2020 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
POWERTRAIN TECHNICAL SESSION
AUGUST 11-13, 2020 - NOVI, MICHIGAN

ECO-DRIVING DRIVER ADVISORY APPLICATION FOR CONNECTED VEHICLES

Piyush Bhagdikar[1], Stas Gankov[1], Sankar Rengarajan[1], Jayant Sarlashkar[1], Scott Hotz[1]

[1]Southwest Research Institute

ABSTRACT

Connected and automated vehicles (CAVs) leverage onboard sensing and external connectivity using Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Vehicle-to-Everything (V2X) technologies to "know" the upcoming operating environment with some degree of certainty, significantly narrowing prior information gaps. These technologies have been traditionally developed and used for driver assistance and safety but are now being used to operate the vehicle more efficiently [1–5]*. The eco-driving algorithm, which leverages the data available through these streams, performs two key functions: (1) acceleration smoothing and (2) eco-approach and departure (Eco-AND) at signalized intersections. The algorithm uses information from neighboring vehicles and signalized intersections to calculate an energy-efficient speed trajectory. This paper presents the development of an Android-based driver advisory application that leverages cellular Internet connectivity and Traffic Technology Services (TTS) data* [6]*, to perform Eco-AND at signalized intersections and discusses the potential for saving energy and time with the eco-driving algorithm.*

Citation: P. Bhagdikar, S. Gankov, S. Rengarajan, J. Sarlashkar, S. Hotz, "Eco-Driving Driver Advisory Application for Connected Vehicles", In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2019.

1. INTRODUCTION

Southwest Research Institute (SwRI), in collaboration with Toyota Motor North America and the University of Michigan, under the Advanced Research Projects Agency-Energy's (ARPA-E) NEXTCAR program, is leveraging connectivity to improve the energy consumption of a 2017 Toyota Prius Prime by 20% [7]. The eco-driving algorithm, developed as part of this program, utilizes a Dedicated Short-Range Communication (DSRC) radio to communicate with neighboring vehicles and infrastructure. The same eco-driving algorithm has been deployed on an Android device with the additional capability of leveraging signal phase and timing information available through a web query. This application gives the driver an energy-efficient velocity trajectory through a signalized intersection. The modularity of the application and the target device allows it to interface with other conventional V2V, V2I, and V2X communication modules and onboard sensors to leverage additional information

streams to perform a full eco-driving advisory that accounts for surrounding vehicles. In the following sections, an overview of the eco-driving algorithm is presented, the advisory application architecture and the development process are described, and finally, data demonstrating the potential for reducing energy consumption by using the algorithm for Eco-AND is shown.

2. ECO-DRIVING ALGORITHM

As described in the previous section, the eco-driving algorithm uses information from neighboring vehicles and signalized intersections to calculate an energy-efficient velocity trajectory. The goal is to generate a trajectory that minimizes accelerations, avoids stopping at intersections, or arrives at intersections efficiently without significantly affecting the surrounding traffic. The cost function for the optimizer includes acceleration, jerk, and deviation from the desired velocity for a segment. The deviation from the desired velocity acts as a forcing function to ensure the vehicle does not drive excessively slowly, affecting traffic around the ego-vehicle.

$$J = \int_0^{S_f} \{a(s)^2 + (da/ds)^2 + (v(s) - v^*)^2\}\, ds$$

$$= {a_0}^2 + \sum_{k=1}^{N} \{ {a_k}^2 + \left(a_k - a_{k-1}\right)^2 + \left(v_k - v^*\right)^2 \}$$

- First term is acceleration
- Second term is jerk (change in acceleration)
- Third term is deviation from desired velocity
- Desired velocity is a property of the road (ex: highway → 65 mph)
- Individual terms are normalized and weighted

Figure 1 illustrates the eco-driving algorithm reacting in an energy-efficient fashion by leveraging information about the lead vehicle and nearby traffic light. Around the 165 second mark, the ego-vehicle performs an intentional deceleration maneuver (top half of Figure 1) to ensure that it arrives at the traffic light location during a green phase (bottom part of Figure 1). The eco-driving algorithm demonstrated a 6% nominal savings in energy consumption based on tests run on a four-mile corridor with signalized intersections in the simulated traffic environment The results are discussed in Section 4.

3. DRIVER ADVISORY APPLICATION

Figure 2 depicts the information needed to realize the eco-driving algorithm on a target device. The eco-driving algorithm needs the Signal Phase and Timing (SPaT) information of the approaching signal and the velocity and location of neighboring vehicles. The four information gateways shown on the left side of Figure 2 are to obtain the following information:

- Global Positioning System (GPS): Ego-vehicle location, velocity, and heading.
- Long-Term Evolution (LTE): Enables Internet connectivity. This communication

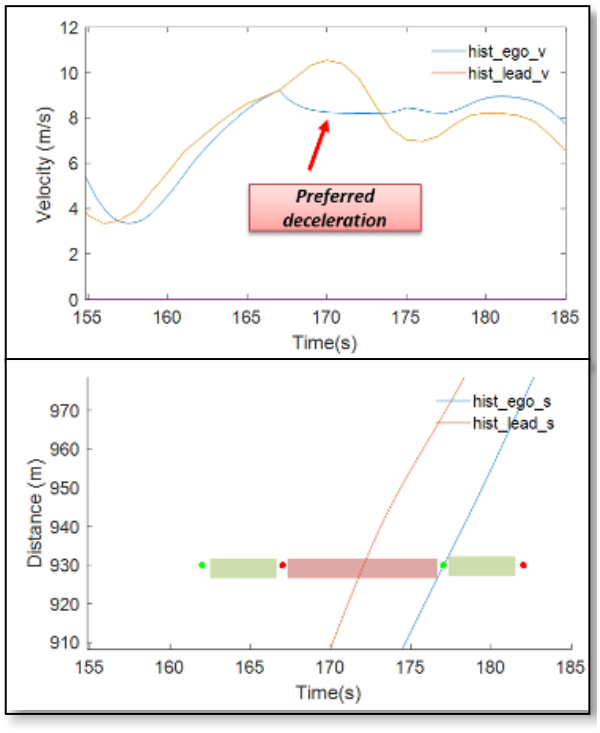

Figure 1: Eco-driving results from vehicle testing on CAV hub dynamometer

pathway is used to gather information on traffic signals, road speed limits, and routing. Any other communication method that enables Internet connectivity can be used instead of LTE.

- Vehicle Controller Area Network (CAN) bus: Vehicle speed, relative velocity, and distance to lead vehicle (if equipped with radar or camera).
- DSRC radios: SPaT and MAP messages from roadside units (RSUs) and Basic Safety Messages (BSMs) broadcast by neighboring vehicles. The BSMs contain location, speed, and heading information of neighboring vehicles within an approximate radius of 500 m.

At the outset, a selection for the operating system (OS) and target hardware was done. Two operating systems were considered: Android and Microsoft Windows. Given the popularity of Android-based mobile devices, the team decided to move forward with building the application for Android OS. Most Android devices already come equipped with GPS and LTE radios. Additionally, several kits are available at online marketplaces that enable interfacing with the vehicle CAN bus [9], and SwRI has experience with interfacing a DSRC radio with an Android device. This makes Android a suitable platform to deploy the algorithm.

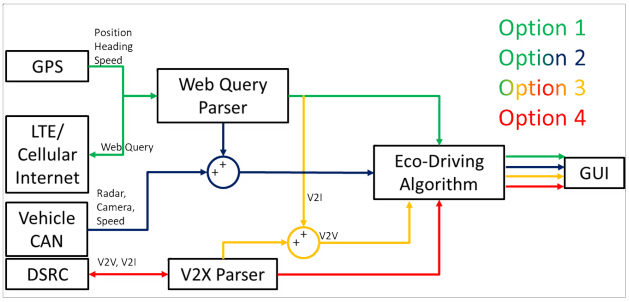

Figure 2: Driver advisory application information flow

3.1. Android Application Development

To make a minimum viable prototype (MVP) for an initial demonstration, the driver advisory is realized based on "Option 1" as depicted in Figure 2. The information available through "Option 1" enables the algorithm to generate energy-efficient velocity trajectory for Eco-AND only. MathWorks' Simulink Support Package for Android [8] was used to compile the Simulink based eco-driving algorithm for the Android target. The toolbox provides functionality to deploy a Simulink model as a standalone Android application and generate an Android application project. The Simulink model is exported as a shared object library (.so) for the project. The toolbox provides gateways to exchange data between the Simulink model and the Java code. This allows the developer to modify the user-interface components, as well as back-end Java code to add functionality to the application. Significant modifications were necessary to a) implement TTS application programming interfaces (APIs) to acquire the traffic light information b) implement Google APIs for route information c) query OpenStreetMap database for the speed limit d) parse and fuse incoming data and e) add more user-friendly UI widgets.

The TTS API provides the signal phase and timing information for an approaching intersection. TTS servers run models to predict how long the traffic signal controller remains in a red or green state. The data obtained using this service contains a confidence value associated with it, which can be used to determine if the eco-driving algorithm should act on the received data or not. To provide traffic light information, the API needs the location and heading of the vehicle as an input. If the vehicle intends to pass the signal without turning, the heading information provided by the device GPS is valid. However, to provide the correct information in cases where the vehicle intends to turn, the maneuver should be known in advance. This information gap is filled by using Google Maps' Directions API [10], which provides turn-by-turn

direction guidance for the entire trip. Assuming the driver follows the advised route, the information can be reliably used to provide correct heading information needed to query TTS. TTS and Google Maps GUIs are depicted in Figure 3. The advisory application utilizes the back-end information these GUIs operate on via APIs and fuses this information to provide to the core algorithm. Additionally, the eco-driving algorithm needs a desired velocity for a road segment to avoid violating the road speed limit, while ensuring that the ego-vehicle does not negatively alter surrounding traffic by driving too slowly. The speed limit information is obtained by querying the OpenStreetMap database through the Overpass API [11]. The development of such an application is not limited to the APIs used here, and there are alternatives for each of them [12,13].

3.2. User Interface

The preliminary user interface, designed primarily for debugging and field testing, is depicted in Figure 4. In addition to the visual aids, audio messages are played when the ego-vehicle approaches an intersection and there is a need to slow down or speed up. The user interface widgets can be broadly grouped into three categories:

- Speed advisory: The dial displays the difference between the target speed computed by the eco-driving algorithm and the vehicle's actual speed. The dial moves towards the blue zone (positive values) when the ego-vehicle travels at a speed slower than the recommendation. Similarly, the dial moves towards the red zone (negative values) when the vehicle travels faster than the recommendation, advising the driver to slow down. The difference is displayed on the top left corner of the screen. The actual (GPS) speed of the ego-vehicle, the target speed as computed by the algorithm, and the road segment speed limit obtained using the Overpass API are displayed just below the dial advisory. Audio messages are played when the ego-vehicle approaches a traffic light and there is a need to slow down or speed up. This message is also overlaid on the speed dial, as depicted in Figure 4.
- Traffic light advisory: Based on the intent of the ego-vehicle, the relevant traffic light phase and time remaining in that phase (as determined by TTS) are displayed. The maneuver is displayed to the left of the traffic light. The name of, and distance to, the approaching intersection is also displayed on the top of the screen. The traffic light advisory uses the current GPS heading, unless the origin and destination coordinates are provided and a route is fixed at the beginning of the trip.

Figure 3: a) TTS GUI b) Google Maps Route

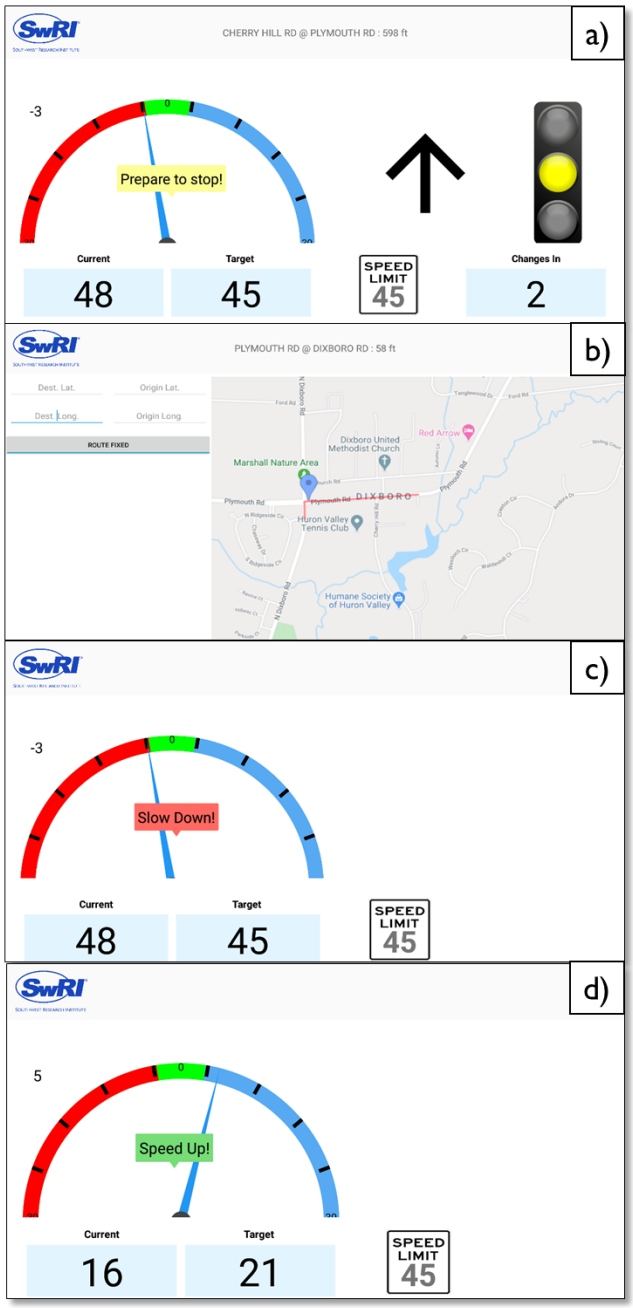

Figure 4: a) Ego-vehicle approaching an intersection and being advised to prepare to stop
b) Second page of the application displaying route options
c)Ego-vehicle being advised to slow down because it is traveling at speed greater than the speed limit on a segment of road with no signalized interaction
d) Ego-vehicle right after passing a signalized intersection being advised to speed up in an economical fashion

- Routing information: The second page of the application, which can be accessed with a swipe, contains fields to input the origin and destination coordinates, and a button to request a route between the two points. A snapshot of the route is displayed as soon as the information is parsed.

There might be scenarios where the algorithm cannot compute a target speed due to the unavailability of a valid road speed limit and/or signal phase and timing information. In such cases, the "Target Speed" field displays no numerical value, and the speed advisory dial stays at zero. Further, when the ego-vehicle is driving on a segment of road, towards a signalized intersection not in the prediction horizon of TTS, all traffic light advisory widgets are hidden.

The developed application was tested on several Samsung and Google phones and tablets.

Safety distance buffers are built into the algorithm, allowing the application to scale to a full eco-driving advisory when the neighboring vehicle information becomes available. However, it should be noted that this application is only an advisory, and a human driver is still in-charge.

4. SIMULATION TESTING OF THE ALGORITHM

A simulation study was performed to quantify the benefits of Eco-AND. Caliper TransModeler, a traffic simulation software, was used to generate testing scenarios. The addition of a custom wrapper created a closed-loop simulation environment where the ego-vehicle maneuvers are computed outside the TransModeler environment (on a real-time controller or in Simulink), based on the traffic and traffic light information provided by the TransModeler at every time step. The computed ego-vehicle target speed is sent to the TransModeler environment. The ego-vehicle is one of the actors in the traffic simulation environment and acts based on this external command. If present, all other vehicles in the traffic simulation

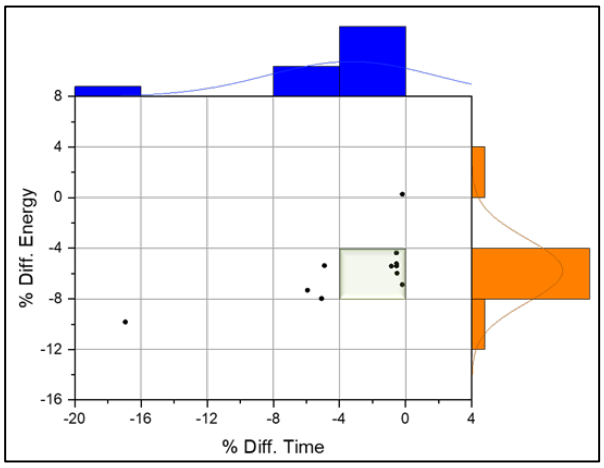

Figure 5: Eco-AND energy and trip time benefits

environment come equipped with a base driver model to react to the vehicle in front and the approaching traffic light state. A high-fidelity forward-looking model of the 2017 Toyota Prius Prime was developed and validated with the real-world trip and regulatory drive cycle data to estimate energy consumption in simulation. The vehicle simulator validation is discussed in detail in [7].

Figure 5 summarizes the results of the simulation study done on a four-mile corridor with no traffic, and eight signalized intersections. This scenario removes the effects of neighboring vehicles to quantify benefits purely from Eco-AND. Twelve runs with different start times were used to generate a bivariate distribution comparing the energy consumption and trip travel times of a vehicle in CAV mode using the eco-driving algorithm to a human driver. For this study, the vehicle simulator was run in charge-sustaining mode for energy estimation. With the eco-driving algorithm controlling the longitudinal speed, most runs show a reduction in energy consumption between 4% and 8%. The weighted average benefit is approximately 6% on energy consumption and 3.2% on time.

5. CONCLUSION

Vehicle connectivity has enabled the development of technologies that not only seek to improve safety, but also efficiency. The Android implementation of the eco-driving algorithm presented in this paper goes beyond using conventional CAV communication channels to operate the vehicle more efficiently and enables its use in human driven vehicles. The team is considering the following future work to improve the application:

- To fully realize the benefit of information streams enabled via V2X communication, the application needs to be interfaced with a DSRC radio.
- The team is also exploring pathways to interface the application with onboard vehicle sensors, such as radar, to optimize speed trajectories when neighboring vehicles are not equipped with CAV technology.
- Driver clinics need to be conducted to refine and improve the user interface.
- The routing features could be made more user-friendly by accepting place names instead of coordinates and displaying dynamically updating navigation instead of a static route overview.
- The application provides an opportunity to integrate the eco-routing framework developed as a part of the NEXTCAR program [7], enabling users to choose a route based on distance, time, and more importantly, energy consumption on each of the possible routes.

1. REFERENCES

[1] D. Shen, D. Karbowski, and A. Rousseau, “Fuel Efficient Speed Optimization for Real-World Highway Cruising,” in *SAE Technical Papers*, 2018, vol. 2018-April, doi: 10.4271/2018-01-0589.

[2]J. Lu *et al.*, “Predictive Transmission Shift Schedule for Improving Fuel Economy and Drivability Using Electronic Horizon,” vol. 10,

no. 2, pp. 680–688, 2017, doi: 10.4271/2017-01-1092.

[3]Z. Wadud, D. MacKenzie, and P. Leiby, "Help or hindrance? The travel, energy and carbon impacts of highly automated vehicles," *Transportation Research Part A: Policy and Practice*, vol. 86, pp. 1–18, Apr. 2016, doi: 10.1016/j.tra.2015.12.001.

[4]J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annual Reviews in Control*, vol. 45, pp. 18–40, 2018, doi: https://doi.org/10.1016/j.arcontrol.2018.04.011.

[5]L. Tate, S. Hochgreb, J. Hall, and M. Bassett, "Energy Efficiency of Autonomous Car Powertrain," in *SAE Technical Papers*, 2018, vol. 2018-April, doi: 10.4271/2018-01-1092.

[6]"Personal Signal Assistant - TTS." https://www.traffictechservices.com/how-it-works.html (accessed May 19, 2020).

[7]S. Rengarajan *et al.*, "Energy Efficient Maneuvering of Connected and Automated Vehicles," in *SAE Technical Papers*, Apr. 2020, vol. 2020-April, no. April, doi: 10.4271/2020-01-0583.

[8]"Simulink Support Package for Android Devices Documentation." https://www.mathworks.com/help/supportpkg/android/index.html?s_tid=CRUX_lftnav (accessed May 19, 2020).

[9]"OpenXC." http://openxcplatform.com/ (accessed May 19, 2020).

[10]"Developer Guide | Directions API | Google Developers." https://developers.google.com/maps/documentation/directions/intro (accessed May 19, 2020).

[11]"Overpass API - OpenStreetMap Wiki." https://wiki.openstreetmap.org/wiki/Overpass_API (accessed May 19, 2020).

[12]"Build apps with HERE Maps API and SDK Platform Access | HERE Developer." https://developer.here.com/ (accessed May 19, 2020).

[13]"Maps SDK | Android | Mapbox." https://docs.mapbox.com/android/maps/overview/ (accessed May 19, 2020).