使用 Sim_Control 仿真器调试 PnC 代码

目录

1 实验引入 1

2 实验简介 2

3 实验目的 2

4 总体设计 2

5 实践流程 2

6 实验总结 12

7 参考资料 12

使用 Sim_Control 仿真器调试 PnC 代码

1 实验引入

自动驾驶汽车在实现落地应用前，需要经历大量的道路测试来验证算法的可行性和系统的稳定性，但道路测试存在成本高昂、极端交通场景复现等各种各样的问题。仿真系统通过模拟各种交通场景生有效的实现了车辆在仿真系统下日行百万公里的测试，为自动驾驶系统开发提供了大量的数据支撑。Sim_Control 是 Apollo 软件交互系统 Dreamview 提供的仿真组件。Sim_control 通过模拟 Chassis、Localization、PerceptionOBstacle 等信息输入，实现对 routing、planning 等算法模块的仿真调试，同时 Apollo 提供了 PnC Monitor、Cyber Monitor 等系统调试工具，可以实时的监控各模块运行数据，有效提升开发者对自动驾驶软件算法的学习与调试。

2 实验简介

Apollo Map 地图下设置了多个典型交通场景，如人行道避让、红绿灯停止线、静态障碍物、减速带、龟速车绕行、施工限速绕行、左转待转、路口避障、自主泊车等场景。本实验将以【静态障碍物绕行】场景为例，介绍通过修改软件参数，实现车辆对静态障碍物绕行横向距离与绕行限速的动态调整。以此希望开发者掌握如何使用 Sim_Control 组件进行 Apollo 软件代码的调试的方法。

3 实验目的

- 熟悉 Apollo 软件系统的启动方法，掌握 Dreamview 交互系统使用
- 熟悉 Apollo 软件系统工作流程，掌握系统调试工具的 Cyber_monitor 使用
- 掌握通过 Sim_Control 模块对 Apollo 软件代码的仿真调试方法

4 总体设计

1> 进入 Docker 环境，打开 DreamView 可视化交互系统

2> 进入 Sim_Control 仿真模式，选择要调试的代码模块

3> 启动场景生成组件 Sim_Obstacle，设置车辆行驶路径

4> 静态障碍物绕行场景调试方法演示

5> Add Default Routing 使用方法介绍

5 实践流程

5.1 打开 DreamView 交互系统

（1）在 Ubuntu 系统中打开命令行工具，进入/apollo 目录，执行启动 docker 环境指令。

```
cd ~/apollo
bash docker/scripts/dev_start.sh
```

启动成功后，会提示【OK】Enjoy! 操作示意如图 1 所示：

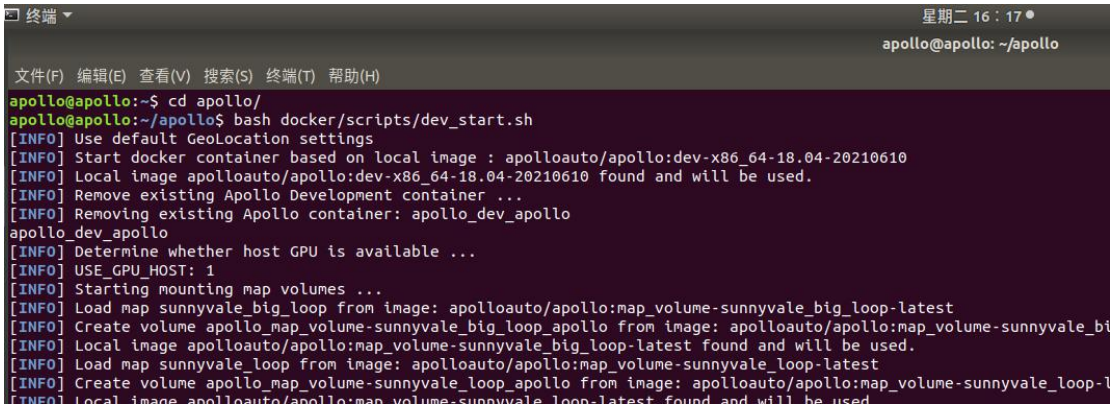

图 1 启动 docker 环境

（2）启动 docker 环境成功后，执行进入 docker 环境指令。

```
bash docker/scripts/dev_into.sh
```

进入成功后，用户名会变为 in-dev-docker，操作示意如图 2 所示：

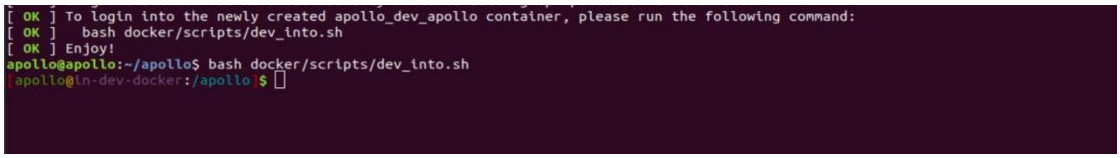

图 2 进入 docker 环境

（3）在 docker 环境中执行启动 dreamview 指令。

```
bash scripts/bootstrap.sh
```

如图 3 所示，启动成功后，终端串口会显示『Dreamview is running at http://localhost:8888』，若出现『Module dreamview is already running - skipping.』提示，则表示上一次运行 Dreamview 没有正常关闭，Dreamview 仍在运行中，此时可执行关闭 Dreamview 指令，待关闭完成之后，重新执行启动 Dreamview 指令即可。关闭 Dreamview 指令如下：

```
bash scripts/bootstrap.sh stop
```

图 3 启动 DreamView

（4）启动成功后，在终端右键然后选择 OpenLink 或在浏览器中输入(http://localhost:8888)回车打开。操作示意如图 4 所示，打开成功后，DreamView 运行界面如图 5 所示：

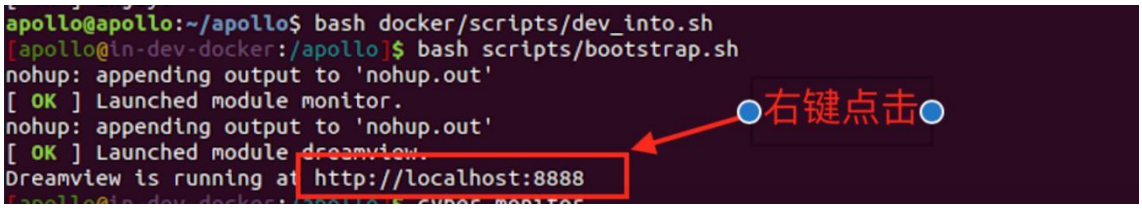

图 4 打开 DreamView

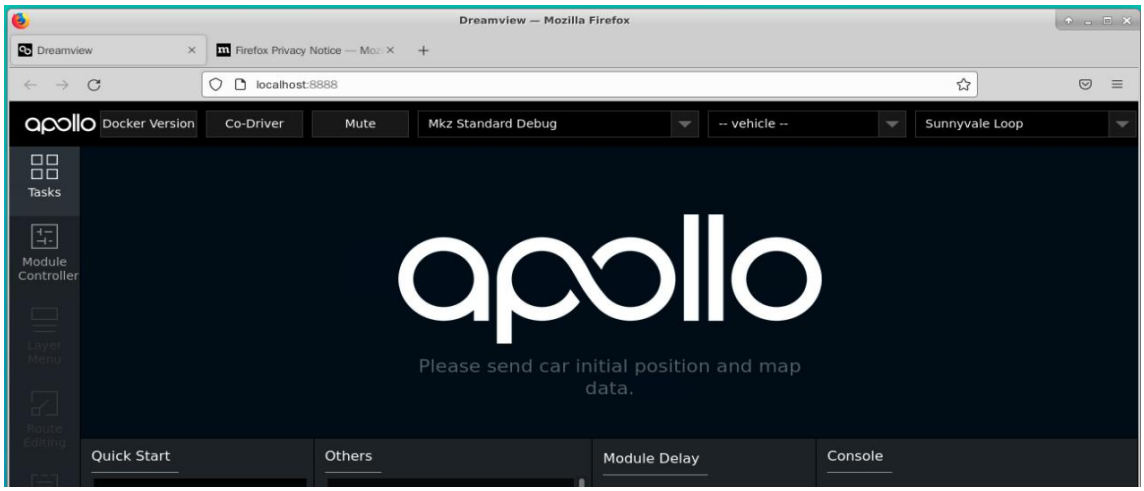

图 5 DreamView 运行成功示意图

注意：

Apollo 代码是运行在 Docker 环境之下，因此本节所有内容均在 docker 环境下执行，bash docker/scripts/dev_start.sh 表示以某个镜像为基准创建一个容器环境，启动之后，在该进程被杀死前都存在，不可多次执行。bash docker/scripts/dev_into.sh 表示进入容器环境，若要在新的终端中进入容器，则需要执行一次该命令。

5.2 进入 Sim_Control 仿真模式

（1）在上方菜单栏选择 Contest Debug > 车辆型号选择 MkzExample > 地图选择 Apollo Map > 点击 Tasks > 选择 Sim Control，即可进入仿真模拟控制，如图 6 所示：

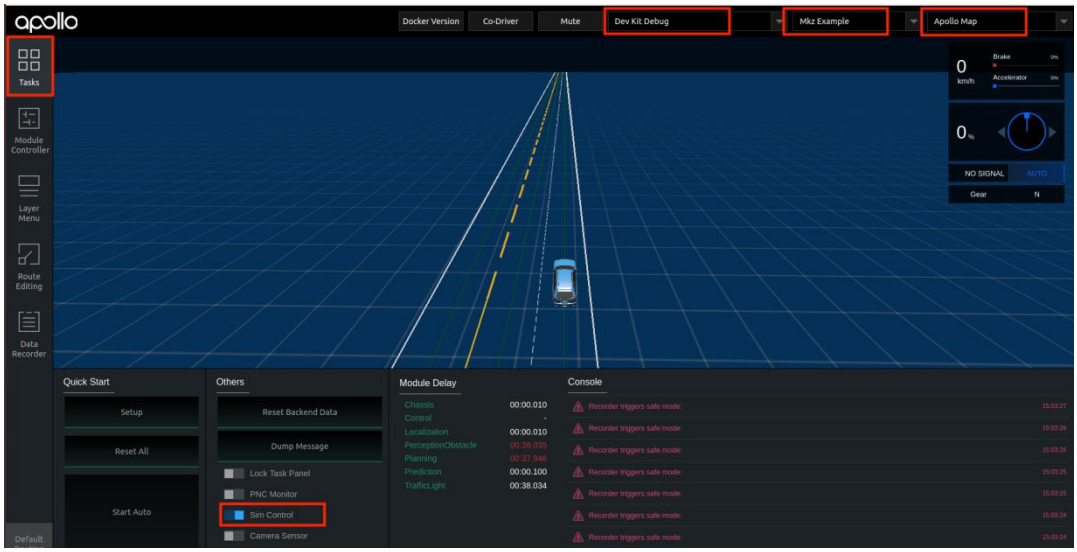

图 6 DreamView 运行成功示意图

（2）点击左侧 Module Control 栏，启动需要调试的模块进程，如图 7 所示，我们选择 Planing、Prediction、Routing 模块：

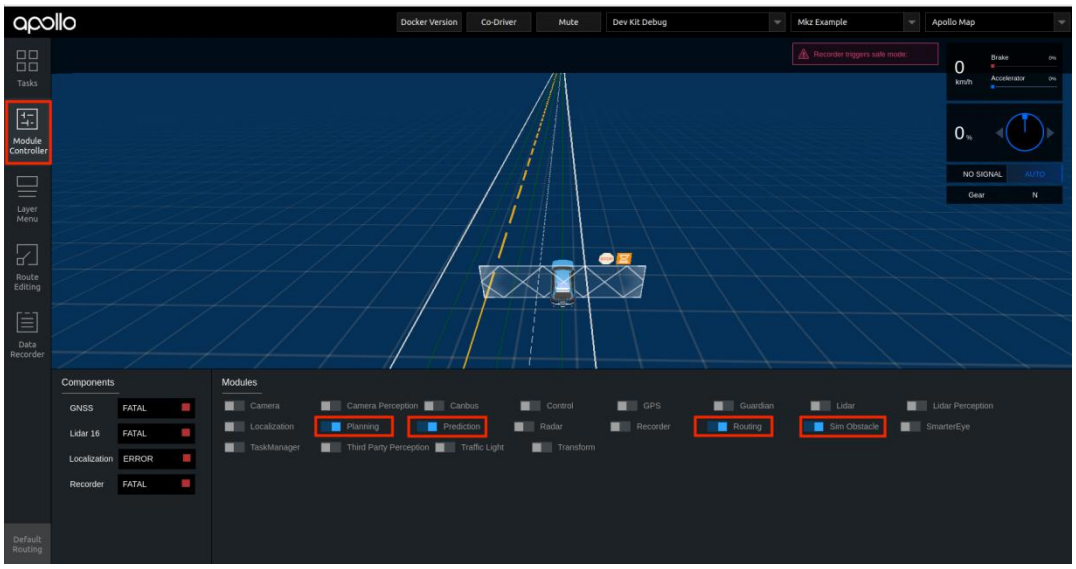

图 7　待调试模块启动示意图

（3）使用 Cyber _monitor 或 Module Delay 组件查看待调试的模块是否有数据输出。如图 8、图 9 所示：

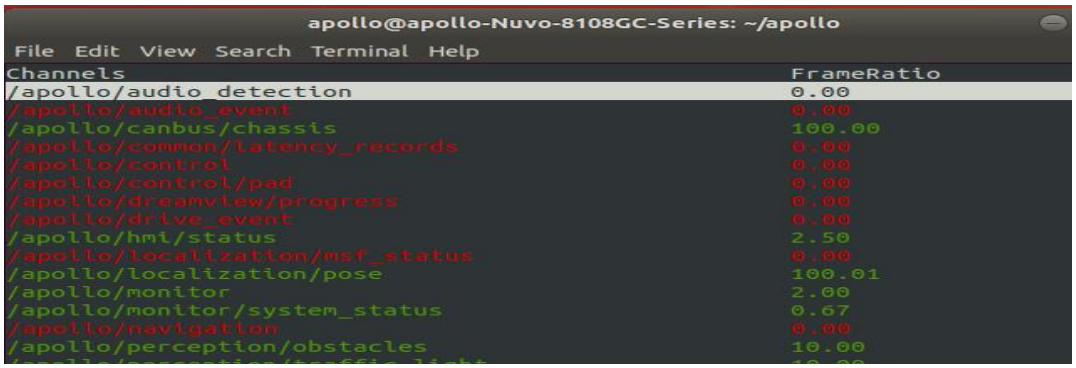

图 8　Cyber _monitor 数据输出示意图

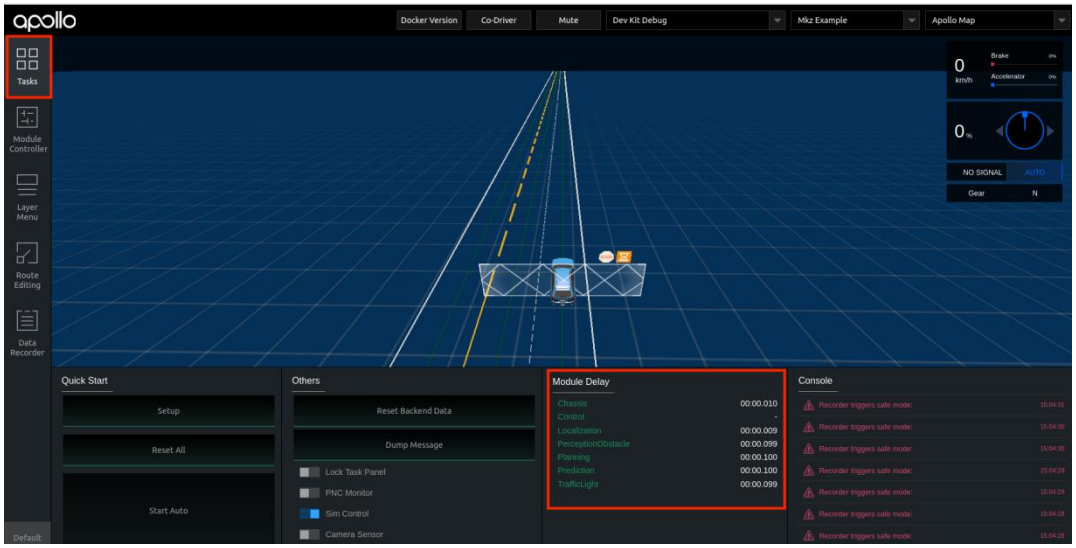

图 9　Module Delay 数据输出示意图

5.3 设置车辆仿真行驶路径

（1）点击左侧 Module Control 栏，启动场景生成组件 Sim_Obstacle，如图 10 所示：

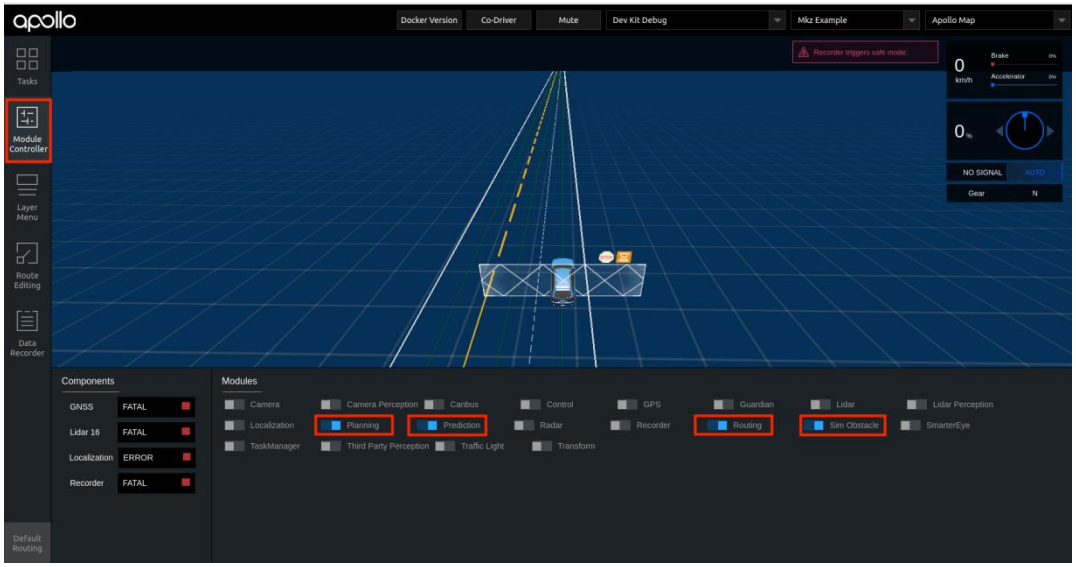

图 10　场景生成组件 Sim_Obstacle 启动示意图

（2）模块启动完成后，点击左侧 Routing Editing，拖动、点击鼠标可以在地图中设置车辆行驶路径（起点-终点或起点-途径点-终点）。如图 12 所示：

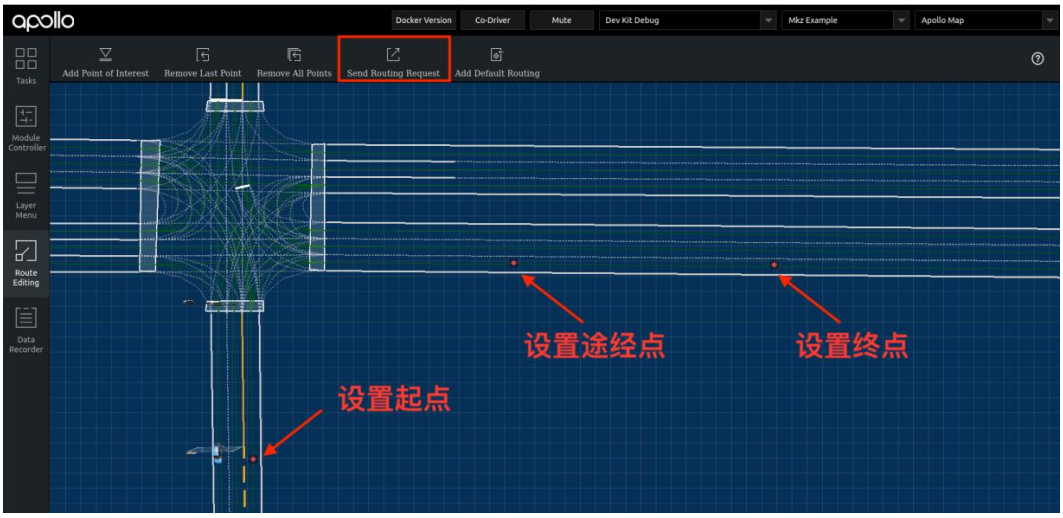

图 12　车辆仿真行驶轨迹设置示意图

（3）当点位设置完成后，点击 Send Routing Request。等待出现如图 13 所示，红线是 routing 模块在地图中搜索出的路径，浅蓝色的轨迹是 planning 模块实时规划的局部路径。

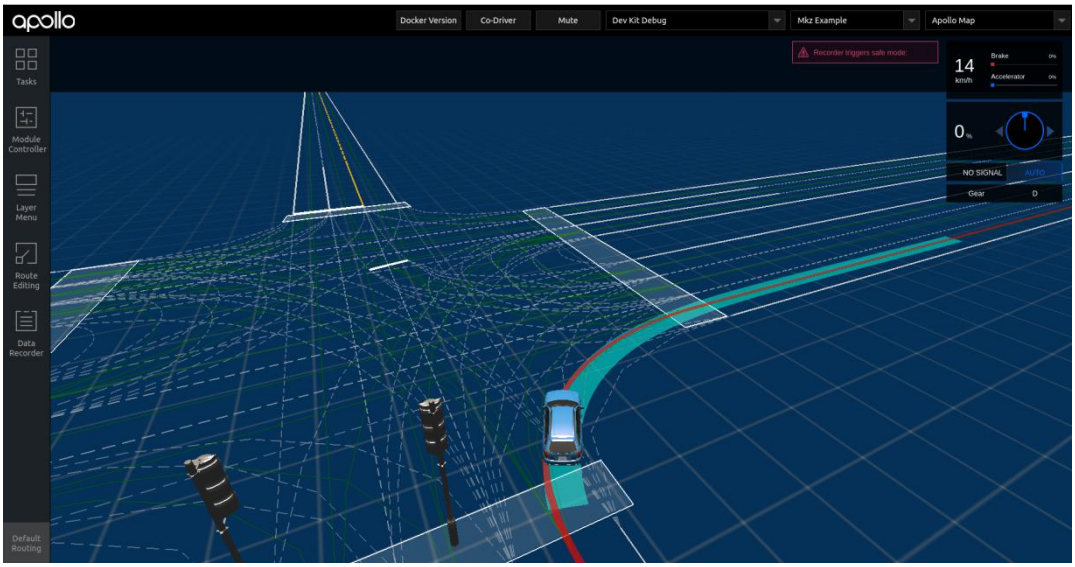

图 13　车辆仿真行驶示意图

（3）点击左任务栏 Tasks，打开 PNC monitor 组件，如图 14 所示，界面右侧就会输出 Planing 和 Control 相关的数据曲线，通过数据曲线的分析为开发者代码调试提供支撑。

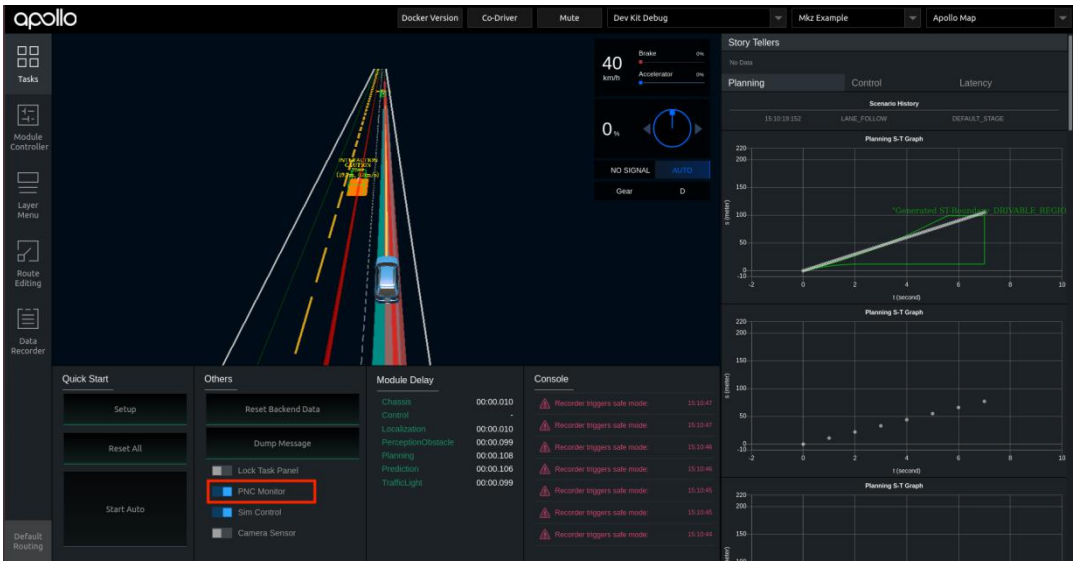

图 14　PNC_monitor 组件展示示意图

5.4 静态障碍物绕行场景调试方法演示

本小节将以【障碍物绕行】场景为例，介绍通过修改软件参数，实现车辆对静态障碍物绕行横向距离与绕行限速的动态调整。以此希望开发者掌握如何使用 Sim_Control 组件进行 Apollo 软件代码的调试的方法。

（1）打开 Sim_Control，进入 Route Editint 模式，如图 16 所示，静态障碍物位于地图上红色标记位置处如图 15 所示，在障碍物前后设置车辆行驶线路（起点-终点）。

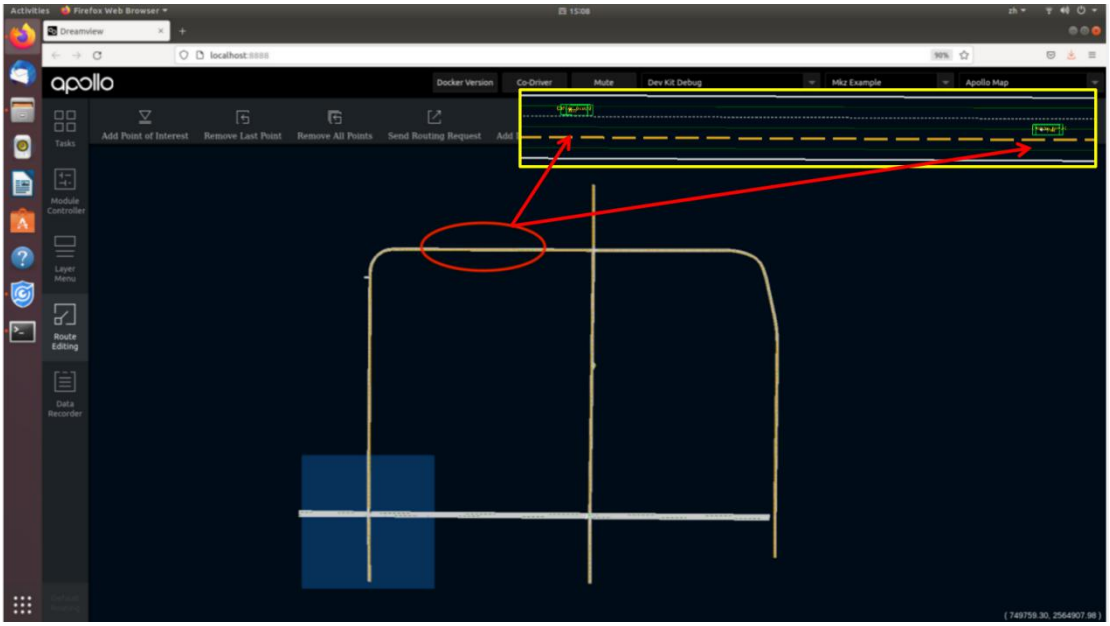

图 15　障碍物位置示意图

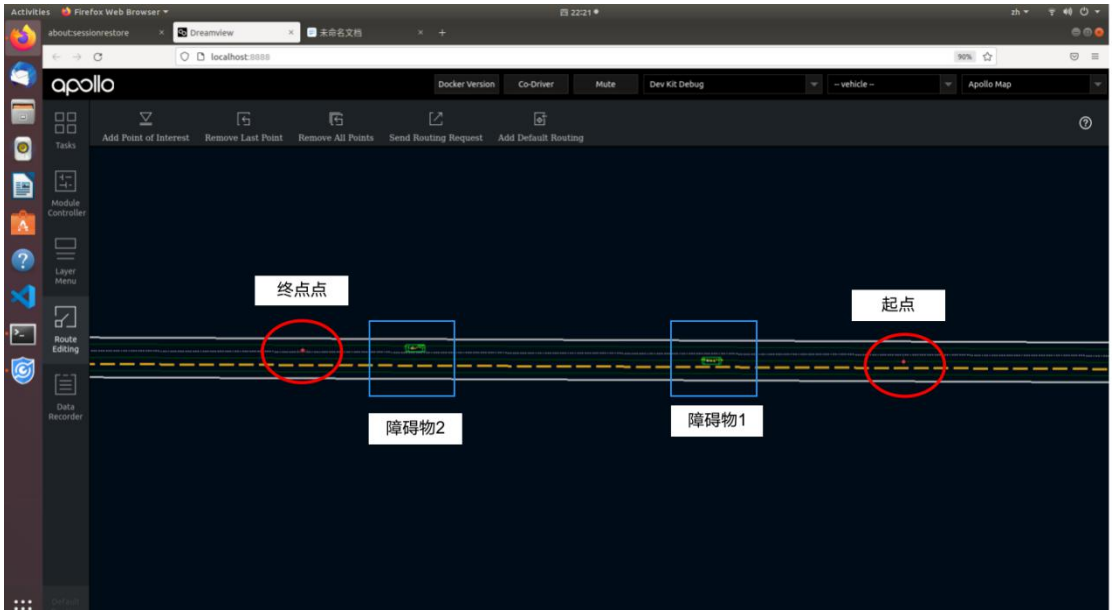

图 16　障碍物位置处车辆行驶路径设置示意图

（2）当点位设置完成后，点击 Send Routing Request。等待出现如图 13 所示，红线是 routing 模块在地图中搜索出的路径，浅蓝色的轨迹是 planning 模块实时规划的局部路径。

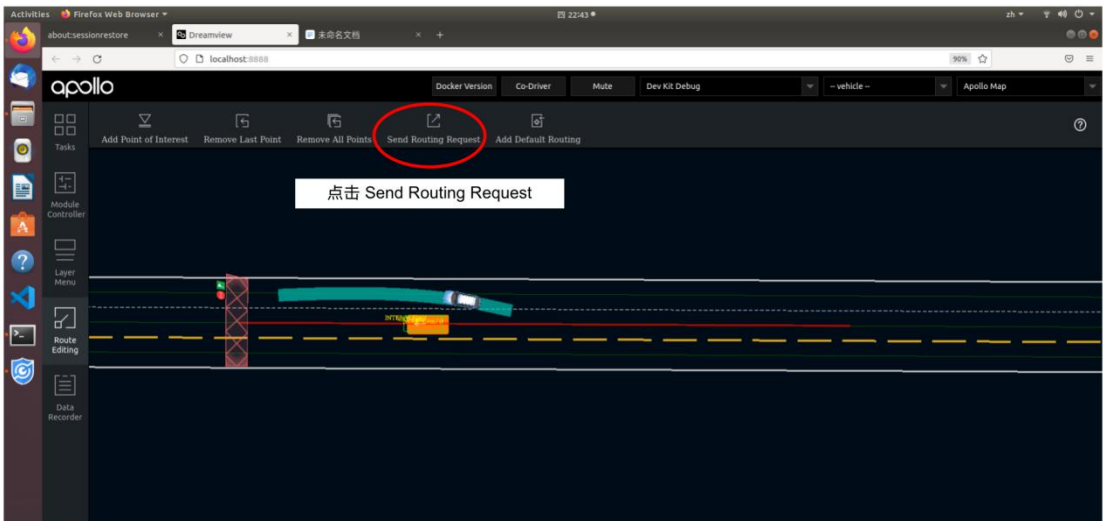

图 17　车辆仿真行驶示意图示意图

（3）在 Visual Studio Code 中打开 Apollo 代码，打开 apollo/modules/planning/conf 找到配置文件 planning.conf 文件。调整【obstacle_lat_buffer】参数可以实现车辆对障碍物绕行横向距离的调整，默认为 1 米。如图 18 所示，我们将其调整为 1.5 米。

（4）在 Visual Studio Code 中打开 Apollo 代码，打开 apollo/modules/planning/conf 找到配置文件 planning_config.pb.tx 文件。【static_obs_nudge_speed_ratio】为绕行限速的百分比，【speed_limit】是地图中约束的限速值（本地图 Apollo Map 全地图限速 60KM/H），调整【obstacle_lat_buffer】参数可以实现车辆对障碍物绕行限速的设置。如图 18 所示，我们将其调整为 0.1，即 10%。

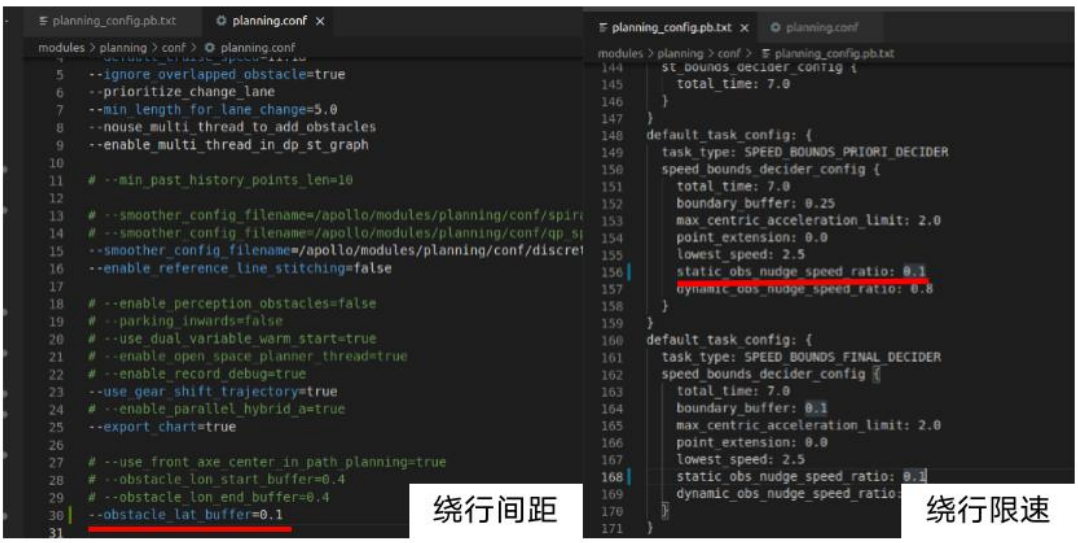

图 18　代码参数修改示意图

（5）修改好代码参数后，保存这两个文件，重启 planning 模块，重新发送 routing request 即可看到车辆运行运行前后发生了变化。如图 19 所示：

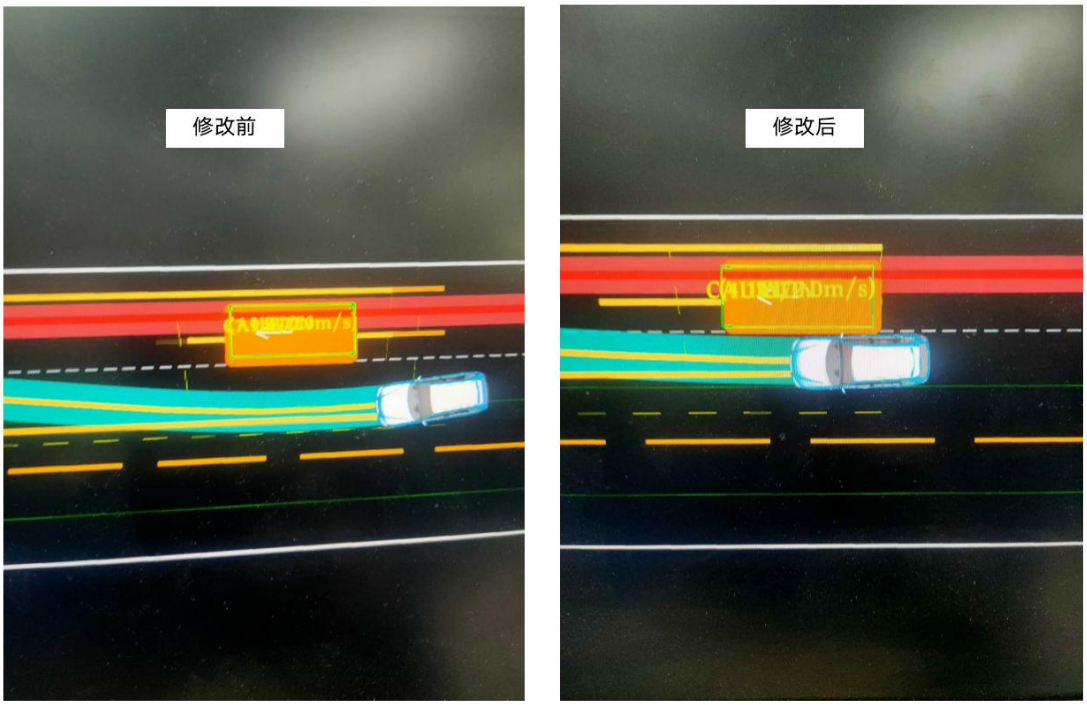

图 19　车辆运行前后对比示意图

5.5 Sim_Control 调试小技巧

通常我们需要对某个场景或者多个场景进行反复的代码调试，为了方便调试可以通过设置 default routing 来记录该场景，具体操作方法如下：

（1）点击任务栏 Route Editing > 点击 Add Default Routing > 在地图上设置起点与终点 > 再次点击 Add Default Routing，在弹出的对话框中为该场景命名，如图 20 所示，我们为静态障碍物绕行的场景命名为 nudge ， 命名完成后点击 save 保存。

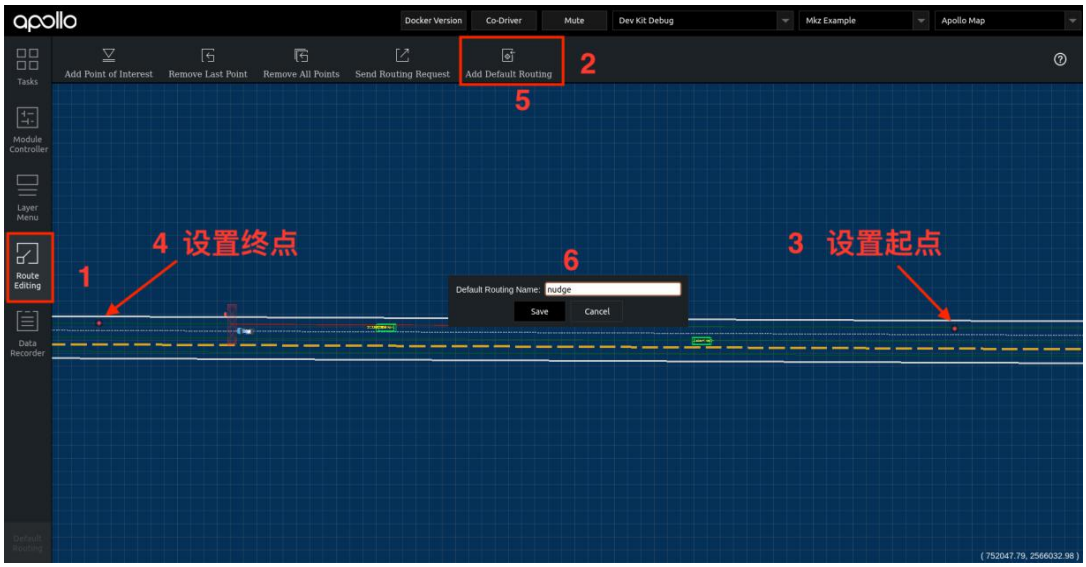

图 20　Default Routing 设置示意图

（2）nudge 场景设置完成后，就可以进行代码或参数的修改，若要再次启动 nudge 场景的仿真任务，可点击任务栏 Tasks ＞ 点击任务栏 Default Routing ＞选择要调试的场景 ＞ 点击 send 发送即可启动该场景的仿真，而无需在 Route Editing 中发送 Routing Request。如图 21 所示：

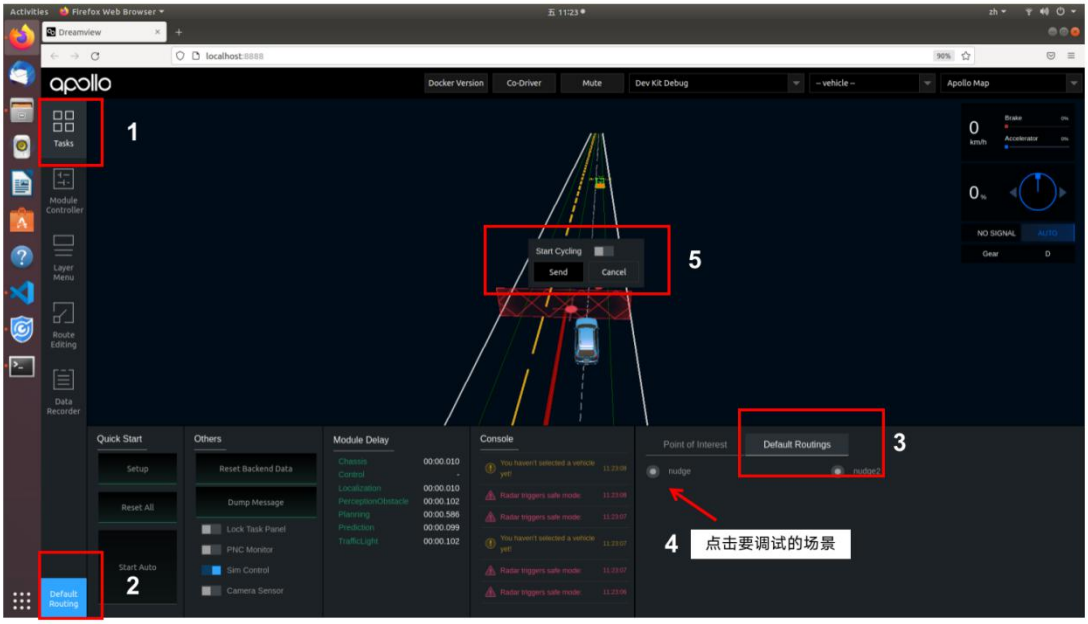

图 21　通过 Default Routing 启动 nudge 场景操作示意图

注意：

Apollo 代码在通过 Sim_Control 进行 PnC 代码调试时，请务必确保 routing、Planing Localization 模块正常启动，另外，若需要调试障碍物场景还需要开启 Sim_Obstacle 模块。

6 实验总结

至此我们通过该示例演示了如何在 Sim_Control 模式下，修改 Planing 代码中的参数实现对车辆运行状态的动态仿真，希望开发者熟悉仿真调试过程，掌握使用 Sim_Control 组件进行 Apollo 软件代码的调试的方法。

7 参考资料

1、https://gitee.com/ApolloAuto/apollo/tree/v6.0_edu/modules/planning

2、https://gitee.com/ApolloAuto/apollo/tree/v6.0_edu/modules/routing

3、智能驾驶入门课，https://apollo.auto/devcenter/coursetable_cn.html?target=1

4、Apollo 开放平台使用文档，

https://apollo.auto/document_cn.html?target=/Apollo-Homepage-Document/Apollo_Doc_CN_6_0/

5、规划决策算法，https://apollo.auto/developer/index_cn.html#/learning?id=3

6、开发者社区技术分析，https://apollo.auto/developer/index_cn.html#/article

7、开发者社区开发者说，https://apollo.auto/developer/index_cn.html#/article