Self-Driving Car Simulator

Ishtiak Zaman, Shrijit Pillai, Tejas Shah
Indiana University Bloomington
107 S Indiana Ave
Bloomington, Indiana 47405

Abstract

The Self-Driving Car is one of the most challenging and interesting topics in the field of Artificial Intelligence today. The problem is challenging as the system operates in a partially observable, continuous and dynamic environment. In such an environment, the system not only has to take care of the driving task but also should be contextually aware of the environment to the maximum possible extent. Our project attempts to simulate the car driving task in such an environment using a hybrid approach to check its feasibility in solving a complex real world problem as quickly as possible.

1. Introduction

The Self-Driving Car has a very rich domain. There are a number of factors that affect the driving process such as the speed of the car, obeying the traffic rules, the position of other vehicles on the road, the speed at which those vehicles are operating and many more. The system is not fully aware of all the factors that affect its operation and many a times it is necessary to respond quickly to avoid any untoward incident. It is because of this reason that the domain is one of the most challenging and intriguing in the field of Artificial Intelligence.

It is interesting to understand the underlying system that performs such complex operations. In our project, we have implemented the system using a hybrid approach of Rule based and Case based systems. The Rule based system handles the basic driving tasks such as stopping on a Red signal or a stop sign, adjusting the speed based on the speed limit sign and handling the lane change. The Case Based system handles the exceptional events such as stopping the car when a pedestrian crosses, taking evasive action upon sensing an emergency such as applying the brakes hard or steering away. The combination of the Rule based and Cased based system integrates the basic driving task with the ability to take accurate and quick decisions.

Our system takes a flat file as input which contains the directions from source to destination along-with some metadata which describes the route. Our system does not make use of any hardware such as the camera or sensors to obtain the state of the environment. To compensate for the lack of computer vision, our system utilizes event generators to generate random events to test the functionality of the system. Our project has two primary goals:-

- Realize the potential of a hybrid system operating in a partially observable, continuous and dynamic environment.
- Evaluate the ability of the system to take accurate and quick decisions in such an environment.

The paper is organized as follows: Section 2 discusses the motivations for our project. Section 3 describes the related work. Section 4 explains the design of our system and the justification of the system design. Section 5 describes the implementation of the system and Section 6 gives an account of the evaluation of our system in terms of performance. Section 7 highlights the future work, Section 8 is the conclusion and Section 9 is the Appendix which briefly describes the output giving an overall picture of the program.

2. Motivation

i. **Reduction in Accidents**
It is estimated that over 33000 people are killed every year in the US alone due to vehicular crashes and over a million people worldwide. Ninety percent of the accidents are due to human error. The Self-Driving Car can reduce the accidents occurring due to human error.

ii. **Environment-Friendly**
The Self-Driving Car can save 10-15% on fuel by operating at optimal speeds thereby saving fuel

iii. **Increased Roadway Capacity**
The cars can maintain a fixed distance between each other thereby increasing the road capacity. An increased road capacity results in more convenient travel and reduction in congestion.

iv. **Increased Productivity**
It is estimated that in the Unites States people who drive cars spent around 157 hours per year driving. The task of driving being taken over by the car, productive work can be done during the travel time such as replying to e-mails, preparing for a meeting, preparing slides for presentation or even take some well deserved nap.

3. Related Work

The paper on Case Base Prediction of Teen Driver Behavior and Skill [1] uses case based techniques to model vehicle control behavior of teens. The paper employed a pure case based reasoning approach to predict the behavior of the driver. The problem of predicting driver behavior is the regression task of predicting the values of steer, throttle and brake for a given situation based on previous experiences of cases. Our system, although hybrid, is similar in the case based reasoning part. Based on the retrieved case, the system adjusts the speed of the car by accelerating or decelerating by an amount which is determined by the severity of the case.

The paper [2] discusses a driver model which models the human driver. The paper describes the sensory limitations, steering control calculation, speed control, path planning and prediction capability of the driver model. Our project accounts for the sensory limitations using the random event generators. The prediction of the possible next state(in case of an exceptional event) and the corresponding action it should perform is handled by the case based reasoning system. The controller component of the system uses the retrieved case to manipulate the speed of the car.

The paper [3] describes a number of near-collision driver behavior models such as avoidance by braking, avoidance by steering and a combination of both. The paper describes the application of these models depending on the situation. For example, if there is a scenario of head-on collision, then the best approach might be to brake as well as steer towards the left or right. This not only reduces the impact of collision(if it occurs) but by moving away from line of impact, the chances of collision are reduced. Some of cases in our case base too follows this approach to avoid a collision and if a collision does happen the severity is reduced due to the reduction in speed.

The paper [4] describes Boss, an autonomous vehicle that uses on-board sensors and a three layered planning system to drive in urban environments. The first layer decides the route to take, the second layer decides the behavior of the car such as when to change lanes and the third layer performs actions to avoid obstacles. This is quite similar to the approach followed by our system. The difference is that our system does not use computer vision; the events are generated randomly to test the system and the case base system handles the functionality of the three layers in Boss.

The paper [5] discusses the techniques to predict the future location and speed of a dynamic object in an urban environment so that handling of exceptional events involving the dynamic object can be performed. Our system performs this future prediction based on the similarity of the retrieved case. The retrieved case produces a plan which allows the system to determine the action to be taken to handle the event.

4. System Design

The Self-Driving Car Simulator that we developed follows a Hybrid approach consisting of the Rule Based and Case Based systems. Such an approach allows for a clear division of tasks between the two systems with the Rule based system handling the basic driving rules and the Case Based system taking care of the exceptional events. Figure 1. shows the architecture of our system.

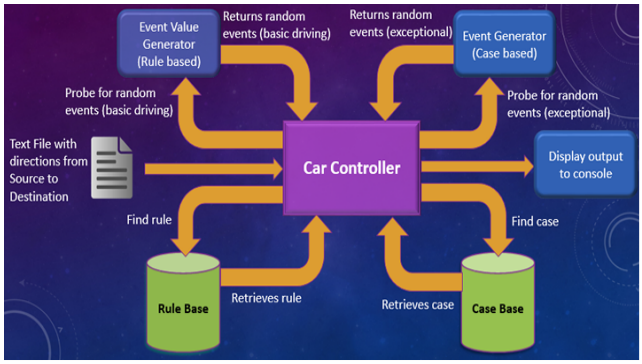

Figure 1: High level system model

The high-level overview of the system is as follows:-
The system starts by taking a text file as input which contains the directions from the source to the destination. The file contains meta-data related to the route such as the number of lanes, whether the lane is single-lane or multi-lane, whether a signal or a stop sign is present at the end of the lane, whether the lane has a speed limit and the lenth of the lane. The event generator generates exceptional events periodically. In case of a random event generated by event generator, the system checks in the case base for similar rules and retrieves the case with a similarity greater than a pre-defined threshold. The actions taken by the system is output to the console.

The components of the system are as follows:-

i. **Car Controller:** The car controller is the core component which drives the system. It processes the lane related information, one lane at a time and co-ordinates the activities between the Rule based and the Case Based system.

ii. **Event Value Generator:** It is named so because it generates values corresponding to the events and not the events. For example, if the signal is visible to the car (which is determined when the car crosses a threshold distance on that lane), the event value generator outputs a random signal value (RED, AMBER or GREEN). If the event value generator is probed again for a value, it outputs the current value based on the previous value. For example, if the previous value output was RED, then the current value will either be RED or GREEN and not AMBER.

iii. **Event Generator:** It generates random exceptional events to test the functionality of the case based system. The event generated by the generator contains information such as the object(pedestrian, vehicle etc.), distance from the car, speed of the object and the direction with respect to the car.

iv. **Rule Base:** It contains the rules to be matched based on the condition.

v. **Case Base:** It contains the pre-loaded cases which will be assessed for similarity with the random case generated by the event generator

It is to be noted that the Controller probes the Event Value Generator for random values corresponding to basic driving events. The data related to these events are available in the lane information. Based on the random values provided by the generator, the rule base is checked and the corresponding action is returned. For exceptional events, the Event Generator generates random cases; the Controller never probes the Event Generator. So when the Event Generator produces the events, the task is directed to the Case Base System.

Rule Based System:

The Rule based system consists of the Controller, the Event Value Generator and the Rule Base. The car controller operates based on the data present in the file. The controller starts the car simulator and for each related lane data in the text file, the controller controls the simulator accordingly. For example, if there is a STOP sign at the end of the lane, the car upon reaching a threshold distance (braking distance), would check the rule base for the corresponding rule and perform the necessary action, in this case, to stop the car. The controller would then probe the event value generator periodically until an "all clear" value is provided by the generator.

Case Based System:

The Case based system consists of the Controller, the Exceptional Event Generator and the Case Base. The exceptional event generator generates events from a random set of events. To avoid cold start problem, the case base is preloaded with cases to start with so that it can handle the random events generated by the event generator. As accurate and quick responses are required in case of an exceptional event, the case based system assesses the similarity between a candidate case and the exceptional event until the first case which is more than 90% similar to the random event is searched. The case base might contain cases with better similarity, however, as quick response is needed, a trade-off between accuracy and retrieval time is employed. If no cases with similarity greater than the minimum threshold of 50% similarity is found, then the system defaults to applying the brakes to stop the car in the minimum possible time. The evaluation section discusses the time taken to search the entire case base and if the car can successfully handle such an exceptional event (an event for which no satisfactorily matching case is found in the case base). Each matching case retrieved from the case base goes to the modifier part of the case based system. The modifier modifies the "priority" and required final speed of the car depending on the current state of the car to the controller. The "priority" is rate of acceleration/deceleration which indicates the degree by which speed of the car should be increased or decreased to handle the event.

Why a Hybrid System?

The main advantage of the hybrid system approach is that there is a clear division of tasks between the rule based and case based systems. This allows the events to be directed towards either the rule based or the case based system depending on whether the events are basic driving rules or exceptional events. The second benefit of this approach is that the search space has been reduced. Had the exceptional events been also included in the rule base, then the number of rules would have increased. As a direct consequence of the reduced search space, the rule/case retrieval time is reduced. The hybrid approach also enabled efficient knowledge representation. Representing the cases as rules would have resulted in a large number of antecendents for the rules and would have also resulted in long chains. The hybrid approach helped to avoid such a scenario. Finally, the approach enables the system to give compelling explanations for its actions.

5. Implementation

The Car simulator has been implemented using the object-oriented methodology. The object-oriented approach allows for modularity and flexibility which is essential in the development of the simulator. The controller, event value generator and the event generator are implemented as objects. The main program starts by reading the input text file and storing the information in the file to be processed by the controller.

```
path = [{Src:'FP',
         Dest:'IMU',
         Dist:0.5,
 event:
 [{turn:'L',move:0.1, lane:2,
   signal:True},
  {turn:'R',move:0.1, lane:2,
   signal:True, sign: 'speed~40'},
  {turn:'', move:0.1, lane:1,
   signal:False, sign:'speed~20'}]}]
```

The data read from the input text file will be stored in the above format. The 'Src' and 'Dest' are the source and destination respectively and the 'Dist' attribute is the total distance between the source and destination. The boolean signal attribute of 'True' indicates the presence of the signal at the end of the lane. A 'False' value indicates a STOP sign. The speed limit sign shows the speed limit of the lane. So speed 40 indicates the speed limit of the lane is 40 kmph. The 'turn' attribute indicates whether the car should take a left ('L'), right ('R') or go straight ('S'). A turn value 'D' indicates that this is the last lane to reach the destination. The 'move' attribute indicates the length of that particular lane.

The default speed of the car is assigned to be 25 kmph or 15 mph and the default sleep time is kept to be 10ms. The sleep time is necessary to display the output to the console in a controlled manner.

Rule Based System

The Rule Based System handles the Lane Change, Speed Limit, Signal and the STOP sign. The Appendix (Section 9) describes the Rule Based System by showing the output of the program. The handling of these rules are explained below in detail:-

Handling Lane Change After the car crosses a particular threshold (65% of lane distance), the controller probes the event value generator to give the status i.e. if it safe to change the lane or not. Based on the value the generator provides, controller searches the rule base and performs the necessary action. The action could be to change the lane, wait or no action required. If the action is to "wait", then the controller again probes the value generator after a set period to give status and this process repeats until the generator gives an "all clear" value or the signal is reached. If "no action is required" then it means that there is no need to change the lane(either the car is on the correct lane or it is a single lane path).

Handling Speed Limit If the lane contains a speed limit sign, then the controller adjusts the car's speed accordingly. The value of the speed limit is extracted from the lane information and then the speed limit is compared to the car's current speed. The rule base is checked to retrieve the necessary action. If the car's speed is less than the speed limit, then the car's speed is increased else it is decreased until the speed is in the range of +1 or -1 of that of the speed limit.

Handling Signal In this case, when the car reaches the braking distance, the controller probes the event value generator to give the status of the signal. If the value is "RED", the rule base is searched for the corresponding action. In this case, the action is to slow down the car. After a set period of time, the controller again probes the value generator for the status. If the value is "RED", the car keeps slowing down. The process is repeated until the car stops or a "GREEN" signal is received. If the value is "GREEN", the action returned from the rule base instructs the controller to increase the speed of the car.

It is to be noted that while slowing down, the car takes a long time to come to a halt. In fact the speed of the car never reaches zero but very close to zero. So to handle this issue and to make the car halt, the speed of the car is made zero once the speed becomes less than 0.35 kmph.

Handling STOP Sign This is similar to handling the signal event. Only in this case, once the car crosses the braking distance, the controller does not probe the event value generator. This is because the controller is aware that there is a STOP sign at the end of the lane. This information is available from the lane data that was parsed when the text file was read. Once the car stops before the STOP sign, it probes the event value generator until an "all clear" value is obtained. After the value is obtained, the car starts and continues its journey.

Case Based System

The cases are stored in the following manner in the case base:-
$[ExceptionalEvent$
target: Bus
Distance: 20m, left
Speed: me: 20kmph, Bus: 30kmph

$Plan$
(accelerate, me, priority, 40)
]

The above case shows an exceptional event that there is a Bus at a distance of 20 meters to the left of our car and travelling at a speed of 30kmph. The speed of the car is assumed to be 20kmph. The corresponding action is to accelerate the car to reach the speed of 40kmph. The action has a 'priority' and based on this value the rate of acceleration is determined. Every case will have a 'priority' value in the action plan. It is depending on this 'priority' value that the controller is able to accelerate or decelerate the car quickly. The Case Based system has the following two modules:-

i. **Retriever:** The retriever module performs linear search on the case base. The maximum similarity threshold limit is 90% and the minimum threshold limit is 50%. The CBR assesses the similarity between the properties (target, distance, speed) of the object in the event to that of a candidate case in the case base. Let D(c), D(q) be the distance of the object from the car in the candidate case and query event respectively. Let S(c), S(q) be the speed of the object in the candidate case and the query case respectively. Let Di(c), Di(q) be the direction of the object wrt the car in the candidate case and the query case respectively. The similarity between the properties is calculated as follows:-

$$\text{Sim(D)} = 1 - |(D(q) - D(c))/D(q)|$$
$$\text{Sim(S)} = 1 - |(S(q) - S(c))/S(q)|$$
$$\text{Sim(Di)} = 1 - |(Di(q) - Di(c))/Di(q)|$$

Sim(Total) = Sim(D) + Sim(S) + Sim(Di)

The Case based system maintains a mapping between the objects with the SimilarityMatrix. For instance, SimilarityMatrix[Bus][Truck]=0.95 or SimilarityMatrix[Pedestrian][Rock]=0.05. A mapping of the directions is also maintained so that if the direction in the candidate case and query event are opposite then the similarity between them will be 0.

ii. **Modifier:** The Modifier module modifies/adapts the retrieved case to better suit the current state. It compares the distance and the speed of the object in the event with that of the object in the retrieved case. If the difference in the distance and the speed is below a threshold, then the Modifier modifies the priority attribute of the retrieved plan to adjust the car's speed.

The Appendix (Section 9) shows the output of the case based system.

6. Evaluation

The system was evaluated for accuracy as well as performance. To measure the ability of the system to take accurate decisions, we modified the random event generator to produce 25 cases per lane. There were four lanes from the source to the destination. So the total number of random cases for one run of the simulator yielded 100 cases. We ran the simulator 3 times. Thus a total of 300 cases were

handled by the simulator. For each event generated, the event, the corresponding retrieved case and the similarity value were displayed to the console. We then manually compared the retrieved case with the event for all 300 cases. Since the information regarding the speed and distance of the object in question with that of our car is available, we were able to determine if an accident would take place or not based on the retrieved case.

Our findings are as follows:-
We found out that the case based system performed well in terms of taking accurate decisions. Out of the total 300 cases, the CBR system successfully handled 298 cases. There were two cases that resulted in a collision. These two cases were the ones where the similarity value was very low. Both cases resulted in the CBR returning the default action of slowing down with the maximum priority. However, it is to be noted that there were other cases(5 cases) generated where the CBR returned the default action. But those cases did not result in a collision indicating that the default action successfully handled some cases.

One of the two cases that failed involved a bus at a distance of 5 meters coming in from the left and travelling at 40 kmph. The speed of the car was 20kmph. The default case instructed the controller to apply brakes with maximum priority.

Rule Based System Performance: The rule based system was evaluated to determine its response time. The response time was evaluated as the difference between the time the instruction to search the rule base was executed to the time the action was returned. It was found that for 100 executions, on average the response time was 0.078ms. The time difference was calculated using the time.clock() function in Python.

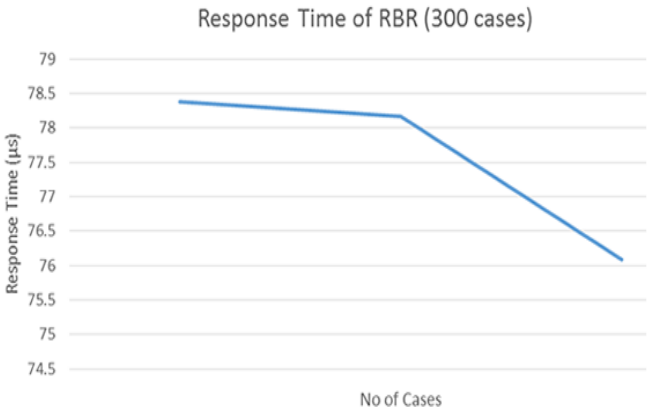

Figure 2: Response Time of Rule Base System

Case Based System Performance: The case based system was also evaluated to determine its response time. The response time was evaluated as the difference between the time the instruction to search the case base was executed to the time the corresponding case was returned. It was found that for 100 executions, on average the response time was around 11ms.

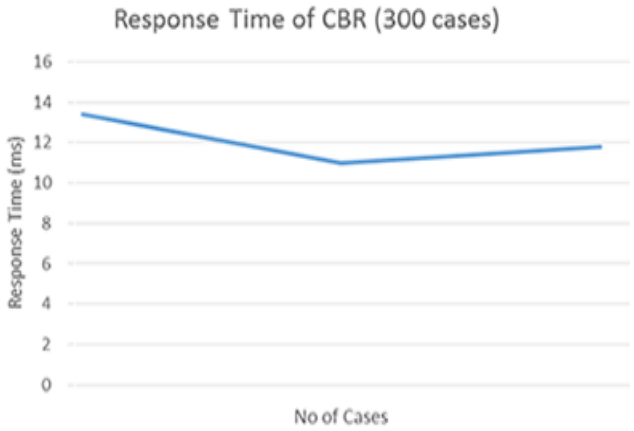

Figure 3: Response Time of Case Base System

Strengths:

i. **Emulates Real World Driving:** System simulates a real world driving scenario to the maximum possible extent. The formula to accelerate/decelerate the car is the classical equation of motion. The case base was constructed based on our experience of driving a car and what we might do in case of an exceptional event. So the case base contains actual previous experiences.

ii. **Robust and Scalable:** The accuracy of the case based system indicates good decision taking ability of the system. The failed cases can be handled by adding more cases to the case base. The priority attribute in the retrieved case in the case base handles the amount by which the speed should increase or decrease. Using this attribute alone, many of the cases can be handled by assigning an appropriate value to priority. This allows the system to be easily scaled.

iii. **Efficient Response Time:** The evaluation of both the Rule Base and Case based system shows that the system has efficient response time. This was possible because of the Hybrid approach which enabled division of tasks between the two systems.

Weakness:

i. The current system is single threaded. A multi-threaded system would enable faster processing of tasks which is ideal for a real world driving scenario.

ii. All the traffic rules are not included in the rule base. The most basic and frequently followed rules are added to the rule base.

iii. There is no learning involved in the current system. Learning can be achieved only after we are certain that the action returned by the case has successfully handled the exceptional event. A multi-threaded system can easily handle this scenario wherein one thread can be assigned to check if there is a need to add the case to the case base or whether adding the case would result in redundancy.

7. Future Work

i. Modify the current system to work as a multi-threaded system.

ii. Incorporate computer vision to capture the images and process them and feed the necessary data to the system. The system can then handle the cases the way it is working now.

iii. Learning can also be incorporated to help the system make more informed decisions.

8. Conclusion

In this paper, we have implemented a hybrid system of rule based and case based components for a Self-Driving Car Simulator. We have shown that such a hybrid system is quite suitable for handling complex tasks by efficiently dividing these tasks between the rule based and case base systems. Through the implementation of the simulator, our goal of evaluating the hybrid system for handling complex tasks as well as measuring the system's decision making capability and response time for reacting to a dynamic event has been achieved. Our findings show that the decisions taken by the system based on previous experiences is very accurate. Also, the response time of both the rule based and the case based systems is very less (0.078ms and close to 11ms respectively), thereby enabling a quick action to handle both basic driving tasks such as obeying traffic rules and complex exceptional scenarios.

The project has helped us understand the working and the potential of a hybrid system in solving the complex task of driving a car. We also realized that it is important for the components of the hybrid system to be complementary to each other so that the strength of one compensates for the weakness of the other. Such an integration of components is the key to solving such complex tasks.

9. Appendix

This section describes the system as a whole with the help of the output of the program.

Rule Base System

Lane Change A part of the output for handling the lane change is shown below:-

```
Lane Distance covered: 62.89542
Speed: 25
Remaining Lane Distance: 37.1045794753

Lane Distance covered: 64.2843
Speed: 25
Remaining Lane Distance: 35.7156905864

Lane Distance covered: 64.3537
Speed: 25
Remaining Lane Distance: 35.646246142

Attempting to change lanes...
Cannot change lane right now...
Attempting to change lanes...
Lane change successful...
```

The above output shows that the car is attempting to change the lane. The change lane event is triggered once the car crosses a set threshold distance. At this point, the controller probes the Event Value Generator to check if it is safe to change the lane. If the value generated indicates that it is not safe, then the controller probes the value generator again after a pre-defined period. The process continues until the value generator gives an "all clear" signal to the controller to change the lane.

Handling Signal A part of the output for handling the signal is shown below:-

```
Braking Distance Reached...
Signal in view...

Current Signal: RED.
Reducing the speed of the car...

Lane Distance covered: 97.1827817027
Speed: 7.65907105728
Remaining Lane Distance: 2.81721829734

Lane Distance covered: 97.203253565
Speed: 7.36987044997
Remaining Lane Distance: 2.79674643498
... ...
Lane Distance covered: 98.548291949
Speed: 0.261645827831
Remaining Lane Distance: 1.45170805096

Lane Distance covered: 98.5490169237
Speed: 0.26099086622
Remaining Lane Distance: 1.45098307633

Current Signal: RED.
Reducing the speed of the car...

Lane Distance covered: 98.563146594
Speed: 0.248486446024
Remaining Lane Distance: 1.43685340595

Lane Distance covered: 98.5638351763
Speed: 0.247889602548
Remaining Lane Distance: 1.43616482372

Lane Distance covered: 98.7155204371
Speed: 0.141649506018
Remaining Lane Distance: 1.28447956289

Lane Distance covered: 98.7186466471
Speed: 0
Remaining Lane Distance: 1.28135335295

Car STOPPED...
Waiting for signal to turn GREEN...

Current Signal: GREEN
```

```
Lane Distance covered: 98.7312953662
Speed: 0.650505555556
Remaining Lane Distance: 1.26870463381

Lane Distance covered: 98.733241323
Speed: 0.700544444444
Remaining Lane Distance: 1.26675867702
... ...
Lane Distance covered: 99.71108461
Speed: 5.95462777778
Remaining Lane Distance: 0.288915389984

Lane Distance covered: 100.013541894
Speed: 6.80528888889
Remaining Lane Distance: -0.0135418939666

Lane Distance Reached
Taking a left turn...
```

Once the car crosses the braking distance and if there is a signal at the end of the lane, the controller probes the event value generator for the signal event. If the value generator provides the value ”RED” or ”AMBER”, the controller reduces the speed of the car and probes the value generator again after a pre-defined period. If the value provided is still ”RED”, then speed is further decreased. The process continues until the car stops. If value generated is ”GREEN”, then the controller increases the speed of the car. The value generator for the signal event is designed in such a way that it mimics an actual signal.

Handling STOP Sign This is similar to handling the Signal, except that event value generator is not probed periodically prior to halting the car. Once the car has stopped, the value generator is probed to give an ”all clear” signal until then the car waits.

```
Lane Distance covered: 96.3676427469
Speed: 25
Remaining Lane Distance: 3.63235725309

Lane Distance covered: 96.4370871914
Speed: 25
Remaining Lane Distance: 3.56291280864

Braking Distance Reached...
STOP sign in view...

Lane Distance covered: 97.1827817027
Speed: 7.65907105728
Remaining Lane Distance: 2.81721829734

Lane Distance covered: 97.203253565
Speed: 7.36987044997
Remaining Lane Distance: 2.79674643498

Lane Distance covered: 97.5011292603
Speed: 4.09660659907
Remaining Lane Distance: 2.49887073973
... ...
Lane Distance covered: 98.7155204371
Speed: 0.141649506018
Remaining Lane Distance: 1.28447956289

Lane Distance covered: 98.7186466471
Speed: 0
Remaining Lane Distance: 1.28135335295

Car STOPPED...
Waiting to move...

Lane Distance covered: 98.7312953662
Speed: 0.650505555556
Remaining Lane Distance: 1.26870463381

Lane Distance covered: 98.733241323
Speed: 0.700544444444
Remaining Lane Distance: 1.26675867702
```

Handling Speed Limit In the output, we see that the speed limit sign is in view, which causes the controller to increase the speed till it reaches the speed limit (20 kmph, in this case). Once the required speed limit is reached, the speed remains constant.

```
... ...
Speed Limit Sign in view...Speed Limit: 20
Increasing speed to 20

Lane Distance covered: 1.31491080247
Speed: 12.8099555556
Remaining Lane Distance: 98.6850891975

Lane Distance covered: 1.38649421296
Speed: 12.9100333333
Remaining Lane Distance: 98.613505787

Lane Distance covered: 2.93491983025
Speed: 14.9115888889
Remaining Lane Distance: 97.0650801698
... ...
Lane Distance covered: 6.91050955247
Speed: 19.1148555556
Remaining Lane Distance: 93.0894904475

Lane Distance covered: 7.01712018519
Speed: 19.2149333333
Remaining Lane Distance: 92.9828798148

Lane Distance covered: 9.22502209877
Speed: 20.0500388889
Remaining Lane Distance: 90.7749779012

Lane Distance covered: 9.33627220679
Speed: 20.0500388889
Remaining Lane Distance: 90.6637277932
... ...
```

Case Based System

Below is a part of the output showing the handling of an exceptional case by the Case Base System

```
... ...
Lane Distance covered: 88.6802145569
Speed: 25
Remaining Lane Distance: 11.3197854431

Lane Distance covered: 90.0691034458
Speed: 25
Remaining Lane Distance: 9.93089655422

Lane Distance covered: 90.1385478902
Speed: 25
Remaining Lane Distance: 9.86145210978

Exceptional Event in Progress...
Object: bus
Object Distance from Car: 14.25
Object Speed: 7.2
Object Direction wrt Car: left

Retrieved Case:
Object: 'bus'
Distance from Car: 18.38
Direction wrt Car: 'left'
Speed of Car and Object:
  'me': 27.609999999999999, 'bus': 4.5

Increasing Speed to speed limit of 28

Lane Distance covered: 90.6941034458
Speed: 25
Remaining Lane Distance: 9.30589655422

Lane Distance covered: 90.7704977359
Speed: 27.5019444444
Remaining Lane Distance: 9.2295022641

Lane Distance covered: 90.8482755137
Speed: 28
Remaining Lane Distance: 9.15172448632

Exceptional event concluded...
```

The above output shows an exceptional event involving a bus in progress. The details of the event such as the distance of the object (bus) from our car, the speed of the object, its direction wrt our car are displayed and the corresponding case retrieved is also displayed. The retrieved case provides an action plan to the controller, which in this case is to 'accelerate' to a speed of 28 kmph. In this case, as the bus is at a distance over 14 meters from the car and the speed of the bus (7.2 kmph) is much less than that of the car(25 kmph), the case based system based on the retrieved case instructs the controller to accelerate to 28 kmph.

10. References

[1] S Ontanon, Y. Lee, S. Snodgrass, D. Bonfiglio, F. Winston, C. McDonald and A. Gonzalez, "Case-Based Prediction of Teen Driver Behavior and Skill" ICCBR 2014, LNCS 8765, pp. 375389, 2014.

[2] C. MacAdam, "Understanding and Modeling the Human Driver" Vehicle System Dynamics 2003, Vol. 40, Nos. 13, pp. 101134

[3] G. Markkula, O Benderius, K. Wolff and M. Wahde, "A review of near-collision driver behavior models"

[4] C. Urmson, J. Anhalt, H. Bae, T. Brown, D. Demitrish, J. Struble, M. Taylor, M. Darms, D. Ferguson ,"Autonomous Driving in Urban Environments: Boss and the Urban Challenge", June 2008.

[5] D. Ferguson, M. Darms, C. Urmson and S. Kolski, "Detection, Prediction, and Avoidance of Dynamic Obstacles in Urban Environments"

[6] C. Riesbeck and R. Schank, "Inside Case-Based Reasoning". Erlbaum, 1989

[7] D. Leake, "Case-Based Reasoning: Experiences, Lessons, and Future Directions". AAAI Press, 1996 [8]D. Ferguson, T. M. Howard, M. Likhachev, "Motion Planning in Urban Environments", IEEE/RSJ International Conference on , vol., no., pp.1063-1069, 22-26 Sept. 2008

[9] J. Paz, S. Rodrguez, J. M. Sanchez, A. Luis, J. Corchado, "Context Aware Hybrid Agents on Automated Dynamic Environments", 2nd International Workshop on Hybrid Artificial Intelligence Systems, 2008