# Defining an Interface for Autonomous Vehicle Steering

Emiko Soroka
PI: Sanjay Lall

January 17, 2021

## Introduction

In the autonomous driving stack, where should the division be made between high-level autonomous behavior and low-level vehicle control? High-level modules that plan the vehicle's route, observe its surroundings, and decide on the best path to a destination can be easily ported from one vehicle to another. However, lower-level steering and acceleration control is more dependent on the vehicle's specific hardware. To optimize fuel efficiency, for example, a gasoline-powered vehicle must be driven differently than an electric-gas hybrid or fully electric vehicle with regenerative braking. Tire size, vehicle weight, engine performance, and other mechanical characteristics inform this optimization, making it clear that this portion of the autonomy stack is much less portable between different types of vehicle.

Defining an interface between easily-interchangeable high level autonomy modules and lower-level controllers enables rapid development of autonomous driving for different vehicle platforms. Currently, autonomous vehicle development programs focus on a specific, sometimes custom, vehicle platform. The ability to transfer this work to other vehicles will enable rapid adoption of autonomous driving by reducing development costs to transfer the autonomy stack onto new platforms.

## Standardizing Autonomous Vehicle Software

Much work has been done on standardizing autonomous vehicle software for safety, system verification, and interoperability between different sensors and software modules. At a system architecture level, the AUTOSAR standard is used for vehicle software, defining how modules should interoperate to provide guidance for more specific frameworks. (citation needed) AUTOSAR focuses on structuring software for verification and compliance with safety standards. At a lower level, protocols are defined to be compatible with AUTOSAR (citation ADAS Middleware paper) At this layer, software is divided into algorithmic modules, independent of vehicle hardware, and low-level modules connected by a hardware abstraction layer (HAL). However, Middleware does not specify what data is actually being transmitted between components of the software sytsem. This remains an open question.

This paper proposes what data should be passed through such an interface; e.g. what data is required to achieve a clean split between hardware-specific (low-level) software and more abstract (high-level) modules, while preserving the vehicle's ability to make hardware-specific optimizations to its driving behavior. We propose this data should contain the road boundaries and optional fixed points: partial or full specifications of the vehicle's state. For example, to specify a complete stop at a stop sign, a zero velocity state would be specified at a specific position. However, the time at which this state occurs does not need to be specified; it can be handled by the hardware-specific controller as it steers to a stop.

## Proposed Interface

Many steering algorithms focus on path tracking: once a path is generated by higher-level autonomy, the goal is to follow it as closely as possible. Paths are commonly represented as a series of waypoints, with the goal to arrive precisely at each specified state. Much work has been done to optimize these waypoint tracking controllers, ranging from optimizing PID gains, to new formulations of the problem for model-predictive control (MPC).

However, if higher-level autonomous modules are interchangeable between vehicles, this reduces their ability to provide an optimal trajectory for the low-level controller while taking a specific vehicle's hardware into account. Clearly, the division between high and low-level (and interchangeable versus vehicle-specific) should be made before this trajectory is generated. To enable optimal steering and acceleration control of different vehicles, is necessary to move beyond waypoint tracking to a more flexible representation of the control problem.

This proposal is for the interface to consist of the road boundaries, along with some desired states to reach. This does not mean the vehicle's state must be fully specified at certain points or times. For example, a vehicle stopping at a stop sign must reach the correct position with the correct velocity (zero), but the acceleration it undergoes to reach that point, as well as the exact time it reaches the point, does not need to be specified. This allows the low-level controller freedom to optimize the vehicle's behavior for fuel efficiency, passenger comfort, or other metrics.

On the low-level, or vehicle-specific, side of this interface, the final path planning can be split into a path planner and a trajectory tracker, or a single nonlinear MPC problem that solves for the control inputs while simultaneously adjusting the vehicle's path.

## Implementation of this Interface

As proof-of-concept, tests have been carried out in simulation to demonstrate this separation between high-level and low-level autonomy. The same planned path is used with different controllers representing different vehicles. Additionally, this demonstrates the ability to implement variable behavior in the low-level controllers. This is a common task: many vehicles today have the ability to switch between 'sport' and 'eco-friendly' modes. In the future, when autonomous vehicles perform deliveries and drive with no human occupants, being able to switch driving behaviors to increase passenger comfort, transport cargo quickly, or maximize fuel efficiency will become even more important.

## Benefits

Clearly defining an interface between interchangeable high-level autonomy modules and hardware-specific modules enables faster development of autonomous driving for different models of vehicles.

This also aids the certification process for autonomous software: the high-level modules need only be developed and certified once. (citation needed?)