2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)
Windsor Oceanico Hotel, Rio de Janeiro, Brazil, November 1-4, 2016

The Development of an Ontology for Driving Context Modelling and Reasoning

Zhitao Xiong, Vinayak V. Dixit,S. Travis Waller
Research Centre for Integrated Transport Innovation (rCITI)
School of Civil and Environmental Engineering,UNSW Australia
See http://www.rciti.unsw.edu.au

Abstract— **In order to use technology to influence human behaviour and promote safer and more fuel efficient behaviour through incentive mechanisms, an instrumented vehicle is developed. The first step is to make it "perceive" the outside world, so extracting knowledge from some data sources such as sensors is crucial. More critically, there is a fundamental need for a standard that would enable knowledge sharing/exchanging among the different entities, e.g., between on-board sensors, in-vehicle controls and traffic management agencies. This paper proposes an Ontology for Context Modelling (OCM) to be used as the world model for driving context representation and reasoning, which can enable a better understanding of traffic context and sensor capability, which is the basis for providing data source to Advanced Driver Assistance Systems (ADAS), V2X (Vehicle-to-Everything) communications and even driving decision making within autonomous vehicles. Through the experiments, we evaluate the capability of the OCM to represent the driving context and the reasoning mechanism to compensate for sensor failures and recognize lane changing and overtaking events. This methodology has significant value for creating standards in autonomous and semi-autonomous cars.**

I. Introduction

With advancement in sensor systems, vehicles gather a large variety of information from different sources such as: 1) internal vehicular communication systems that rely on the CAN system; 2) on-board sensors and 3) external messages and communication relayed through V2I. Moreover, driver assistance and control systems need to parse through this extensive information and extract useful information such as road users' positions and speed. This extracted information is also critical to conduct online verification quickly and efficiently.

Therefore, knowledge extraction from the wide variety of data sources is crucial. More critically there is a fundamental need for a standard that would enable knowledge sharing/exchanging among the different entities, e.g., between on-board sensors, in-vehicle controls and traffic management agencies. To this end, a human-readable and machine processable knowledge base is needed. Moreover, a reasoning mechanism to evaluate internal inconsistency as well as verification of the data is also needed to ensure reliability of in-vehicle controls.

Ontologies have been used as a way of describing domain concepts and their relationships in a logical and programming language-independent manner. This is achieved by: 1) identifying and sharing common knowledge in a domain (knowledge representation) and 2) reasoning about or analyzing those domain knowledge (knowledge reasoning).

By formally describing concepts and properties, ontologies have been used for a variety of purposes [2], including the driving context modelling for 1) representing driving context for information exchange between vehicles (Knowledge Representation) and 2) assisting decision making by providing spatio-temporal reasoning (Knowledge Reasoning). A model for scene representation [3] for DAS (Driver Assistance Systems) included not only spatio-temporal representation, but also some additional information about traffic rules. In addition, ontologies have also been used to assist path planning [4] by reasoning about obstacle. Recently, researchers used ontologies to assist multi-vehicle collaboration in route determination [5], in order to deal with intersection negotiation and safe-headway maintenance. Moreover, researchers developed an ontology with rules to assist the navigation of a vehicle when approaching intersections [6]. To sum up, ontologies have be used to imitate human-like reasoning by modelling driving contexts associated to, e.g., intersection negotiation and route guidance. In fact, based on data generated from on-board sensors, ontologies can be extended to include more driving situations. That is why ontology can a solution for knowledge sharing and information exchanging among different entities, e.g., between on-board sensors, in-vehicle controls and traffic management agencies.

In order to develop a formal knowledge base for driving context that can be read by humans and shared, re-used, reasoned by computing systems, this paper proposes an ontology for driving context modelling. This paper is organized as follows: Section II will introduce the instrumented vehicle that has been used to conduct this research. Section III will describe the Ontology for Context Modelling (OCM) in detail. Section IV provides a description of the experiments conducted and the relevant results. This is followed by Section V, which concludes this paper with a discussion.

II. Instrumented Vehicle Setup

The vehicle of interest in this research was developed by rCITI[1] in collaboration with GoGet CarShare[2] in order to examine driving behaviours and promote safer and fuel efficient driving styles. The vehicle is a Toyota Yaris that

[1]Research Centre for Integrated Transport Innovation

[2]GoGet CarShare is the first car sharing company based in four cities in Australia with over 1000 cars

978-1-5090-1889-5/16/$31.00 ©2016 IEEE

Fig. 1. Instrumented Vehicle Setup

has been equipped with four sets of sensors as illustrated in Figure 1.

The installed sensors include a Nax 5.1, which is an integrated Inertial Measurement Unit (IMU) and a Global Positioning System (GPS) from Navexlorer[3], it provides information about the vehicles velocity, yaw rate and GPS coordinates, which is also required by the other onboard sensors. The Intelligent Forward View (IFV) camera from Delphi[4] is used to track lanes and recognize speed signs. The Electronically Scanning Radar (ESR) from Delphi is a radar sensor that is used to provide detailed velocity and position information of road users in front. The Rear and Side Detection System (RSDS) from Delphi, is used to detect vehicles at the rear and in adjacent lanes.

All sensors have been integrated through a mobile computer using a CAN bus (ESR, IFV and RSDS) and serial port (Nax 5.1). A program, written in C++, has been hosted on the mobile computer to control all sensors and collect data. All the equipment except the IFV, ESR and RSDS have been installed in the rear of the car. The hardware and software system has been termed as Integrated Intelligent Vehicular System (IIVS).

Ontology for Context Modelling was first developed in a prototyping stage by examining sensor specifications along with the data collected by experimenters for calibration purposes. Its evaluation was conducted in a second stage, where the data collection was performed in conjunction with a forward looking video camera that visually recorded the driving environment. As an example, consider the sensors detect a leader vehicle with speed 80 *kmh*, this data will be generated in the OCM as an individual of *Vehicle* with corresponding speed property specified by *hasSpeed*. In the next section, OCM will be first introduced which will be followed by details regarding its evaluation.

III. Ontology for Context Modelling

In general, there are three major components in an ontology: classes, individuals and properties.

Classes specify the concepts in a domain such as *RoadUser*. Narrower concepts can be described within subclasses, e.g., *Vehicle* is a subclass of *RoadUser*.

[3]http://navexplorer.net/

[4]http://www.delphi.com/

Individuals specify the specialized instances of classes. For instance, an individual of the class *Vehicle* can be *vehicle_123456*, which is a vehicle with a license plate number "123456".

Properties specify the characteristics of classes, whose individuals have the same features. For instance, property *hasSpeedReading* can be used to indicate the speed reading from a specific sensor, which is an individual of the class *Sensor*. There are four types of properties [1]: 1) intrinsic properties, e.g., *hasSpeed* of a vehicle; 2) extrinsic properties, e.g., *hasVehicleModel* of a vehicle; 3) parts, e.g., *hasLane* of a road and 4) relationships between two individuals, which may not necessarily belong to the same class. A property is defined by two aspects: objects and data types.

An object property specifies the relationship between two individuals, while a data property specifies the relationship between an individual and a data type, which can be a string, Boolean, float, etc. For instance, *hasSpeedReading* is a data property that specifies the relationship between an individual of class *Vehicle* and a float value; *hasLane* is an object property that specifies the relationship between two individuals of class *Road* and *Lanes* respectively.

Properties also have a domain and a range. The domain of a property describes the class. The range of a property identifies the feasible classes from which individuals can be instantiated. For instance, the domain of property *hasLane* can be the class *Road*, and the range of *hasLane* can be the class *Lane*. An ontology for driving simulation was also proposed [9], [10]. It was termed as Ontology for Scenario Orchestration (OSO) and is used to generate formal action script of simulated vehicles along with relevant driving and action contexts to guide the simulated vehicle to interact with the human participant's vehicle. By using OSO, knowledge can be also shared and reused between and within research groups.

In this research, the Ontology for Context Modelling (OCM) has been designed to include common concepts (classes) and their properties that have been identified when developing the instrumented vehicle. This includes the following contextual information: 1) physical scenes, including lane information, traffic signs, traffic lights, speed limits; 2) road users, including cars, bikes, motorcycles, pedestrians; 3) the instrumented vehicle and its corresponding properties and 4) sensor features and specifications, including sensor model, sensor functionalities and raw data specification of a specific sensor.

That is, by covering concepts and their relationships that a driver could come across or an autonomous car could be used for decision making, OCM has been designed to include some essential contextual information that can be of interest to not only the vehicle, but also other entities such as traffic management agencies. Moreover, the selection of concepts and corresponding relationships between them conforms to relevant industrial standards, especially for those sensors. Concepts that will be utilised by IIVS directly are represented by abstracted state variables based on an object's physical and internal properties. For instance, the colour of a vehicle,

the speed of a vehicle.

OCM was developed using Protégé[7] and it contains 76 classes and 50 properties as illustrated in Figure 4. Protégé is an ontology editor and knowledge acquisition system developed by Stanford University and the University of Manchester. In the next section, we elaborate on the key features of context modelling.

Figure 4 goes here.

Protégé is an ontology editor and knowledge acquisition system developed by Stanford University and the University of Manchester. It is free, Open Source, widely used and well supported, so Protégé has been chosen as the editor for OCM. Moreover, Protégé also provides reasoners, e.g, HermiT[5], to check a particular ontology's subsumption: if one class can be a subclass of another class or its consistency: if there are any classes that cannot have any individuals.

In this section, some essential concepts will be introduced. *Classes*, *Properties* and *individuals* are represented with italic fonts.

A. Context Modelling

1) NeighbourHood:

The concept of "Neighbourhood" is represented by the class *Neighbourhood*, which is derived from a superclass *Formation*, whose details can be found in references [9], [10].

Basically, this class includes information regarding the surrounding road users. It contains 12 positions as illustrated in Figure 2, therefore, 12 sub-classes of *Neighbourhood* are modelled, e.g., "Follower". As a relative position around the vehicle of interest, each position will be related with a road user if a road user is found at that position.

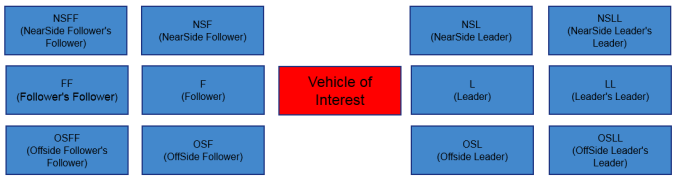

Fig. 2. Neighbourhood of Road Users around a Vehicle of Interest

2) Entity:

Entities include all the physical objects that can appear in a driving environment, e.g., a traffic sign, lane markings or other vehicles. It has two major subclasses: *RoadFacility* and *RoadUser*. The former indicates objects used for road networks and the latter indicates objects that use the road.

RoadFacility has a subclass of *SpeedSign*, which has the property of *hasIndidatedSpeed*. This property relates a specific speed sign with an integer (the speed limit observed).

RoadUser has four subclasses: *Bicycle*, *Motorcycle*, *Pedestrian* and *Vehicle*. Each of them has four properties:

hasLaneOccupancy indicates their relative lane position with respect to the subject vehicle. with The subject vehicle is always assigned the lane id *0*. $\pm n$ means lane No. n on the left/right of lane 0. The bigger the number, the further the lane is from the subject vehicle.

hasPosition_X_Pixel and *hasPosition_Y_Pixel* indicate the X-axis and Y-axis positions of a road user in the video taken from the camera. If the cameras[6] resolution is 640 $\times$ 480, then the domain of values for *hasPosition_X_Pixel* and *hasPosition_Y_Pixel*, are 0-640 and 0-480, respectively.

hasFormationPosition indicates the neighbourhood positions of the road user. For instance, if a road user occupies the "Follower" position, then the road user will be related with class *Follower* by using property *hasFormationPosition*.

3) Sensor:

The class *Sensor* includes all the sensors in the subject vehicle. It includes four subclasses: *LaserSensor*, *PoseSensor*, *RadarSensor* and *VisionSensor*. *GPS* is a subclass of *PoseSensor*, with the property *hasSpeedReading*, that is defined by exactly one floating point number. In other words, the GPS can provide exactly one speed reading. Individuals of *Sensor* identifies a specific sensor. For instance, individual *delphiIFV200* indicates the IFV from Delphi, which is a specific *VisionSensor*.

4) StateVariable:

State variables are used to describe the states of a reference object, which can be any entity included in the class *Entity*. It contains the following major subclasses: *SV_AccelerationRate* for acceleration rate reading (negative for deceleration), *SV_Colour* for the colour of the vehicle, *SV_Shape* for the type of vehicles (wagon, sedan, SUV, etc) and *SV_Speed* for the speed reading. Those subclasses can be also used to indicate what information a sensor can provide. For instance, *delphiIFV200* can be related with *SV_Position_Lane* by using a property named *hasReading*.

B. Context Reasoning

As this OCM was designed for real-time applications, the reasoning mechanism was integrated with a rule engine for offline reasoning (JESS[7]) and an expert system for real-time reasoning (CLIPS[8]). The reasoning can be described by three main rules. All instances of these rules are instantiated either by the rule engine or by the expert system based on the real-time data feed from the IIVS.

1) NeighbourHood:

Neighbourhood reasoning requires the ability to query about which sensor can provide the required information needed (for eg. Speed), and to query the states of the surrounding road users'. These two querying procedures are critical for for neighbourhood building, which are used for querying states of the Neighbourhood entities. For instance, the following statement can be used to query about all the vehicles that travel in front of the vehicle of interest:

(defquery qLaneLeaders1
"Query for all the leaders of in each lane"
(declare (variables ?lane))

[5]http://protegewiki.stanford.edu/wiki/HermiT

[6]As stated in Section II, in this research a video camera was used

[7]http://herzberg.ca.sandia.gov

[8]http://clipsrules.sourceforge.net

(Vehicle (hasLaneOccupancy ?lane)
(hasPosition_X_Pixel ?x) (hasPosition_Y_Pixel ?y)))

(bind ?result (run-query qLaneLeaders 0))*

As the vehicle of interest always occupies lane 0, querying vehicle in front will be performed by finding all vehicles in lane 0. The results of which will be stored in a variable. As indicated by the rule stated above, the results also include the positions of the vehicle in the IFV coordinates.

2) Sensor Compensation:

During our experiment, we found that the sensors may fail[9]. Examples of potential failures include but are not limited to: ESR producing inaccurate readings under rainy weather conditions, IFV produces erroneous lane marking results under limited lighting. More generally, we also found that the majority of the errors came from the fact that the sensor did not remember the context. For example, the IFV sensor may report the left/right lane as a road edge, when in fact it was a large vehicle such as a truck. Though, it would be impossible to tackle all these failures at a hardware level, we implemented a rule to identify and compensate for such errors. A simplified version with only essential information is presented below.

(defrule rTruckEdge1
"identify and compensate for road-edge identification error"
t1 ← (Instant)
t2 ← (Instant (hasClosePrecedence ?t1))
?x ← (Vehicle (hasType TRUCK))
?self ← (Vehicle (hasLaneMarking ?y) (hasNearSideLeader ?x) (hasTimeStamp ?t1))
?self ← (Vehicle(hasLaneMarking ROADEDGE) (hasTimeStamp ?t2)))
⇒
?self ← (Vehicle (hasLaneMarking ?y) (hasTimeStamp ?t2)))

The rule presented above, check if a truck was found to be at the nearside of the leader at time $t1$ and at time $t2$, and if the road marking transferred from a non-edge reading to edge reading during this period, then IIVS should consider this to be a fault. Then the road edge found at time $t2$ should be changed to a non-edge reading unless a new reading is confirmed. In this research, $t2$ has been identified to be the predecessor of $t1$ and is assumed to be 20 milliseconds, based on the updating frequency of the sensor data.

3) Overtaking and Lane-changing Recognition:

Presently there are no sensors on the steering wheels to capture indicator readings as well as lane-changing maneuvers. This makes the detection of lane-changing challenging. In this research, we implement rules to detect lane-changing movement as well as overtaking maneuvers. Overtaking recognition is achieved by detecting two lane-changing procedures. A simplied version with essential information is as follows.

[9]For example, due to poor lighting, obscure lane marking and weather

(defrule rLaneChanging1
"recognise Lane changing movement"
t1 ← (Instant)
?x ← (Vehicle (hasLaneOccupancy 0))
?self ← (Vehicle (hasLeader ?x) (hasTimeStamp ?t1) (hasLaneMarking ?y))
?self ← (Vehicle (hasAngleSpeed ?z) (hasAngleSpeedReference ?x) (hasTimeStamp ?t1)))
⇒
(?self (isLaneChanging TRUE)))

That is to say that if the IIVS detects that the subject vehicle is now moving left or right with regard to its leader, it is believed to do lane changing.

IV. Evaluation

The development of OCM complies with the On-To-Knowledge methodology [11] which requires: feasibility study, kick-off, refinement, evaluation and application & evolution. For instance, classes are being deleted or added (e.g., *PoseSensor* is added because of the adoption of Nax IMU/GPS sensor) according to specific requirements. Moreover, the development of OCM also complies with the OntoClean [13] methodology by identifying the meaning of the properties, classes and their relations during the development and evolution. Finally, OCM has been checked by the HermiT reasoner as mentioned in Section III in Protégé and was found to be consistent.

In order to evaluate OCM, an experiment was conducted by utilising a video camera to verify the output of context modelling based on OCM. A pilot study was carried out on a 2 km long 2-lane motorway with sunny weather as is illustrated in Figure 5. Specifically, this experiment evaluated the real-time application of the OCM to:

- represent, derive and query knowledge;
- identify and compensate sensor failures;
- recognise lane-changing and overtaking events and;
- satisfy the ontology engineering criteria [14].

Based on the data collect a visual inspection was carried out to evaluate the performance of the IIVS with respect to the first three criteria, and finally the ontology engineering criteria was applied. Overall the OCM was found to be robust.

Figure 5 goes here.

A. Knowledge Representation, Derivation and Query

Knowledge representation concerns about the correct initialization of each instance. Knowledge Query is used to query states of Neighbourhood entities. Knowledge deviation is used to enquire about the information providers. A class named *InfoProvider* was created in OCM to provide this service and has three subclasses: *AccProvider*, *ObjectDetector* and *SpeedProvider*. For instance, the class *SpeedProvider* is equivalent to a class who has speed reading, which can be represented as "*hasSpeedReading*". Hence, as GPS can provide speed reading, GPS is automatically classified as a subclass of *SpeedProvider*. As a result, in real-time

situations, if IIVS needs to find speed, it will directly look for raw data from any speed providers such as the ESR radar.

In this experiment, leaders and lanes have been properly represented as shown in Figure 5. Relevant individuals have been created in the expert system based on CLIPS.

B. Sensor Failure Identification and Compensation

At time T_1, which is 20 seconds after the beginning of this scenario, a truck was recognised as road edge. As the left lane marking was recognised as "dotted" before the compensation, IIVS treated the marking still as "dotted" until new road marking was identified as illustrated in Figure 3.

Fig. 3. Sensor Failure Compensation

C. Lane-Changing and Overtaking Recognition

At time T_2 and T_3, which are 26 and 48 seconds after the beginning of this scenario, the vehicle of interest performed two lane-changing manoeuvres. As the leader vehicles in both time steps were slower than the vehicle of interest according to the data fed to IIVS (5 kmh and 3 kmh respectively), IIVS recognised the lane-changing movements and treated them as overtaking manoeuvres.

D. Ontology Engineering Criteria

The context modelling and ontology engineering criteria from Krummenacher et al. (2007) are largely satisfied, e.g., 1) Applicability: OCM can be also used in areas that need to model driving context, e.g., driving simulation, so it is not limited to instrumented vehicle application only and 2) extensibility: new concepts can be added into OCM with relevant classes and properties without the need of changing the existing model in OCM;

V. Conclusion

In this paper, the Ontology for Context Modelling (OCM) is introduced along with an expert system built upon it. It is an abstract model of driving context. It has been used with field data from an instrumented vehicle and be evaluated with respect to some general criteria. Further research is being undertaken to adopt rules for real-time processing in order to encode more driving situations. Hence, rules and queries will be updated along with the OCM manipulation.

To sum up, OCM, although in its early phase, has shown its capacity in bringing a formal, standardized and thus shareable knowledge base for driving-related activities. However, there is a need for future research to develop rules for more complex situations and test it against varieties of real-world situations. Since OCM is built upon relevant technical manuals or standards, it is believed to benefit driving context modelling with on-board sensors, knowledge sharing between different entities and knowledge reasoning for driving assistance.

VI. Acknowledgement

The authors would like to express their sincere gratitude to the GoGet team and their leadership. We would like to especially thank Mr. Bruce Jeffreys, Ms. Rachel Moore and Mr. Nic Lowe. This research has received assistance from Mr Sunny Gunawan, who did his honour thesis with this project.

References

[1] Noy, N.F. & Mcguinness, D.L. (2001). Ontology development 101: A guide to creating your first ontology. Tech. rep., Stanford University, Stanford, CA, USA.

[2] Corcho, O., Fernández-López, M. & Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies: where is their meeting point? Data Knowl. Eng., 46, 41-64.

[3] Fuchs, S., Rass, S. & Kyamakya, K. (2008). Integration of ontological scene representation and logic-based reasoning for context-aware driver assistance systems. Electronic Communications of the EASST, 11.

[4] Provine, R., Schlenoff, C., Balakirsky, S., Smith, S. & Uschold, M. (2004). Ontology-based methods for enhancing autonomous vehicle path planning. Robotics and Autonomous Systems, 49, 123-133.

[5] Hamilton, A., González, E., Acosta, L., Arnay, R. & Espelosín, J. (2013). Semantic-based approach for route determination and ontology updating. Engineering Applications of Artificial Intelligence, 23, 1174-1184.

[6] Armand, A., Filliat, D., and Ibanez-Guzman, J. (2014). Ontology-Based Context Awareness for Driving Assistance Systems. In Proc. of the IEEE Intelligent Vehicles Symposium, pages 16, E tats-Unis.

[7] Protégé (2013). Protégé-OWL Ontology Editor. http://protege.stanford.edu.

[8] Hitzler, P., Krotzsch, M. & Rudolph, S. (2011). Foundations of semantic web technologies. Chapman and HallsCRC.

[9] Xiong, Z., Cohn, A. G., Carsten, O., & Jamson, H. (2012). Autonomous local manoeuvre and scenario orchestration based on automated action planning in driving simulation. In Proceedings of driving simulation conference Europe 2012, pages 233 - 244, Arts Et Métiers Paristech Paris, France, September 2012.

[10] Xiong, Z., Jamson, H., Cohn, A. G. & Carsten, O. (2013). Ontology for Scenario Orchestration (OSO): a Standardised Scenario Description in Driving Simulation. In Proceedings of the 2nd International Conference on Transportation Information and Safety 2013, pages 1572 - 1577, Wuhan, China, June 2013.

[11] Sure, Y. (2003). Methodology, Tools and Case Studies for Ontology based Knowledge Management. Phd thesis, University of Karlsruhe, Department of Economics and Business Engineering.

[12] Jarrar, M. (2005). Towards Methodological Principles for Ontology Engineering. Phd thesis, Vrije Universiteit Brussel.

[13] Guarino, N. & Welty, C. (2002). Evaluating ontological decisions with ontoclean. Communications of the ACM, 45, 61-65.

[14] Krummenacher, R. & Strang, T. (2007). Ontology-based context modeling. In Proceedings Third Workshop on Context-Aware Proactive Systems (CAPS 2007)(June 2007).

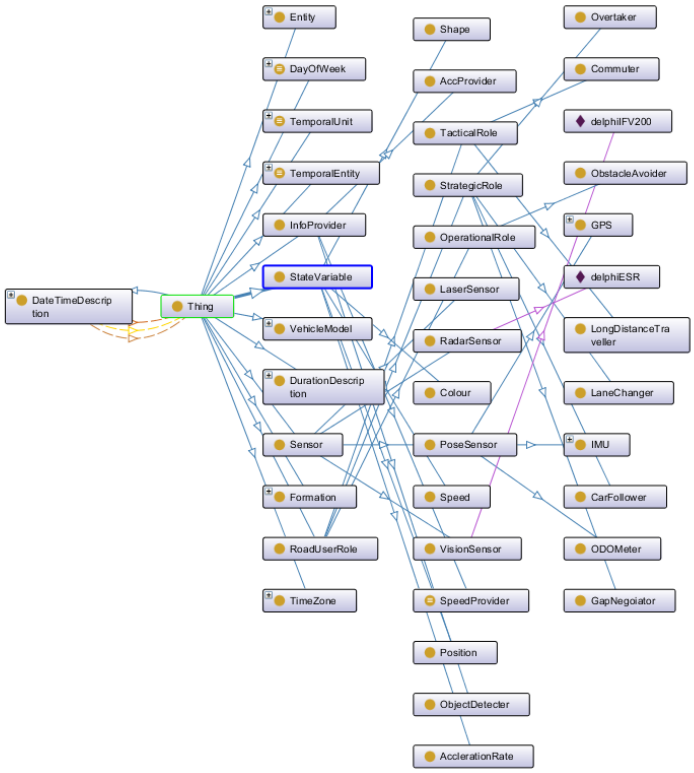

Fig. 4. Ontology for Context Modelling as of 07/2016

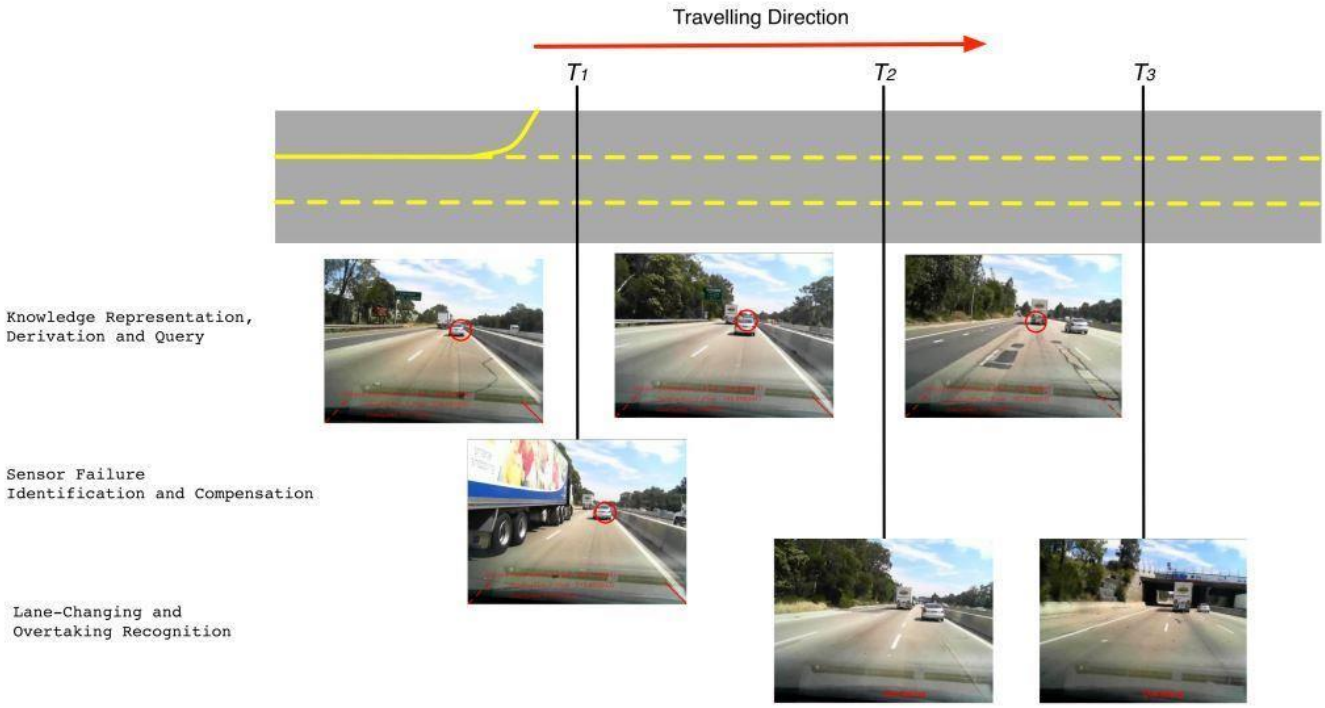

Fig. 5. Test Scenario and Relevant Events