Apollo自动驾驶系统的接入

51VR自动驾驶仿真平台开发的目主要目的是为了测试和训练自动驾驶系统。我们选取了百度Apollo平台接入系统，演示仿真平台的运作方式。

阿波罗是百度发布的名为“[Apollo](https://baike.baidu.com/item/Apollo/20625961)（阿波罗）”的向汽车行业及自动驾驶领域的合作伙伴提供的软件平台。发布时间是2017年4月19日，旨在向汽车行业及自动驾驶领域的合作伙伴提供一个开放、完整、安全的软件平台，帮助他们结合车辆和硬件系统，快速搭建一套属于自己的完整的自动驾驶系统。

百度开放了环境感知、路径规划、车辆控制、车载操作系统等功能的代码或能力，并且开发一直很活跃，这是我们选择Apollo的主要原因。

## Apollo的软件架构

Apollo 5自动驾驶系统当前包含以下核心模块：

* 感知模块 （Perception）用于检测自动驾驶车辆周围的环境。包含两个重要的子模块：障碍物检测和红绿灯检测；
* 预测模块 （Prediction）用于预测检测到的障碍物的未来运动轨迹；
* 路由模块 （Routing）用于告知自动驾驶车辆到达目的地所经过的道路/车道；
* 规划模块 （Planning）给出一条带有时间/位置信息的轨迹以供自动驾驶车辆行驶；
* 控制模块 （Control）根据规划模块给出的轨迹得到油门/刹车/转向的控制指令；
* 总线模块 （CanBus）传递控制指令给车辆硬件的接口；同时将底盘信息传递给软件系统；
* 高精地图 （HDMap）提供道路相关信息以供各个软件模块查询；
* 定位模块 （Location）融合各种传感器，如GPS，IMU，LiDAR以估算自动驾驶车辆的当前位置。

这些模块的关系如下图所示：

**感知模块**

障碍物检测

红绿灯检测

**路由模块**

**控制模块**

**规划模块**

**定位模块**

**高精地图**

**预测模块**

**总线模块**

## 仿真引擎与Apollo的接口

接入方式一 （从决策系统接入）

**障碍物、红绿灯数据**

51VR仿

真

引

擎

**路由模块**

**控制模块**

**规划模块**

**定位模块**

**高精地图**

**预测模块**

**GPS、IMU数据**

**油门、刹车和方向控制数据**

接入方式二 （从感知系统接入）

**感知模块**

障碍物检测

红绿灯检测

**路由模块**

**控制模块**

**规划模块**

**定位模块**

**高精地图**

**预测模块**

51VR仿

真

引

擎

**点云、照相机数据**

**GPS、IMU数据**

**油门、刹车和方向控制数据**

## Apollo系统高精地图的扩展实现

Apollo系统使用的是百度自定义的高精地图数据，这些数据都是百度自己采集的。51VR仿真引擎支持的是更加标准的OpenDrive格式的数据。为此，需要将标准OpenDrive地图转换为Apollo能识别的地图数据。我们实现了标准OpenDrive文件的读取和转换模块，以提供高精地图服务。当前，我们已经实现了道路、车道、路网和停止线、红绿灯等信息的转换。

## Apollo 对接Cyber消息接口

Apollo Cyber RT是百度自研的一个专门为自动驾驶场景开发的开源框架。51VR仿真引擎通过Cyber消息与Apollo自动驾驶对接。对接消息主要包括两种，一种是51VR仿真引擎向Apollo发送的传感器信息或者感知结果信息，另一种是51VR仿真引擎从Apollo获取的控制信息和其他辅助信息。

51VR仿真引擎 发往Apollo的传感器消息和感知结果信息主要包括：

* GPS （Cyber Topic: /apollo/sensor/gnss/odometry， 60HZ）  
  GPS消息中包括主车当前位置信息和朝向信息
* IMU（Cyber Topic: /apollo/sensor/gnss/corrected\_imu， 60HZ）  
  IMU消息中包括主车角速度和线性加速度信息
* 底盘（Cyber Topic: /apollo/canbus/chassis， 60HZ）  
  底盘消息中包括了车辆当前的一些底盘状态，包括驾驶模式、车速和发动机转速等
* 障碍物（Cyber Topic: /apollo/perception/obstacles， 30HZ）  
  51VR仿真系统可以绕过感知，直接向Apollo发送障碍物和红绿灯信息，以便快速测试自动驾驶的决策和控制模块。障碍物消息包括障碍物类型（车、行人等）、位置、朝向、速度和包围盒等信息。
* 红绿灯（Cyber Topic: /apollo/perception/traffic\_light， 10HZ）  
  红绿灯消息包括当前红绿灯的颜色。
* 毫米波雷达（Cyber Topic: /apollo/sensor/conti\_radar， 10HZ）  
  毫米波雷达消息包括探测到的障碍物的纵向位移、横侧向位移、纵向速度、横侧向速度以及雷达波强度等。
* 激光雷达点云（Cyber Topic: /apollo/sensor/velodyne64/compensator/PointCloud2， 10HZ）  
  激光雷达消息包括点云中每个点的位置和强度信息。激光雷达和毫米波雷达的消息会在Apollo中融合共同完成对障碍物的识别。
* 广角摄像机（Cyber Topic: /apollo/sensor/camera/front\_6mm/image， 5HZ）  
  广角摄像机焦距相对较小，拥有较大视场角, 便于识别近处的红绿灯。摄像机输出为BGR格式。
* 长焦摄像机（Cyber Topic: /apollo/sensor/camera/front\_12mm/image， 5HZ）  
  长焦摄像机焦距相对较大，拥有较小视场角, 便于识别远处的红绿灯。长焦摄像机和广角摄像机共同完成对红绿灯和障碍物的识别。
* 路由 (Cyber Topic: /apollo/routing\_request)

主车起点，终点，途径点，案例开始的时候发送一次，给Apollo以规划主车的全局路径

51VR仿真引擎从Apollo获取的消息包括：

* 控制信息（Cyber Topic: /apollo/control， 60HZ）  
  51VR仿真系统的基本控制信息包括油门、刹车和转向。
* 局部规划轨迹（Cyber Topic: /apollo/planning， 1HZ）  
  Apollo规划的主车轨迹线信息，以可视化的方式显示在51VR仿真系统中。

## Apollo对接Cyber消息具体描述

* GPS，从仿真端发给Apollo

/apollo/sensor/gnss/odometry

GPS信息。从仿真端发送给Apollo。

定义在： localization/proto/gps.proto

类型： apollo::localization::Gps

使用到的字段：

message Gps {

optional apollo.common.Header header = 1;

// Localization message: from GPS or localization

optional apollo.localization.Pose localization = 2;

}

Pose协议中使用到的字段，只使用了部分：

message Pose {

// Position of the vehicle reference point (VRP) in the map reference frame.

// The VRP is the center of rear axle.

optional apollo.common.PointENU position = 1;

// A quaternion that represents the rotation from the IMU coordinate

// (Right/Forward/Up) to the

// world coordinate (East/North/Up).

optional apollo.common.Quaternion orientation = 2;

}

* IMU，从仿真端发给Apollo

/apollo/sensor/gnss/corrected\_imu

定义在： location/proto/imu.proto

类型：apollo::localization::Imu

使用到的字段：

message CorrectedImu {

optional apollo.common.Header header = 1;

// Inertial Measurement Unit(IMU)

optional apollo.localization.Pose imu = 3;

}

Pose协议中使用到的字段，只使用了部分：

message Pose {

// Linear acceleration of the VRP in the map reference frame.

// East/north/up in meters per second.

optional apollo.common.Point3D linear\_acceleration = 4;

// Angular velocity of the vehicle in the map reference frame.

// Around east/north/up axes in radians per second.

optional apollo.common.Point3D angular\_velocity = 5;

}

* 底盘，从仿真端发给Apollo

/apollo/canbus/chassis

定义在：canbus/proto/chassis.proto

类型： apollo::canbus::Chassis

使用到的字段，只使用了部分：

Message Chassis {

enum DrivingMode {

COMPLETE\_MANUAL = 0; // human drive

COMPLETE\_AUTO\_DRIVE = 1;

AUTO\_STEER\_ONLY = 2; // only steer

AUTO\_SPEED\_ONLY = 3; // include throttle and brake

// security mode when manual intervention happens, only response status

EMERGENCY\_MODE = 4;

}

// Engine speed in RPM.

optional float engine\_rpm = 4 [default = nan];

// Vehicle Speed in meters per second.

optional float speed\_mps = 5 [default = nan];

// Real throttle location in [%], ranging from 0 to 100.

optional float throttle\_percentage = 8 [default = nan];

// Real brake location in [%], ranging from 0 to 100.

optional float brake\_percentage = 9 [default = nan];

// Real steering location in [%], ranging from -100 to 100.

// steering\_angle / max\_steering\_angle

// Clockwise: negative

// CountClockwise: positive

optional float steering\_percentage = 11 [default = nan];

optional DrivingMode driving\_mode = 21 [default = COMPLETE\_MANUAL];

}

* 障碍物，包括车辆，自行车和行人，当不使用Apollo的感知模块时发送给Apollo。

/apollo/perception/obstacles

定义在：perception/proto/perception\_obstacle.proto

类型：apollo::perception::PerceptionObstacles

使用到的字段，只使用了部分：

message PerceptionObstacles {

repeated PerceptionObstacle perception\_obstacle = 1; // An array of obstacles

optional apollo.common.Header header = 2; // Header

}

message PerceptionObstacle {

optional int32 id = 1; // obstacle ID.

// obstacle position in the world coordinate system.

optional Point position = 2;

optional double theta = 3; // heading in the world coordinate system.

optional Point velocity = 4; // obstacle velocity.

// Size of obstacle bounding box.

optional double length = 5; // obstacle length.

optional double width = 6; // obstacle width.

optional double height = 7; // obstacle height.

repeated common.Point3D polygon\_point = 8; // obstacle corner points.

enum Type {

UNKNOWN = 0;

UNKNOWN\_MOVABLE = 1;

UNKNOWN\_UNMOVABLE = 2;

PEDESTRIAN = 3; // Pedestrian, usually determined by moving behaviour.

BICYCLE = 4; // bike, motor bike

VEHICLE = 5; // Passenger car or truck.

};

optional Type type = 10; // obstacle type

optional double timestamp = 11; // GPS time in seconds.

optional double confidence = 13 [deprecated = true];

enum ConfidenceType {

CONFIDENCE\_UNKNOWN = 0;

CONFIDENCE\_CNN = 1;

CONFIDENCE\_RADAR = 2;

};

optional ConfidenceType confidence\_type = 14 [deprecated = true];

}

* 红绿灯，当不使用Apollo的感知模块时发给Apollo

/apollo/perception/traffic\_light

红绿灯信息。当不使用Apollo的感知模块时发送给Apollo。

定义在： perception/proto/traffic\_light\_detection.proto

类型：apollo::perception::TrafficLightDetection

使用到的字段，只使用了部分：

message TrafficLightDetection {

optional apollo.common.Header header = 2;

repeated TrafficLight traffic\_light = 1;

}

message TrafficLight {

enum Color {

UNKNOWN = 0;

RED = 1;

YELLOW = 2;

GREEN = 3;

BLACK = 4;

};

optional Color color = 1;

// Traffic light string-ID in the map data.

optional string id = 2;

// How confidence about the detected results, between 0 and 1.

optional double confidence = 3 [default = 1.0];

}

* 毫米波雷达，当使用Apollo的感知模块时从仿真端发送给Apollo

/apollo/sensor/conti\_radar

定义在：drivers/proto/conti\_radar.proto

类型：apollo::drivers::ContiRadar

使用到的字段，只使用了部分：

message ContiRadar {

optional apollo.common.Header header = 1;

repeated ContiRadarObs contiobs = 2; //conti radar obstacle array

}

message ContiRadarObs {

optional apollo.common.Header header = 1;

optional bool clusterortrack = 2; // 0 = track, 1 = cluster

optional int32 obstacle\_id = 3; // obstacle Id

// longitude distance to the radar; (+) = forward; unit = m

required double longitude\_dist = 4;

// lateral distance to the radar; (+) = left; unit = m

required double lateral\_dist = 5;

// longitude velocity to the radar; (+) = forward; unit = m/s

required double longitude\_vel = 6;

// lateral velocity to the radar; (+) = left; unit = m/s

required double lateral\_vel = 7;

// obstacle Radar Cross-Section; unit = dBsm

optional double rcs = 8;

// 0 = moving, 1 = stationary, 2 = oncoming, 3 = stationary candidate

// 4 = unknown, 5 = crossing stationary, 6 = crossing moving, 7 = stopped

optional int32 dynprop = 9;

}

* 激光雷达，点云信息，当使用Apollo感知模块时从仿真端发送给Apollo

/apollo/sensor/velodyne64/compensator/PointCloud2

定义在：driver/proto/pointcloud.proto

类型：apollo::drivers::PointCloud

message PointXYZIT {

optional float x = 1 [default = nan];

optional float y = 2 [default = nan];

optional float z = 3 [default = nan];

optional uint32 intensity = 4 [default = 0];

optional uint64 timestamp = 5 [default = 0];

}

message PointCloud {

optional apollo.common.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;

}

* 摄像头，摄像头图像，当使用Apollo感知模块时从仿真端发送给Apollo

/apollo/sensor/camera/front\_6mm/image

/apollo/sensor/camera/front\_12mm/image

定义在 drivers/proto/sensor\_image.proto

类型：apollo::drivers::Image

message Image {

optional apollo.common.Header header = 1;

optional string frame\_id = 2;

optional double measurement\_time = 3;

optional uint32 height = 4; // image height, that is, number of rows

optional uint32 width = 5; // image width, that is, number of columns

optional string encoding = 6;

optional uint32 step = 7; // Full row length in bytes

optional bytes data = 8; // actual matrix data, size is (step \* rows)

}

* 控制信息，从Apollo发送给仿真端，控制主车的运动。

/apollo/control

定义在： control/proto/control\_cmd.proto

类型： apollo::control::ControlCommand

使用到的字段，只使用了部分：

message ControlCommand {

// target throttle in percentage [0, 100]

optional double throttle = 3;

// target brake in percentage [0, 100]

optional double brake = 4;

// target steering angle, in percentage of full scale [-100, 100]

optional double steering\_target = 7;

}

* 局部规划路径

/apollo/planning

有两个作用：

1. 从Apollo发送给仿真端，显示用。
2. 当不接入Apollo控制系统及仿真端动力学仿真系统时可以发送给仿真端，驱动主车进行完美控制

/apollo/planning

定义在： planning/proto/planning.proto

类型： apollo::planning::ADCTrajectory

使用到的字段，做显示使用时只使用部分：

message ADCTrajectory {

optional apollo.common.Header header = 1;

// path data + speed data

repeated apollo.common.TrajectoryPoint trajectory\_point = 12;

}

message PathPoint {

// coordinates

optional double x = 1;

optional double y = 2;

optional double z = 3;

// direction on the x-y plane

optional double theta = 4;

// curvature on the x-y planning

optional double kappa = 5;

// accumulated distance from beginning of the path

optional double s = 6;

// derivative of kappa w.r.t s.

optional double dkappa = 7;

// derivative of derivative of kappa w.r.t s.

optional double ddkappa = 8;

// The lane ID where the path point is on

optional string lane\_id = 9;

}

message TrajectoryPoint {

// path point

optional PathPoint path\_point = 1;

// linear velocity

optional double v = 2; // in [m/s]

// linear acceleration

optional double a = 3;

// relative time from beginning of the trajectory

optional double relative\_time = 4;

}

* 路由，从仿真端发送给Apollo，供Apollo规划全局路径

/apollo/routing\_request

定义在： routing/proto/routing.proto

类型： apollo::routing::RoutingRequest

使用到的字段，只使用部分：

message RoutingRequest {

optional apollo.common.Header header = 1;

repeated LaneWaypoint waypoint = 2;

//for 51VR L2 case demonstration

optional int32 switchcontrol\_waypoint\_id = 7 [default = 0];

repeated LaneWaypointL2Control waypoint\_control = 8;

}

message LaneWaypoint {

optional string id = 1;

optional apollo.common.PointENU pose = 3;

}