技术文档 | 如何添加新的控制算法

无人驾驶车辆的自动驾驶行为由纵向云顶和横向运动共同组成。由于车辆纵向、横向运动之间存在很强的耦合关系，任何方向上的运动状态变化都会影响另一方向上的运动控制效果，进而影响车辆的自动驾驶安全性。

因此，车辆运动控制相应包括纵向运动控制和横向运动控制两个方面。

车辆纵向运动控制是指在行车方向上对车辆的行驶速度进行自动控制，是被控车辆与前后车辆保持理想的间距，并防止与其他障碍物发生碰撞。

目前应用于车辆上的纵向运动控制系统包括防抱死制动系统、驱动防滑系统、自适应巡航系统、走——停巡航系统和主动避障系统等电子装置。

这些电子装置通过控制发动机节气门开度、制动液压及自动变速器，使车辆实现巡航、跟踪和避障等控制目标。

车辆横向运动控制是指在不同的车速、载荷、路况及风阻等条件下，协助驾驶员或直接控制车辆自动跟踪真实或虚拟的理想行车路线，并使车辆满足一定的舒适性和稳定性要求。

横向运动控制主要通过控制车辆的前轮转向角和横摆力矩共同实现该目标。按照车辆横向运动过程目标轨迹的不同，可以将横向运动控制分为车道保持控制和车道变换控制两种类型。

Apollo中的控制算法由一个或多个控制器组成，可以轻松更改或替换为不同的算法。 每个控制器将一个或多个控制命令输出到CANbus。

Apollo中的默认控制算法包含横向控制器（LatController）和纵向控制器（LonController）。 它们分别负责横向和纵向的车辆控制。

新的控制算法不必遵循默认模式，例如，一个横向控制器+一个纵向控制器。 它可以是单个控制器，也可以是任意数量控制器的组合。

添加新的控制算法的步骤：

> - 创建一个控制器；
> - 在文件control_config 中添加新控制器的配置信息；
> - 注册新控制器。

为了更好的理解，阿波君将在下面对每个步骤进行详细的阐述：

所有控制器都必须继承基类Controller，它定义了一组接口。

以下是控制器实现的示例：

```
namespace apollo {
namespace control {

class NewController : public Controller {
 public:
  NewController();
  virtual ~NewController();
  Status Init(const ControlConf* control_conf) override;
  Status ComputeControlCommand(
      const localization::LocalizationEstimate* localization,
      const canbus::Chassis* chassis, const planning::ADCTrajectory* trajectory,
      ControlCommand* cmd) override;
  Status Reset() override;
  void Stop() override;
  std::string Name() const override;
};
}  // namespace control
```

```
18  }  // namespace apollo
```

按照下面的步骤添加新控制器的配置信息：

1.根据算法要求为新控制器配置和参数定义 Proto。

作为示例，可以参考以下位置的LatController的Proto定义：

```
modules/control/proto/ lat_controller_conf.proto
```

2.定义新的控制器 Proto 之后，例如

```
new_controller_conf.proto ,
```

输入以下内容:

```
1  syntax = "proto2";
2
3  package apollo.control;
4
5  message NewControllerConf {
6      double parameter1 = 1;
7      int32 parameter2 = 2;
8  }
```

3.参考如下内容更新 。

```
modules/control/proto/control_conf.proto
```

文件：

```
optional apollo.control.NewControllerConf new_controller_conf = 15;
```

4.参考以下内容更新ControllerType

`modules/control/proto/control_conf.proto`

中：

```
enum ControllerType {
    LAT_CONTROLLER = 0;
    LON_CONTROLLER = 1;
    NEW_CONTROLLER = 2;
  };
```

Protobuf定义完成后。

开发者可在

`modules/control/conf/control_conf.pb.txt`

中相应更新控制配置文件。

注意：上面的 "control/conf" 文件是 Apollo 的默认文件。您的项目可能使用不同的控制配件文件。

要激活Apollo系统中的新控制器。

请在如下文件中的“ControllerAgent”中，

注册新控制器：

modules/control/controller/controller_agent.cc

按照如下示例添加注册信息：

```
void ControllerAgent::RegisterControllers() {
  controller_factory_.Register(
      ControlConf::NEW_CONTROLLER,
      []() -> Controller * { return new NewController(); });
}
```

在完成以上步骤后，您的新控制器便可在Apollo系统中生效。

END