Development and Verification of Collision Avoidance and Obstacle Dodge Algorithm for Autonomous Vehicles

Wen-Han Lu
National Taiwan University
Taipei, Taiwan
R02522503@ntu.edu.tw

Cheng-Ho Li
Industrial Technology Research Institute
Hsinchu, Taiwan
ChengHoLee@itri.org.tw

Yu-Yin Peng
Industrial Technology Research Institute
Hsinchu, Taiwan
yypeng@itri.org.tw

Jung-Ho Cheng
National Taiwan University
Taipei, Taiwan
jhcheng@ntu.edu.tw

Abstract: This paper proposes a collision avoidance and obstacle dodge algorithm for an intelligent autonomous driving vehicle. Three stages developments have been put forth in this study. We began with model-in-the-loop (MiL). A dynamic model is constructed for obstacle avoidance control which is implemented on the CarSim software platform for simulation of vehicle dynamics analysis and verification. Then a control-in-the-loop (CIL) process is adopted to simulate the controller with the algorithms developed in this study. Using the test results, the control parameters can be tuned and optimized. All of the software and hardware are integrated and verified on a hardware-in-the-loop (HiL) platform before actual implementation on the real vehicle. Finally, a three-wheeled motorcycle is equipped with the necessary sensors, actuators and controllers for autonomous accelerating, braking and steering control system. The vehicle has been successfully tested and verified to fulfill some preliminarily designated traffic scenarios.

Keywords: Autonomous Vehicle, Obstacle Avoidance, Active Braking, Model in the Loop, Control in the Loop, Hardware in the Loop

1 Introduction

Autonomous driving has become an important branch of the development of intelligent vehicles, but there are still many problems on control need be overcome, such as autonomous vehicle speed control, steering control, obstacle avoidance control and vehicle stability control. Self-driving vehicles require more considerations on driving condition and situation judgment to guarantee the safety. Therefore a control strategy to ensure driving safety must be established.

In recent years, the use of obstacle avoidance technology was investigated for small wheel robots [1] and self-driving vehicles. Although they cannot completely replace human drivers, if we can develop obstacle avoidance control strategy to be used on general vehicles, the developed technologies can be used as a driver assistance system for the driver to avoid dangers due to lack of attention to the road condition. The obstacle avoidance system involved in vehicle operation to reduce the occurrence of accidents will be a huge contribution to human society.

In fact, the process of overtaking is very similar to dodge obstacles, but in the process of overtaking, the obstacle may be moving. So in this case we need to consider the relative speed between two vehicles. Reference [2] provides a method to assess overtaking vehicles using Rendezvous-Guidance Law, while finding the best path to the target vehicle to avoid collision. There are also studies [3-4] which use model predictive control (MPC) in path planning and obstacle avoidance strategies. The advantages of MPC are robustness and more suitable for non-linear dynamic control, but compared to the linguistic control methodology, MPC is more complex.

From the references [5-8], we can find that the fuzzy logic control (FLC) method is a robust algorithm and can be used on real vehicle control. FLC is a linguistic algorithm, so the programmer is easy to use language to define the strategy, and has better adaptability, robustness and fault tolerance.

Based on the review of existing methods, Peng et al. [7] proposed three modes of system operation and Njah et al. [8] proposed a two-layer control algorithm that will be used for the establishment of control algorithm in this research as a basic design.

This paper is structured as follows. In Section 2 the overall system structures are developed. Section 3 details the controller structure for obstacle avoidance. In Section 4, we present the simulation results showing the behaviour of the proposed controller. We then demonstrate the CIL and HiL simulations as well as the experimental results on real vehicle in Section 5. Finally, Section 6 provides some concluding remarks and future work.

2 Overall Structure

The most important thing of this research topic for autonomous driving vehicle obstacle avoidance control is to establish a framework for the overall control strategy. By reviewing previous studies of autonomous driving control for vehicle through the research results, we sort out suitable strategies used in this study. The proposed control strategies are shown in Figure 1, which contains several important substructures such as controllers, actuators, sensors, and the vehicle dynamics model. In this study, a tricycle AEON 3D with two front steering wheels and one rear driving wheel is used to implement and validate the developed obstacle avoidance algorithm. Figure 2 shows the relation among the sensor, controller and actuator as a MATLAB/Simulink model. The actuator is responsible for commands sent from controller transmitting to the steering mechanism, accelerator pedal, and brake pedal of the vehicle to give virtual model control commands and return sensor feedback information to the controller. The controller determines the current road conditions to determine whether the vehicle should turn to avoid obstacles encountered or choose brake operation to stop vehicle because of the insufficient space.
The decision making process in the controller is detailed in Figure 4 in which possible situations of the vehicle state and road condition are considered. First when the sensor detects an obstacle in front of vehicle on the driveway and has no sufficient distance to make a turn action, a braking command to stop the vehicle is then chosen. Even when the obstacle appears too abruptly and the vehicle cannot completely stop, it can still manage to lower the speed of the vehicle so that to avoid a severe accident. The second condition is when the obstacle can be avoided by steering action yet the left lane of driveway does not have enough space for lane change. The controller will choose to brake to stop the vehicle's movement. The third situation is when the left lane allows for executing the steering movements, the vehicle will change lane to avoid the obstacle. After lane changing, there may be two possibilities. First, no obstacle is encountered, the vehicle should move back to the right lane. However, if there is no room on the right, then the vehicle will stay in the same left lane until the circumstance changes. On the other hand, if a front obstacle is detected, the same action will be taken similar to the first and second situations discussed before. The controller will eventually guide the vehicle to return to the right lane and the decision cycle starts all over again.

The above flowchart is implemented in Figure 5 on a Simulink software platform. In the above decision making process, the most critical judgment of whether to stop the vehicle or dodge the obstacle is based on the vehicle's stopping distance at different speeds. In this study, CarSim software is adopted to construct the vehicle dynamics model and a look-up table of braking distances at various vehicle speeds is generated from simulation.

The vehicle and virtual road models for simulation are shown in Figure 6 and 7. Since there is only a vehicle model with one front wheel and two rear wheels in the CarSim software, we take a four-wheel vehicle model and change it into our tricycle, which is driven by a rear wheel and steered with the two front wheels. The basic specifications of the vehicle are included in Figure 8. The road model is built as a long straight dry asphalt surface with two drive lanes. A total width is eight meters (a standard construction in Taiwan urban area) and the road surface friction coefficient is 0.85. In this case, the obstacle is assumed to be a stationary traffic cone. The resulting look-up table of stopping distance versus vehicle speed from CarSim simulation is given in Table 1.

<table>
<thead>
<tr>
<th>Speed (km/h)</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITRI Model</td>
<td>2.7</td>
<td>4.0</td>
<td>5.6</td>
<td>7.4</td>
<td>9.4</td>
<td>11.6</td>
</tr>
<tr>
<td>Vehicle</td>
<td>3</td>
<td>-</td>
<td>6</td>
<td>-</td>
<td>11</td>
<td>-</td>
</tr>
<tr>
<td>Scooter</td>
<td>-</td>
<td>-</td>
<td>4.5</td>
<td>-</td>
<td>8</td>
<td>-</td>
</tr>
</tbody>
</table>
Controller for Obstacle Avoidance and Path Planning

This section will detail the steering path planning for lane change, which consists of three steps: (1) For a given speed, set a tentative path. In CarSim, we design a smooth turning path and then let the vehicle follow the designated path. (2) Observe CarSim output data and plot diagrams from simulation. Care need be taken so that excessive tire sideslip angle will not occur to cause the vehicle to rollover. In this study, the angle is chosen to be less than four degrees [9]. We go back to step (1) and re-plan a new path until a suitable one is obtained. (3) Output the steering angle of the tire and repeat the process for various vehicle speeds. The appropriate paths data are stored as MATLAB matrices, one for each speed, which can be used as a command set for further study. The above process uses the Lane-Change simulation in CarSim software.

Figure 9 demonstrates CarSim path planning configuration interface. By inputting the point coordinates, CarSim can automatically turn them into a smooth curve, shown as the red curve in the figure. Following the above lane change simulation procedure, a typical tire angle versus time diagram at speed 45 km/h can be obtained as shown in Figure 10.

Figure 11 contains the information of reaction distance at different vehicle speeds of our target tricycle. This data set can be used as steering and braking commands (command set) at different circumstances for obstacle avoidance.

Simulation Results

The vehicle model and road model implemented in the CarSim software are transferred to MATLAB/Simulink.
platform, which as shown in Figure 12 can then be integrated with the controller including scenario judgment sequence of Figure 4 as well as steering and braking commands (path planning instruction set shown in Figure 11) together with the calculated model parameters to constitute a complete model for simulating obstacle avoidance function of a three-wheeled vehicle.

![Figure 12: The vehicle model (on the right) with controller (on the left) in MATLAB/Simulink platform](image)

Figure 13 illustrates the sequential diagram for verifying the correctness of steering and braking commands shown in Figure 11. To execute a lane change event, the controller sends a steering command (steering angle) from the path planning instruction set to CarSim, which generates a vehicle driving path and an equivalent steering angle curve, which can then be used to compare with the input steering command (such as the one shown in Figure 10). Figure 14 depicts a typical comparison result. The four curves in the figure are the input and output curves of two front wheels steering command. The close overlapping of the curves indicates that the dynamic behaviour of the vehicle tires is in accordance with the input command. The above verification process is repeated for each of the steering commands in Figure 11.

![Figure 13: Steering command verification flowchart](image)

Utilizing the vehicle model and controller established in Figure 12 together with the proven steering commands of Figure 11, we can now verify the proposed control strategies for various scenarios. For example, suppose that there is an obstacle in front of the vehicle 40 meters away and the vehicle speed is 45 km/h. The controller obtains the signal from the sensors and judges that the vehicle has no sufficient space to dodge the obstacle; the brake control command is issued. The vehicle dynamics model implemented in CarSim computes the brake response shown in Figure 15. It can be seen that vehicle has stopped to avoid hitting the obstacle in front of it (less than 40 meters). The vehicle model can also be used to tune brake torque control parameters via PID control, giving the required torque-time curve shown in Figure 16, which has resulted in the vehicle standing still 3 meters ahead of the obstacle.

We can also verify the context of vehicle steering to avoid obstacles. The results given in Figure 17 are the (tire) steering angles of the case assuming that the vehicle travelling at 45 km/h encounters an obstacle 40 meters ahead. In this case, the controller, by using the information of current speed and the distance to the obstacle sent from the sensors, judges that there is space on left lane and sends steering commands to the vehicle dynamics model. Figure 18 shows the resulting lateral position of the vehicle calculated from Simulink. At 7 seconds after the simulation, the obstacle is detected. The vehicle starts to change lane by moving 3.8 meters to the left in 1.5 seconds. It overshoots for about 0.3 meter and tries to stabilize in this lane. The vehicle then successfully returns to the original lane when the situation calls for sufficient space there.
Experiment Results

In order to verify the proposed control strategies and controllers for autonomous vehicles, we implement a simulated driving and braking system on a hardware-in-the-loop (HiL) platform, on which we initiate controller-in-the-loop (CIL) process. The CIL simulation is carried out by combining the Simulink model with dSPACE systems, CarSim vehicle dynamics software and RT-LAB as shown in Figure 19. The RT-LAB, fully integrated with MATLAB/Simulink, is a real-time simulation software environment that has simplified the way model-based design is performed [10]. First, we compile the Simulink model of the controller onto the MicroAutoBox through dSPACE real time toolbox in MATLAB/Simulink as depicted in Figure 20. Then we connect the RT-LAB system, CarSim vehicle model and the controller model in MicroAutoBox via Ethernet and CAN-BUS.

Many different scenarios have been verified using this CIL platform. In the following, only one will be described in details. The scenario is to issue a brake command to stop the vehicle from colliding with obstacles 40 meters in front of the vehicle. The results of the CIL simulation are illustrated in Figures 21, 22 and 23, which show the variations of vehicle speed, vehicle longitudinal position and vehicle brake torque respectively during the braking process. We can see from Figure 22 that the vehicle successfully stops behind the obstacle within two meters.
With the control algorithm and parameters on the MicroAutoBox being verified, this is then used as a controller, and is incorporated into an experimental tricycle as shown in Figure 24. The experimental vehicle is designed to be capable of autonomous driving. It is equipped with an ibeo LIDAR, throttle actuator and linear brake actuator, which are connected to the MicroAutoBox, are shown in Figure 25.

The setup of active brake experiment is shown in Figure 26. The test scenario is that there is a static obstacle (traffic cone) in front of the vehicle without sufficient space to dodge it. So the motion controller issues a brake command to the actuator to stop the vehicle from collision. The $X_1$, $X_2$ and $X_3$ given in Figure 26 represent the locations that the vehicle start to brake, the end of brake and the location of obstacle, respectively. Moreover, $V$, $S$ and $d$ represent the vehicle speed, the brake distance and the distance between vehicle and obstacle when the vehicle stops.

We conduct this experiment with different vehicle speeds and the result is shown in Table 2. From the table, we can see that the value of $S$ proportionally increases as the vehicle speed increases whereas the value $d$ decreases. Although we can avoid the collision now, for the future work, we will need to tune the controller PID parameter or to reduce the reaction time of linear brake actuator in order to satisfy the function of active brake system with higher vehicle speeds.
6 Conclusions

This study described a collision avoidance and obstacle dodge algorithm for an intelligent autonomous driving vehicle. We began with the establishment of vehicle and road models and obstacle avoidance strategies for autonomous vehicles. All of the models and algorithms were implemented in CarSim, MATLAB/Simulink, RT-LAB and dSPACE platform. The models and strategies were validated both in the MiL and CiL simulations. It was shown that the virtual vehicle with dynamics model not only successfully avoids obstacles or brakes to stop without collision, but also can be used to optimize the control strategies or to fine tune the parameters. After the strategies were verified with MiL and CiL simulations, we then implemented the control strategies in a tricycle equipped with active braking control system to test the proposed control strategies. The vehicle has been successfully tested and verified to satisfy the function of active brake in some designated traffic scenarios.

Acknowledgment

The authors gratefully acknowledge the financial and technical supports of the Industrial Technology Research Institute of Taiwan under contract No. 104-MSL02.

References


