Octopus

常见问题

文档版本	01
发布日期	2025-01-16

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 关于 Octopus

1.1 什么是 Octopus

自动驾驶云服务（Octopus）是面向车企、研究所的全托管平台，在华为云上提供自动驾驶数据云服务、自动驾驶标注云服务、自动驾驶训练云服务、自动驾驶仿真云服务、自动驾驶大模型云服务、配置管理服务，帮助车企以及研究所快速开发自动驾驶产品。

“一站式”是指自动驾驶产品开发的各个环节，包含数据资产、数据处理、数据标注、增量数据集、模型训练、仿真测试等操作都可以在Octopus上完成，支持用户从数据到应用的全流程开发；从技术上看，Octopus底层支持各种异构计算资源，开发者可以根据需要灵活选择使用，而不需要关心底层的技术，让自动驾驶开发变得更简单、更方便。

产品架构

自动驾驶云服务（Octopus）是一个一站式的开发平台，能够支撑开发者从数据收集到仿真应用的全流程开发过程。整体由数据资产、数据服务、标注服务、训练服务、仿真服务、智驾模型服务、公共配置管理组成。

图 1-1 Octopus 基础功能架构图

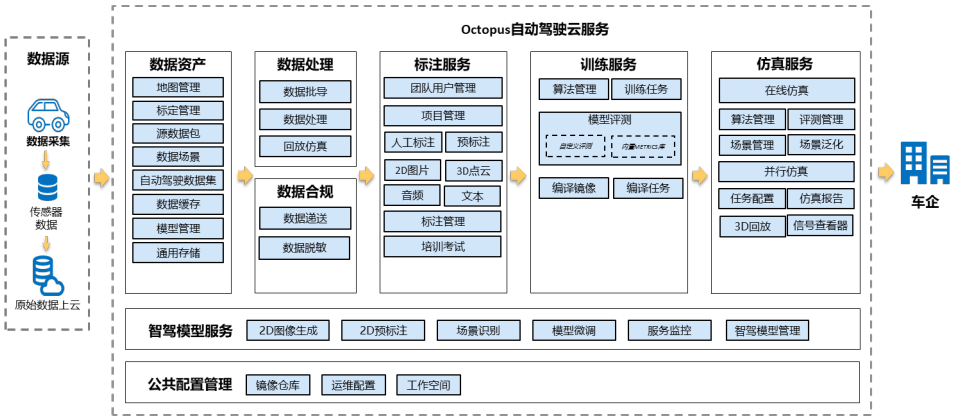
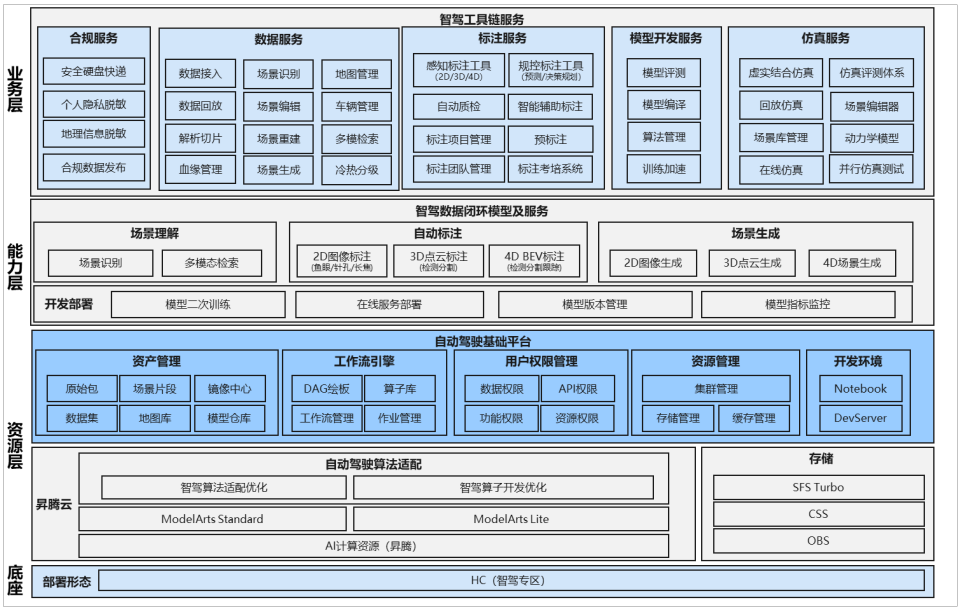


图 1-2 Octopus 高级功能架构图



产品优势

- **一站式**

  开箱即用，全托管的自动驾驶云服务，用户无需从零搭建一套复杂的自动驾驶大数据平台、AI平台、仿真平台、标注平台等多个工具平台，只需聚焦于核心价值（自动驾驶算法、标注数据、仿真场景），快速开展自动驾驶业务，跟上瞬息多变的市场节奏。

- **海量数据**

  平台可支持PB级数据存储和亿级数据秒级检索。

- **软硬件加速**

  感知算法训练和仿真需要使用大量算力资源，Octopus依托华为自研软硬件能力提供的强大算力支持，满足每天百万公里仿真测试和算法训练。

- **自动化标注**

  自动驾驶算法的持续提升依托于持续增加的高质量标注数据集，平台提供预标注范例模型，能对常用的物体如乘用车、大巴车、行人、骑行人、交通灯、可行驶区域等进行预标注，同时通过难例挖掘持续提升标注数据集质量。

- **仿真场景库**

  提供场景库管理和分布式运行能力，覆盖大部分驾驶路况，提升自动驾驶安全性。

- **并行仿真**

  实车测试成本高，危险系数高，提供并行仿真能力，能够利用云端资源快速回归仿真场景，提供上千个并行仿真节点，完成日行百万公里虚拟里程。

- **合规性**

  脱敏算子对数据包进行脱敏处理（包括人脸、车牌、gnss高程），保证用户数据上传的合规性，避免个人隐私泄露，保护用户的数据信息和财产安全。

- **昇腾云助力**

  昇腾云具有稳定安全的底层算力，提供极致性价比。昇腾双栈AI算力，支持万节点计算集群管理。全流程昇腾迁移工具链，典型自动驾驶感知算法适配昇腾，可

以大幅度缩短迁移周期（迁移周期<2周）。统一资源调度，资源极致利用，可以大幅度提升综合分配率（综合分配率达90%），弹性调度、训练和推理融合调度，大幅度缩减资源发放时间（资源发放<30分钟）。

- **大模型赋能**

  盘古大模型赋能自动驾驶，分钟级完成数据处理。自动驾驶场景理解代替人工打标签分类，万段视频片段分钟级处理完成。自动驾驶场景生成，通过NeRF技术实现车型变换、车道变换、场景组合渲染等应用。自动驾驶预标注，代替人工标注，支持2D、3D、4D自动标注，准确率超过90%。自动驾驶多模态检索支持以文搜图、以图搜图等多维检索能力，实现百万图片分钟级检索。

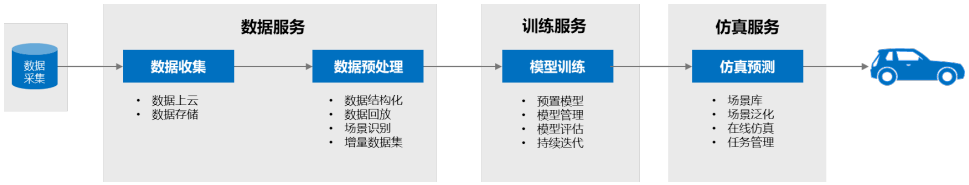
如何访问 Octopus

Octopus提供了简洁易用的云管理控制台，包含数据收集、场景挖掘、数据标注、算法管理、模型训练、模型评估、在线仿真、仿真算法和场景管理等功能，用户可以在管理控制台完成用户的自动驾驶开发。

1.2 Octopus 开发基本流程?

Octopus是一站式自动驾驶开发平台，从数据收集上云，到自动化处理数据，自动或手动标注数据，创建并增量更新数据集，并将数据集用于模型训练，以及基于特定场景的在线仿真，用户都可以在Octopus平台上完成。

图 1-3 Octopus 开发流程



- **采集数据**

  指的是数据采集车辆各传感器的原始数据，是使用Octopus平台前的准备工作。当前支持使用Rosbag数据格式收集采集数据。

- **上传数据**

  原始数据采集完毕后，在平台上创建数据收集任务，通过多种方式上传数据文件至Octopus平台。

- **数据处理**

  通过用户自定义算子对Rosbag数据包进行处理，最终将原始数据结构化，解析出各种不同传感器详细数据，如摄像头录制的图像数据、雷达的点云数据、车辆行驶轨迹等。生成的图片可以直接用于标注。

- **标注数据**

  对于图片和点云数据，可以通过自动或人工的方式，标注图像中特定物体。标注后的图片和点云图片可用于模型训练，高质量的标注数据有利于模型精准度提升，并持续迭代。

- **增量数据集**

  将标注后的数据根据数据类型、标注、标签等，建立不同种类的数据集，同时支持数据集增量更新，可针对性用于不同算法和模型的训练。

- **模型训练**

  基于平台上创建好的数据集，可对自定义算法或内置算法进行训练，并对生成的模型进行评估，也可进一步用于预标注。

- **模型评估**

  在建模过程中，由于偏差过大导致的模型欠拟合以及方差过大导致的过拟合的存在，因此需要一套评价体系，来评估模型的泛化能力。

- **在线仿真**

  仿真即通过软件模拟车辆行驶的路况和场景，不需要真实的环境和硬件，极大节省训练和测试的成本和时间。Octopus仿真服务预置了智能驾驶、主动安全、危险场景等六大场景实例，覆盖大部分驾驶路况，用户可直接在线使用，持续迭代提升自动驾驶安全性。

2 数据服务常见问题

2.1 标定文件上传失败的原因?

1. **检查标定类型**

   标定文件必须上传到对应类型标定项，如激光雷达标定文件，只能上传到激光雷达标定项，平台会对文件标定类型和关键参数进行校验。

2. **检查是否重名**

   标定文件可以重名，但上传到平台的标定项名称不可重名。

3. **检查文件内容**

   平台会对每种类型标定文件中的关键参数进行检查，确保类型匹配、标定文件内容完整。

2.2 平台对接数据格式有哪些要求?

2.2.1 上传数据格式

在使用Octopus平台收集数据前，请仔细阅读本章节，确保上传数据格式符合平台要求，有助于用户更快速的完成数据收集以及数据格式转换。

上传数据格式：Rosbag包+与数据包同名的yaml文件，单包上传大小小于100G。

转换后数据格式：OpenData格式（内必含Octopus_data_collection.yaml配置文件）。

使用场景

Octopus平台接收Rosbag格式数据包，没有转换OpenData格式时，可用于算子作业输入。

命名规范

用户可将多个数据包存放在同个一级目录下，作为同一批次采集数据上传至Octopus平台。

一级目录的结构可根据业务情况自定义数据包名称，示例参考如下：


```
.
├─ octopus
   ├─ Rosbag
      ├─{导入id}
         ├─{数据包名}
            ├─ {数据包名}.bag
            ├─ {数据包名}.yaml
            ├─ xxxxx.mp4          #非必填，大小不限制，时长限制10min，命名大小不可超过128位，不可有特
殊字符。
```

与数据包同名的 yaml 配置文件说明

数据包中必须含有与数据包同名的yaml配置文件主要包括车辆名称、传感器信息和标定ID等信息，详情参考如下：

```
# 华为八爪鱼自动驾驶云服务数据采集说明
project: '项目名称'
module: '感知'
cardrive:
   collect_time: 2020-11-01T08:00:00+08:00     #数据包采集日期，精确到小时即可
   station: '腾飞'                        #选填 数据采集地点名称，站点名称
   car:
      vehicle_name: 'test'             #车辆名称，仅支持在八爪鱼平台创建的车辆
   route: 'shuttlebus_30km'                 #选填 车辆行驶路线
   speed：10km/h                          #选填 车速
   mode: 'auto'                          #选填 路线驾驶意图, auto代表自动驾驶, manual代表人工驾驶采集
   tags: ['主车直行','主车倒车']           #选填 标签，标签个数不超过50个 例：沙尘天，正向设计，驾驶模式
   description: '强风沙天,车辆空载在排土区自动驾驶到接土区前等待长坡道'   #选填 车载情况
   segments:                             #选填 数据包场景片段
      -
        tags: ['晴天','直行']
        time: 2021-08-27T11:43:07~2021-08-27T11:43:47
data_type: Rosbag                        #必填 数据类型
map_id: MAP1134                          #选填，高精地图ID，字符串类型，配备后才可在回放数据界面展示高精
地图信息。
preprocessor:                           #转OpenData算子信息
   id: 10105                            # 算子id
   resource_spec: 4Core_8GiB              # 资源规格
```

2.2.2 转换后数据格式

Octopus平台支持将上传的Rosbag格式转换为OpenData格式。

数据类型

Octopus平台对数据有以下要求：

- 数据类型：包括各传感器数据、车辆数据、目标推理数据、自车坐标姿态以及标签记录数据等。
- 数据格式：Octopus OpenData格式。其中相机采集数据文件后缀为“.jpg”，激光雷达采集数据文件后缀为“.pcd”，其他采集数据文件后缀为“.pb”（谷歌定义的protobuf格式文件）。

  详情请参考**表1 数据类型和消息topic对应关系**。
- 消息topic具体格式要求请参考“**消息topic格式规范**”。
- 接收到的消息topic示例请参考“**消息topic格式示例**”。

除上述数据外，数据包中必须含有“Octopus_data_collection.yaml”配置文件。

说明

- 自车相关或每个传感器设备，都对应一个消息topic。
- 采集数据的topic名称支持自定义，包含中英文、数字、“_”“-”，不得超过64个字符。

表 2-1 数据类型和消息 topic 对应关系

分类	数据类型	消息topic（示例）	文件后缀	备注
传感器	相机（**camera**）	camera_front	.jpg	录制车辆路况图像数据。
	激光雷达（**lidar**）	lidar_roof_0	.pcd	以发射激光束探测目标的位置、速度等特征量的雷达系统，探测车辆周围的目标位置，监测移动速度。
	位置数据（**gnss**）	gnss_raw	.pb	通过卫星导航系统，定位车辆位置。
	毫米波雷达（**radar**）	RADAR_FRONT	.pcd	工作在毫米波波段探测的雷达，探测车辆周围的目标位置，监测移动速度。
车辆数据	自车坐标和姿态数据(**ego_tf**)	ego_tf	.pb	定位自车所处位置以及当前车辆姿态。
	车辆数据（**vehicle**）	vehicle	.pb	车辆底盘信息。
规划推理数据	目标推理数据（**object_array_vision**）	object_array_vision	.pb	感知数据信息。
标签数据	标签记录数据（**tag_record**）	tag_record	.pb	在车端标记驾驶过程中人工和自动驾驶路段以及其他重要信息。
控制数据	控制指令（**control**）	control	.pb	自车的方向盘转角、加速度值等控制数据。
规划路径	规划轨迹（**planning_trajectory**）	planning_trajectory	.pb	自车规划行驶路径。
预测路径	预测跟踪(**predicted_objects**)	predicted_objects	.pb	感知目标的预测路径。
全局规划	全局路径（**routing_path**）	routing_path	.pb	自车全局规划路径。
交通灯	交通灯信息（**traffic_light_info**）	traffic_light_info	.pb	红绿灯。

使用场景

Octopus平台接收到原始数据（Rosbag包）后，将对数据进行解包、轨迹和接管分析等操作，用于数据总览、数据场景、数据回放、标注服务等模块，请用户结合实际需求，准备好相应模块所需数据。

Octopus平台转换后的OpenData数据服务模块所需数据请见下表：

表 2-2 数据和模块对应关系

类型	消息	数据总览	数据场景	数据回放	标注服务
相机	camera	-	-	√	√
激光雷达	lidar	-	-	√	√
位置数据	gnss	√	-	√	-
自车坐标姿态	ego_tf	-	√	√	-
车辆数据	vehicle	-	√	√	-
感知推理	object_array_vision	-	√	√	-
接管及打标签信息	tag_record	-	-	√	-
控制指令	control	-	-	√	-
规划轨迹	planning_trajectory	-	-	√	-
预测跟踪	predicted_objects	-	-	√	-
全局规划	routing_path	-	-	√	-
交通灯	traffic_light_info	-	-	√	-
毫米波雷达	radar	-	-	√	-

采集数据命名规范

用户可将多个数据包存放在同个一级目录下，作为同一批次采集数据上传至Octopus平台。

一级目录的结构可根据业务情况自定义数据包名称，示例参考如下：

```
.
├─ raw-data
├─ package01
├─ OpenData
├─ 2020-04-28-08-00
└─ ...
```

Octopus OpenData 格式数据说明

Octopus OpenData格式数据目录结构可根据实际采集节点种类及数量进行修改，示例参考如下：

```
.
├─ OpenData
```

```
├─ camera_0
    ├─xxx.jpg
    ├─xxx.jpg
    ├─timestamp.jpg
    ├─...
├─ lidar_roof_0
    ├─xxx.pcd
    ├─xxx.pcd
    ├─timestamp.pcd
    ├─...
├─ gnss
    ├─episode-1.pb
├─other sensor
    ├─episode-1.pb
...

├─ Octopus_data_collection.yaml
```

说明

1. Octopus平台对Octopus OpenData数据包内文件大小限制如下：
   - “.yaml”文件小于10kb。
   - “.jpg”文件小于2MB。
   - “.pcd”文件小于10MB。
   - “.pb”文件小于50MB。
2. MP4命名大小不可超过128位，不可有特殊字符。MP4的时间需要和Rosbag包中的时间匹配。
3. 数据名称仅包含中文、大小写英文、数字、“-”“_”，不超过64位。
4. 数据包名称仅包含中文、大小写英文、数字、“-”“_”，不得出现其他字符，且长度不超过64个字符。

“Octopus_data_collection.yaml”配置文件说明

数据包中必须含有“Octopus_data_collection.yaml”配置文件，且配置文件中采集时间、车辆名称、ego_tf关键字为必填项，各类型传感器的名字必须和文件夹名称一致，格式也必须与规范相匹配，否则会导致数据上传失败。

配置文件，主要包括车辆名称、传感器信息和标定ID等信息，详情参考如下：

```
# 华为八爪鱼自动驾驶云服务数据采集说明
cardrive:
    collect_time: 2020-11-01T08:00:00+08:00     #数据包采集日期，精确到小时即可
    station: '腾飞'                          #选填 数据采集地点名称，站点名称
    car:
        vehicle_name: 'test0805'            #车辆名称，仅支持在八爪鱼平台创建的车辆
    route: 'shuttlebus_30km'                 #选填 车辆行驶路线
    mode: 'auto'                           #选填 路线驾驶意图, auto代表自动驾驶, manual代表人工驾驶采集
    tags:                                 #选填 数据包对应标签ID
    description: ''                         #选填 数据包描述
data_type: OpenData                          #必填 数据包类型，转换后的OpenData数据中包含
ocotopus_data_collection.yaml文件
map_id:''                                  #选填，高精地图ID，字符串类型，配备后才可在回放数据界面展示高精地图信
息。
folders:                                  #必填，传感器信息（硬盘递送选填，obs导入和本地直传必填）
    camera:                                #camera类型传感器  数量不超过20个
        -
            name: camera_03encode
            format: jpg
    lidar:                               #lidar类型传感器  数量不超过10个
        -
            name: pandar
            format: pcd
    gnss:                                 #gnss类型传感器  数量不超过1个
```



```
    -
        name: inspvax
        format: proto3
radar:                              #radar类型传感器  数量不超过10个
    -
        name: pandar
        format: pcd
vehicle:                            #vehicle类型传感器  数量不超过1个
    -
        name: holo_VehicleInfoMagotan
        format: proto3
ego_tf:                             #ego_tf类型传感器  数量不超过1个
    -
        name: localization_info
        format: proto3
object_array_vision:
    -                               #object_array_vision类型传感器  数量不超过5个        -
        name: tracked_objects
        format: proto3
tag_record:                          #tag_record类型传感器  数量不超过1个
    -
        name: tag_record
        format: proto3
planning_trajectory:                  #planning_trajectory类型传感器  数量不超过1个
    -
        name: planning_trajectory
        format: proto3
predicted_objects:                    #predicted_objects类型传感器  数量不超过1个
    -
        name: prediction_prediction_obstacles
        format: proto3
control:                            #control类型传感器  数量不超过1个
    -
        name: holo_ControlCommand
        format: proto3
routing_path:                         # routing_path类型传感器  数量不超过1个
    -
        name: routing_routing_response_viz
        format: proto3
traffic_light_info:                   # traffic_light_info类型传感器  数量不超过1个
    -
        name: traffic_light
        format: proto3
```

2.2.3 消息 topic 格式规范

Vehicle

对于车辆自身基本数据录制的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-3 vehicle 消息格式规范

格式名称	说明
VehicleInfo	车辆信息

⚠ 注意

消息格式中部分参数为必选，如使用该数据类型，则不可缺少该参数字段，否则会导致数据上传Octopus平台失败。

```
/**********************************
   content: Octopus 输入数据格式
   version: 0.1
**********************************/
syntax = "proto3"。

package Octopusdata。

message VehicleFrame {
   uint64 stamp_secs = 1。                #必选。时间戳，单位：秒
   uint64 stamp_nsecs = 2。               #必选。时间戳，单位：纳秒
   uint32 autonomy_status = 3。           #非必选。自动驾驶状态
   sint32 gear_value = 4。                #必选。只应从枚举常量中赋值
   float vehicle_speed = 5。              #必选。行驶速度，如果齿轮是倒挡，值为负。
   float steering_angle = 6。             #必选。转向，以角度表示。顺时针或向右为正，0为垂直或直角。
   float yaw_rate = 7。                  #Unit: deg/s
   float interior_temperature = 8。       #Unit: Celsius
   float outside_temperature = 9。        #Unit: Celsius
   float brake = 10。                    #必选。刹车制动按压百分比（0代表不按，1代表完全按下）。
      uint64 timestamp = 11。             #必选。时间戳。
   int32 turn_left_light=12。            #必选。左转灯。
   int32 turn_right_light=13。           #必选。右转灯。
   float longitude_acc=14。              #必选。纵向加速度。
   float lateral_acc=15。                #必选。横向加速度。

}

message VehicleInfo {
   repeated VehicleFrame vehicle_info = 1。
}
```

Camera

采集的camera数据通过转换工具可以保存为”.jpg”图片数据。

Lidar

采集的点云数据通过转换工具可以保存为标准的pcd格式数据。

Gnss

对于卫星导航系统数据录制的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-4 gnss 消息格式规范

格式名称	说明
GnssPoints	gps点

⚠ 注意

消息格式中部分字段为必选，如使用该数据类型，则不可缺少该参数字段，否则会导致数据上传Octopus平台失败。

```
/**********************************
   content: Octopus 输入数据格式
   version: 1.0
```

```
**********************************/
syntax = "proto3"。

package Octopusdata。

message GnssPoint {
    uint64 stamp_secs = 1。     #必选。时间戳，单位：秒
    uint64 stamp_nsecs = 2。    #必选。时间戳，单位：纳秒
        float latitude = 3。        #必选。纬度
        float longitude = 4。       #必选。经度
        float elevation = 5。       #必选。海拔高度，单位：米
        uint64 timestamp = 6。      #必选。时间戳
}

message GnssPoints {
    repeated GnssPoint gnss_points = 1。
}
```

Radar

雷达采集的点云数据通过转换工具可以保存为标准的pcd格式数据。

Ego_tf

对于自车角度位置数据录制的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-5 ego_tf 消息格式规范

格式名称	说明
LocalizationInfo	主车信息

⚠ 注意

消息格式中部分字段为必选，如使用该数据类型，则不可缺少该参数字段，否则会导致数据上传Octopus平台失败。

```
/**********************************
    content: Octopus 输入数据格式
    version: 1.0
**********************************/
syntax = "proto3"。

package Octopusdata。


message LocalizationInfoFrame {
    uint64 timestamp = 1。                  #必选。时间戳。
    uint64 stamp_secs = 2。                 #必选。时间戳，单位：秒
    uint64 stamp_nsecs = 3。                #必选。时间戳，单位：纳秒
    float pose_position_x = 4。             #必选。自车x轴坐标
        float pose_position_y = 5。             #必选。自车y轴坐标
        float pose_position_z = 6。             #必选。自车z轴坐标
    float pose_orientation_x = 7。          #必选。自车四元数x值
        float pose_orientation_y = 8。          #必选。自车四元数y值
        float pose_orientation_z = 9。          #必选。自车四元数z值
    float pose_orientation_w = 10。         #必选。自车四元数w值
    float pose_orientation_yaw=11。         #必选。朝向角，单位：rad
    float velocity_linear=12。              #必选。速度，单位：m/s
```

```
    float velocity_angular=13。              #必选。角速度，单位：rad/s
    float acceleration_linear=14。           #必选。加速度，单位：m^2/s
    float acceleration_angular=15。          #必选。角加速度，单位：rad^2/s
}

message LocalizationInfo {
    repeated LocalizationInfoFrame localization_info = 1。
}
```

Object_array_vision

对于目标推理数据录制的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-6 object_array_vision 消息格式规范

格式名称	说明
TrackedObject	感知目标

⚠ 注意

消息格式中部分字段为必选，如使用该数据类型，则不可缺少该参数字段，否则会导致数据上传Octopus平台失败。

```
/**********************************
    content: Octopus 输入数据格式
    version: 1.0
**********************************/
syntax = "proto3"。

package Octopusdata。


message Object {
    uint64 id = 1。                    #必选。目标推理数据object数组id
    string label = 2。                 #必选。标记物体类型
    float pose_position_x = 3。          #必选。目标物x轴坐标
        float pose_position_y = 4。        #必选。目标物y轴坐标
        float pose_position_z = 5。        #必选。目标物z轴坐标
    float pose_orientation_x = 6。       #必选。目标物四元数x值
        float pose_orientation_y = 7。     #必选。目标物四元数y值
        float pose_orientation_z = 8。     #必选。目标物四元数z值
    float pose_orientation_w = 9。       #必选。目标物四元数w值
    float pose_orientation_yaw = 10。    #必选。朝向角，单位：rad
    float dimensions_x = 11。            #必选。目标物x方向尺寸（长）
    float dimensions_y = 12。            #必选。目标物y方向尺寸（宽）
    float dimensions_z = 13。            #必选。目标物z方向尺寸（高）
    float speed_vector_linear_x = 14。   #必选。目标物x方向速度
        float speed_vector_linear_y = 15。   #必选。目标物y方向速度
        float speed_vector_linear_z = 16。   #必选。目标物z方向速度
        float relative_position_x = 17。     #必选。目标物相对于主车x方向位置
        float relative_position_y = 18。     #必选。目标物相对于主车y方向位置
        float relative_position_z = 19。     #必选。目标物相对于主车z方向位置
}
message TrackedObjectFrame {
    uint64 timestamp = 1。             #必选。时间戳
    uint64 stamp_secs = 2。            #必选。时间戳，单位：秒
    uint64 stamp_nsecs = 3。           #必选。时间戳，单位：纳秒
    repeated Object objects = 4。      #必选。object数组
}
```

```
message TrackedObject {
    repeated TrackedObjectFrame tracked_object = 1。
}
```

Tag_record

对于标签记录数据录制的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-7 tag_record 消息格式规范

格式名称	说明
ScenarioSegments	场景片段

```
/**********************************
    content: Octopus 输入数据格式
    version: 1.0
**********************************/
syntax = "proto3"。

package Octopusdata。

message ScenarioSegment {
 uint32 scenario_id = 1。                #必选。场景id
 string source = 2。                    #必选。片段的来源
 uint64 start = 3。                     #必选。片段的开始时间（时间戳）
 uint64 end = 4。                       #必选。片段的结束时间（时间戳）
}

message ScenarioSegments {
 repeated ScenarioSegment segments = 1。
}
```

Control

对于控制数据的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-8 control 消息格式规范

格式名称	说明
ControlCommand	控制命令

```
/**********************************
    content: Octopus 输入数据格式
    version: 1.0
**********************************/
syntax = "proto3"。
package Octopusdata。
message CommandFrame {
    uint64 stamp_secs = 1。
    uint64 stamp_nsecs = 2。
    uint64 timestamp = 3。                    #必选，时间戳
    float acceleration=4。                    #必选，加速度值
    float front_wheel_angle=5。               #必选，方向盘转角
    int32 gear=6。
```

```
}
message ControlCommand {
    repeated CommandFrame command_frame = 1。
}
```

Predicted_objects

对于预测路径数据的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-9 predicted_objects 消息格式规范

格式名称	说明
PredictionObstacles	预测障碍物

```
/**********************************
   content: Octopus 输入数据格式
   version: 1.0
**********************************/
syntax = "proto3"。
package Octopusdata。
message PathPoint {
   float x = 1。          #必选，预测轨迹点x坐标
   float y = 2。          #必选，预测轨迹点y坐标
   float z = 3。          #必选，预测轨迹点z坐标
   float theta = 4。
   float kappa = 5。
   int32 lane_id= 6。
   float v=7。
   float a=8。
   float relative_time=9。
}
message PredictionTrajectory {
  repeated PathPoint path_point = 1。   #必选，预测轨迹多个点
}
message Obstacle {
   uint64 obstacle_timestamp = 1。
   int32 id=2。               #必选，预测目标的id
   float x = 3。  #非必选，预测目标的x坐标
   float y = 4。  #非必选，预测目标的y坐标
   float z = 5。  #非必选，预测目标的z坐标
   repeated PredictionTrajectory prediction_trajectory = 6。   #必选，预测目标的多条轨迹
}
message PerceptionObstacle {
  uint64 stamp_secs = 1。
  uint64 stamp_nsecs = 2。
  uint64 timestamp = 3。   #必选，预测目标的时间戳
  repeated Obstacle obstacle_info= 4。  #必选，多个目标的预测信息
}
message PredictionObstacles {
  repeated PerceptionObstacle perception_obstacle= 4。  #必选，多条帧数据
}
```

Planning_trajectory

对于规划路径数据的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-10 planning_trajectory 消息格式规范

格式名称	说明
PlanTrajectory	规划路径

```
/**********************************
    content: Octopus 输入数据格式
    version: 1.0
**********************************/
syntax = "proto3"。
package Octopusdata。
message TrajectoryPoint {
    float x = 1。    #必选，轨迹点x坐标
    float y = 2。    #必选，轨迹点y坐标
    float z = 3。    #必选，轨迹点z坐标
    float theta = 4。
    float kappa = 5。
    int32 lane_id=6。
    float v=7。    #必选，速度
    float a=8。    #必选，加速度
    float relative_time=9。  #必选，相对时间
}
message Trajectory {
    uint64 stamp_secs = 1。
    uint64 stamp_nsecs = 2。
    uint64 timestamp = 3。     #必选，时间戳
    float total_path_length = 4。
    float total_path_time=5。
    int32 gear=6。       #非必选，档位
    int32 trajectory_type=7。
    int32 vehicle_signal=8。
    repeated TrajectoryPoint trajectory_points = 9。  #必选，轨迹
}
message PlanTrajectory {
     repeated Trajectory trajectory_info= 1。
}
```

Routing_path

对于全局规划路径数据的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-11 routing_path 消息格式规范

格式名称	说明
RoutingFrames	规划路径

```
/**********************************
    content: Octopus 输入数据格式
    version: 1.0
**********************************/
syntax = "proto3"。

message Point{
    float x = 1。
    float y = 2。
    float z = 3。
}
```



```
message Path{
    uint64 id = 1。
    repeated Point path_point = 2。
}

message RoutingPath{
    uint64 timestamp = 1。
    uint64 stamp_secs = 2。
    uint64 stamp_nsecs = 3。
    repeated Path routing_path_info = 4。
}

message RoutingFrames{
    repeated RoutingPath routing_frame = 4。}
```

Traffic_light_info

对于交通灯数据的消息格式，需遵循一定规范，其中部分字段为必选，其他请根据实际需要自由选取。

表 2-12 traffic_light_info 消息格式规范

格式名称	说明
TrafficLightInfo	交通灯

```
/**************************************
    content: Octopus 输入数据格式
    version: 1.0
**************************************/
syntax = "proto3"。

package Octopusdata。

message Light {
    uint64 id = 1。
    uint64 color = 2。
    uint64 state = 3。
    uint64 type = 4。
    float location_x = 5。
    float location_y = 6。
    float location_z = 7。
}

message Lights {
    uint64 timestamp = 1。
    uint64 stamp_secs = 2。
    uint64 stamp_nsecs = 3。
    repeated Light lights = 4。
}

message TrafficLightInfo {
    repeated Lights trafficlight_info = 1。
}
```

2.2.4 消息 topic 格式示例

Vehicle

Vehicle格式示例：


```
vehicle_info {
    stamp_secs: 1604996332
    stamp_nsecs: 847945211
    autonomy_status: 0
    gear_value: 4
    vehicle_speed: 43.93000030517578
    steering_angle: 0.699999988079071
    yaw_rate: 0.0
    interior_temperature: 0.0
    outside_temperature: 0.0
    brake: 0.0
    timestamp: 1604996332847
    turn_left_light: 0
    turn_right_light: 0
    longitude_acc: -0.03125
    lateral_acc: 0.0
}
```

Gnss

Gnss格式示例：

```
gnss_points {
    stamp_secs: 1604996332
    stamp_nsecs: 855145358
    latitude: 30.18027687072754
    longitude: 120.21341705322266
    elevation: 7.392796039581299
    timestamp: 1604996332855
}
```

Ego_tf

Ego_tf格式示例：

```
localization_info {
    timestamp: 1604996332855
    stamp_secs: 1604996332
    stamp_nsecs: 855301408
    pose_position_x: 1165.5460205078125
    pose_position_y: -479.2198486328125
    pose_position_z: -1.48505699634552
    pose_orientation_x: 0.003883248195052147
    pose_orientation_y: -0.0031167068518698215
    pose_orientation_z: 0.7017714977264404
    pose_orientation_w: 0.7123847603797913
    pose_orientation_yaw: 1.5557808876037598
    velocity_linear: 12.21684455871582
    velocity_angular: 0.014540454372763634
    acceleration_linear: 0.23571151494979858
    acceleration_angular：  0.0
}
```

Object_array_vision

Object_array_vision格式示例：

```
tracked_object {
    timestamp: 1604996332862
    stamp_secs: 1604996332
    stamp_nsecs: 862911489
    objects {
      id: 26175
      label: "Car"
      pose_position_x: 1154.59912109375
      pose_position_y: -496.5350646972656
```



```
  pose_position_z: -1.8222997188568115
  pose_orientation_z: 0.714431643486023
  pose_orientation_w: 0.6997052431106567
  pose_orientation_yaw: 1.5916229486465454
  dimensions_x: 4.513162136077881
  dimensions_y: 1.7747581005096436
  dimensions_z: 1.628068208694458
  speed_vector_linear_x: 0.012852923013269901
  speed_vector_linear_y: -9.972732543945312
  relative_position_x: -17.48011016845703
  relative_position_y: 10.685434341430664
  relative_position_z: -0.17673441767692566
}
objects {
  id: 26170
  label: "Pedestrian"
  pose_position_x: 1180.902099609375
  pose_position_y: -504.7625732421875
  pose_position_z: -1.3601081371307373
  pose_orientation_z: -0.7057344317436218
  pose_orientation_w: 0.7084764242172241
  pose_orientation_yaw: -1.5669186115264893
  dimensions_x: 0.7922295331954956
  dimensions_y: 0.7891787886619568
  dimensions_z: 1.6868246793746948
  speed_vector_linear_x: 0.13573257625102997
  speed_vector_linear_y: 1.5281875133514404
  relative_position_x: -25.306795120239258
  relative_position_y: -15.737456321716309
  relative_position_z: 0.39350399374961853
}
objects {
  id: 26169
  label: "Pedestrian"
  pose_position_x: 1175.647216796875
  pose_position_y: -506.730712890625
  pose_position_z: -1.569373607635498
  pose_orientation_z: 0.6943609118461609
  pose_orientation_w: 0.7196269631385803
  pose_orientation_yaw: 1.5350627899169922
  dimensions_x: 0.8029457330703735
  dimensions_y: 0.7876891493797302
  dimensions_z: 1.6028095483779907
  speed_vector_linear_x: 0.06551000475883484
  speed_vector_linear_y: 0.0022428608499467373
  relative_position_x: -27.355571746826172
  relative_position_y: -10.512933731079102
  relative_position_z: 0.19844147562980652
}
objects {
  id: 26168
  label: "Pedestrian"
  pose_position_x: 1173.3189697265625
  pose_position_y: -507.2300109863281
  pose_position_z: -1.6026556491851807
  pose_orientation_z: 0.717462956905365
  pose_orientation_w: 0.6965966820716858
  pose_orientation_yaw: 1.600306749343872
  dimensions_x: 0.7922430038452148
  dimensions_y: 0.7811086177825928
  dimensions_z: 1.6341478824615479
  speed_vector_linear_x: -0.04817964881658554
  speed_vector_linear_y: -0.21502695977687836
  relative_position_x: -27.89008903503418
  relative_position_y: -8.192517280578613
  relative_position_z: 0.16775710880756378
}
objects {
  id: 26155
```



```
  label: "Bus"
  pose_position_x: 1172.106689453125
  pose_position_y: -478.5303039550781
  pose_position_z: -0.48812994360923767
  pose_orientation_z: -0.7203028798103333
  pose_orientation_w: 0.6936596632003784
  pose_orientation_yaw: -1.6084778308868408
  dimensions_x: 11.322981834411621
  dimensions_y: 2.9294095039367676
  dimensions_z: 3.1415622234344482
  speed_vector_linear_x: -0.017722932621836662
  speed_vector_linear_y: 0.1302066147327423
  relative_position_x: 0.7977913022041321
  relative_position_y: -6.548437118530273
  relative_position_z: 0.9966707229614258
}
objects {
  id: 26153
  label: "Bus"
  pose_position_x: 1148.1876220703125
  pose_position_y: -490.8350524902344
  pose_position_z: -0.954763650894165
  pose_orientation_z: 0.6907882690429688
  pose_orientation_w: 0.7230570912361145
  pose_orientation_yaw: 1.5251574516296387
  dimensions_x: 10.779899597167969
  dimensions_y: 2.856076717376709
  dimensions_z: 2.811084508895874
  speed_vector_linear_x: 0.03153659775853157
  speed_vector_linear_y: 0.23439916968345642
  relative_position_x: -11.868709564208984
  relative_position_y: 17.1827335357666
  relative_position_z: 0.6278138756752014
}
objects {
  id: 26141
  label: "Bus"
  pose_position_x: 1171.7779541015625
  pose_position_y: -512.5936889648438
  pose_position_z: -0.9443151354789734
  pose_orientation_z: -0.7186583876609802
  pose_orientation_w: 0.6953632831573486
  pose_orientation_yaw: -1.6037421226501465
  dimensions_x: 10.841312408447266
  dimensions_y: 2.9661808013916016
  dimensions_z: 3.2250704765319824
  speed_vector_linear_x: 0.0513402484357357
  speed_vector_linear_y: 0.006104861851781607
  relative_position_x: -33.26952362060547
  relative_position_y: -6.731308937072754
  relative_position_z: 0.8776476979255676
}
objects {
  id: 26133
  label: "Bus"
  pose_position_x: 1146.657958984375
  pose_position_y: -508.7508239746094
  pose_position_z: -0.883571445941925
  pose_orientation_z: 0.7007946968078613
  pose_orientation_w: 0.713362991809845
  pose_orientation_yaw: 1.5530219078063965
  dimensions_x: 12.186415672302246
  dimensions_y: 2.824420690536499
  dimensions_z: 3.292656183242798
  speed_vector_linear_x: 0.005901232361793518
  speed_vector_linear_y: 0.013970088213682175
  relative_position_x: -29.803848266601562
  relative_position_y: 18.443498611450195
  relative_position_z: 0.8749525547027588
```



```
  }
  objects {
    id: 26120
    label: "Bus"
    pose_position_x: 1170.993408203125
    pose_position_y: -525.5801391601562
    pose_position_z: -1.104852318763733
    pose_orientation_z: -0.7154129147529602
    pose_orientation_w: 0.6987019181251526
    pose_orientation_yaw: -1.5944297313690186
    dimensions_x: 10.749905586242676
    dimensions_y: 2.7170863151550293
    dimensions_z: 3.0421104431152344
    speed_vector_linear_x: 0.016746148467063904
    speed_vector_linear_y: -0.23609620332717896
    relative_position_x: -46.26727294921875
    relative_position_y: -6.141877174377441
    relative_position_z: 0.8449855446815491
  }
}
```

Tag_record

Tag_record格式示例：

```
segments {
  scenario_id: 100000000
  source: "takeover"
  start: 1617336642300
  end: 1617336652300
}
segments {
  scenario_id: 100000000
  source: "vehicle"
  start: 1617336672300
  end: 1617336692300
}
```

Control

Control格式示例：

```
stamp_secs: 1617336640
stamp_nsecs: 795364700
timestamp: 1617336640795
acceleration: 0.7146586179733276
front_wheel_angle: 2.9919793605804443
```

Predicted_objects

Predicted_objects格式示例：

```
stamp_secs: 1617336640
stamp_nsecs: 971891550
timestamp: 1617336640971
obstacle_info {
  obstacle_timestamp: 1617336640699
  id: 6711
  x: -123.08731842041016
  y: 486.83221435546875
  z: 0.575542688369751
  prediction_trajectory {
    path_point {
      x: -103.26817321777344
      y: 486.0815734863281
      theta: -0.007839304395020008
```



```
      v: 4.405668258666992
      relative_time: 4.5
    }
    path_point {
      x: -102.82765197753906
      y: 486.0737609863281
      theta: -0.00746726430952549
      v: 4.405668258666992
      relative_time: 4.599999904632568
    }
    ......
  }
}
obstacle_info {
  obstacle_timestamp: 1617336640699
  id: 6744
  x: -145.0320587158203
  y: 491.35015869140625
  z: -0.40381166338920593
  prediction_trajectory {
    path_point {
      x: -145.0320587158203
      y: 491.35015869140625
      theta: -2.9442124366760254
      v: 1.0038001537322998
    }
    path_point {
      x: -145.1304931640625
      y: 491.3304748535156
      theta: -2.9442124366760254
      v: 1.0038001537322998
      relative_time: 0.10000000149011612
    }
    ......
  }
}
obstacle_info {
  obstacle_timestamp: 1617336640699
  id: 6760
  x: -138.3047332763672
  y: 489.9286193847656
  z: -0.12651222944259644
}
```

Planning_trajectory

Planning_trajectory格式示例：

```
stamp_secs: 1617336640
stamp_nsecs: 809739351
timestamp: 1617336640809
trajectory_points {
  x: -151.27487182617188
  y: 486.55096435546875
  theta: 0.0023324606008827686
  kappa: -0.0017824547830969095
}
trajectory_points {
  x: -151.21182250976562
  y: 486.5510559082031
  theta: 0.0022713469807058573
  kappa: -0.0017127590253949165
}
......
```

Routing_path

Routing_path格式示例：


```
timestamp: 1630057162125
stamp_secs: 1630057162
stamp_nsecs: 125769156
routing_path_info {
  id: 1
  path_point {
    x: -203.34230041503906
    y: 125.63516998291016
    z: -0.5
  }
  path_point {
    x: -203.34915161132812
    y: 125.72517395019531
    z: -0.5
  }.....}
```

Traffic_light_info

Traffic_light_info格式示例：

```
timestamp: 1630057508000
stamp_secs: 1630057508
lights {
  id: 1
  color: 1
  location_x: -206.60186767578125
  location_y: 459.9820861816406
  location_z: 3.0
}
lights {
  id: 2
  color: 2
  location_x: -74.1282958984375
  location_y: 484.984619140625
  location_z: 4.0
}
lights {
  id: 3
  color: 3
  location_x: 59.96036911010742
  location_y: 473.6038513183594
  location_z: 5.0
}
```

3 仿真服务常见问题

3.1 如何一键恢复在线仿真功能?

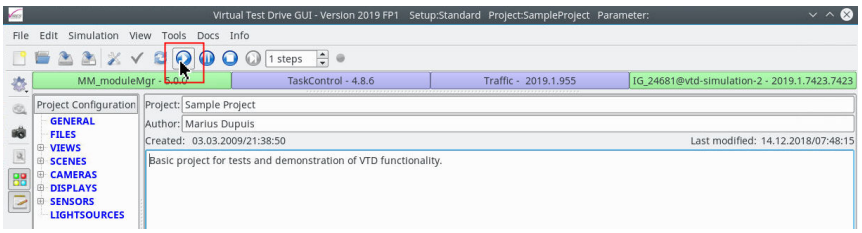
现象：使用在线仿真功能时，场景损坏导致加载失败，或在线仿真软件所在机器系统发生故障导致数据丢失或其他不可预知问题。

解决办法：

1. 重启在线仿真软件并重新加载场景。

   关闭在线仿真软件并重新启动，先单击 √图标，再单击在线仿真软件播放按钮。

   图 3-1 在线仿真软件播放按钮

2. 恢复系统镜像至所需版本。

   在线仿真所在机器，提供了系统镜像的备份、恢复、删除功能，建议用户在环境配置稳定后，就备份一个初始版本。并根据实际业务需要，定期备份系统镜像，以保障业务稳定和数据安全。

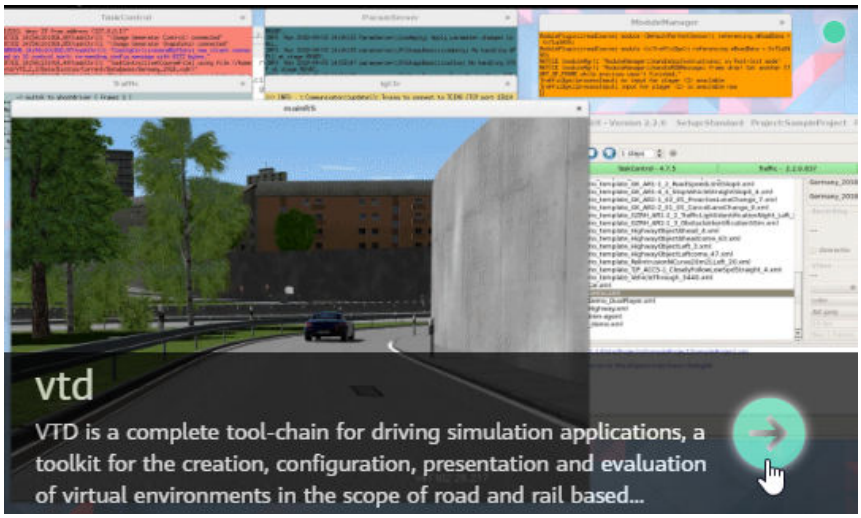
3.2 如何解决不小心释放在线仿真机器的问题?

现象：使用在线仿真时，操作过程中不小心关闭页面、退出登录或释放机器。

解决办法：

步骤1 再次选择“仿真服务 > 在线仿真”，找到指定仿真机器，单击进入在线仿真页面。

图 3-2 进入在线仿真页面入口



步骤2 进入仿真机器后，单击  等待刷新，约10s后桌面会启动完成。

说明

如果仿真机器被释放掉，重新进入会有桌面初始化的状态显示，完成初始化即可进入桌面

----结束

3.3 仿真场景终止条件有几种?

仿真终止的条件有以下几种：

1. 超时。
   创建任务时设置的“最大运行时长”，超过运行时间时，仿真任务停止。

图 3-3 超时

2. 场景文件中设置终点。

   a. xml类型场景。

      Read the destination of Ego in xml end triggers。

```
<Scenario>
 ...
 <Action Name="ReachPositionCondition">
  <PosAbsolute
     CounterID=""
     CounterComp="COMP_EQ"
     Radius="8.0000000000000000e+00"     （关键行，检测半径）
     X="2.4439516235319047e+02"          （关键行，x坐标）
     Y="-9.1187807788708923e+00"         （关键行，y坐标）
     Z="0"                               （关键行，z坐标）
     NetDist="false"
     CounterVal="0"
     Pivot="Ego"
  />
  <SCP
     ExecutionTimes="1"
     ActiveOnEnter="true"
     DelayTime="5.0000000000000000e+00"   （关键行，延时执行，建议5s）
   ><![CDATA[<SimCtrl><Stop/></SimCtrl>]]></SCP
  >
 </Action>
 ...
</Scenario>
```

      说明

      为了到达终点的评测，一般建议延迟5s执行，等待评测检测成功。此处对应前文中的DelayTime。

   b. xosc类型场景，OpenSCENARIO 1.0版本如下：

      Read the destination of Ego in xosc end triggers。

```
<StopTrigger>
 <ConditionGroup>
  <Condition
     name=""
     delay="0.000000"     （关键行，延时执行，建议5s）
     conditionEdge="rising">
   <ByEntityCondition>
    <TriggeringEntities triggeringEntitiesRule="any">
     <EntityRef entityRef="Ego"/>
    </TriggeringEntities>
    <EntityCondition>
     <ReachPositionCondition tolerance="2.000000">（关键行，检测半径）
      <Position>
       <WorldPosition
        x="0.178000"    （关键行，x坐标）
        y="5.621000"    （关键行，y坐标）
        z="0.000000"    （关键行，z坐标）
        h="0.000000"
        p="0.000000"
        r="0.000000"/>
      </Position>
     </ReachPositionCondition>
    </EntityCondition>
   </ByEntityCondition>
  </Condition>
 </ConditionGroup>
</StopTrigger>
```

      说明

      为了到达终点的评测，一般建议延迟5s执行，等待评测检测成功。此处对应前文中的DelayTime。

c. xosc类型场景，OpenSCENARIO 0.9.1的示例如下：

Read the expected destination of Ego in xosc。

```
<OpenSCENARIO>
   <FileHeader author="Octopus/Simulation" date="2022-07-07T07:27:18"
description="AcquirePositionExample" revMajor="0" revMinor="9"/>
   <Catalogs>
      <VehicleCatalog>
         <Directory path="Distros/Current/Config/Players/Vehicles"/>
      </VehicleCatalog>
      <DriverCatalog>
         <Directory path="Distros/Current/Config/Players/driverCfg.xml"/>
      </DriverCatalog>
      <PedestrianCatalog>
         <Directory path=""/>
      </PedestrianCatalog>
      <PedestrianControllerCatalog>
         <Directory path=""/>
      </PedestrianControllerCatalog>
      <MiscObjectCatalog>
         <Directory path="Distros/Current/Config/Players/Objects"/>
      </MiscObjectCatalog>
      <EnvironmentCatalog>
         <Directory path=""/>
      </EnvironmentCatalog>
      <ManeuverCatalog>
         <Directory path=""/>
      </ManeuverCatalog>
      <TrajectoryCatalog>
         <Directory path=""/>
      </TrajectoryCatalog>
      <RouteCatalog>
         <Directory path=""/>
      </RouteCatalog>
   </Catalogs>
   <RoadNetwork>
      <Logics filepath="Projects/Current/Databases/Germany.2018/Germany_2018.xodr"/>
      <SceneGraph filepath="Projects/Current/Databases/Germany.2018/
Germany_2018.opt.osgb"/>
   </RoadNetwork>
   <Entities>
   </Entities>
   <Storyboard>
      <Init>
         <Actions>
         </Actions>
      </Init>
      <EndConditions>
         <ConditionGroup>
            <Condition delay="0" edge="rising" name="End Condition">
               <ByEntity>
                  <TriggeringEntities rule="any">
                     <Entity name="Ego"/>
                  </TriggeringEntities>
                  <EntityCondition>
                     <ReachPosition tolerance="10">
                  <Position>
                    <World h="3" p="1" r="2" x="100" y="200" z="300"/>
                  </Position>
                </ReachPosition>
                  </EntityCondition>
               </ByEntity>
            </Condition>
         </ConditionGroup>
      </EndConditions>
   </Storyboard>
</OpenSCENARIO>
```


3.4 同一个任务配置运行多次仿真任务都可以改变什么?

基于同一个任务配置运行多次仿真任务，可以更改“算法版本”。

不支持修改任务配置和场景库、测试套件的关联关系，但是可以继续往场景库以及套件中增删场景或用例。新运行的任务，则会读取当下场景库或用例中的场景数据。如果清空里面的有效场景或用例，会导致任务运行失败。

不支持修改任务配置和评测的关联关系，但是评测自身属性，可以在评测管理页面进行调整。再次启动任务时，将使用新的评测方式。

3.5 采样方式有几种?

蒙特卡洛采样

蒙特卡洛采样是一种简单的随机抽样，根据概率分布进行采样，如对样本服从μ=0，δ=1的正态分布，通过通过蒙特卡洛采样进行采样，采样得到的点能满足正态分布要求，但如下图所示，采样得到的点会集中μ=0附近，要想采样得到更边界的点，需要进行大量采样。

图 3-4 蒙特卡洛采样

拉丁超立方采样

拉丁超立方采样的目的是用较少的采样次数，来达到与多次蒙特卡洛采样相同的结果，并且涵盖更全面的边界点。

如下图所示，同样对于μ=0，δ=1的正态分布，可以利用更少的采样点得到相同的分布，并且不会产生明显的聚集现象，边界值也能更容易获取到。

图 3-5 拉丁超立方采样

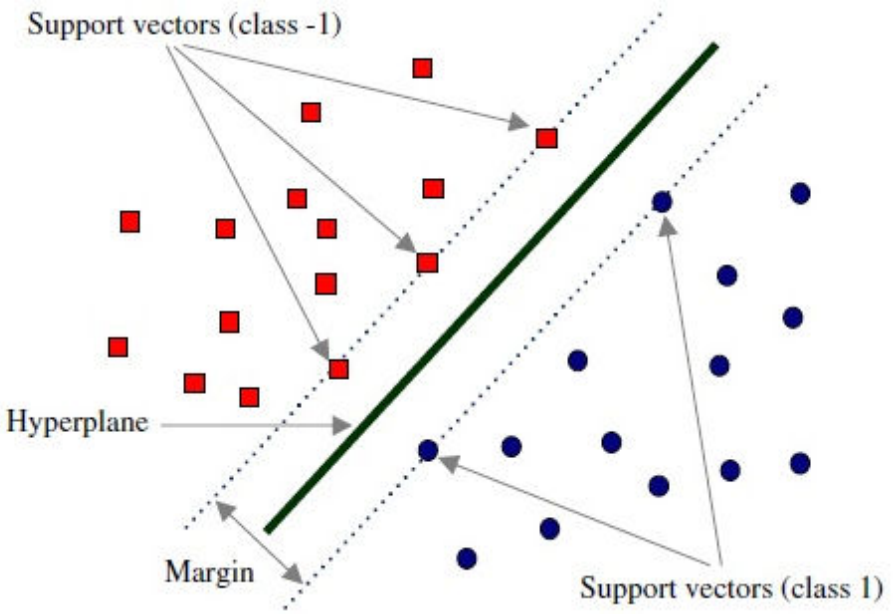
联合概率分布采样

联合概率分布采样假设连续型参数符合正态分布，支持录入连续型参数之间的相关系数（值为1时，表示变量完全正相关。值为0时，表示变量间独立。值为-1时，表示变量完全负相关。），并根据参数分布和相关系数进行联合概率分布采样。而离散型参数根据给定的取值列表进行随机采样。

重要型采样

重要性采样是在优化目标边界附近进行采样，利用上一次泛化场景仿真后得到的评测分数进行训练拟合，找到边界后不断在边界附近进行采样。

图 3-6 重要型采样



采样结果

如下图所示，对于某个逻辑场景进行拉丁超立方和蒙特卡洛采样，参数都是符合随机特性，但采样得到的值不相同。

对于同样的采样点数，拉丁超立方采样的结果会更加分散，并且边界值会更多。

图 3-7 蒙特卡洛采样结果

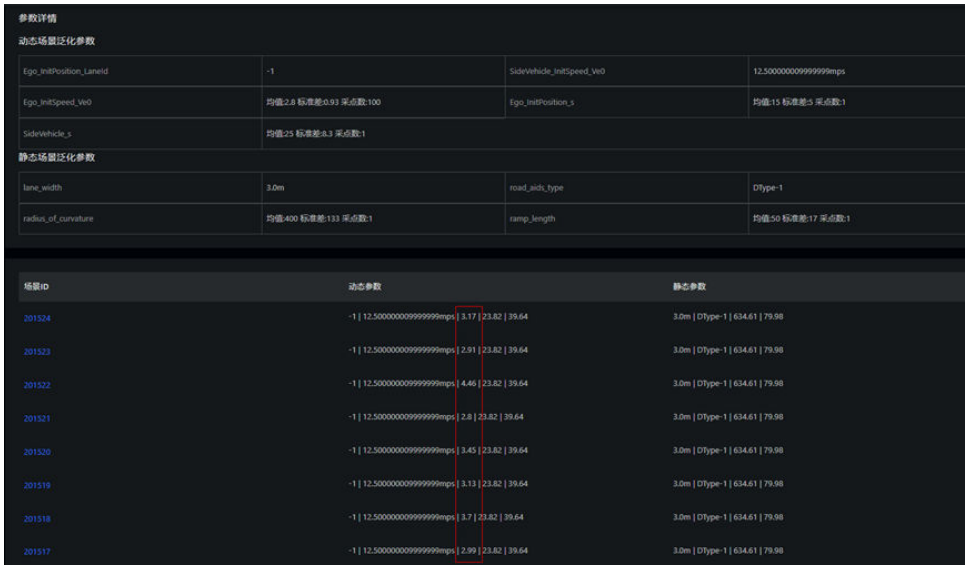
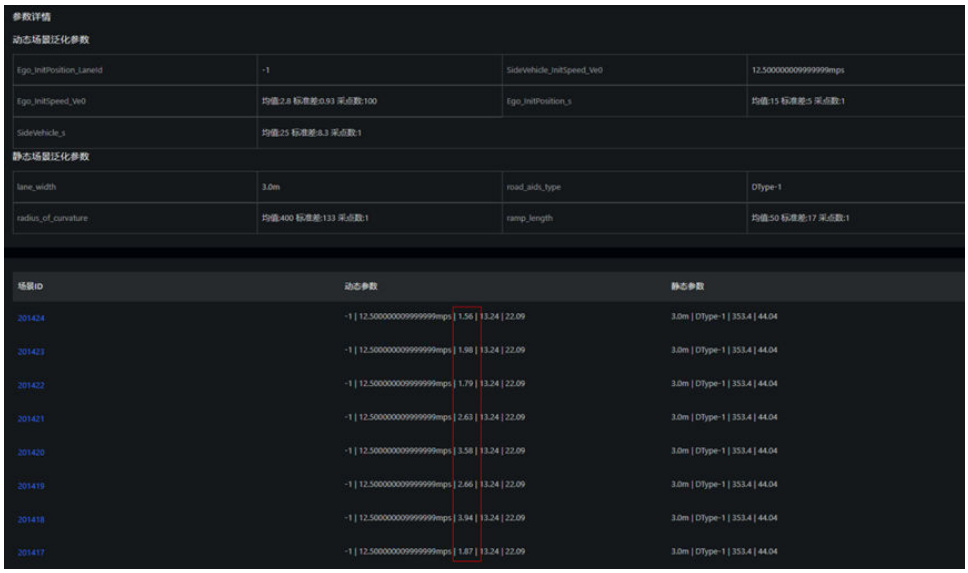
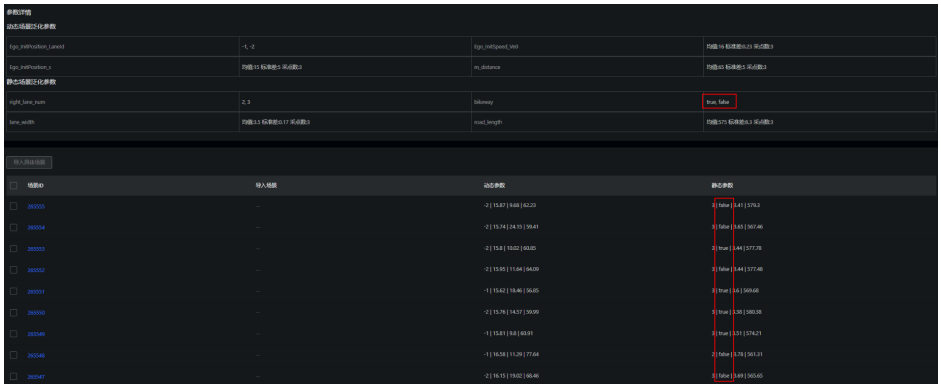


图 3-8 拉丁超立方采样结果



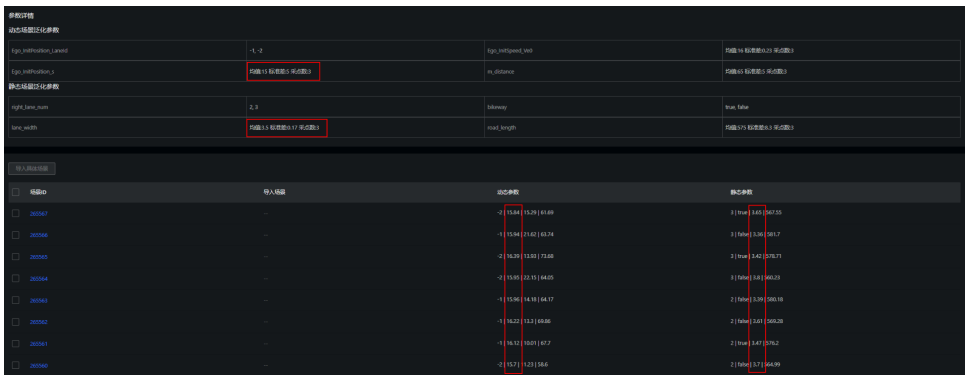
对于**离散型参数**，联合概率分布采样会根据给定的取值列表进行随机采样。

图 3-9 离散型参数-联合概率分布采样结果



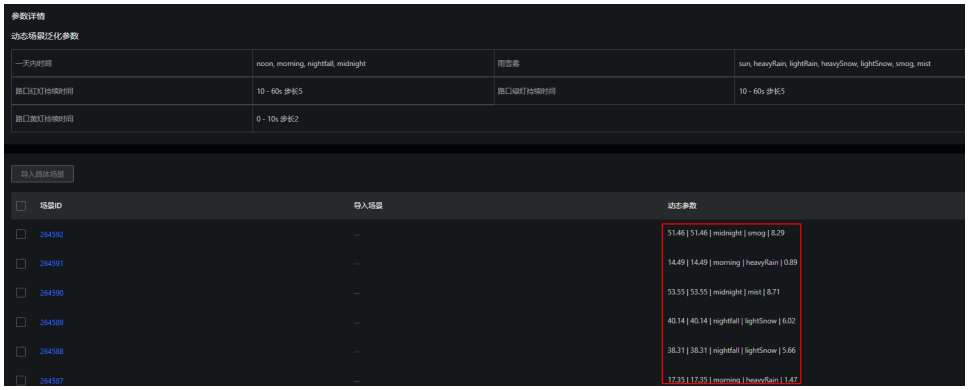
对于**连续型参数**，联合概率分布采样会根据参数分布和相关系数进行采样。

图 3-10 连续型参数-联合概率分布采样结果



对于**重要性采样**，联合概率分布采样会在边界附近进行采样。

图 3-11 重要性采样结果



4 其他常见问题

4.1 如何上传数据至 OBS?

使用Octopus进行自动驾驶开发时，您需要将数据上传至对象存储服务（OBS）桶中。

您可以登录OBS管理控制台创建OBS桶，并在您创建的OBS桶中创建文件夹，然后再进行数据的上传。

OBS上传数据的详细操作请参见**《对象存储服务快速入门》**。

4.2 如何查看账号 ID 和 IAM 用户 ID?

1. 使用IAM账号登录华为云。
2. 在页面左上方单击“控制台”，进入华为云管理控制台。
3. 在控制台右上角的账户名下方，单击“我的凭证”，进入“我的凭证”页面。
4. 在API凭证页面获取IAM用户名、用户ID、账号名和账号ID。

4.3 如何获取访问密钥 AK/SK?

1. 使用IAM账号登录华为云。
2. 在页面左上方单击“控制台”，进入华为云管理控制台。
3. 在控制台右上角的账户名下方，单击“我的凭证”，进入“我的凭证”页面。
4. 在“我的凭证”页面，选择“访问密钥 > 新增访问密钥”。
5. 填写该密钥的描述说明，单击“确定”。根据提示单击“立即下载”，下载密钥。
6. 密钥文件会直接保存到浏览器默认的下载文件夹中。打开名称为“credentials.csv”的文件，即可查看访问密钥（Access Key Id和Secret Access Key）。

4.4 提示“上传的 AK/SK 不可用”，如何解决?

问题分析

AK与SK是用户访问OBS时需要使用的密钥对，AK与SK是一一对应，且一个AK唯一对应一个用户。如提示不可用，可能是由于账号欠费或AK与SK不正确等原因。

解决方案

1. 使用当前账号登录OBS管理控制台，确认当前账号是否能访问OBS。
   - 是，请执行步骤2。
   - 否，请执行步骤3。
2. 如能访问OBS，单击右上方登录的用户，在下拉列表中选择“我的凭证”，确认当前AK/SK是否是当前账号创建的AK/SK，可参考**如何获取访问密钥AK/SK?** 。
   - 是，请联系提交工单处理。
   - 否，请更换为当前账号的AK/SK。
3. 请确认当前账号是否欠费。
   - 是，请给账号充值。。
   - 否，且提示资源已过保留期，需要提交工单给OBS开通资源。

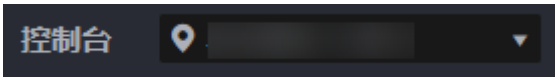
4.5 如何查看 Octopus 与 OBS 桶是否在同一区域?

在使用Octopus各功能时，如创建训练作业、创建数据集等，涉及到需要指定OBS目录时，都需要保证此OBS桶与Octopus在同一区域。

查看 OBS 桶与 Octopus 是否在同一区域

1. 查看创建的OBS桶所在区域。
   a. 登录OBS管理控制台。
   b. 进入“对象存储”界面，可在桶列表的“桶名称”列查找，或在右上方的搜索框中输入已经创建的桶名称搜索，找到您创建的OBS桶。
      在“区域”列可查看创建的OBS桶的所在区域。
2. 查看Octopus所在区域。
   登录Octopus控制台，在控制台左上角可查看Octopus所在区域。

   图 4-1 Octopus 控制台

3. 比对您创建的OBS桶所在区域与Octopus所在区域是否一致。务必保证OBS桶与Octopus所在区域一致。

4.6 如何查看用户拥有的权限?

1. 使用账号登录华为云。
2. 在页面左上方单击“控制台”，进入华为云管理控制台。
3. 在控制台右上角的账户名下方，单击“统一身份认证”，进入“统一身份认证服务”页面。
4. 查找用户。
5. 单击用户名进入用户详情，查看用户所属用户组。
6. 单击用户组，查看用户组授权记录。
7. 单击权限（以Octopus FullAccess为例），查看具体的权限策略内容。了解策略内容请参考**策略语法**。

图 4-2 查看权限策略内容

基本信息

名称	Octopus FullAccess	作用范围	项目级服务
类型	系统策略	描述	Octopus服务所有权限

策略内容

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "octopus:*:*"
            ],
            "Effect": "Allow"
        }
    ]
}
```