Autonomous Vehicle Control: Teaching Tool and Simulation

Marie O'Brien
Faculty of Applied Science
University of British Columbia Okanagan
Kelowna, Canada
marie.obrien@ubc.ca

Kashish Gupta
Faculty of Applied Science
University of British Columbia Okanagan
Kelowna, Canada
kashish.gupta@ubc.ca

Bara Emran
Faculty of Applied Science
University of British Columbia Okanagan
Kelowna, Canada
bara.emran@ubc.ca

Homayoun Najjaran
Faculty of Applied Science
University of British Columbia Okanagan
Kelowna, Canada
homayoun.najjaran@ubc.ca

Abstract— **This Research to Practice Category Full Paper presents a teaching tool for users to design and practice different features and parameters of the longitudinal control of autonomous vehicles. The proposed platform is composed of four modules: vision-based perception, decision-making, speed control and simulation. The teaching tool provides the ability for users to work with individual modules and then integrate these modules to visualize their effect on the vehicle performance in simulation. Each module has its own MATLAB graphical user interface for users to simplify the learning procedure and intuitively update system parameters. The output of the teaching tool, autonomous vehicle longitudinal control, is visualized using the MATLAB 3D World Simulator. The proposed teaching tool was presented in a fourth-year engineering control system class and the students' feedback was used to improve the teaching tool for a better learning experience.**

Keywords—teaching tool, fuzzy logic, image processing, longitudinal control, autonomous vehicles

I. INTRODUCTION

Due to the rapid emergence of autonomous vehicle (AV) technology and the industry's need for trained engineers and technologists, incorporating educational AV related tools into existing undergraduate engineering curricula is essential. Research of AV technology can be separated into the following components: perception of static and dynamic obstacles, local and global localization and mapping, path planning, high-level decision making, vehicle control, and connected vehicle technology [1]–[3]. Aspects of these components are being introduced separately in undergraduate and graduate engineering courses such as modern and digital control, and data and sensor fusion. However, as this sector continues to expand, greater emphasis on applying theory through modeling and simulation to real-life applications is essential.

Many different software tools and programming environments are available to teach students engineering topics, but MATLAB is one of the most popular programs for engineering applications due to its availability of pre-built toolboxes, Simulink, and 3D simulation and modeling [4]. In addition, educational games and toolboxes for modeling and simulation in a variety of engineering areas are being developed in MATLAB [5]–[8]. Studies have shown that integrating modeling and simulation into curricula provides opportunities for students to further analyze and visualize complex problems and interactively test and compare models to gain an in-depth understanding of the course content [9],[10]. MATLAB's toolboxes typically focus on one main topic such as fuzzy logic or image processing. One of the remaining challenges is understanding how these toolboxes can be related to one another for a real-world engineering application such as AVs.

Our application has been designed to solve this problem by providing user friendly graphical user interfaces (GUIs) which integrate three main AV concepts perception, decision making, and control into a teaching tool. The authors have developed a modular educational tool for undergraduate students to model and simulate an adaptive cruise control system. As these concepts are currently not included in detail in the undergraduate engineering curricula we have separated the educational tool into four modules namely vison-based perception, decision-making, speed control and simulation to accommodate a range of undergraduate student knowledge levels. As an overview the users access the individual module GUIs through the main GUI screen (Fig. 1). The user works through the first three modules separately which are then connected through the Simulink model for simulation using the 3D World Simulator. For the vision-based perception, decision-making, and speed control modules the users can save, load and modify different versions of their attempts and perform comparisons of their systems against the default systems. As well, each module includes a "Help" button, where the user can access a detailed manual of the module.

The expected learning outcomes of the educational tool are for students to gain an overall understanding of the three AV technology components through the process of modeling and analyzing results and then visualize the combined outcomes as an adaptive cruise control system. Students will experience the design process of building a model based on the course theory and develop a deeper understanding of concepts through testing and verification of the models based on real applications. The educational tool was developed in MATLAB and Simulink for seamless integration into the existing engineering curriculum.

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

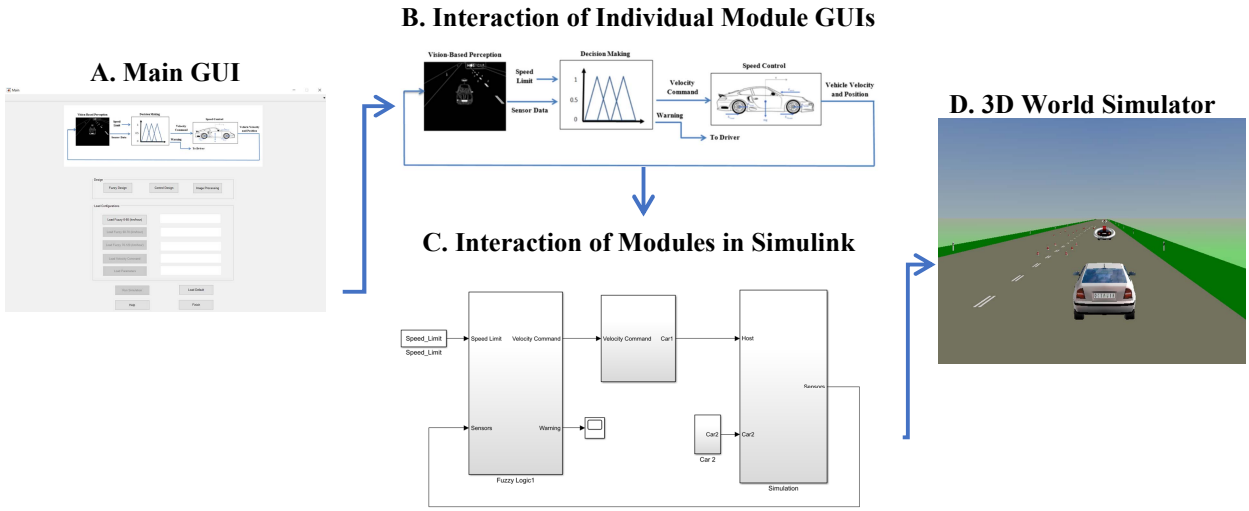

Fig. 1. Proposed Teaching Tool Overview

The results of each module are evaluated based on self-assessment through comparison with the default model results.

The remainder of the paper is organized as follows. Section 2.0-5.0 discuss the vision-based perception, decision making, speed control, and simulation modules. The implementation of the teaching tool is presented in Section 6.0 followed by the conclusions and future work in Section 7.0.

II. VISION-BASED PERCEPTION MODULE

Visual perception in humans has long been the motivation behind visual data gathering systems for autonomous vehicles. While machine vision has proved its efficiency and achieved many milestones to make self-driving cars a reality [11], we are still a long way from level five autonomy. With the trend shifting towards intelligent systems in global research, it is of paramount importance to educate the upcoming students and generate their interest in the field to promote quality research. The vision-based perception module of the teaching tool enables the user to analyze and extract useful information from image data gathered using a simulated front facing camera as a part of the on-board sensor system.

In a real-world application, the sensor system or the perception module gathers and analyzes information about the 3D world that acts as input to the decision-making and control modules. In this case however, data is gathered through a simulation. Thus, the module allows input from the simulation in the form of images and classifier training data to enhance user experience with general image operations. The objective of the section is to familiarize the user with the steps involved in an image processing pipeline from image acquisition to post-processing to get a desired output for successful lane detection and to learn about different kinds of classification techniques through MATLAB's classification learner app [12]. The vision-based perception module enables the user to learn through experimentation on raw image data gathered through the Simulink 3D viewpoint. At any stage during the process, the intermediary image output can be visualized and saved into a file. The app consists of five major sections (Fig. 2).

A. Pre-processing

In this section the users can select one of the Median or Gaussian Blur filters and input the threshold and corresponding parameters for the inputted image. As the image is simulated, the general gradient change through the image is steep which contributes to the high frequency noises. Gaussian blur smoothens the edges while median blur is chosen for salt and pepper type noise.

B. Edge Detection

This section features the four most widely used edge detection techniques for image processing: Sobel, Canny, Prewitt and Roberts. In general, edge detection is a mathematical operation on an image to find areas of sharp changes or discontinuities in brightness values. The related threshold and direction for a chosen technique can be manipulated to observe the change in the image in real time. Users can input the parameters corresponding to the chosen technique or use the "Auto Thresh" button to calculate the best threshold for the inputted image and technique type. A short explanation of the available edge detection techniques is provided below:

1) Sobel – This method uses 3X3 convolution kernels along the specified direction (horizontal, vertical or both) in the image to find gradient change. The kernels obtained after convolution operation are decomposed to find gradient magnitude and direction. Magnitude above a specific threshold is marked as edges.

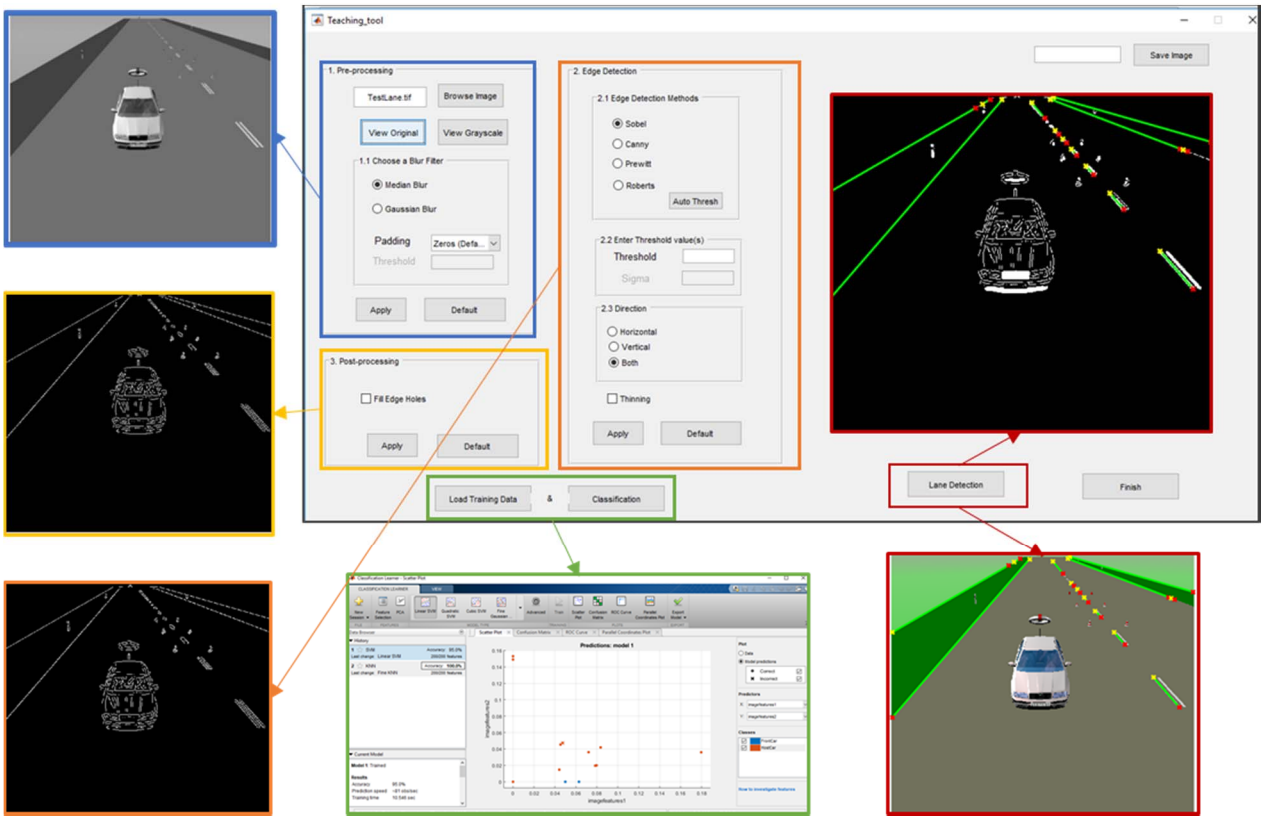

Fig. 2. Vision-based perception (Machine Vision) GUI

2) Canny – This method uses intensity gradients to classify pixel points as edges based on double threshold. A gaussian kernel filter is applied to the image to calculate the intensity gradients. The Canny operator can find a wider range of edge pixels in an image and uses non-maximum suppression to remove false edges.

3) Prewitt – This method uses the derivative of pixel data in the specified direction to find potential edges. The process is similar to Sobel and uses a 3X3 convolution kernel.

4) Roberts – This method uses two pre-defined 2X2 kernels for the convolution operation.

For all the techniques described above, the gradient for a pixel point (x,y) is given by (1).

$$G(x,y) = \sqrt{{G_x}^2 + {G_y}^2} \qquad (1)$$

Where G_x is the point formed by convolution operation with the first kernel at (x,y) and G_y is the point formed by convolution operation with the second kernel at (x,y).

C. Post-processing

This section enables the user to choose the thinning parameter for edge detection and visualize its effect on the edge detection output. The thinning parameter removes the spurious points from the edge detection result.

D. Lane detection

The purpose of this section is to find and mark lanes on the output of the edge-detection image produced after the post-processing section. Hough line transformation is used to identify straight continuous edges as lanes and marks the identified start and end points with different markers for visualization (Fig. 2). The section uses pre-defined parameters for computation and does not allow user input. The output of this section reflects on the parameters chosen in the previous sections A-C. The user can update the corresponding parameters to attain a better lane detection result.

E. Classification

In this section of the module the user is able to load a given set of training images into the workspace. The training data consists of multiple sample images of the two cars – front and host, from different Simulink generated viewpoints. The app calculates 200 features from each image using the "bag of words" model and connects to MATLAB's classification learner app. The user can learn about different classification techniques offered by the app through experimentation and visualization of the training and verification output in the form of a confusion matrix, ROC curve, etc.

III. DECISION-MAKING MODULE

Due to technological advancement and availability of large databases, decision-making systems for autonomous vehicles have evolved drastically from traditional mathematical models to machine learning and heuristic algorithms [13]. Fuzzy logic, a type of heuristic algorithm, allows for modeling of vague and imprecise values typically involving human reasoning that traditional mathematical models are not able to properly capture [14]. Fuzzy logic uses fuzzy sets providing a membership over a [0,1] interval which differs from the traditional mathematical models that use crisp sets [14]. Since the introduction of fuzzy sets by Zadeh in 1965 [15] fuzzy logic has been used for a wide variety of applications such as medicine [16], fatigue management [17], vehicle control [18] and unmanned aerial vehicle control [19]. Fuzzy logic has been used for car-following models since the early 1990's [20]–[22]. These studies illustrate early on the benefit of using fuzzy inference systems over classical mathematical models for complex problems such as driving. Improvements to early car-following models are still being explored through adaptive neuro-fuzzy inference systems and evolving neuro-fuzzy models [23], [24].

A. Decision-making module overview

The learning objective of this module is to understand the overall process of developing a fuzzy inference system (FIS). The decision-making module provides a guided user interface which intuitively allows the users to develop (i) a driver warning model and (ii) a velocity command model. A GUI was built for each model (Driver Warning GUI and Velocity Command GUI). The GUIs connect to MATLAB's Fuzzy Logic Toolbox which uses the selected parameters to build the fuzzy model [25]. When the user clicks the "Fuzzy Design" button on the main GUI screen (Fig. 1) both decision-making GUIs open. The look and utility of the GUIs are identical with the exception of the input and output parameters.

In this teaching tool, a Mamdani fuzzy logic model is implemented. As an overview, this type of fuzzy logic model has five main phases: (i) the development of the input and output membership functions, (ii) input fuzzification, (iii) development of the rule-base, (iv) aggregation, and (v) output defuzzification [14]. The decision-making GUIs encompass these phases through four main sections: input and output parameters, membership function design, rule-base development and visualization and assessment (Fig. 3 A-D).

B. Input and Output Variables

The main purpose of this section is to familiarize the users with the input and output variables used in each GUI. For the driver warning model three FISs are built to calculate the driver warning. Each have three input values: relative velocity, time headway and ego vehicle velocity and one output value: the warning level. Three FISs are required in order to apply the Insurance Corporation of British Columbia's (ICBC) safe following distance rules for all speed ranges (0-60,60-70,70-120) [26]. The driver warning module provides a numerical warning depicting if the driver is a safe, satisfactory or an unsafe distance from the lead vehicle based on ICBC driving rules. For the velocity command model, the FIS calculates the increase or decrease velocity percentage (the output) based on two input parameters: relative velocity and the difference between the actual relative distance and the required relative distance. The purpose of this FIS is to calculate an appropriate increase or decrease velocity percentage which is then used to calculate the velocity command. The velocity command is provided to the speed control module as the desired speed input. The number and type of input and output parameters for both models have been pre-set in the application. The speed limit and the input values for both models are provided to the decision-making module through Simulink from the sensors block. The sensor block simulates the collection of information received from an on-board sensor system such as vehicle position and velocity.

The user will first load the default FIS or previously made FIS into the GUI to initialize the input and output parameters. Then the user saves their own FIS version which allows them to adjust the parameters to create their own system. If the user attempts to load an FIS which does not have the correct number and name for input and output parameters a "pop-up message" informs the user, why it cannot load the FIS. In addition, the user is able to visualize the input and output variables from the loaded FIS by clicking the "Display" button.

C. Develop Membership Functions

Fuzzy logic uses linguistic variables to describe parameters, referred to as fuzzy sets, in the input and output space [14]. Each of these fuzzy sets have a membership function assigned to it which provides the degree of membership from [0, 1] using functions such as triangular or trapezoidal (Fig. 3-B) [14]. The users are provided with default membership functions for each input and output variable (Fig. 3-A). For each variable the users can update existing membership functions, add or remove membership functions or start from scratch (Fig. 3-B). To simplify the development of membership functions and enhance the learning experience only triangular and trapezoidal membership function options are available. Also, to ensure realistic driving values are maintained each variable has a maximum and a minimum value.

D. Rule base development

The fuzzy sets are then used to develop a rule base using if-then statements which presents the connection between the input and output model space. An example rule is shown below where the italicized words are the system variables and the bolded words are the fuzzy sets. A rule is created for each input output relationship.

If the *relative velocity* is **slow** and the *time headway* is an **unsafe** distance and the *ego vehicle velocity* is **slow** then the *warning* is **high**

The rule base is created or updated after the user adjusts the membership functions to show the new relationships between the input and output model space (Fig. 3-C). The user updates the rules by selecting the "Create or Update Rule Base" button which opens the Fuzzy Logic Toolbox Rule Editor [25].

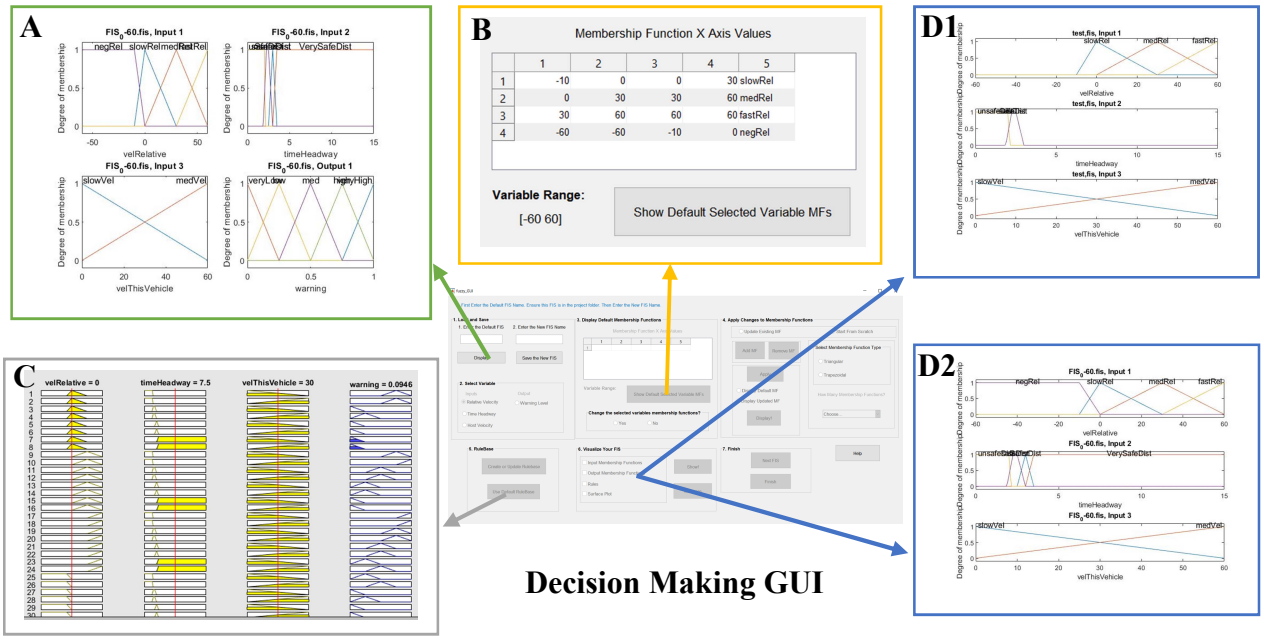

Fig. 3. This figure presents the Driver Warning GUI used in the decision-making module. The GUI is divided into four main sections: input and output parameters (A), development of membership functions (B), rule base development (C) and visualization and assessment (D)

E. Visualization and Assessment

In this section the users can visualize their new FIS and compare the input and output membership functions and rule base to the default (Fig. 3-D1 new FIS, Fig. 3-D2 default). The membership functions and rule base are visualized through plots and the Fuzzy Logic Toolbox Rule Viewer, and Surface Viewer [25]. Once the user has completed one FIS they can click the "Next FIS" button if they would like to develop another FIS or "Finish" to close the GUI. The input fuzzification, aggregation and output defuzzification methods are currently non-configurable in the application. This is to reduce the amount of options to allow the user to focus on the overall understanding of developing a FIS. Common input defuzzification, aggregation and output defuzzification operations were chosen which are min, max and center of area respectively [14].

IV. SPEED CONTROL MODULE

In general, speed control refers to the control of the longitudinal motion of a vehicle through the manipulation of the gas throttle and brake. This includes controlling the vehicle's acceleration, speed and distance from the vehicles ahead. The objective of this module includes: (i) learn how to use various tools to develop a longitudinal speed control and (ii) compare the performance of different control techniques. For this purpose, the module provides a guided user interface that allows the users to develop different speed control systems in an intuitive way. The GUI provides the users the ability to develop their own speed control directly or use various available MATLAB toolboxes such as Control System Designer and PID Tuner [27]. Fig. 6 shows the GUI of this module.

A. Cruise Control approach

A very basic speed control is the standard cruise control (SCC) system that sustains a desired speed by using the vehicle's brake and throttle. Clearly, SCC cannot take vehicle safety into consideration while traveling at the preset velocity; it is up to driver's experience to decide when to reduce the velocity or disengage the cruise control entirely. An advanced version of the SCC system is an adaptive cruise control (ACC) system. This system has two functions: speed control and spacing control. The speed control acts exactly like a SCC system; it maintains the speed of the vehicle at a desired speed. The spacing control modifies the speed of the vehicle to keep a certain distance from a downstream vehicle. The adaptation involves the switching procedure between the speed and spacing control. The ACC system uses additional sensors such as a camera or a radar to measure the distance to downstream vehicles. In the case of far or no downstream vehicle, the ACC switches to the SCC mode. In the case of a close downstream vehicle, the ACC switches to the spacing control [28]. In the simulation, the sensor block uses the information received from an on-board sensor system, such as vehicle velocity and distance to the downstream vehicle, and feeds it back to the ACC.

There are different approaches to design an ACC system [29]. A typical ACC system architecture is a two-tier architecture; an upper and a lower level controller (Fig. 4). The goal of the upper level controller is to generate a desired acceleration from a speed set-point, while the lower controller's goal is to follow the desired acceleration through manipulation of the throttle.

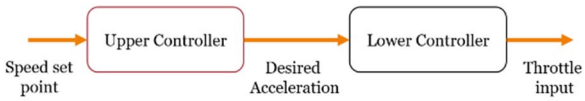

Fig. 4. An illustration indicates the ACC approach composed of the upper and lower-level loops

In this module, the procedure to design an ACC system is divided into three parts: (i) vehicle dynamics, (ii) upper level control and (iii) lower level control. Dividing the ACC into parts simplifies the design procedure and enhances the user's learning. In each subsection, the user can modify the default parameters and learn their effect on the vehicle performance and their influence on the overall system.

B. Vehicle Dynamics

In this teaching tool, a simple longitudinal vehicle model is considered [30]. It consists of the vehicle dynamics and ignores the powertrain dynamics. When driving on a flat road, the vehicle experiences several external longitudinal forces; namely tire forces, rolling friction and aerodynamic forces as shown in Fig. 5. Hence, the vehicle dynamics are expressed as follows:

$$m\ddot{x} = F_t - R_t - F_{drag} \tag{2}$$

where F_t is the longitudinal traction forces acting upon the tires from the ground and the primary force moving the vehicle forward in newtons [N], R_t is the force from the tire rolling resistance proportional to the normal force, tire coefficient and brake coefficient in newtons [N], F_{drag} is the equivalent longitudinal aerodynamic drag force proportional to the aerodynamic drag coefficient and the vehicle velocity squared in newtons [N] and m is the vehicle mass in kilograms [kg].

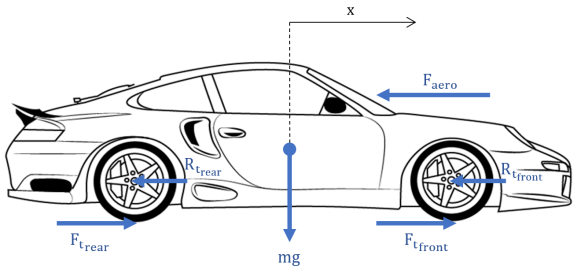

Fig. 5. An illustration indicates the longitudinal forces acting on a vehicle moving on a flat road

Through the Car Parameters section in the GUI (Fig. 6), the user can modify the default parameters such as the mass and the aerodynamic, tire and brake friction gains. In addition, the user can set maximum and minimum force thresholds that are generated by the car's engine.

C. Upper Level Control

The main objective of the upper level control is to make the actual vehicle speed converge to the desired speed set by the driver or, in case of the teaching tool, the decision-making system. The upper level control section assumes a simple dynamic model which is essentially a first order lag used to ensure that the lower level controller has enough time to track the desired acceleration. It is expressed as follows:

$$a = \frac{1}{\tau s + 1} a_d \tag{3}$$

where the value of a is the vehicle's longitudinal acceleration in meters per second [m/s^2], the value of τ is a designed time in seconds [s] and the value of a_d is the desired acceleration and considered as the control input.

The upper level has some desirable performance specifications including fast response, zero steady state error and a small overshoot. The user can begin the design procedure by adjusting the time constant and acceleration threshold to understand their influence on the control system. Then, the user can specify different control systems [30], including proportional-integral (PI) controller, lead-lag controller or pole placement method. The GUI provides the user the ability to tune the control parameters to achieve the desirable control performance. In addition, the user can check the control response by comparing the open and closed loop responses.

D. Lower Level Control

The objective of the lower level control is to track the desired acceleration determined by the upper level control, where a simplified vehicle dynamics model is used as the system model. In this level, the controller generates the vehicle dynamics by generating a proper throttle and brake signal. The users have the option to select different control technique to design the control system and meet the required performances; namely a simple proportional control or an adaptive control. In the last decade, adaptive control systems have become more popular in wide range of autonomous vehicles [31]. The GUI gives the users the opportunity to experience such a comprehensive control system and compare its performance with the traditional proportional controller. Users can then modify the parameters of the selected control systems such as propositional gain, mass initial guess and estimation gain.

E. Cruise Control Performance

To ensure a reasonable control performance, each modified parameter in the GUI has a maximum and a minimum value. The "Check Parameters" button validates the user's parameters and informs the users by a "pop-up message" whether the parameters are within their allowable ranges. The "Load Default Parameters" buttons load the default vehicle parameters and control systems which the user can use and modify. Finally, users can check the overall performance of their ACC system using the "Run Simulation" button. This will run the user's designed ACC system and generate the system performance and compare it against the default ACC system.

V. SIMULATION MODULE

In order to provide the user with a method to analyze and visualize the interaction of the decision-making and speed control modules a car-following simulation was built in MATLAB's 3D World Simulator. After the user has completed the vision-based perception, decision-making, and control modules the parameters are loaded into the Simulink model using the main GUI screen.

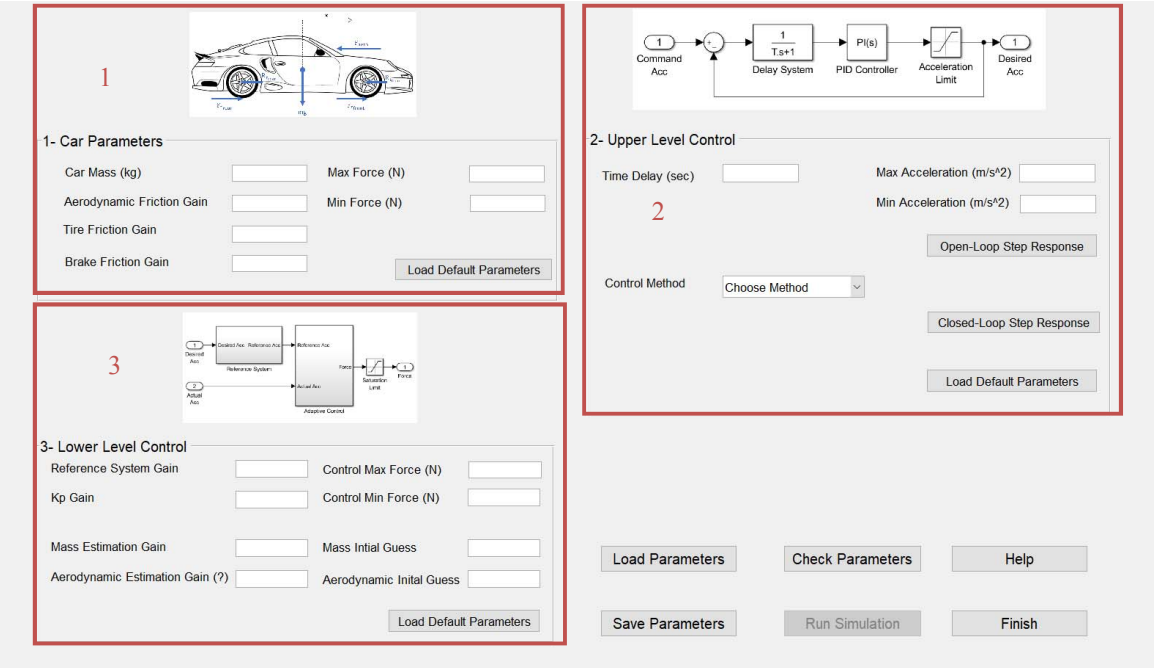

Fig. 6. The Speed Control Module's GUI divided into three sections: Car Parameters, Upper Level Control and Lower Level Control.

As discussed previously, the vision-based perception module allows input from the simulation in the form of image classifier training data to enhance user experience with general image operations. The on-board sensor system providing the velocity and distance information is simulated in Simulink. The user has the flexibility to load default parameters, their own parameters or a combination.

To run the simulation the user clicks the "Run Simulation" button which loads the Simulink program and running the Simulink program will open the autonomous vehicle longitudinal control simulation (Fig. 7). The 3D simulation allows the users to visually see the impacts of their design choices through the vehicles driving patterns such as approaching too quickly, slowly or crashing into the car in front. From this simulation, the user can also generate different viewpoints of either the world or the car in front, to use them as inputs for the vision-based perception module. In addition, the velocity command, speed limit, ego-vehicle velocity, lead vehicle velocity, driving warning and relative distance are graphically visualized using a scope in Simulink (Fig. 8). This allows the user to graphically compare results based on their design choices.

VI. IMPLEMENTATION

Last fall the proposed work was presented in a fourth-year engineering control course for the purpose of quality improvement and assessment. The MATLAB files and instruction manuals required for the application were provided to the students in a zipped file before the session. Instructions on how to download and start the program were given verbally and shown on the classroom projector screen. The undergraduate students were provided with a brief introduction of each module along with the module manual. Due to time constraints, the students were split into three groups and asked to each complete one teaching tool module and provide feedback for quality improvement and assessment.

After the students completed their assigned module they were asked the following questions for quality improvement of the application:

- What section(s) of the app did you enjoy the most?
- What section(s) of the app did you find difficult?
- Was the app user friendly?
- Was the flow of the app clear?
- Do you think this app has helped you understand the concept?
- Was the instruction manual clear and concise?
- Was there any topic that was not explained clearly in the manual?
- Were you able to identify the role of the section for autonomous vehicle applications? If yes, briefly explain the role of the section. If no, what could be improved to clarify the role of this section?
- Do you have any suggestions to improve the app?

Some of the comments and suggestions from the students included:

- Provide detailed instructions on the MATLAB version and toolboxes required for the application, how to download the main application, where to save the downloaded files, and clear steps in starting the application.
- Some students experienced issues downloading and running the program with a MAC operating system.
- Students reported that the GUI windows were not automatically sizing to their computer screen and suggested including the ability to resize the window manually.
- Due to the time limits in the class each module was briefly introduced. Some students recommended that we include more background information about each module.
- Develop a tutorial section in the instruction manuals for the user to follow along when developing their first simulation.
- Integrate troubleshooting tips into the GUI to assist with common errors.

This provided the authors with comments to improve the teaching tool for a formal scientific evaluation in the future.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, the authors proposed a teaching tool solution for the rapid expansion of technology related to autonomous vehicles through four modules namely vision-based perception, decision-making, speed control, and simulation. The vision-based perception module allows the user to (i) visualize the effect of different image processing techniques and edge detection methods and (ii) calculate features of a given training set of two cars from simulated viewpoints. The decision-making module introduces users to the concept and implementation of fuzzy logic to (i) develop a driver warning model and (ii) calculate the velocity command for the ego vehicle. The speed control module introduces users to the concept of designing a control system to maintain the vehicle speed to a set value calculated by the decision-making module. The proposed teaching tool provides a platform for users to first learn individual modules and then visualize the integrated modules through simulation using MATLAB's 3D World Simulator. The anticipated learning outcomes are for students to gain an overall understanding of the three AV technology components through the process of modeling and integration using an autonomous vehicle longitudinal control simulation. Future work includes the implementation of a formal study to gain a greater understanding of how the application improves learning through integration of three engineering applications.

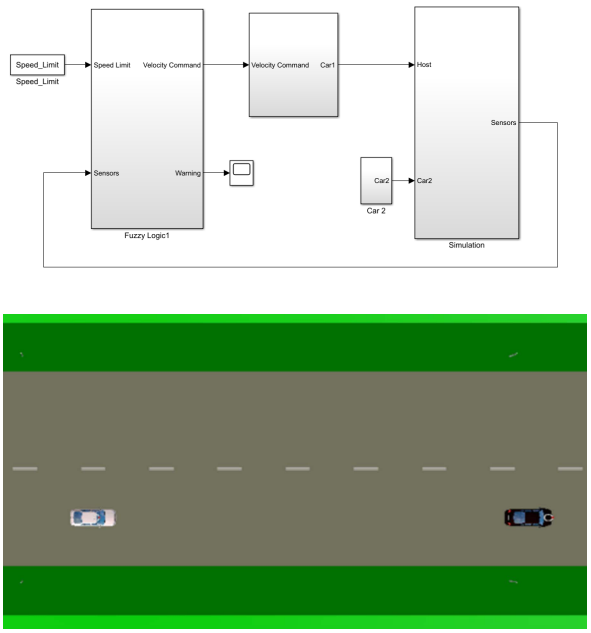

Fig. 7. The top image presents the Simulink model which integrates the decision-making and speed control modules with the simulation. The bottom image is a top view of the autonomous vehicle longitudinal control simulation.

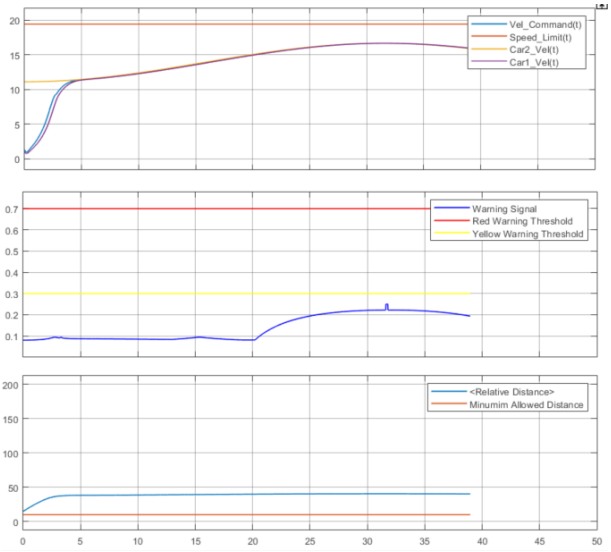

Fig. 8. Scope output from the Simulink model. The top plot shows the velocity command (blue), speed limit (red), ego-vehicle's velocity (purple) and the lead vehicle's velocity (yellow) in meters per second. The middle graph shows the warning value (blue), the red warning threshold (red) and yellow warning threshold (yellow) from 0-1. The bottom plot shows the ego-vehicle's relative distance (blue) and minimum allowable distance (red) in meters.

ACKNOWLEDGMENT

Funding for this study was provided by the Natural Sciences and Engineering Research Council (NSERC) Canada under the Discovery Grant Program.

REFERENCES

[1] V. Gill, B. Kirk, P. Godsmark, and B. Flemming, "Automated vehicles: the coming of the next disruptive technology," The Conference Board of Canada, Ottawa, 2015.

[2] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," Transp. Res. Part C Emerg. Technol., vol. 89, pp. 384–406, Apr. 2018.

[3] H. Cheng, Autonomous Intelligent Vehicles. London: Springer London, 2011.

[4] MATLAB R2018a. Natick, MA, USA: The MathWorks®, Inc.

[5] A. Gil, O. Reinoso, J. M. Marin, L. Paya, and J. Ruiz, "Development and deployment of a new robotics toolbox for education," Comput. Appl. Eng. Educ., vol. 23, no. 3, pp. 443–454, May 2015.

[6] C. A. Bodnar, D. Anastasio, J. A. Enszer, and D. D. Burkey, "Engineers at play: games as teaching tools for undergraduate engineering students," J. Eng. Educ., vol. 105, no. 1, pp. 147–200, Jan. 2016.

[7] J. M. Ramirez-Cortes, V. Alarcon-Aquino, P. Gomez-Gil, A. Diaz-Mendez, M. Ibarra-Bonilla, and I. García-Enriquez, "Interactive educational tool for compensators design in MATLAB® using frequency response analysis," Computer. Appl. Eng. Educ., vol. 22, no. 4, pp. 699–707, Dec. 2014.

[8] K. Nikolaou and D. Pitilakis, "SoFA: A matlab-based educational software for the shallow foundation analysis and design," Comput. Appl. Eng. Educ., vol. 25, no. 2, pp. 214–221, Mar. 2017.

[9] C. V. Schwarz and B. Y. White, "Metamodeling knowledge: developing students' understanding of scientific modeling," Cogn. Instr., vol. 23, no. 2, pp. 165–205, Jun. 2005.

[10] W. Feurzeig and N. Roberts, Eds., Modeling and Simulation in Science and Mathematics Education. New York, NY: Springer New York, 1999.

[11] B. Ranft and C. Stiller, "The role of machine vision for intelligent vehicles," IEEE Trans. Intell. Veh., vol. 1, no. 1, pp. 8–19, Mar. 2016.

[12] "Statistics and Machine Learning (TM) User Gude (R2018a)," The MathWorks®, Inc, Natick, MA, USA, 2018.

[13] M. Brackstone and M. McDonald, "Car-following: a historical review," Transp. Res. Part F Traffic Psychol. Behav., vol. 2, no. 4, pp. 181–196, Dec. 1999.

[14] R. Yager and D. Filev, Essentials of Fuzzy Modeling and Control. John Wiley & Sons, Inc., 1994.

[15] L. A. Zadeh, "Fuzzy sets," Inf. Control, vol. 8, no. 3, pp. 338–353, 1965.

[16] S. Barro, M. Roque, and editors, Fuzzy Logic in Medicine, vol. 83. Physica, 2013.

[17] T. Azim, M. A. Jaffar, and A. M. Mirza, "Fully automated real time fatigue detection of drivers through Fuzzy Expert Systems," Appl. Soft Comput., vol. 18, pp. 25–38, May 2014.

[18] J. P. Rastelli and M. S. Peñas, "Fuzzy logic steering control of autonomous vehicles inside roundabouts," Appl. Soft Comput., vol. 35, pp. 662–669, Oct. 2015.

[19] B. Erginer and E. Altuğ, "Design and implementation of a hybrid fuzzy logic controller for a quadrotor VTOL vehicle," Int. J. Control Autom. Syst., vol. 10, no. 1, pp. 61–70, Feb. 2012.

[20] S. Kikuchi and P. Chakroborty, "Car-following model based on fuzzy inference system," Transp. Res. Rec., pp. 82–82, 1992.

[21] R. Holve, P. Protzel, J. Bernasch, and K. Naab, "Adaptive fuzzy control for driver assistance in car-following," Proc 3rd EUFIT, pp. 1149–1153, 1995.

[22] J. E. Naranjo, C. Gonzalez, J. Reviejo, R. Garcia, and T. de Pedro, "Adaptive fuzzy control for inter-vehicle gap keeping," IEEE Trans. Intell. Transp. Syst., vol. 4, no. 3, pp. 132–142, Sep. 2003.

[23] A. Ghaffari, A. Khodayari, R. Braunstingl, F. Alimardani, and R. Kazemi, "Improved adaptive neuro fuzzy inference system car-following behaviour model based on the driver–vehicle delay," IET Intell. Transp. Syst., vol. 8, no. 4, pp. 323–332, Jun. 2014.

[24] L. Li, S. Yang, and W. J. Cao, "Driver's speed decision-making model based on ANFIS," Appl. Mech. Mater., vol. 488–489, pp. 955–960, Jan. 2014.

[25] "Fuzzy Logic Toolbox (TM) User's Guide (R2018a)," The MathWorks®, Inc, Natick, MA, USA, 2018.

[26] Learn to drive smart: Your guide to driving safely. North Vancouver, B.C., Canada: Insurance Corporation of British Columbia (ICBC), 2015.

[27] "Control System Toolbox (TM) User's Guide R2018a," The MathWorks ®, Inc, Natick, MA, USA, 2018.

[28] G. Marsden, M. McDonald, and M. Brackstone, "Towards an understanding of adaptive cruise control," Transp. Res. Part C Emerg. Technol., vol. 9, no. 1, pp. 33–51, 2001.

[29] L. Xiao and F. Gao, "A comprehensive review of the development of adaptive cruise control systems," Veh. Syst. Dyn., vol. 48, no. 10, pp. 1167–1192, 2010.

[30] R. Rajamani, Vehicle Dynamics and Control, 2nd ed. Springer, 2012.

[31] B. J. Emran, J. Dias, L. Seneviratne, and G. Cai, "Robust adaptive control design for quadcopter payload add and drop applications," in Chinese Control Conference, CCC, 2015, vol. 2015–Septe, no. July, pp. 3252–3257.