传感器配置


Vlp16 * 3
Camera * 1
Imu * 1
Gnss *1
Radar * 2


Innoviz * 2
Camera * 2
Ultrasonics * 12

Pandar40 * 1
Vlp16 * 1
Camera * 1
Panoramic * 4

Innoviz * 4
Camera * 4

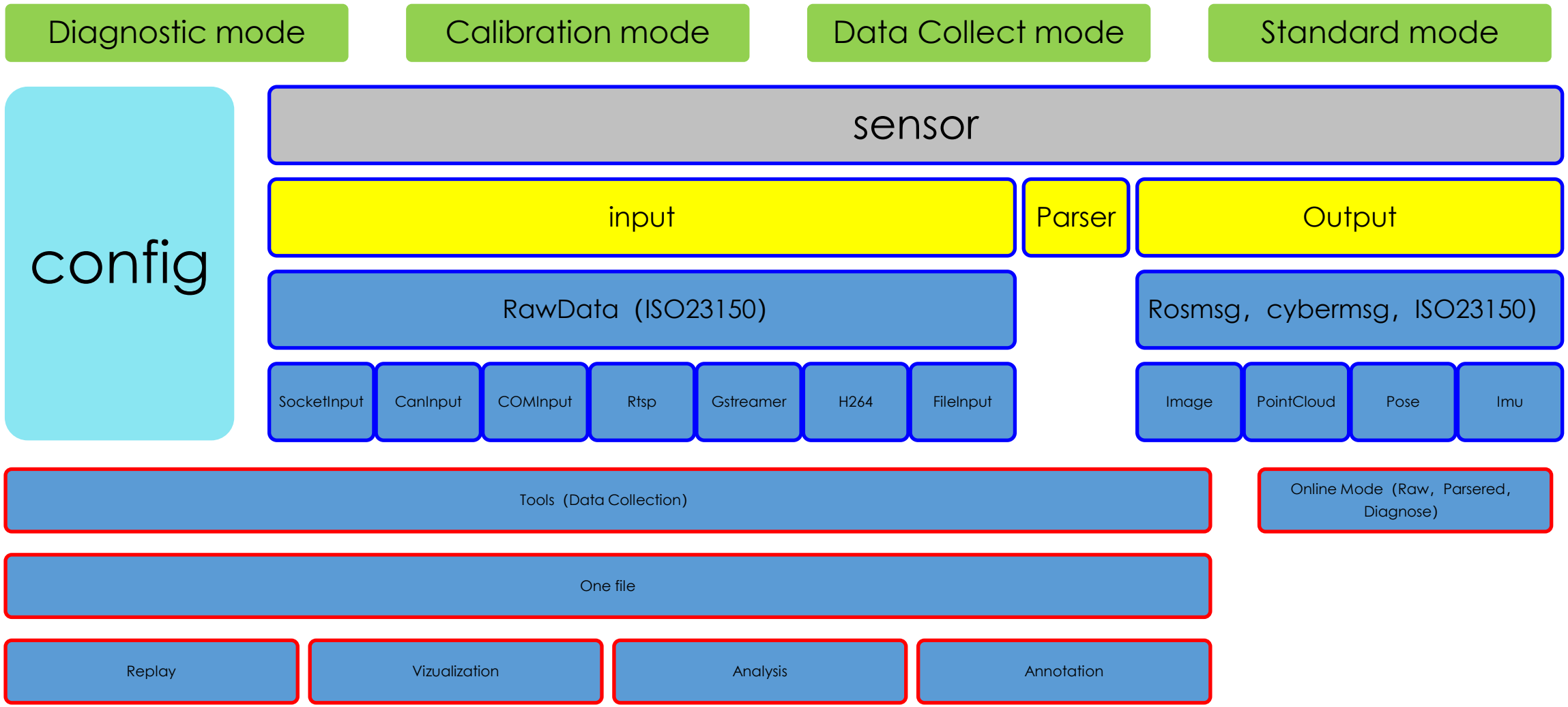


。 。 。 。
。 。

痛点：

1. 不同项目中的传感器配置差异较大，遇到新项目需要从输入输出重新开发驱动，且由于人员更迭很难形成统一接口进行迭代；（**统一框架**）
2. 不同传感器的流数据有差异，以太网，can，mipi等，数据同步采集困难，采集到各种格式的数据都需要重新开发解析脚本，且很难在软件层面做同步或进行数据回灌；（**数据接口统一**）
3. 针对传感器故障无法进行有效诊断，标定，在线或者离线所采用到的实际数据会有一定差异，不同模式不能区分需要数据，从而造成输出数据的冗余，浪费资源；例如数采需要采集原始数据以保证数据量小，而实车感知模块需要驱动解析好的数据；（**故障诊断**）

SensorHub 架构 3



SensorHub 架构-配置

配置

使用.proto文件定义配置结构，使用.prototxt文件输入具体配置，约束整个结构，新增传感器只进行拓展开发，使用工厂进行注册，而不修改该原有程序。

```
frame_id: ""
channel_name: ""
header_frame_id: ""
header_channel_name: ""
stricted_max_time_gap: 50000 # us
latest_max_time_gap: 500000 # us
priotiry: -15

blind_zone {
    delete_point: true
    min_x: 0
    max_x: 0
    min_y: 0
    max_y: 0
    min_z: 0
    max_z: 0
}

compoment_config {
    sensor_position_id: 1
    frame_id: ""
    channel_name: ""
    raw_data_channel_name: ""
    priotiry: -15
    stricted_bundle: true
    config_file: "param/lidar/test/test.prototxt"
}
```

Sensor list

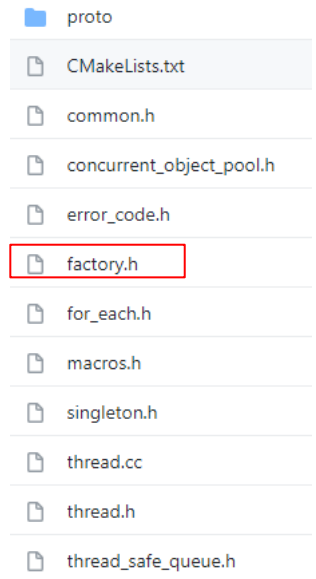
每个项目或者每个车型对应一个sensorlist。包括了盲区大小，同步方式，有什么样得传感器，传感器数据输入输出通道，数据通道的id名称，优先级等

```
input_config {
    name: "SocketInput"
    pool_size: 10
    packet_size: 1500
    lidar_port: 6699
    gps_port: 7788
    socket_config {}
}

parser_config {
    name: "VLP16Parser"
    pool_size: 4
    split_azimuth: 18000
    time_zone: 8
    max_points: 100000
    use_local_time: false
    lidar_packet_config {
        size: 1248
        check_sum: 0x55AA05
        blocks: 12
        block_offset: 42
        block_size: 100
        block_check_sum: 0xFFEE
        firing_offset: 4
        lasers: 32
        point_size: 3
    }

    gps_packet_config {
        size: 1248
        check_sum: 0xA5FF005A
    }
}
```

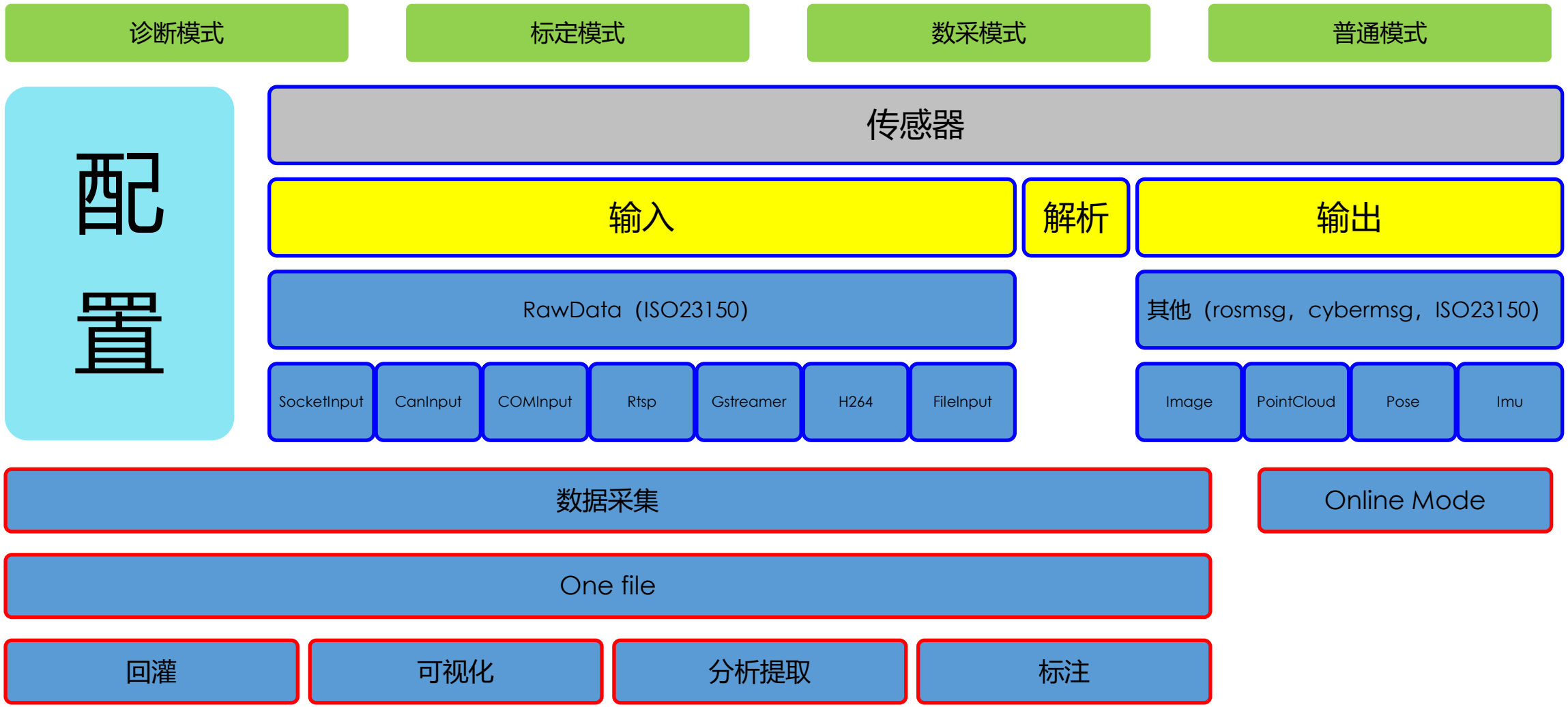
Sensor Config

InputConfig
用来配置每一种传感器的输入类型（Socket，File，SocketCan，fpga，gstreamer等等）

PaserConfig
解析数据的基本配置等



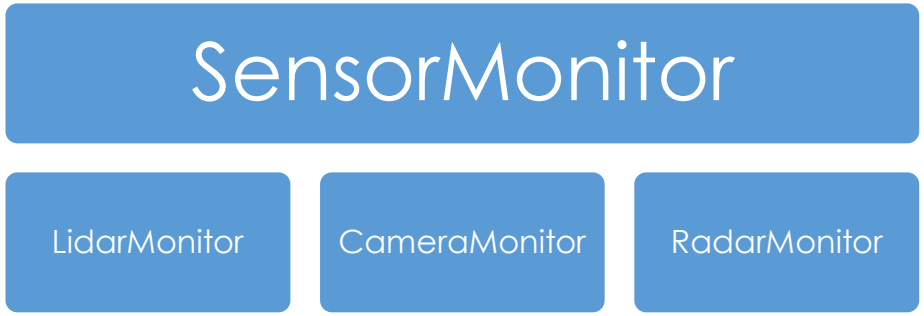
SensorHub 架构

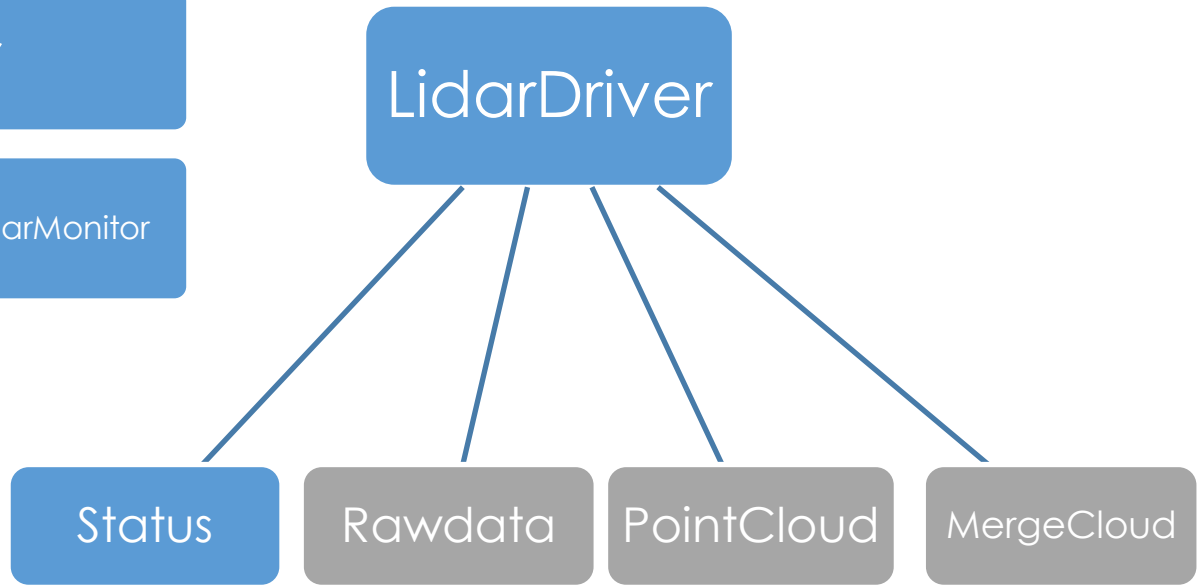


SensorHub 架构-模式

诊断模式 | 标定模式 | 数采模式 | 普通模式

构建单独的**诊断模块**内聚诊断和监控功能，以单例形式管理所有传感器的状态，防止出现竞争，可以按照**激光**，**相机**，**毫米波**等构建实例

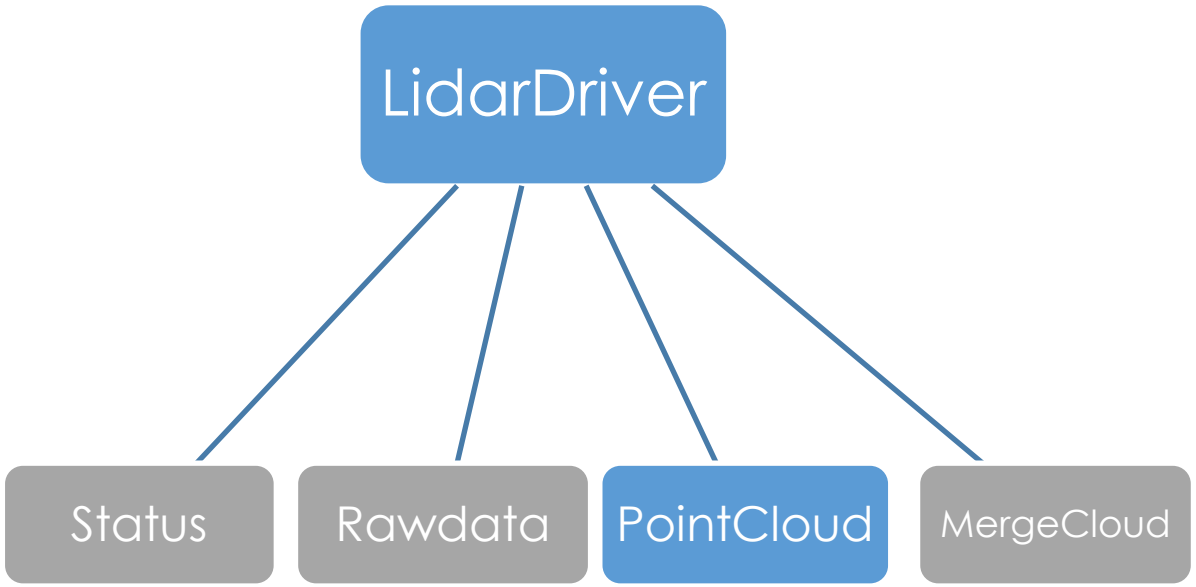




Status 每个lidar的状态通道数据输入诊断模块，其他通道不发送数据

SensorHub 架构-模式

诊断模式 | 标定模式 | 数采模式 | 普通模式



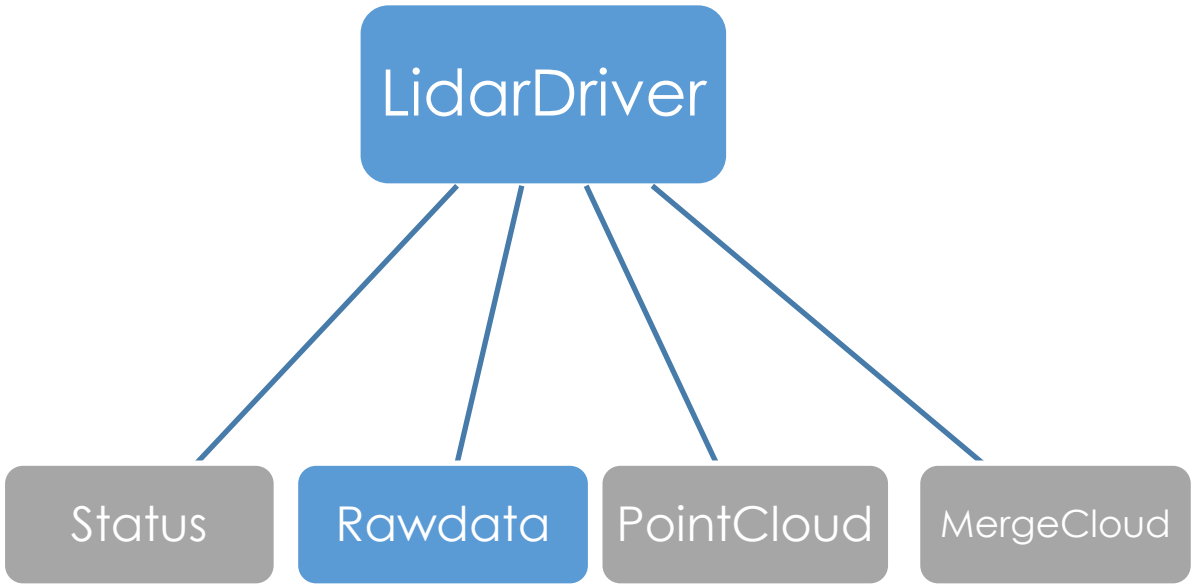

```
message PointXYZIT {
  optional float x = 1 [default = nan];
  optional float y = 2 [default = nan];
  optional float z = 3 [default = nan];
  optional uint32 ring = 4 [default = 0];
  optional float azimuth = 5 [default = nan];
  optional float elevation = 6 [default = nan];
  optional uint32 intensity = 7 [default = 0];
  optional uint64 timestamp = 8 [default = 0];
  optional uint32 semantic_flag = 9 [default = 0];
}

message PointCloud {
  optional crdc.airi.Header header = 1;
  optional string frame_id = 2;
  optional bool is_dense = 3;
  repeated PointXYZIT point = 4;
  optional double measurement_time = 5;
  optional uint32 width = 6;
  optional uint32 height = 7;
}
```

Calibration

每个lidar解析好的点云通道数据输入标定模块，其他通道不发送数据

SensorHub 架构-模式

诊断模式 | 标定模式 | **数采模式** | 普通模式



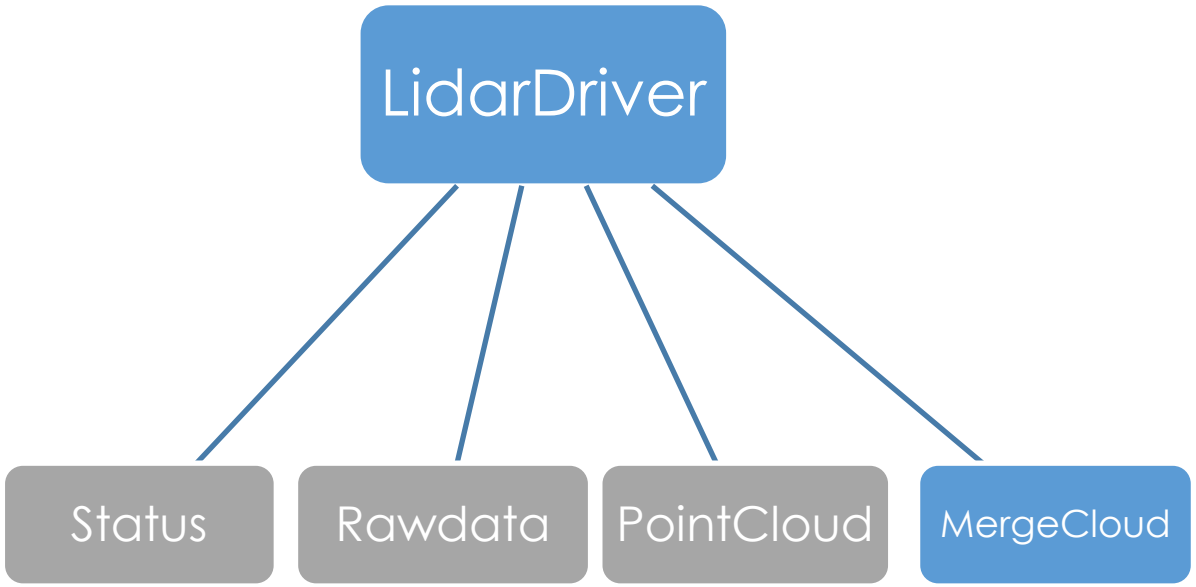
DataRecorder

```
message Packet {
    optional uint32 version = 1 [default = 1];
    optional bytes data = 2;
    optional uint32 size = 3;
    optional uint32 port = 4;
    optional uint64 time_system = 5;
}

message Packets {
    repeated Packet packet = 1;
}
```

每个lidar的原始数据输入数据采集模块，其他通道不发送数据

SensorHub 架构-模式

诊断模式 | 标定模式 | 数采模式 | 普通模式

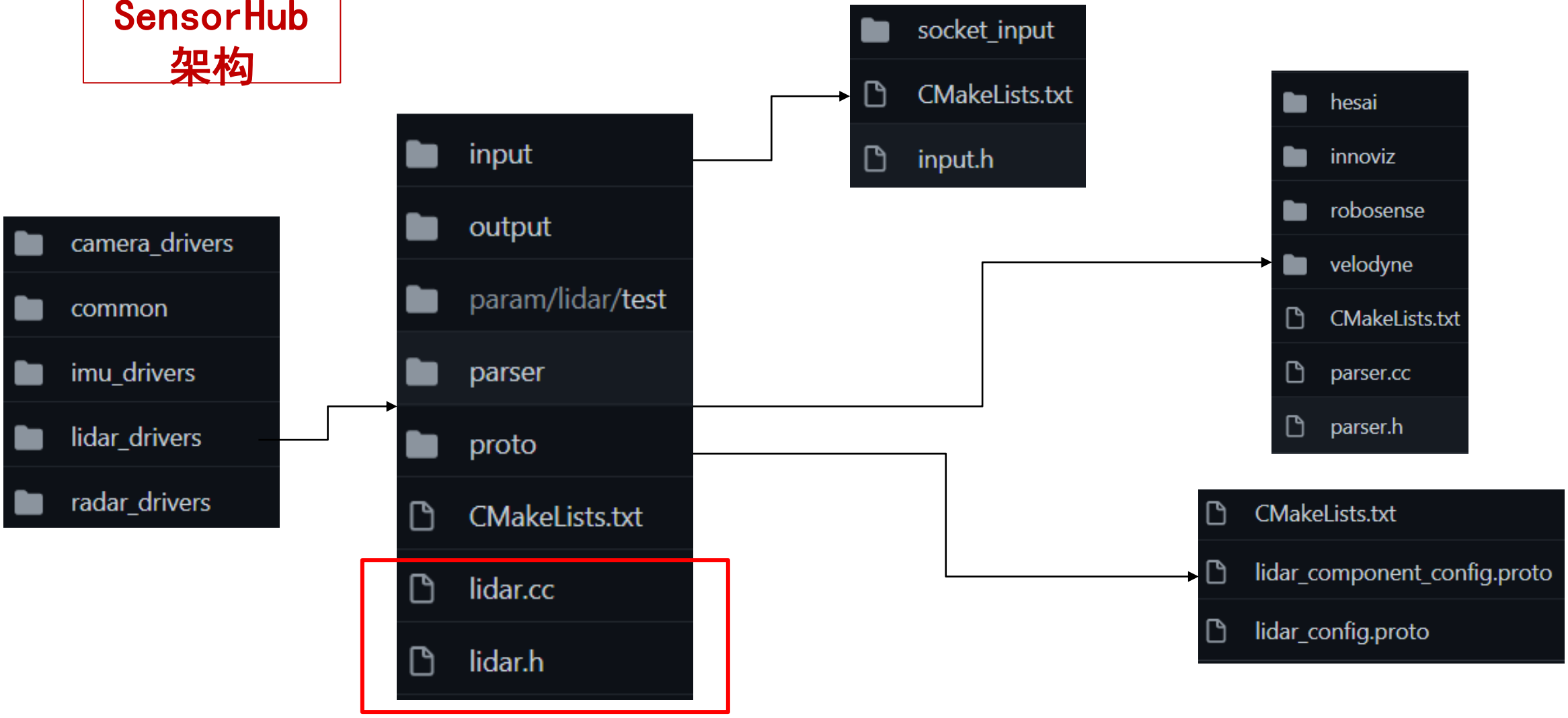


Perception

```
message PointClouds {
  optional crdc.airi.Header header = 1;
  repeated PointCloud cloud = 2;
}
```

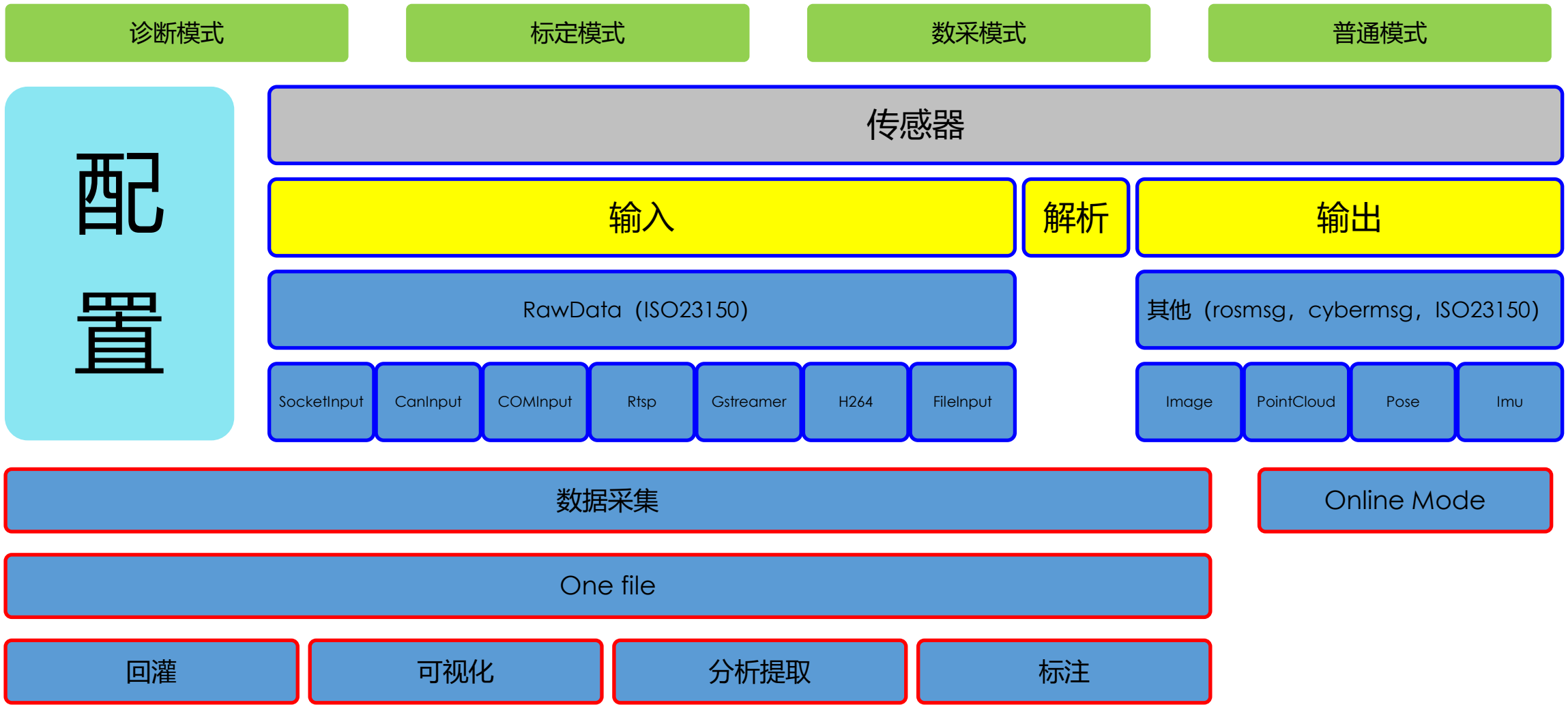
拼接+运动补偿的数据输入感知模块，其他通道不发送数据

SensorHub 架构

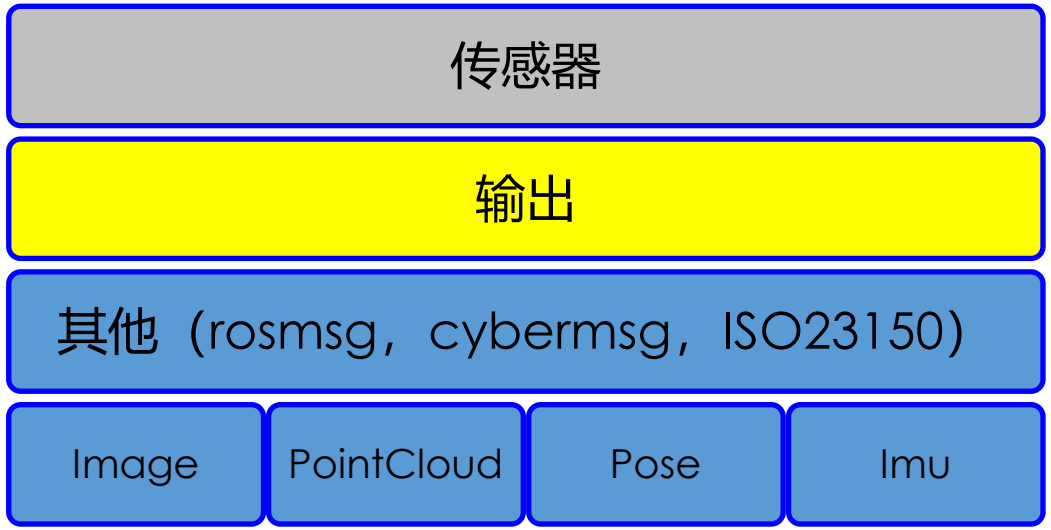


继承thread 每个sensor 驱动是一个单独的thread

SensorHub 架构



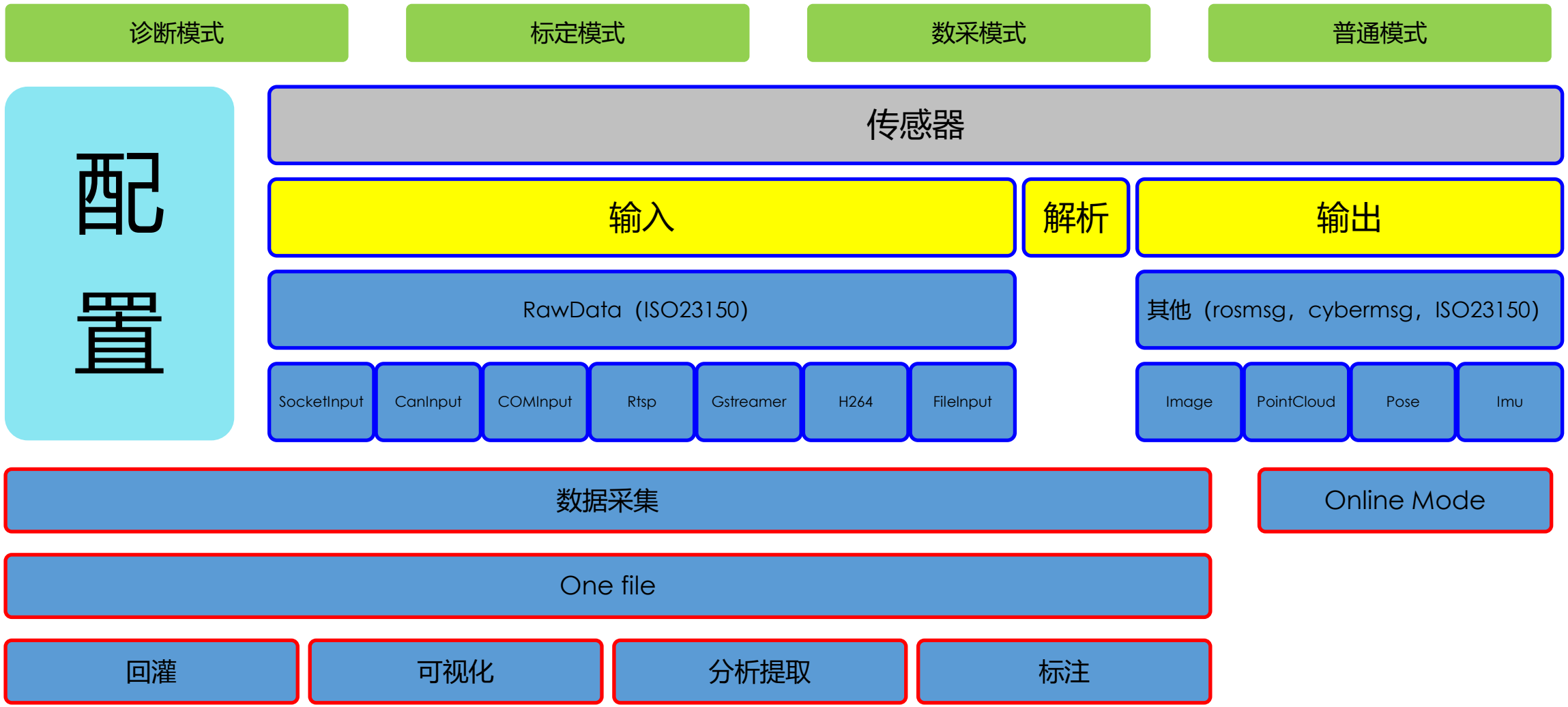
SensorHub 架构-输出



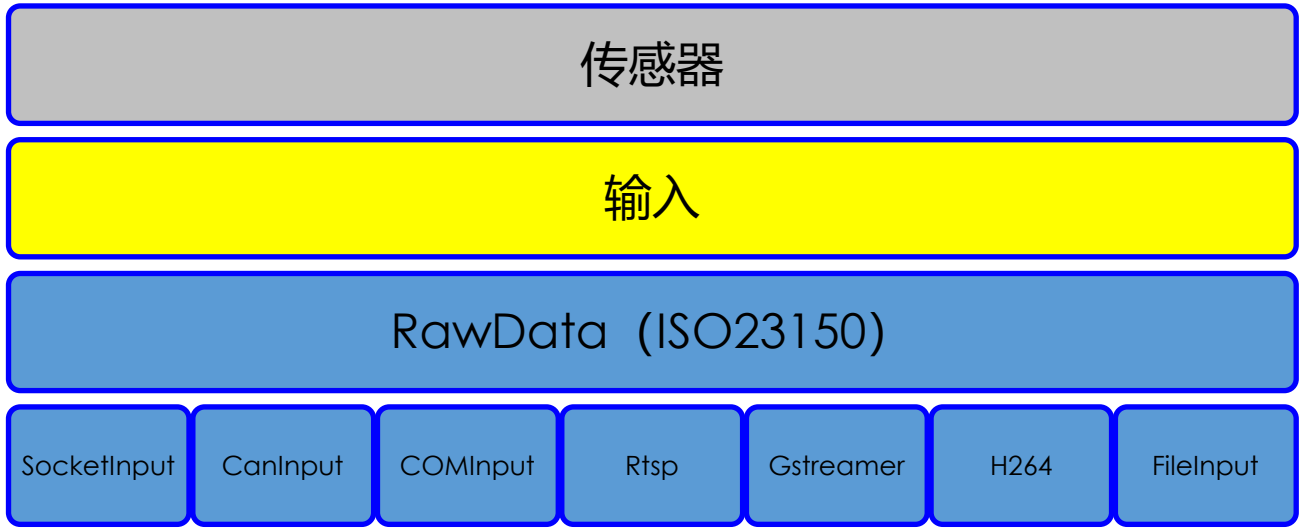
- CMakeLists.txt
- error_code.proto
- geometry.proto
- gnss.proto
- gps.proto
- header.proto
- image.proto
- imu.proto
- ins.proto
- lidar.proto
- localization_status.proto
- measure.proto
- pnc_point.proto
- pose.proto
- radar.proto
- ultrasonic.proto

使用proto定义好所有数据的类型和接口，统一数采，标定，存储解析的格式。后续无论更换什么样的协议栈都可以使用序列化好的码流进行通讯

SensorHub 架构

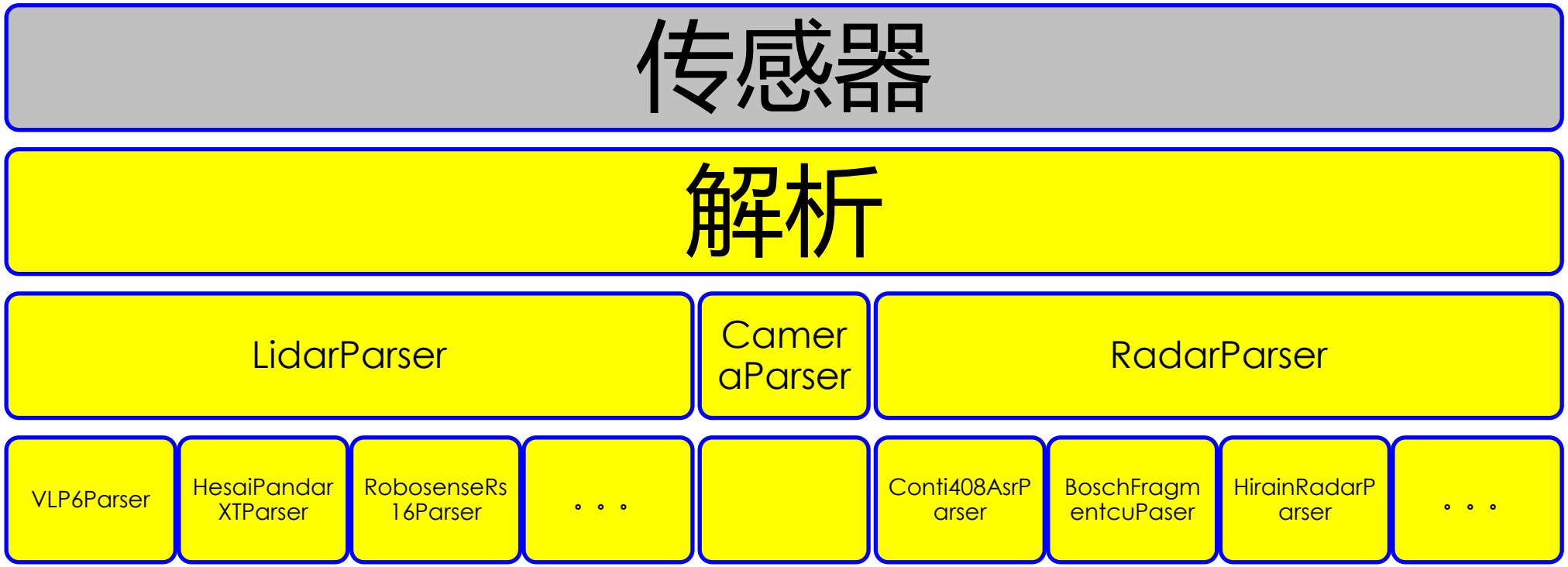


SensorHub 架构-输入

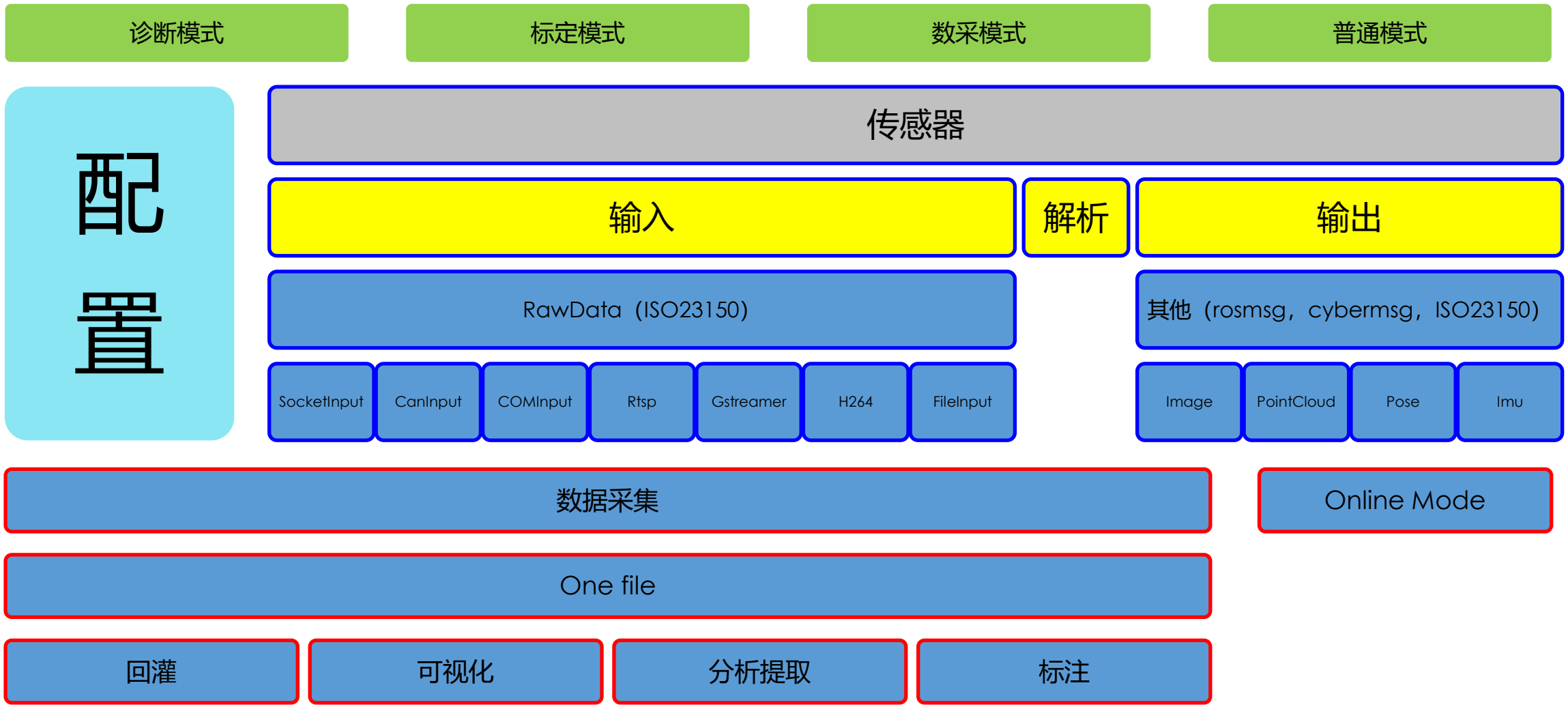


由A核统一管理输入，以方便数据统一采集

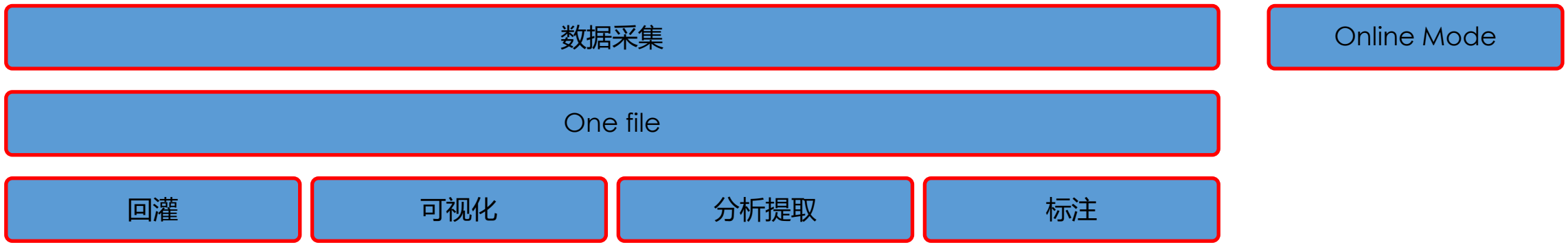
SensorHub 架构-解析



SensorHub 架构



SensorHub 架构



单例封装通讯模块，通讯模块可根据需要使用ROS，CyberRT，someip等，但是如果使用someip，围绕数据解析和整个可视化，回灌的工具链都需要自己开发。数据采集不单单只是拿到数据。不合理的数据会增加大数量的数据后处理工作。