Experimental platform for autonomous collision avoidance strategies

Xiao Diyin, Fredrik Granum, Johan Gustafsson, Adam Fjeldsted and G. R. Campos

***Abstract*—Comfortability and safety is two of the main aspects when building a modern car. Both aspect can be improved by designing fully automated vehicles. This article is focusing on the problem of collision avoidance concerning fully automatic vehicles. Two decentralized solutions is purposed. To test the solutions an experimental platform is needed, This platform is constructed by designing a simulator called CAS (Collision Avoidance Simulator) which has the capability of interacting with a real test vehicle and run simulations with the test vehicle as hardware in the loop. Both algorithms were simulated without any collisions occurring between vehicles. One of them was implemented in the experimental platform, along with the test vehicle, with promising results. The results indicates that the experimental platform will be a valuable tool in upcoming research.**

I. Introduction

Many car manufacturers today are working on fully automated vehicles. Both to be able to offer a more pleasant driving experience and to improve the traffic safety. Today, driver behaviour is a contributing factor in over 90 percent of all accidents [1]. Hence a huge step in improving traffic safety is to implement proactive safety systems to avoid collisions between vehicles. The technology for implementing such a system is available today, but to be able to implement it in vehicles, rigorous tests are needed which is expensive. That is why the aim of this article is to highlight the possibility of lowering test costs by simulation.

A simulator named CAS (Collision Avoidance Simulator) is created and is later used to set up a *experimental platform* using a real test vehicle as hardware in the loop for verifying functionality of collision avoidance strategies. In the platform, the real and virtual vehicles are able to detect each other and hence react as if a real collision is about to occur. In this way, collision avoidance strategies can be tested both in a cheaper and more efficient way. The platform contains a simulator and a test vehicle as hardware in the loop.

The platform, with use of CAS, is able to make the test vehicle and the virtual vehicles detect each other and apply a decentralized strategy for collision avoidance by only controlling the longitudinal movement of the vehicles.

The environment that the platform includes is a three or four way intersection on flat ground. Vehicles interacting with the platform are assumed to be fully automated with predefined paths. The article also highlights proposed strategies for collision avoidance, these have been implemented and tested and is used to both show the capability of the platform and to function as groundwork for a future solutions.

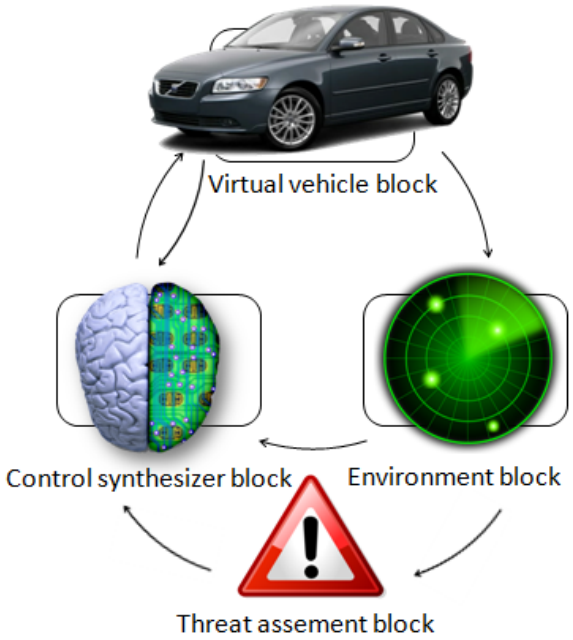

Fig. 1. Simplified flow chart of how the platform works. The different blocks are the virtual vehicle block, the environment block and the controller synthesizer blocks.

II. Platform design

One of the subject of this article is to develop a platform, containing a simulator and a test vehicle as hardware in the loop, that can be used to verify collision avoidance strategies. To keep the efficiency at a high level and increase the usability, the simulator is built in separately upgradeable modules. It is possible to switch out one of these blocks to achieve new functionality. The following sections will attempt to explain what the specification of the platform is, how the simulator is interfaced with a test vehicle, the general structure of the simulator and some in depth descriptions of how the different blocks work.

A. Interface with real vehicle

The virtual vehicles in CAS are developed to behave in a similar way as the test vehicle, hence it is easy to go from using the simulator to using the experimental platform. The test vehicle is equipped with a real time computer and the communication between the simulator and the test vehicle is done using *User Datagram Protocol* (UDP). CAS uses the *Real Time Windows Target* [4] toolbox in Simulink to simulate

real time. To make sure that the communication is performed with minimal delay it is important that CAS reads data faster than it is being sent from the test vehicle, so CAS will always work with the most recent data. Reading at 25 Hz and the sending data between the test vehicle and the simulator at 10 Hz produces an average round time delay of about 100 ms. This problems occurs due to the fact that personal computers are incapable of running in real time and the simulator, when connected to a test vehicle, is forced to run at real time.

B. General structure of the CAS

The main blocks in the platform is the virtual vehicle blocks, the environment block, the controller synthesizer blocks and the threat assessment block. An illustration of the system is given in Figure 1.

Each virtual vehicle block includes a model of the vehicle's dynamics, sensors and broadcasting module. The input to the block containing the vehicle dynamics is reference acceleration and it outputs the speed. This speed is used to position the car in the simulation according to it's current distance from the intersection generate the cars *Global Position System* (GPS) information.

Let s(t) denote the vehicles current distance from the center point of the intersection and i stands for the i_{th} vehicle data, then the GPS sensor block does the following

$$\begin{bmatrix} x_i(t) & y_i(t) & \theta_i(t) \end{bmatrix} = GPS_{Block}(s_i(t)) \qquad (1)$$

The signal is built up in a way so that the package starts with the vehicle's *identification*(ID) followed by position, heading speed, maximum WiFi range, intended path on the form of Left turn = 1, straight = 2, right turn = 3, the vehicle's width, length and weight and lastly the vehicles maximum deceleration and acceleration. The signal components can be seen in Table I.

TABLE I
SHOWING SIGNAL COMPONENTS THAT ALL VEHICLES BROADCAST

ID	x	y	θ	Speed	WiFi Range
Path	Width	Length	Weight	Max DeAcc.	Max Acc.

More data can be added by the user to this signal inside CAS if needed. This signal is broadcast to all vehicles where in each one is a block that determines which part of the signal, i.e the detected vehicles, it has access to. This is the signal that the control synthesizer has access to and uses to make informed decisions.

1) Graphical interface: To input data to the simulator, and representing the result coming out of it, a visual interface is included. The interface is designed with MatLab, visualized in Figure 2, and helps the user to reach all the settings and data from previous simulation. To setup a new simulation the user chooses the preferred scenario in the tab *Simulation Settings* and click *Open Excel File*. The available scenarios are a four way intersection *X intersection*, a three way intersection *T intersection* and the possibility to start the vehicles in platoons in either a four or three way intersection. This will bring the user to a customized Excel sheet where the initial condition for all the vehicles can be set up. After setting all the initial conditions the user start the simulation by pressing the *Run Simulation* button. After a simulation is done press the *Playback* button in the bottom of the application menu to show the simulation result, when doing so a pop-up will appear asking the user if you want to store the playback or perform a fast replay. A snapshot from the replay is shown in Figure 3.

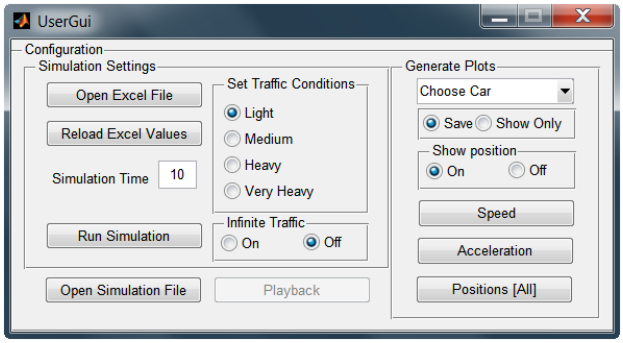

Fig. 2. Picture showing the Application Menu used by the user to set-up, run and evaluate results from simulations.

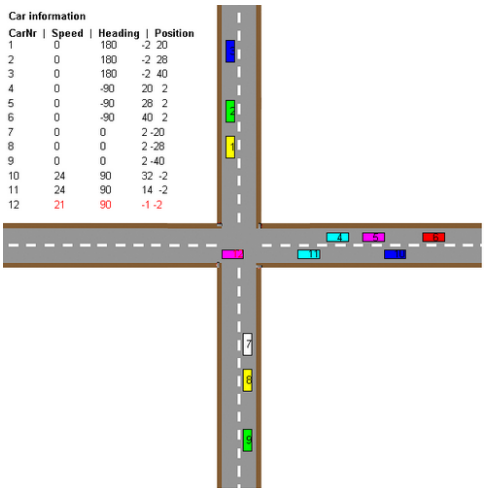

Fig. 3. An example of how a replay of a simulation can look when using the experimental platform for collision avoidance.

In the *Simulation Settings* tab you can also do settings for infinite traffic flow. This will make the vehicles regenerate at a random position once it has left the screen, different traffic densities can also be selected inside of this tab. The last tab *Generate plots*, is for further analysis of the behavior in the simulation, this can be done by setting the platform to generate different plots showing properties like speed, acceleration and position of all vehicles during the simulation.

2) Vehicle model: For the CAS to be reliable, the strategies for collision avoidance tested in CAS will be applicable in real life the dynamics of the virtual vehicles needs to be verified. The dynamics needs to be similar to the test vehicle's

performance at the same time as the models representing the virtual vehicles should not be too computer heavy since the platform needs handle multiple vehicles at the same time. To keep the number of computations to a reasonable level an optimized model for testing vehicles with automatic transmission from Mathworks Example Library is used [2]. This model is extended with a neutral gear and a proportional integration (PI) controller is implemented with reference acceleration as input.

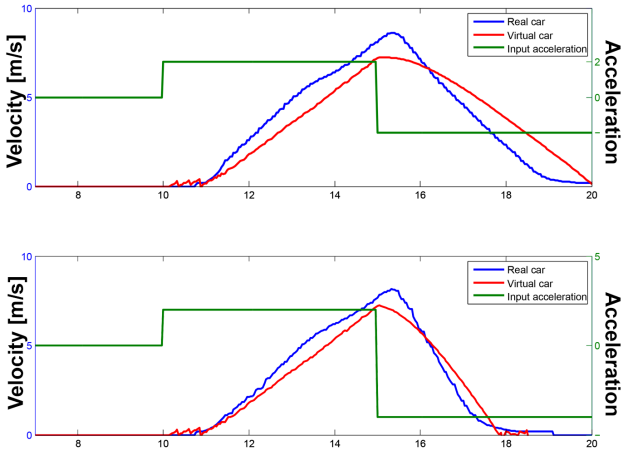

Fig. 4. Shows pulse response of two different acceleration pulses.

To verify the dynamics of the virtual vehicles two different acceleration pulses was tested both in a real vehicle and in the extended model. Figure 4 shows that the model, although reaching a slightly lower top speed, has a similar response as the test vehicle both during acceleration and braking.

Note that the test vehicle is not equipped with fast responding acceleration controller and the virtual vehicles are modeled in a similar way. This makes tuning the controller for use with the test vehicle easier but this needs to tuned to different test vehicles.

3) Information acquisition: For this modular approach, each vehicle has to have adequate information about their own environment and other vehicles. The vehicles broadcasts their sensor data and own information that other vehicles need and at the same time they listen for information from others. Each vehicle communicates their ID, position and other information shown in Table I. In this way the vehicles are able to get a good understanding on the surrounding environment and the controller can make a decision that best avoids collisions based on the sensors and static information from other vehicles.

To simplify the problem all vehicles are simulated with the same or similar sensors and are transmitting it to other vehicles. This could very well prove to be the case in the future if all vehicle manufacturers follow the same standard.

- **GPS :** The GPS is simulated by using the vehicle's coordinates and Gaussian noise with variance and frequency specified by the user can be added to the model. The positions and heading are generated from the GPS signal to retrieve both position and direction of travel.
- **Radar :** In a fully autonomous vehicle it is safe to assume that that they all have radar sensing capabilities to avoid obstacles, with that in mind all vehicles are equipped with a front facing radar to sense cars heading in the same general direction. This radar information is not sent to other vehicles, but is used as a safety precaution since the GPS doesn't always give accurate position. To avoid heavy calculations involving convexes the radar sensor simulation has been simplified. The radar can only measure the distance to vehicles in front that are heading in the same direction. The user can add Gaussian noise to this signal as well.
- **WiFi Module :** Each vehicle is equipped with a WiFi module with a user specified range. This module continuously outputs sensor data which other vehicles can receive via their own WiFi module. The inner workings of a wireless signal transmitter/receiver is not simulated. It is assumed that the vehicles will have no packet loss or interference from other vehicles. This is a valid assumption if the vehicles are using *Self-Organized Time Division Multiple Access*(STDMA) scheduling protocol. When the transmitting and receiving vehicle are within 100m of each other it is safe to assume a very good packet reception probability [3]. The user can specify the frequency of transmissions or even turn of a transmission of cars by setting the transmission range to zero.

4) Threat assessment: The threat assessment block is designed to help the user to evaluate if the vehicle is currently causing or being subject to any threat. It uses the current velocity, preferred deceleration and maximum deceleration to calculate parameters that can help the user to evaluate the current threat. A threat could be that the vehicle will, at a certain point in time, be unable to stop before the intersection limits. These calculated parameters, which are listed below are readily available for use in the control synthesizer.

- **Deceleration :** The needed deceleration such that the vehicle will come to a full stop at the intersection limit
- **Comfortable :** A boolean variable which is high if it is possible to use the users specified comfortable deceleration to stop before the intersection.
- **Uncomfortable :** A boolean variable that has the same function as stated above except using highest possible deceleration.

5) Control synthesizer: The control synthesizer block is where the collision avoidance strategy is implemented. A skeleton is provided for users that filters out available signals and their algorithm can be written in. A memory buffer is also available for saving variables between runs to allow functionalities such as integration, derivation and more.

III. Proposed Control Strategies

Two different control strategies has been implemented. The first algorithm (*Algorithm 1*) is a simplified control strategy

mainly focusing on verifying the functionality of the simulator, and then the second algorithm (*Algorithm 2*) is a more advanced control strategy, more focusing on optimizing the flow through the intersection.

A. Algorithm 1

Algorithm 1 implements *First come - First serve* policy. The first vehicle that reaches the intersection is the one that has highest priority. In general a vehicle compares its own distance to that of other vehicles that are detected.

Let d_0 denote the distance from the intersection of a vehicle determining if it should go through it, D_i the distance of other vehicles that it can detect, where i denotes the i_{th} car and ID_i stores the ID of those detected vehicle. The pseudo code for the algorithm is shown in Algorithm 1.

```
while CheckNow do
    D = getDetected();
    for i = 1:detectedVehicles do
        if d_0 < D(i) then
            WaitFor(i) = ID(i);
        else
            RemoveWaitFor(ID(i));
        end
    end
    if isEmpty(WaitFor) then
        StartDriving();
    else
        WaitAtIntersection() ;
    end
end
```

Algorithm 1: Pseduo code for Algorithm 1

The *CheckNow* parameter is a boolean that is *true* when the vehicle is driving towards an intersection and is at a given distance from it. This ensures that the vehicle only checks for other vehicles when nearing an intersection. Similarly, the *getDetected* function only collects data about cars that are moving towards an intersection. In cases that the distance of the two vehicles is equal they use the ID's of the vehicles, which is never the same, where the higher ID has priority.

B. Algorithm 2

Algorithm 2 takes into account the current speed, distance to intersection and desired turn for each vehicle. In this way, it is possible to optimize the flow of the vehicles based on the momentum of each vehicle and what path the vehicles are traveling.

Several vehicles are allowed to go in the intersection at the same time, since the vehicles broadcast their intended path a set of condition can be defined. Two vehicles coming from opposite direction would be able to go straight, or four vehicles coming from all four different directions would all be able to turn right without any risk of collision.

Traditional traffic rules regulates who has precedence among two vehicles based on a set of criteria. These only takes into account the direction of travel, not the current speed of the vehicles. The advances collision strategy works in a similar way and lets the vehicles, decide independently calculate who has precedence depending on a set of criterias including direction, distance to intersection and speed.

The pseudo-code can be seen below in Algorithm 2. When a vehicle is approaching the intersection, it runs the algorithm and start fetching information about other vehicles in the surrounding. The algorithm goes through all detected vehicles to find out who has a higher priority in the intersection. If a high priority vehicle is found the algorithm checks whether these vehicles are in risk of colliding and makes the required adjustment in the vehicles acceleration. The algorithm calculates the maximum deceleration required to stop before the intersection to allow all vehicles with higher priority than itself to pass. This is done using current velocity V and distance to stop d. In this way, the vehicles starts to reduce their speed a long time before the intersection, making the strategy more environmentally friendly, since the vehicles don't always have to stop and increases the flow in the intersection.

```
while CheckNow do
    D = getDetected();
    for i = 1:detectedVehicles do
        if priority(D(i)) then
            if riskOfCollision(D(i)) then
                refAcc = min(V^2/d, refAcc);
            end
        end
    end
end
```

Algorithm 2: Pseduo code for Algorithm 2

The prediction, $riskOfCollision$, calculates the if there is a risk of collision between the two vehicles by estimating their position forward in time based on current speed and position.

Making the assumption that any vehicle that will turn would need to slow down, it is preferable to let vehicles going straight keep their speed and thus reducing the accelerations and decelerations among the vehicles as a whole.

The function $priority$ favors car going straight and with a higher velocity. As more vehicles are added to the system, the vehicles are deciding the priorities among themselves two and two to build the entire system of all vehicles. Therefore, the function $priority$ is constructed in a way so to avoid circular priorities in the case of more than 2 vehicles. Assuming two vehicles having identical speed, distance to intersection and both want to go straight, both vehicle would need to slow down to avoid a collision. To avoid the deadlock situation that occurs, the vehicle with a higher ID number is allowed precedence, which is the only unique data separating the vehicles.

Before the car enters the intersection, it makes sure that there are no cars in the path and that the exit is free. This is important as there may be cases in which some vehicles do not behave the way they are suppose to.

Unlike the Algorithm 1, in this strategy each vehicle is only keeping track of vehicles with a higher priority. Once there are no other vehicles with a higher priority, the vehicle is free to move on.

IV. Results

Both simulation results and experimental result has been produced. Algorithm 1 has been implemented and tested both in the simulator and in the experimental platform. Algorithm 2 has not been tested with the hardware in the loop, but has been tested in CAS.

A. Simulation results

This sections includes the result from the implementation of both the Algorithm 1 and 2 in the simulator.

1) Algorithm 1: Simulation results of Algorithm 1 can be seen in Figure 5. Five vehicles are approaching intersection at the same time and if the control synthesizer had been disabled then a collision between vehicles 4 and 2 would have occurred as soon as the they enter the intersection area. With the control synthesizer enabled the vehicles get assigned priorities according to Algorithm 1. It can be seen that at around 7 seconds vehicles 3 assigns higher priority to vehicle 2 thus avoiding possible collision. This happens also between vehicles 3 and 1 and shows that the algorithm was definitely able to divert possible collisions.

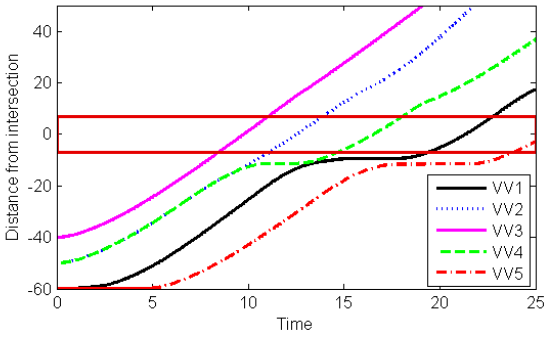

Fig. 5. Five vehicles is approaching the intersection area. Different priorities of passing is assigned to the different vehicles by Algorithm 1. The result is that no vehicle is entering the intersection area (shown with a black square) at the same time as the other vehicles. VV = Virtual Vehicle.

2) Algorithm 2: Algorithm 2 efficiently avoids collisions with other vehicles, at the same time as the flow in the intersection is increased. The advantages of having more vehicles in the intersection at the same makes it possible for the increased flow, and by promoting vehicles with a higher velocity makes the energy usage more efficient as compared with Algorithm 1. As can be seen in figure 6, the vehicles that needs to yield for other vehicles do slow down before, but do not come to a full rest at any point. It can also be seen that the order is different, because the algorithm promotes the vehicles that go straight and can keep their velocity, rather than the vehicles that turn and would need to slow down anyway.

The drawbacks of promoting vehicles with higher velocity is significant in heavy traffic, as the vehicles that are standing still needs to wait for when there are no other vehicles entering the intersection. This could be solved using another parameter, which promotes vehicles with a long waiting period.

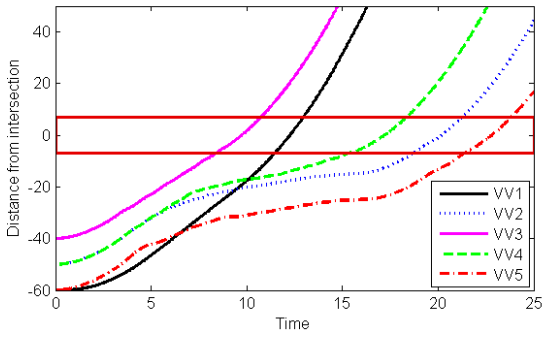

Fig. 6. Results using Algorithm 2 in the same scenario as was used in Figure 5

B. Experimental results

CAS was connected to a modified Volvo S60 using the interface described in section *Interface with real vehicle*(Section II.A) and the Algorithm 1 was tested. The experiment was successfully executed and the S60 was able to avoid collision with virtual cars. The same scenario was implemented as was done in simulation and discussed in *Algorithm 1*(Section III.A).

In all test runs there was a driver behind the wheel for safety and steering of the vehicle. The driver never influenced the speed of the car unless the simulation was over or when failure occurred and thus the speed of the S60 inside simulations were strictly generated from the control synthesizer. A first order transfer function was put on the reference to filter out jittery signal from the controller to the S60 since the on board computer disconnected the speed controller if the signal was to jittery. The first runs with this setup generated collisions

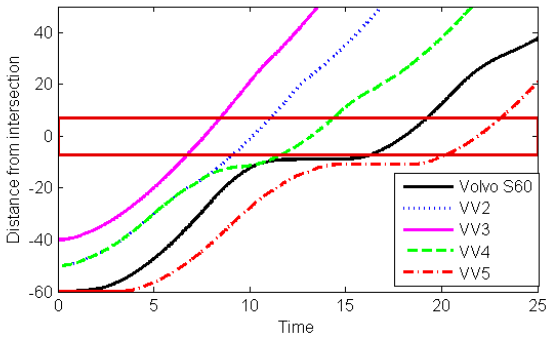

Fig. 7. Same scenario as in *Algorithm 1* where vehicle one has been switched out for the real car. VV = Virtual Vehicle.

between the virtual vehicles and the S60 which the controller was unable to prevent. This was traced to the fact that the speed controller interface in the S60 didn't respond in the same way as the virtual cars did to low decelerations and thus a gain of 1.15 was added to the reference signal from the

controller to the S60. This allowed the controller to influence the real car much better and resulted in consecutive successful runs. The distance traveled in the simulation and on the real road seemed to be close to 1:1 in scale but more accurate measurements are needed.

Figures 7 and Figure 5 both have very similar results and shows that CAS along with a real vehicle can be a valuable and realistic tool in evaluating collision avoidance strategies.

V. Conclusion

Collision avoidance has been a hot topic ever since people started using vehicles. In recent years the technology required to implement proactive fully automatic collision avoidance systems in modern vehicle has been made available. The availability has triggered both the academic world and the industries interest concerning this topic. In this article both new solutions to the problem of collision avoidance and new tools to analyze collision avoidance strategies with has been suggested. The tools suggested in the form of the simulator CAS offers improvements by making the design, the verification and the implementation process more efficient. It also offers gains concerning safety since the strategies can be tested thoroughly in simulation before implemented in a real vehicle and once implemented in a vehicle the simulator can still be used to increase safety and lower costs by providing virtual vehicles instead of using multiple real vehicles. The first proposed solution has been tested both in the simulator and in connection with a test vehicle. The result of both test were successful when no collisions occurred in either case. The control strategy is a bit conservative, but does in a safe way let all the vehicles enter and exit the intersection zone. The second proposed strategy, has only been tested with the simulator but does show promising result. Compared to the first solution Algorithm 2 is more optimized in concern of flow and energy conversion, which is to aspects that are of great importance if the strategy were to be implemented in a larger scale.

Comparing these result with similar work, like the research performed by R. Naumann [6] at the University of Paderborn, the results of this article is more reliable since they have been more extensively tested. R. Naumann's work could greatly benefit from using CAS when verifying his algorithm since it gives him the opportunity to test it in a lot of more cases with continuous traffic. Using CAS can also create benefits for other types of research projects, like the one performed by M. R. Hafner [7] at University of Michigan, where the goal is to prevent collision between two vehicles. In this case CAS capability of going from simulation to real life control of vehicles could have greatly improved the time consumption and the safety of testing.

With the former paragraph implying a great need for a capable experimental platform for collision avoidance strategies and the results shown in this article, it can be concluded that the experimental platform developed has a great potential. It has already been used to successfully implement a collision avoidance strategy in a real vehicle and can hopefully make further research and realization of fully automatic vehicles easier also in the future.

VI. Future work

As the cooperative driving and safety system still remains a hot topic in the vehicle technology field, CAS for testing has a bright future but there is always room for improvement.

At present, CAS only provides single intersection scenario. It would greatly benefit the simulator to have the ability to simulate more than one intersection for example to test traffic flow optimization in parallel with collision avoidance. The vehicle models currently don't include a model of steering which in later releases should be added. More details to the priorities functionalities would be interesting, such as ambulances and cop cars being allowed complete priorities through traffic. A challenging addition would be to include more than one test vehicles at once in the simulation. This would involve much synchronization and needs to be very well done to be usable.

For the most part, implementing CAS in to a xPC Target [4] would allow much more reliable real time simulation and is definitely something that should be done.

VII. Acknowledgements

The authors would like to thank Gabriel Rodrigues de Campos and Robert Hult for amazing patience and help during the projects development. Without their guidance this would not have been possible. Our thanks also goes to Paolo Falcone who supplied us with access to the Volvo S60 for testing.

References

[1] Volvo Cars. *Vision 2020*. [Online]. Available: http://www.volvocars.com/intl/top/about_volvo/corporate/volvo-sustainability/safety/pages/vision-2020.aspx

[2] MathWorks. *Modeling an Automatic Transmission Controller*. [Online]. Available: http://www.mathworks.com/products/simulink/examples.html?file=/products/demos/shipping/simulink/sldemo_autotrans.html#2

[3] Katrin Sjöberg, *Medium Access Control for Vehicular Ad Hoc Networks*. Ph.D. dissertation, Department of Signals and Systems, Chalmers University of Technology, Göteborg. 2013.

[4] Mathworks. (2013, September 19). *Real-Time Windows Target*. [Online]. Available: http://www.mathworks.se/products/rtwt/

[5] R. Naumann, R. Rasche, J. Tacken and C. Tahedl, *Validationand simulation of a decentralized intersection collision avoidance algorithm*. Mechatronic Laboratory and Department of Computer Science, University of Paderborn, Germany. 1997.

[6] R. Naumann, R. Rasche, J. Tacken and C. Tahedl, *Validation and simulation of a decentralized intersection collision avoidance algorithm*. Mechatronic Laboratory and Department of Computer Science, University of Paderborn, Paderborn. 1997.

[7] M. R. Hafner, D. Cunningham, L. Caminiti and D. Del Vecchio, *Automated Vehicle-to-Vehicle Collision Avoidance at Intersections*. Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, Michigan. 2011.