参数无效（如端口为负数） 2. `to_numpy` 转换时点云为空	检查参数合法性，确保点云非空后再转换
`KeyboardInterrupt`	用户按 `Ctrl+C` 中断程序	在 `finally` 块中调用 `reader.stop()`，确保资源释放

联系我们

- 学校：浙江财经大学
- 地址：浙江省杭州市下沙高教园区学源街18号
- 团队：浙江财经大学信息技术与人工智能学院点云组
- 邮箱：2789632062@qq.com
- GitHub: https://github.com/GoldenPigeon123

附录

rs_driver_update基于rs_driver进行了小部分修改，修改点如下：

rs_driver修改点

1. rs_driver/driver/driver_param.hpp

```
inline InputType strToInputType(const std::string& type)
{
  static const std::unordered_map<std::string, InputType> strInputTypeMap = {
      {"ONLINE_LIDAR", InputType::ONLINE_LIDAR},
      {"PCAP_FILE", InputType::PCAP_FILE},
      {"RAW_PACKET", InputType::RAW_PACKET}
  };

  auto it = strInputTypeMap.find(type);
  if (it != strInputTypeMap.end()) {
      return it->second;
  } else {
    RS_ERROR << "Wrong input type: " << type << RS_REND;
    RS_ERROR << "Please give correct type: ONLINE_LIDAR, PCAP_FILE, RAW_PACKET." << RS_
    exit(-1);
  }
}
```

```
struct DeviceInfo{
    void print(){
    RS_INFO << "-------------------------------------------------------" << RS_REND;
    RS_INFO << "             RoboSense Device Info " << RS_REND;
    RS_INFOL << "state: " << state << RS_REND;
    RS_INFOL << "sn: " << sn << RS_REND;
    RS_INFOL << "mac: " << mac << RS_REND;

    RS_INFOL << "top_ver: " << top_ver << RS_REND;
    RS_INFOL << "bottom_ver: " << bottom_ver << RS_REND;
    RS_INFOL << "-------------------------------------------------------" << RS_REND;
  }
}


struct DeviceStatus{
    void print(){
    RS_INFO << "-------------------------------------------------------" << RS_REND;
    RS_INFO << "             RoboSense Device Status " << RS_REND;
    RS_INFOL << "state: " << state << RS_REND;
    RS_INFOL << "voltage: " << voltage << RS_REND;
    RS_INFO << "-------------------------------------------------------" << RS_REND;
  };
}
```