Zenoh-flow/Pylot 移植工作初步开发计划

1、 开发内容

通过对 pylot 框架进行分析，以**最小化**原则，利用 zenoh-flow 框架，完成**至少** 15 个 pylot operator 的开发任务。

模块名称	Operators	备注
初始化模块	CarlaOperator CarlaCameraDriverOperator CarlaLidarDriverOperator CarlaIMUDriverOperator CarlaGNSSDriverOperator	由于需要与 carla 进行实时交互，这些 operators 开发有一定困难
定位模块	LocalizationOperator	可先通过 pylot 生成输入、输出数据进行开发
感知模块	**DetectionOperator** **TrafficLightDetOperator** LaneDetectionOperator ObstacleTrackerOperator ObstacleLocationFinderOperator	可先通过 pylot 生成输入、输出数据进行开发
预测模块	LinearPredictorOperator	可先通过 pylot 生成输入、输出数据进行开发
规划模块	BehaviorPlanningOperator PlanningOperator	可先通过 pylot 生成输入、输出数据进行开发
控制模块	PIDControlOperator	可先通过 pylot 生成输入、输出数据进行开发

2、开发原则

（1）基于原始 pylot 框架进行开发，边开发、边测试；

（2）以 zenoh-flow 中的 source-operator-sink 形成一个 pylot operator 的开发、测试单元；

（3）利用 pickle 对 message 对象进行序列化及反序列化（相比 protobuf，具有开发复杂性低，传输性能高的优势）。

3、开发计划

1）测试数据生成阶段（3.7-3.27）

（1）Pylot 环境搭建；

（2）zenoh-flow 环境搭建；

（3）测试数据生成。

2）单元开发测试阶段（3.28-4.30）

完成 15 个 operators 的开发测试，以一个 operator 单元的开发、测试工作量为例：

（1）operator.py，对 pylot operator 代码进行修改，生成 zenoh-flow operator：

a. 删除基于 erdos 框架的相关代码；

b. 增加 zenoh-flow 框架代码；

c. flags 相关代码改为由 zenoh-flow configuration 进行配置；

d.message 对象相关代码修改；

（2）source.py：基于输入数据，构建 message 对象；

（3）sink.py：接受 operator 的处理结果，进行正确性验证；

（4）pipeline.yml，对 source、operator、sink 的属性及数据流图进行定义：

a.注意路径设置；

b.命名问题；

c.Configuration 参数配置。

3）集成开发测试阶段（5.1-5.15）

完成 15 个 operators 的整体集成开发、测试。

目前，已完成 DetectionOperator、TrafficLightDetOperator 的开发及测试，后续可参照这两个样例进行开发。