

利用Matlab/Simulink实现ACC的Test bench

描述

目标识别

关于目标识别已经有大量的案例和教程被公布。使用Tensorflow API或Matlab/Simulink 计算机视觉工具箱，结合各种类型的传感器数据(如3D激光雷达云点和/或相机拍摄的照片)，可以从摄像机视频流中识别出目标物体。

当然，即便通过ML/DL技术可以识别出目标物体，仍然远远不能满足一个简单的ADAS功能的开发。自动驾驶汽车首先要借助传感器数据正确地理解现实环境，然后还要具备思考、规划和反应的能力。更具体来说，就是系统需要能够正确地控制车辆。

模型预测控制

基于简单的自行车模型，可以将运动学和动力学控制方程输入模型预测控制（MPC）算法。



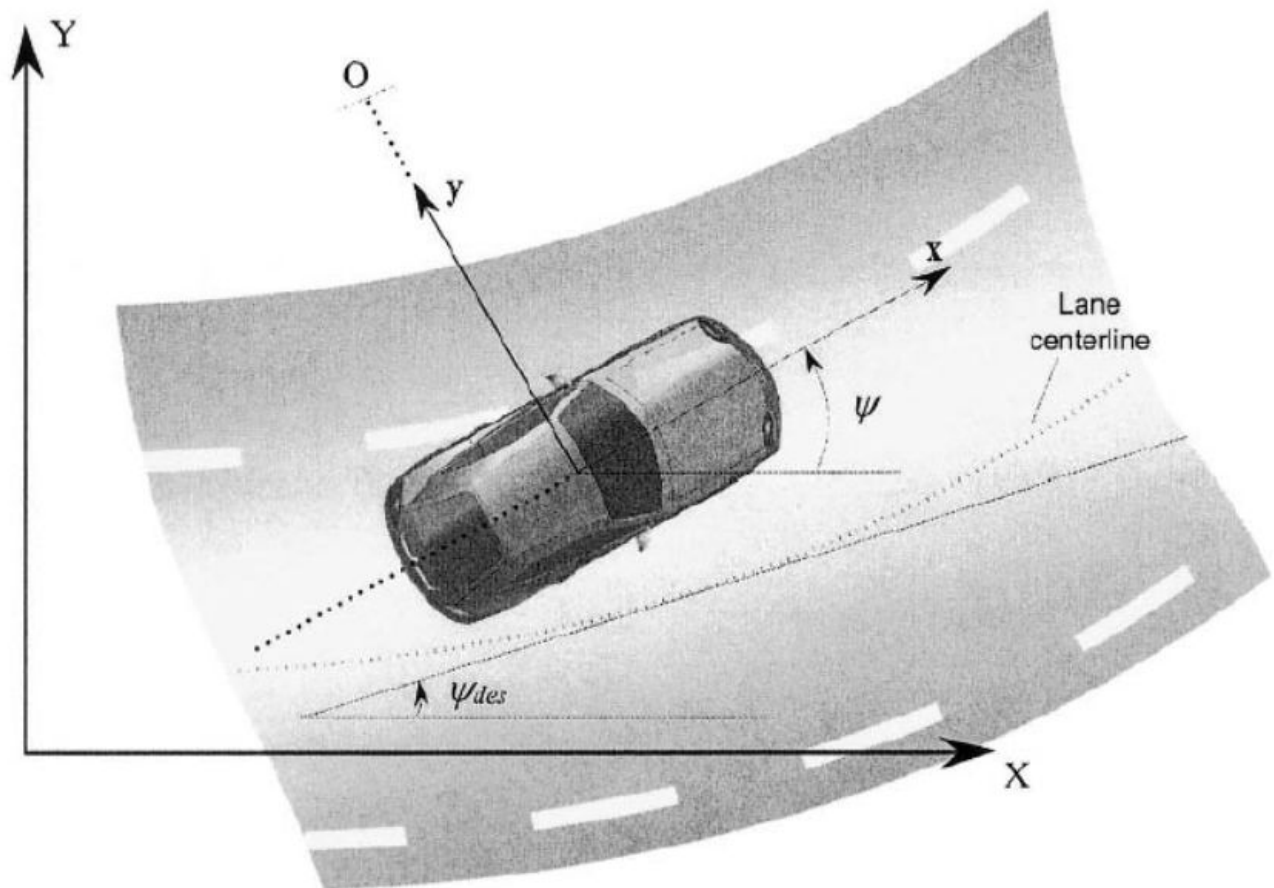


Figure 2-6. Lateral vehicle dynamics

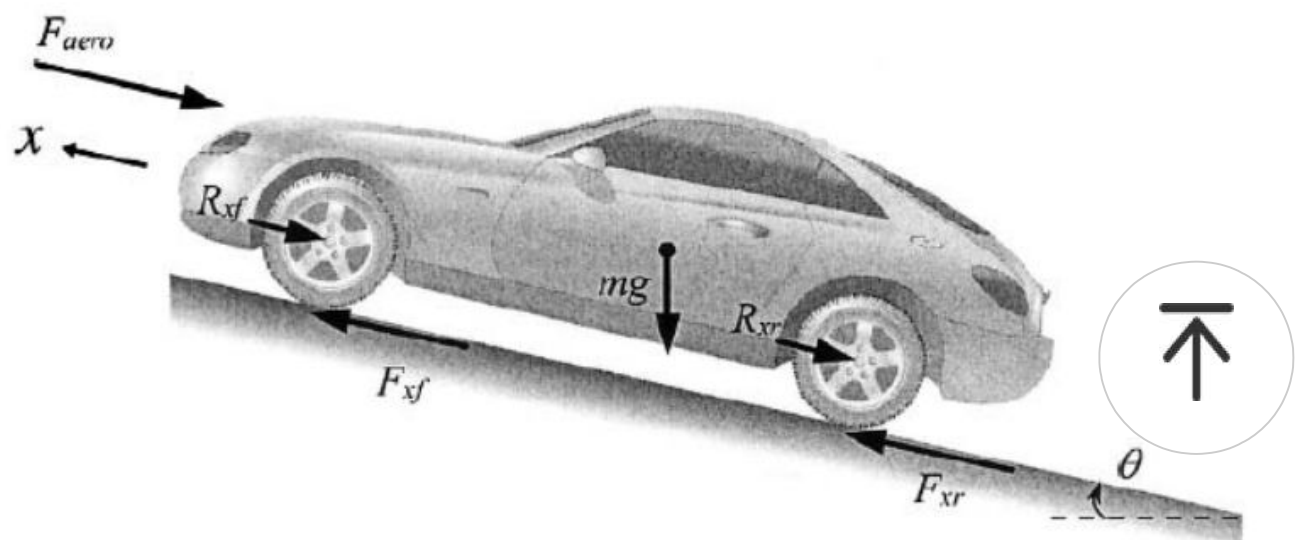



Figure 4-1. Longitudinal forces acting on a vehicle moving on an inclined road

Primary Vehicle Dynamic Equation

$$m\ddot{x} = F_{xf} + F_{xr} - F_{aero} - R_{xf} - R_{xr} - mg \sin(\theta)$$

在当今所有的过程控制中，MPC只是其中一种控制技术。PID当然是用的最多的控制方法，但由于MPC具有多输入/输出的优化能力和约束条件，使MPC也超过了10%的占有率。

MPC是一种基于模型的闭环优化控制策略，大量的预测控制权威性文献都无一例外地指出，预测控制最大的吸引力在于它具有显式处理约束的能力，这种能力来自其基于模型对系统未来动态行为的预测，通过把约束加到未来的输入、输出或状态变量上，可以把约束显式表示在一个在线求解的二次规划或非线性规划问题中。

模型预测控制具有控制效果好、鲁棒性强、点，可有效地克服过程的不确定性、非线性和时变性，并能方便的处理过程被控变量和操纵变量中的各种约束。

线性时不变(LTI)控制系统，连续状态空间模型可以这样描述。

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

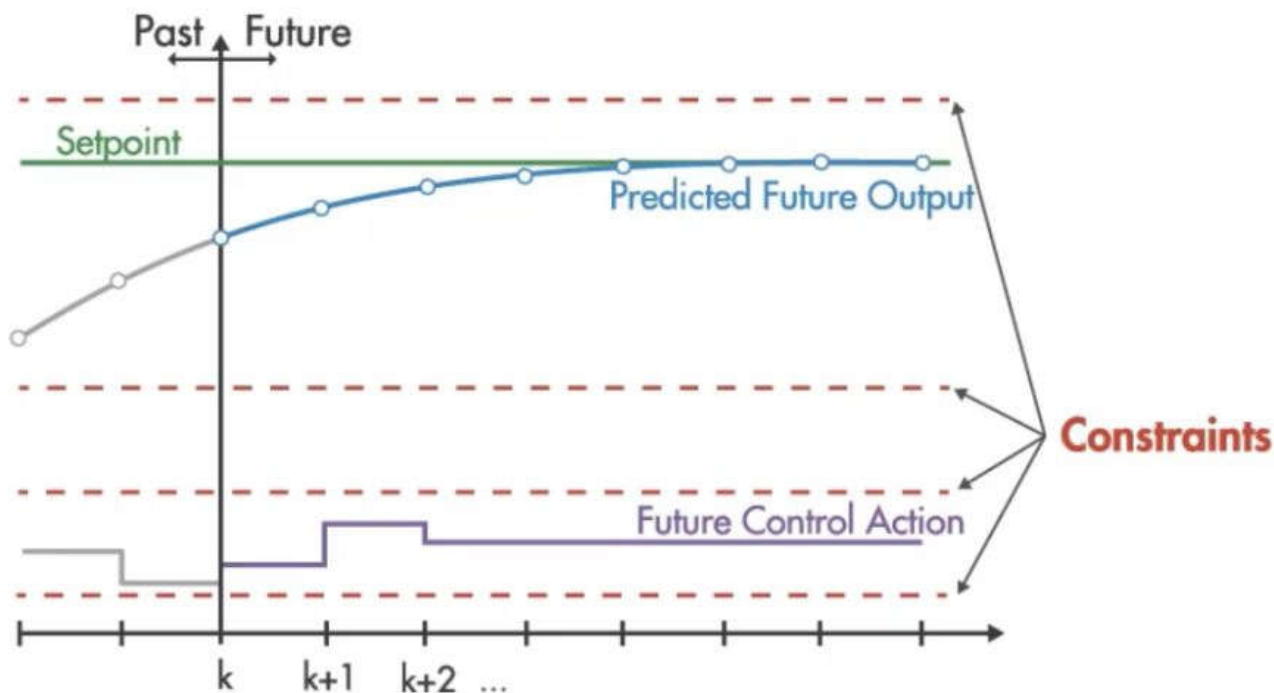
连续状态空间模型。A、B、C、D是常数状态空间矩阵。x是状态向量，y是输出，u是输入/控制变量

基于一个简单的自行车模型，状态函数可以写成：

$$\frac{d}{dt} \begin{bmatrix} V_y \\ \dot{\psi} \\ \dot{V}_x \end{bmatrix} = \begin{bmatrix} -\frac{2C_f+2C_r}{mV_x} & 0 & -V_x - \frac{2C_f\ell_f-2C_r\ell_r}{mV_x} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -\frac{2C_f\ell_f-2C_r\ell_r}{I_z V_x} & 0 & -\frac{2C_f\ell_f^2+2C_r\ell_r^2}{I_z V_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{\tau} & 0 \end{bmatrix} \begin{bmatrix} V_y \\ \dot{\psi} \\ V_x \\ \dot{V}_x \end{bmatrix} + \begin{bmatrix} \frac{2C_f}{m} \\ 0 \\ \frac{2C_f\ell_f}{I_z} \\ 0 \end{bmatrix} \delta + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\tau} \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ V_y\dot{\psi} \\ 0 \end{bmatrix}$$

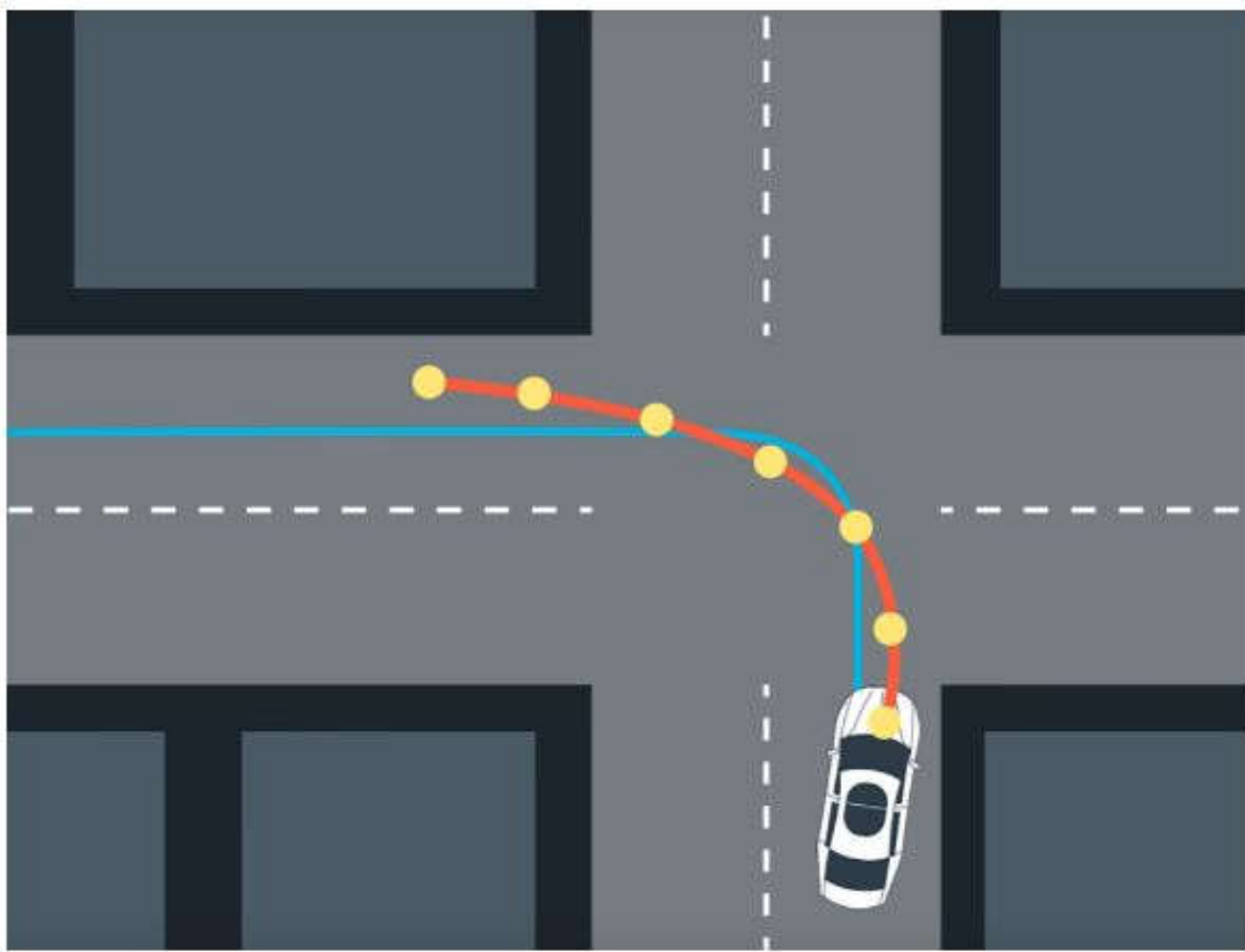
V_y, \dot{V}_y 用于横向控制， $\psi, \dot{\psi}$ 用于转向控制， V_x, \dot{V}_x 用于纵向控制。详细信息可从Matlab文档页面找到。

MPC的主要思路是预测工厂输出的产量，优化器会找到控制输入的最优序列，使工厂的产量尽可能接近设定值。



如下图中展示了一个典型的场景，图中是一辆行驶在十字路口的汽车。MPC将考虑到道路图的曲率，并将道路图和工厂路径之间的误差最小化。MPC的主要优点之一就是具有硬约束和软约束能力的多输入多输出，非常适合ADAS函数中的控制策略。



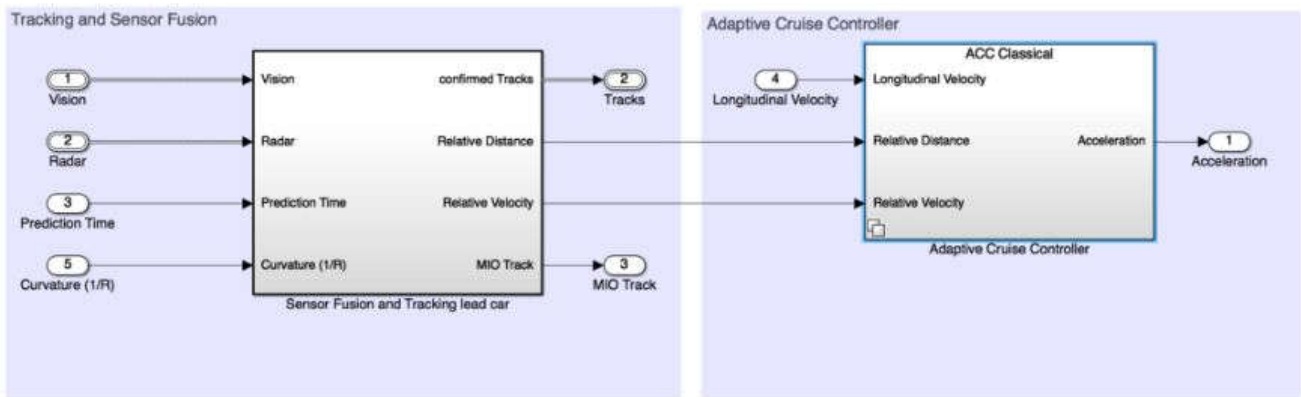


ACC

以下的示例演示了这样一个场景：前方一辆汽车从右边进入了自车的车道，雷达和相机传感器识别到了前车，并已确认。为了安全起见，自车必须估算与出前方车辆的相对距离w.r.t.，如果距离小于安全距离，则自车必须拉开距离，并保持安全距离直至完全停车。直到距离前车足够远，然后自车逐渐加速，直到达到预期的速度。



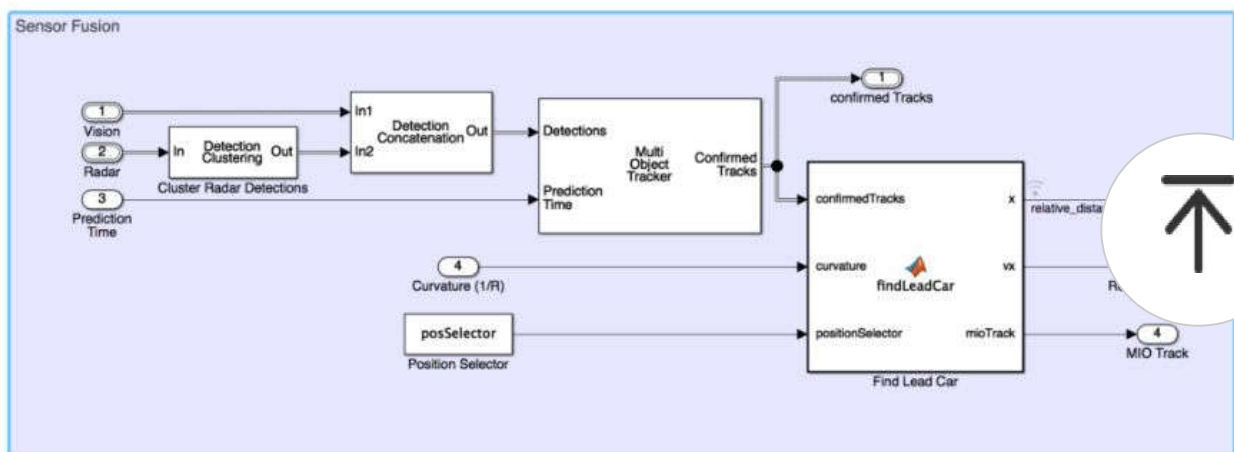
Adaptive Cruise Controller with Sensor Fusion



ACC传感器融合

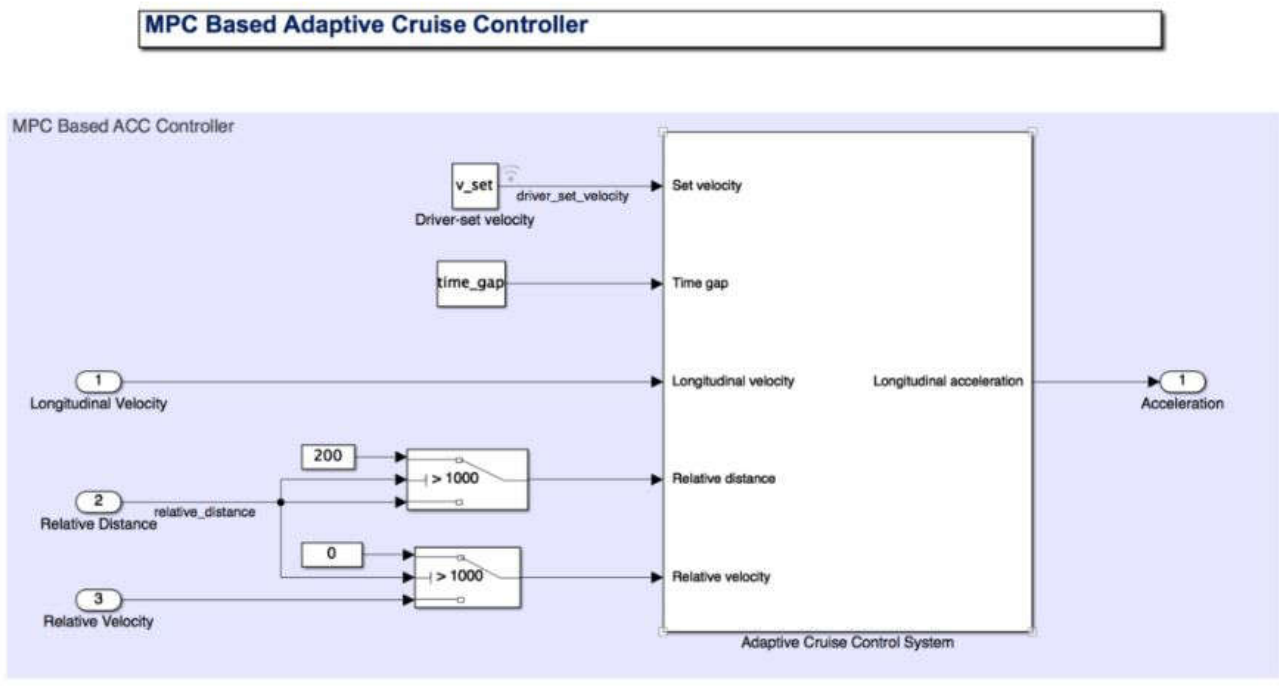
在该Test Bench中，ACC的传感器融合模块具有检测同一车道(以及传感器检测范围内的其他车道)是否有前车的功能，融合测验(去除冗余)，将检测传递给MPC，MPC将会根据实际情况相应地减慢或加速自车。

Tracking and Sensor Fusion



由下图可见，视觉和雷达识别的对象、仿真时间、自车的纵向速度和路面曲率为输入参数。传感器数据融合和前车跟踪子模块，包含由于雷达噪声引起

的第一次雷达探测聚类，并将来自视觉和雷达的探测结合输入到多目标跟踪器。使用卡尔曼滤波器精确估计检测的状态并融合检测，然后，利用确定的轨道和道路信息确定自车与前车之间的相对距离和相对速度，实现ACC的功能。

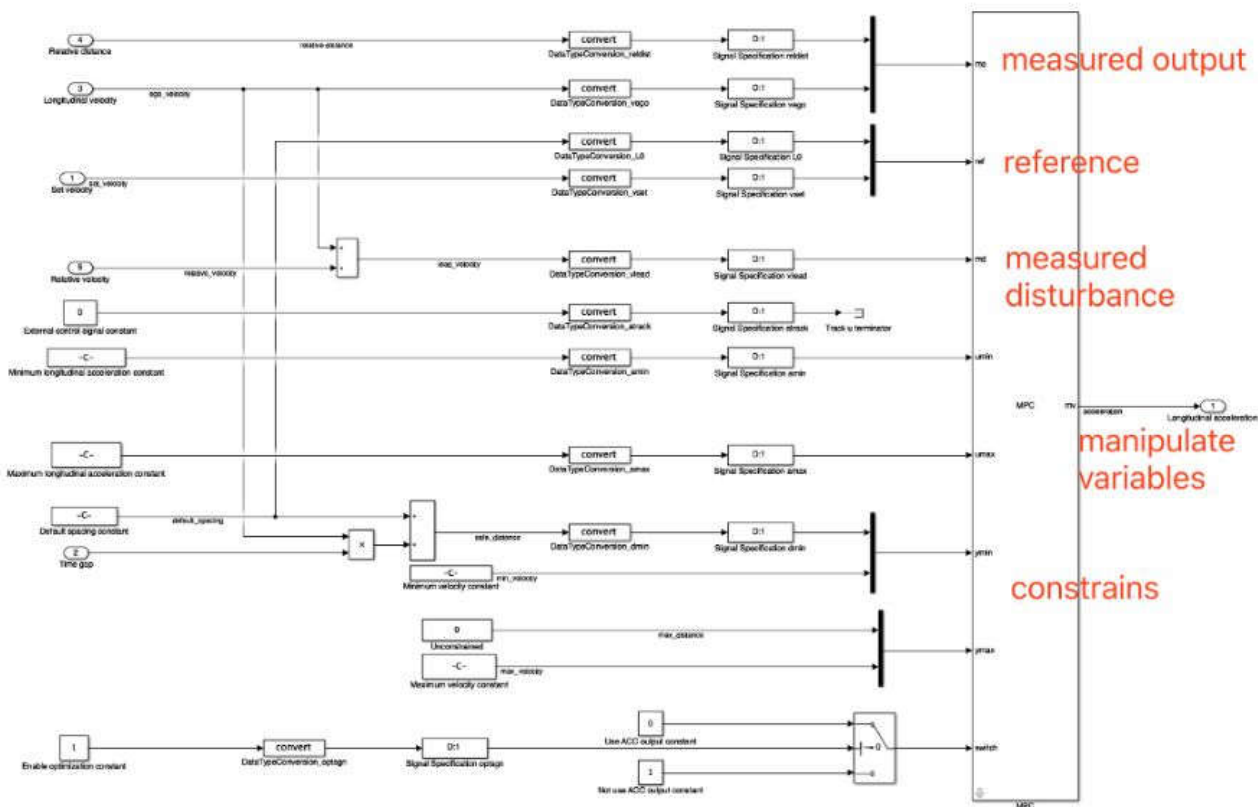


利用MPC算法方案，将时间间隔(可以是驾驶员的反应时间)、纵向速度和驾驶员设定的速度与相对距离、相对速度一起加入自适应巡航控制系统。这个测试台上，使用了预构建的ACC控制模块。也可以构建特定于用户的MPC模块。

该ACC模块的主要功能是跟踪驾驶员设定的速度，并通过调整自车的纵向加速度来保持与前车的安全距离。该模块使用了模型预测控制（MPC）计算

出最佳控制动作，同时还满足了安全距离和速度，并约束了一定的加速度。

算法结构细节如下图所示。然后用户可以从Matlab中对原始ACC模块进行相应的修改。

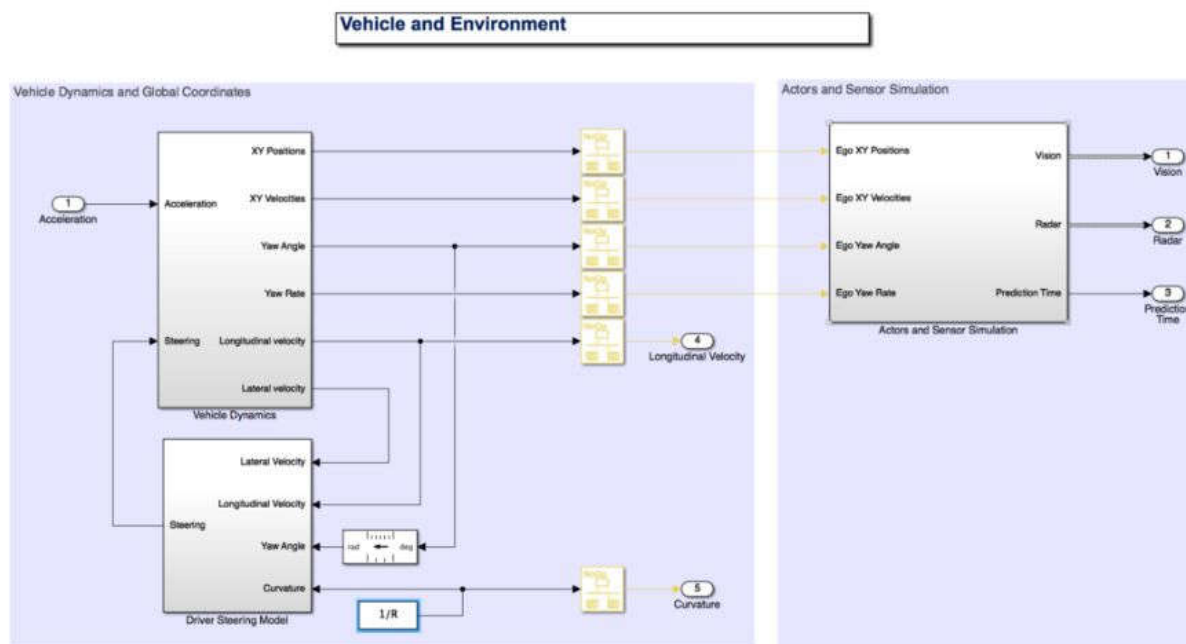


Adaptive Cruise Control using Model Predictive Control

到目前为止，ACC的主要控制已经基本完成。然而，车辆在行驶时，司机还必须要一直保持上。因此，车道跟踪功能（也就是转向控制）也需要考虑在内。

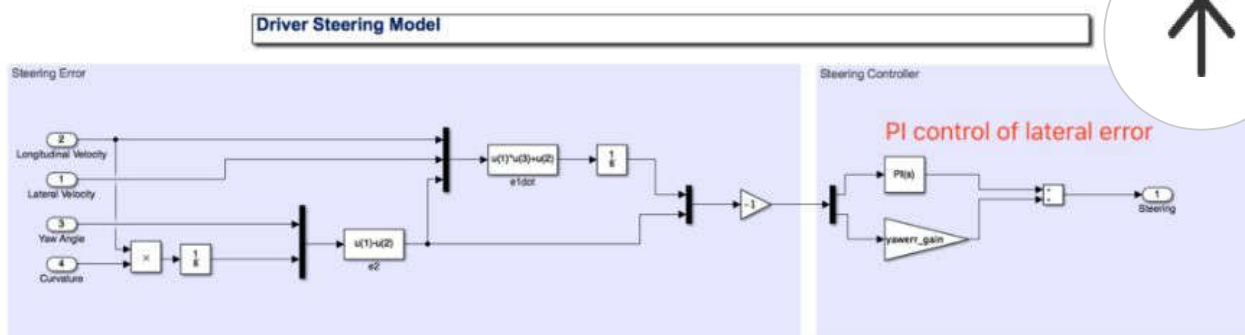
随着MPC纵向加速度的调节，Simulink块中必须要输入道路(地图)信息。在本次的测试用例中，道

路几何简单地通过常曲率 $1/R$ 来描述，并已创建于 Matlab的工作空间中，可以直接从子系统中使用。



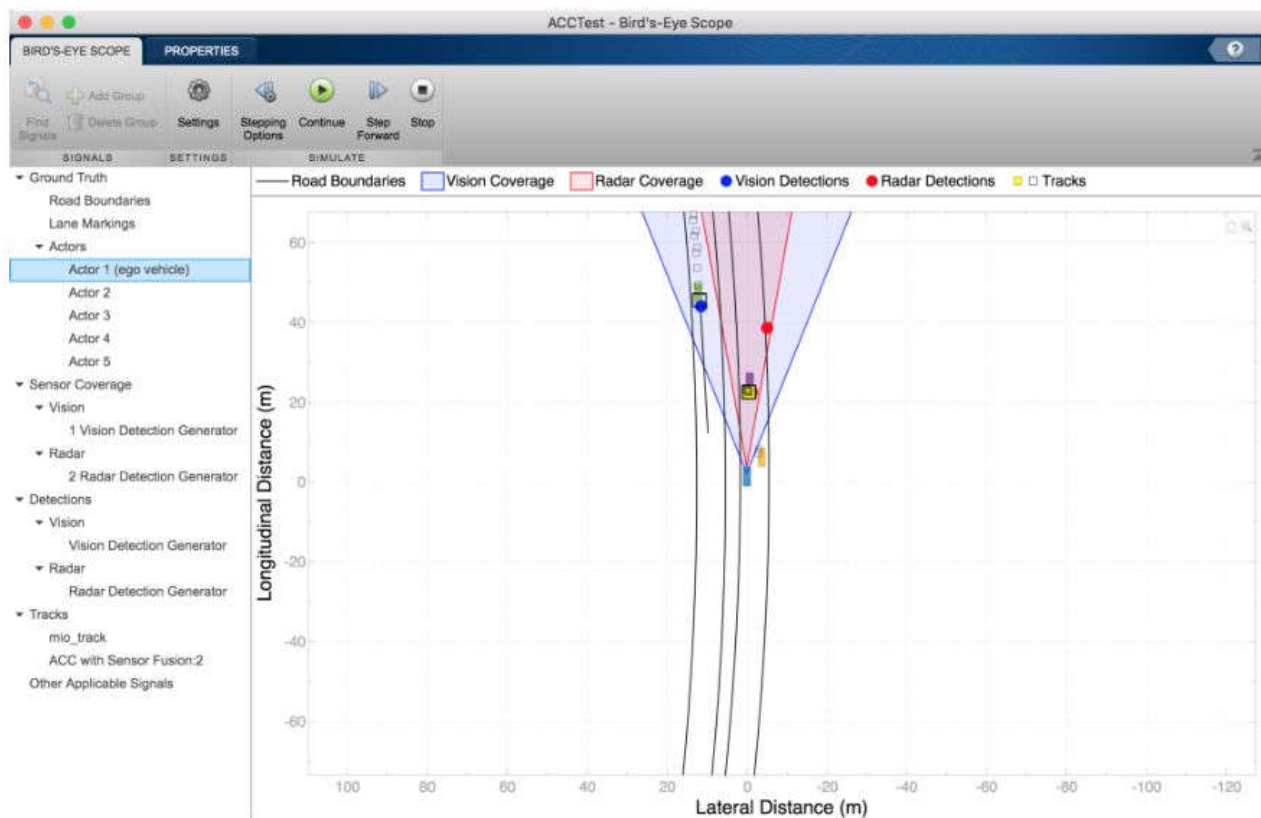
使用MPC调节纵向加速度和曲率的道路更新的位置和偏航角的自车，转向控制采用PID控制。

MPC能够对自车的加速度进行调节，结合采用比例积分微分(PID)控制方案，将可以实现转向控制。

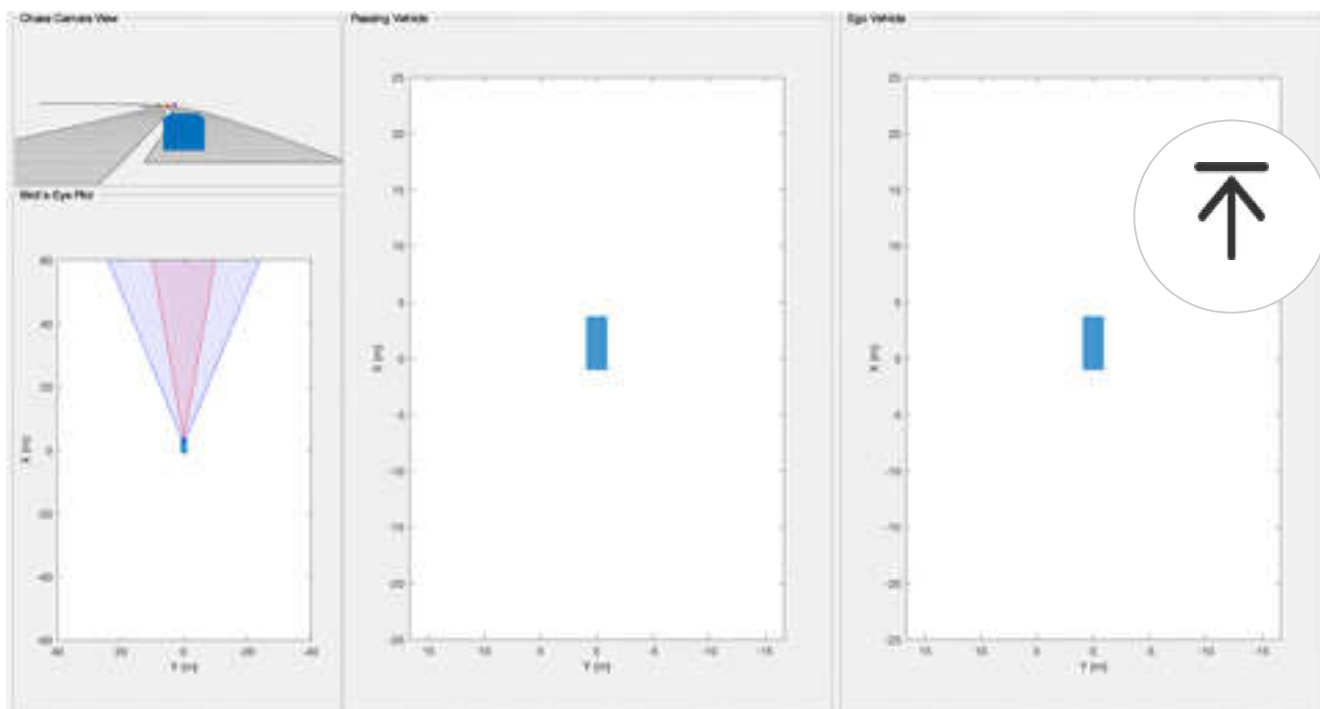


基于自行车模型，可以模拟出自车的位置和偏航角。

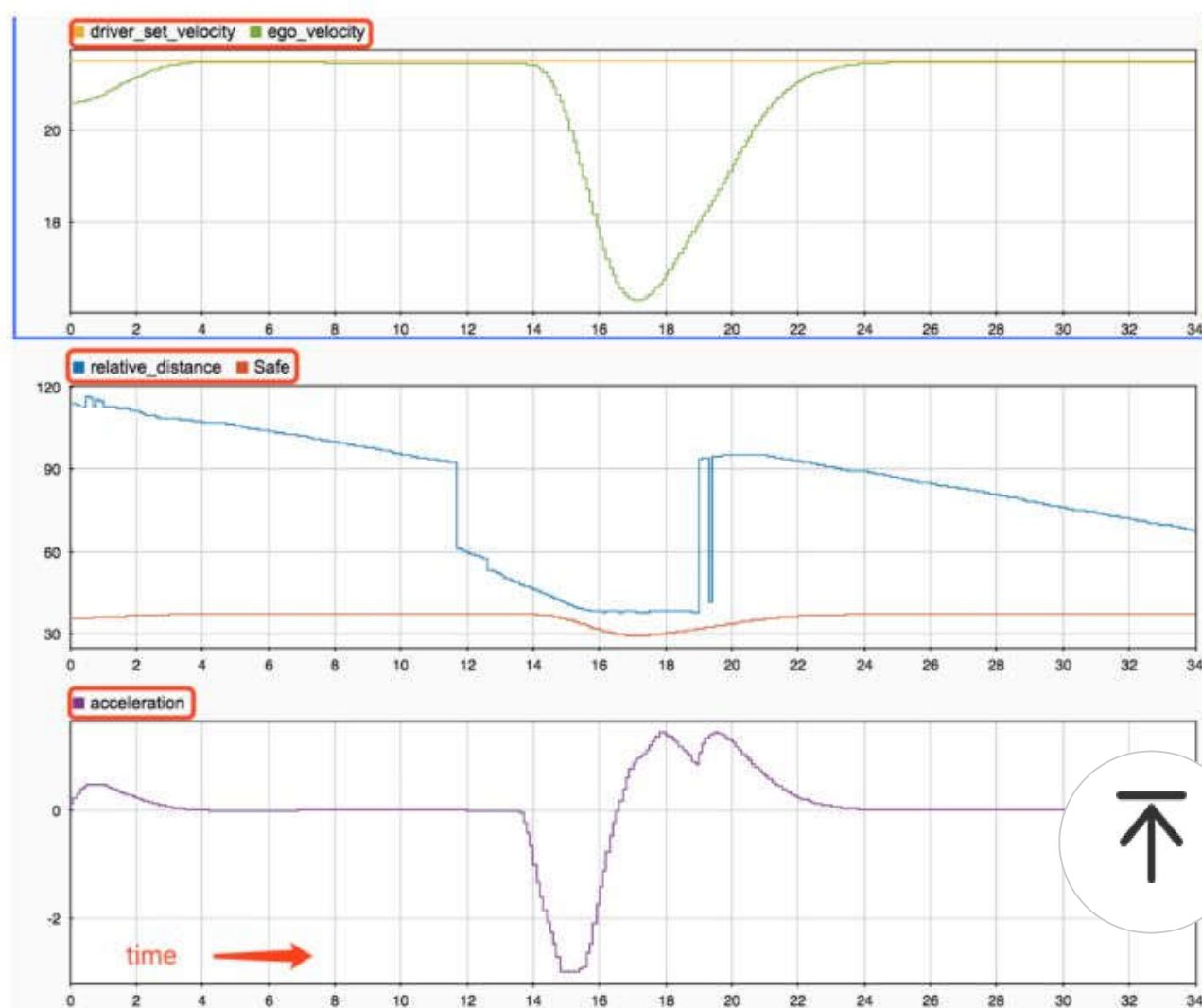
到这里，我们已经具备了运行ACC模拟的所有必要条件。单击run按钮，可以查看结果，如下图所示（只显示一帧）。



整个仿真结果如下：



当一辆他车低速车行驶进入速度较快的自车的车道时，只要传感器检测到低速行驶的前车，在MPC控制的帮助下，自车会先减速以保持安全距离。当前车离开同一车道时，自车再次加速，加速直到驾驶员设定的速度。自车的速度和驾驶员设定的速度如下图所示，并显示了自车的相对距离和加速度曲线。



最后这点很重要，ADAS必须使用C或C++部署到特定的ECU上，而Matlab提供了代码生成器工具

箱，可以轻松实现这一点。如果需要添加或进一步修改C算法，则可以基于生成的C/C++代码继续编写。

以上回顾了利用Matlab/Simulink实现ACC的Test bench。对于更复杂的或不同的驾驶场景，各位朋友们可以使用本文描述的类似方案来进行进一步的功能开发。

