

An Object-Oriented Design of a World Model for Autonomous City Vehicles

Andrei Furda and Ljubo Vlacic

Abstract—This paper presents an object-oriented world model for the road traffic environment of autonomous (driverless) city vehicles. The developed World Model is a software component of the autonomous vehicle's control system, which represents the vehicle's view of its road environment. Regardless whether the information is a priori known, obtained through on-board sensors, or through communication, the World Model stores and updates information in real-time, notifies the decision making subsystem about relevant events, and provides access to its stored information. The design is based on software design patterns, and its application programming interface provides both asynchronous and synchronous access to its information. Experimental results of both a 3D simulation and real-world experiments show that the approach is applicable and real-time capable.

I. INTRODUCTION

Autonomous (driverless) city vehicles capable of driving safely through city traffic, sharing the roads with other traffic participants, such as human-driven cars, pedestrians, bicycles, etc., have been a vision for many years [1].

One of the most challenging remaining research tasks regarding the vehicles' control software, is the autonomous vehicles' ability to make intelligent real-time driving decisions, i.e. their ability to perform the most appropriate driving maneuver for any urban traffic situation. Similar to a human driver, the real-time decision making subsystem makes driving decisions based on perceived information using on-board sensors, knowledge about the planned route, and any other relevant information describing the vehicle's surrounding traffic environment.

This paper addresses the research question regarding how to enable autonomous vehicles to store, manage, and access the information about their surrounding road traffic environment, including other road vehicles, static obstacles, pedestrians, road lanes, traffic signs, intersections, etc., as well as the relationships between these.

A. State of Research

Although numerous solutions have been developed for modeling the environment of autonomous robots [2] and/or

We would like to thank INRIA's team IMARA for the financial support provided towards conducting the experimental work at their test track in Rocquencourt, France. We are particularly grateful to Dr. Michel Parent, Laurent Bouraoui and Francois Charlot for their effort and assistance in performing the experiments.

A. Furda and L. Vlacic are with the Intelligent Control Systems Laboratory (ICSL), Institute of Integrated and Intelligent Systems, Griffith University, Brisbane QLD 4111, Australia, a.furda@griffith.edu.au; l.vlacic@griffith.edu.au

off-road autonomous vehicles [3], [4], no solution or proposal has yet been published, which is capable of modeling all relevant aspects of urban roads, which are required for autonomous city vehicles safely operating in non-simplified urban traffic conditions. Such aspects are for instance perceived and classified objects, such as vehicles, static obstacles, pedestrians, roads, intersections, traffic lanes, and traffic signs, but also the relationships between these objects (e.g. obstacle on left lane, intersection connecting roads, traffic sign valid for one lane, etc.).

A closely related approach for autonomous city vehicles is [5], however it limits the modeled traffic features to static and dynamic obstacles, without including additional perceived traffic information such as traffic lanes, intersections, or traffic signs. Other related research areas with similar requirements for world model information are for instance driver assistance systems, such as Safespot [6].

The modeling solutions developed for the DARPA Urban Challenge 2007 were focused on simplified requirements, and were therefore not sufficient for enabling autonomous vehicles to operate in public urban traffic. For example, the developers of the DARPA Urban Challenge winning vehicle "Boss" observed that their traffic representation was not sufficient to make intelligent driving decisions compared to human drivers [7]. Furthermore, Junior's developers (2nd place) noticed that their vehicle, and probably any other vehicle competing in this race would not be able to cope with a realistic city traffic environment [8].

The remainder of this paper is organized as follows. Section II gives an overview of the autonomous vehicle's control system and the role of the developed World Model, section III presents the World Model, section IV elaborates on the World Model integration, simulation, and real-world experimental tests. Section V concludes this work.

II. AUTONOMOUS VEHICLE CONTROL SOFTWARE ARCHITECTURE

The autonomous vehicle control software consists of the following four functional subsystems (Figure 1):

- Perception Subsystem,
- Real-Time Decision Making & Driving Maneuver Control,
- Driving Maneuvers,
- Vehicle Interface.

The purpose of the Perception Subsystem is to collect available information about the vehicle's road traffic environment, to manage and process it, and to provide it in a

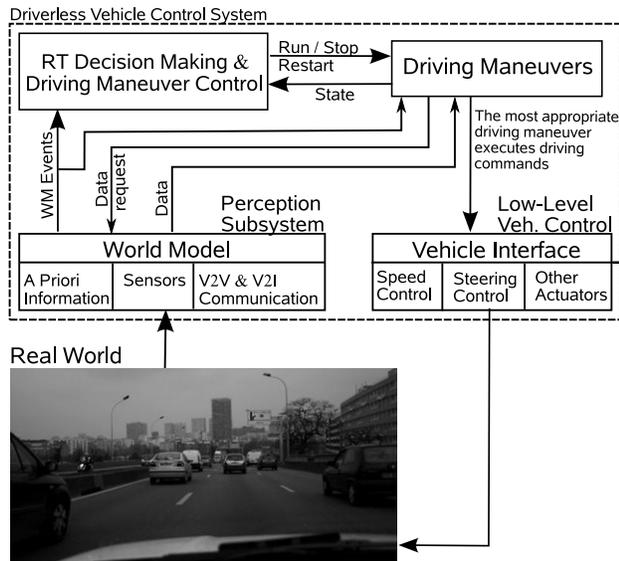


Fig. 1. Simplified view of the autonomous vehicle control software architecture and the flow of data.

adequate form to the Real-Time Decision Making & Driving Maneuver Control, and Driving Maneuvers.

Based on the information provided by the Perception Subsystem, the Real-Time Decision Making & Driving Maneuver Control subsystem makes driving decisions. This software subsystem decides about the activation and the execution of the most appropriate driving maneuver for any given traffic situation.

The Driving Maneuvers subsystem contains a set of closed-loop control algorithms, each able to maneuver the vehicle in a specific traffic situation. The driving maneuvers direct their output to the Vehicle Interface subsystem.

The Low-Level Vehicle Control subsystem contains hardware and software components, which control the vehicle's speed, steering angle, and other actuators (e.g. transmission).

III. THE WORLD MODEL

A. Requirements

In order to enable autonomous driving, the following types of input information about the traffic environment are required (Figure 2):

- a priori information,
- information obtained from on-board sensors in real-time during the vehicle's movement,
- information obtained through communication.

The a priori information (Figures 1 and 2) includes all of information which is available in advance, before the autonomous vehicle starts its journey. This includes for example a planned travel path, coordinates of intersections and roundabouts, and/or other relevant information about the road infrastructure, such as the number of traffic lanes. This a priori information is specified for the planned route in a similar way as in the DARPA Urban Challenge 2007 (i.e. Route Network Definition File RNDf) [9].

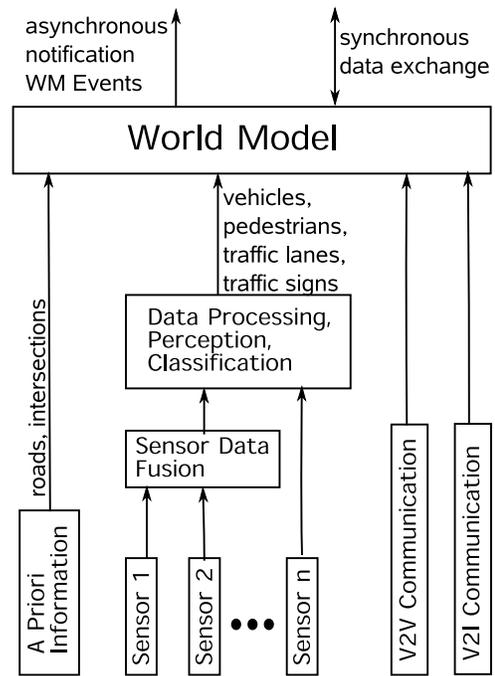


Fig. 2. World Model Input and Output.

In addition to the a priori information, the autonomous vehicle continuously obtains real-time information about its traffic environment. This information is obtained either from the vehicle's own on-board sensors (e.g. cameras, LIDAR, RADAR), through Vehicle-to-Vehicle (V2V) communication, or through Vehicle-to-Infrastructure (V2I) communication (e.g. a traffic management centre). Components for sensor data fusion, processing, perception and classification deal with sensor-related problems such as sensor noise, and uncertainties. Therefore, it is assumed that the information coming from these components is correct.

The purpose of the World Model is to merge this information and to provide at any given time an accurate and up-to-date representation of traffic environment, which is used as input information for the real-time decision making subsystem.

1) *Functional Requirements:* The following are the World Model's most relevant functional requirements:

- Stores a priori information, such as roads, intersections, and traffic signs (a priori known).
- Stores information provided by sensors, such as obstacles, traffic lanes, and perceived traffic signs.
- Stores information obtained through communication with other vehicles or a traffic management centre.
- Cyclicly merges and updates the a priori information with the information obtained continuously from Sensor and Communication components.
- Calculates cyclicly the relationships between the stored entities. For example, if sufficient data is available, the World Model determines the positions of the current road, the current traffic lane, obstacles on the current road, the type of obstacles on the current lane/road, and

distances to obstacles.

- Notifies other subsystems of relevant events in the traffic environment through an asynchronous mechanism.
- Provides other subsystems complete access to all stored information by replying to synchronous data requests (Figure 2).

2) *Non Functional Requirements:* Besides the non functional requirements which are relevant for the entire autonomous vehicle control system, such as reliability and robustness, non functional requirements for the World Model are:

- Real-Time capable,
- Interoperability with various types of sensor components or perception subsystems,
- Modular design,
- Ease of integration through a flexible interface.

The modular decomposition of the control system architecture into the five functional subsystems (Figure 1) is common in many current control system architectures for autonomous city vehicles, such as those developed for the DARPA Urban Challenge [10], [8], [11]. Therefore, the decoupling of the World Model from other control system functionalities as a monolithic subsystem enables its integration and reusability in a variety of autonomous vehicle control architectures. Nevertheless, in order to ensure its usability and real-time execution on dedicated hardware, the internal architecture of the World Model needs to be structured in a modular way, and provide a well-defined interface to its functionality.

B. The World Model Software Architecture

In order not to exceed the autonomous vehicle’s limited real-time computing capabilities, the modeled entities are restricted to a minimum set considered to be necessary for autonomous driving, while maintaining an extendable object-oriented design for possible future needs to include more details. The modeled entities are those which need to be stored as specified in the list of functional requirements (subsection III-A.1): roads, traffic lanes (including one or two lane boundaries), intersections, obstacles (vehicles, pedestrians, static obstacles, obstacles of unknown type), and traffic signs.

A generic and extendable software representation of the traffic environment is defined by decomposing the traffic features, such as roads, traffic lanes and traffic signs, into several entities and specifying their relationships to each other. In the World Model data structure, each entity is represented by a class.

Figure 3 shows a simplified¹ UML (Unified Modeling Language) class diagram of the software component “World Model”. The main class of this software component is called “WorldModel” and is associated with the class “VehicleData”, which contains the vehicle’s current position, steering angle, speed, etc.

¹For the sake of readability, the UML diagram is simplified. Attributes are defined *public* and only a few of the required operations are specified.

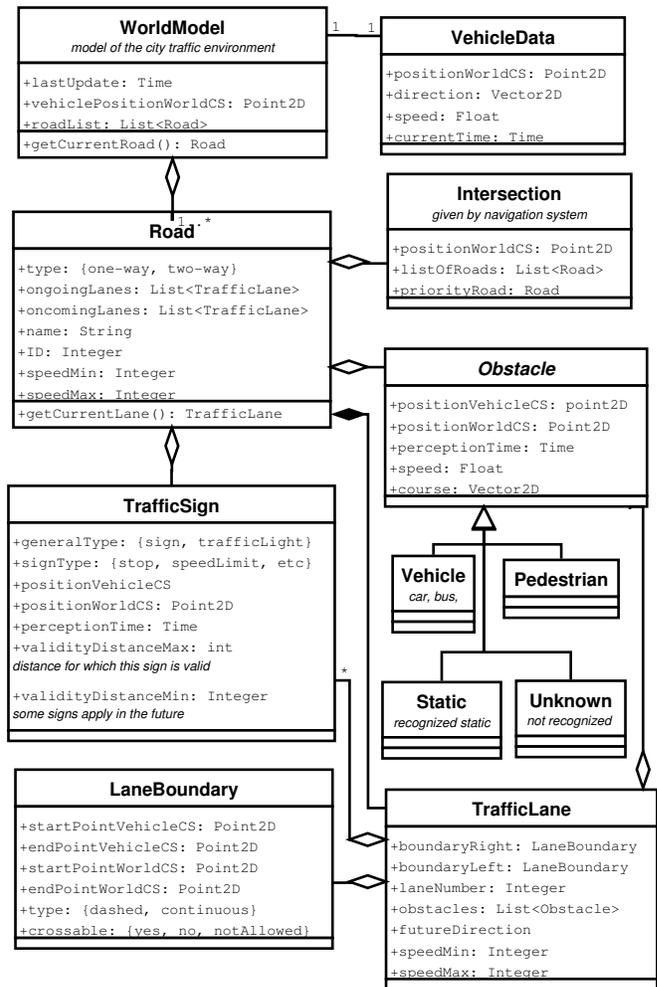


Fig. 3. World Model UML class diagram (simplified diagram).

The class “WorldModel” is an aggregation of roads (class “Road”); the “WorldModel” contains at least one instance of the class “Road”. It can be assumed that at least the current road is always known during the vehicle’s movement, using localization systems such as GPS and INS (Inertial Navigation System).

Each road contains two lists of references to traffic lanes: ongoing lanes and oncoming lanes.

A traffic lane (class “TrafficLane”) is defined by its left and right lane boundaries (class “LaneBoundary”), a lane number, a list of known obstacles, the future direction of the lane and speed limits. A traffic lane is only defined as part of a road.

Traffic signs (class “TrafficSign”) can belong to a traffic lane, or, if they are valid for the whole road, to the road. A traffic sign is described by a general type (e.g. road marking, sign), the traffic sign type (e.g. stop, give way), its position and distance of validity.

The abstract class “Obstacle” encapsulates common properties of all obstacles, such as position, speed and moving course. Subclasses of the class “Obstacle” represent recognized obstacles, such as vehicles, pedestrians, etc. Similar

to traffic signs, obstacles can belong to traffic lanes or to roads. The class “Static” describes static obstacles (e.g. tree), while the class “Unknown” is used for obstacles which were detected but could not be further classified.

Intersections (class “Intersection”) are specified by their positions, a list of roads belonging to the intersection and, if known, the priority road. On the other hand, each road contains references to its intersections. The referencing between the classes “Intersection” and “Road” enables both the extraction of an intersection’s roads and the calculation of the intersection sequence on a road.

C. A Priori Information

Prior to the autonomous vehicle’s journey, the predefined a priori information is loaded from an XML (Extensible Markup Language) file into the World Model structure.

The XML file structure reflects the World Model’s object-oriented data structure (Figure 3). Each XML element corresponds to a class, and each XML attribute corresponds to a class attribute. The part-of relationships (i.e. aggregation, composition) between classes correspond in the XML structure to child elements.

D. Cyclic Updating

On its journey, the autonomous vehicle continuously senses the traffic environment, and updates the World Model data structure with perceived information, such as detected obstacles, detected traffic signs, detected traffic lanes, etc. Accordingly, the World Model data structure needs to be cyclicly updated. The World Model updating operation can be divided into the following three steps:

1. adding newly perceived information,
2. updating existing information,
3. removing obsolete information.

In the first step, newly perceived information, which is provided by various detection algorithms (e.g. obstacle detection) is added to the World Model data structure. It is assumed that such algorithms cover the required steps to provide accurate and consistent information (i.e. they include solutions for conflicting or uncertain sensor data).

The second step, the updating of existing information, is required when perceived information matches an already existing one. In this case, only the perceived objects’ positions in the World Model need to be updated.

Obsolete information, such as static obstacles which have already been passed by the vehicle, can be removed cyclicly (step 3).

E. The World Model Application Programming Interface (API)

The World Model API consists of two entities, each implementing a different concept:

- World Model Events
- Object-oriented data structure.

1) *World Model Event Interface*: A World Model Event represents a discrete event which signals the availability of certain information, that certain conditions are suddenly met, or the occurrence of a certain event happening in the real world. The principle is along the line of Discrete Event Systems [12]. The state of a *World Model Event* is modeled as a boolean variable.

World Model Events are used to notify other software components (observers) about the state of certain predefined conditions, which are relevant for the execution of driving maneuvers.

The state of all defined World Model Events is provided as a k -tuple:

$$WM_{events} = (w_1, w_2, \dots, w_k) : w_l \in \{true, false\}, \quad (1)$$

where each element w_l ($l = 1, 2, \dots, k$) represents an event.

The k -tuple WM_{events} is updated cyclicly, and other software components are actively notified about the occurrence of certain conditions as defined for each event. A timestamp assigned to the event tuple may be used to ensure that the update operation is performed within the required time intervals.

2) *Object-Oriented Data Structure Interface*: Besides World Model Events, the World Model provides other software components full access to all its data in the following way. The control software application contains one single instance of the class “WorldModel”. Beginning with this single instance of “WorldModel”, all data can be obtained by successively retrieving contained or related classes. For instance, the “WorldModel” operation *getCurrentRoad()* returns the instance of the class “Road” which models the road on which the vehicle is currently driving. The current traffic lane can be retrieved by calling the operation *getCurrentLane()* on the current road instance, and so on.

F. Applied Software Design Patterns

1) *Factory Method*: The loading procedure of a priori information from an XML file (section III-C) requires the processing of information stored in certain XML nodes and accordingly the creation and instantiation of a large number of various class instances. Consequently, for this task, the *Factory Method* [13] design pattern is most adequate, as it enables to embed the source code for the creation of class instances into the classes itself.

In order to divide and simplify the process of loading the XML file, the XML file is divided into XML element nodes. Each XML element is delegated as a parameter to its corresponding World Model class. Therefore, each class contains only the implementation required to create an instance of itself from an XML element node.

2) *Singleton*: Multiple identical representations of the traffic environment at the same time are not necessary, and would lead to wasted computing resources. Therefore, in order to ensure that there exists only one single instance of the class “WorldModel”, this class is implemented as a *Singleton*. The *Singleton* design pattern [13] impedes the

recreation of a class instance if it already exists, while however enabling the easy access to this single instance.

3) *Observer*: One of the non-functional requirements for the World Model is a flexible way to exchange information with other subsystems. The World Model actively notifies other subsystems (e.g. Decision Making subsystem) about events of relevance happening in the real world. The World Model Event interface is based on the *Observer* design pattern, as this pattern is best suited for implementing a notification mechanism for multiple subsystems.

Observer is a behavioral design pattern which is ideal for notifying many dependent objects (observers) when the state of one object (subject) has changed.

In this context, the World Model plays the role of the subject, while observers are any software components interested in receiving notifications from the World Model, such as a decision making subsystem. Figure 4 shows the interaction diagram between the World Model and its observers. When the World Model changes its state, it informs all the registered observers. The observers, in turn, request more information if needed.

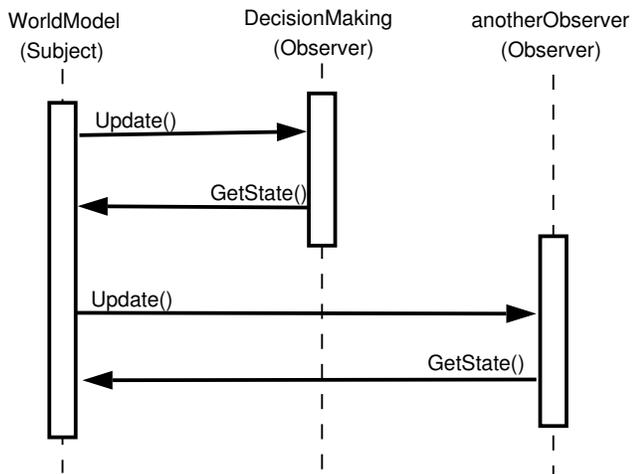


Fig. 4. UML interaction diagram of the *Observer* design pattern.

IV. INTEGRATION AND EXPERIMENTAL RESULTS

The presented World Model is part of a control software for autonomous city vehicles, developed at ICSL, which can be used to maneuver either a simulated autonomous vehicle, or a real one.

A. 3D Simulation

A 3D simulation environment [14] (Figure 5) has been used to test the integration of the World Model software component. The 3D simulation includes the simulation of a autonomous vehicle and its on-board sensors, as well as static and dynamic obstacles. The vehicle interface to the simulated vehicle is identical to the vehicle interface of a real experimental vehicle. The simulated traffic environment is the same as the real traffic environment at the test location for the real experimental vehicle at Griffith University, Nathan

campus. Therefore, the simulation results reflect real road traffic situations.

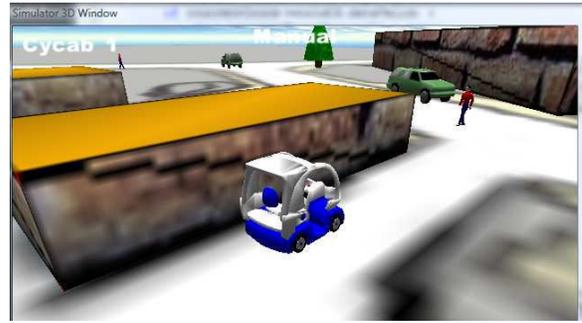


Fig. 5. The autonomous vehicle in the 3D simulation.

B. Experimental Tests using CyCabs

Experimental tests have been carried out at INRIA (France), using two CyCabs (Electric autonomous vehicles manufactured by Robosoft, France) [15] and a conventional car (Citroen C3). Although these experiments were focused on testing the real-time decision making approach developed at ICSL, due to the fact that the decision making subsystem fully relies on the World Model to receive its input information, the decision making results reflect the World Model capabilities. Details about the decision making approach have been published in [16], [17].

All vehicles, including the conventional car, were equipped with differential GPS (DGPS) and were able to communicate over a wireless communication network, exchanging information about their position and speed. In addition to its own GPS position, the autonomous vehicle received the GPS positions of the other two vehicles. Furthermore, the autonomous vehicle's World Model contained a priori information, such as roads, the position of intersections and positions of imaginary stop signs. All tests were conducted at low speed (around 1m/s).

Three different traffic scenarios have been set up, all showing a common decision situation:

- Scenario 1: passing a stopped vehicle without oncoming traffic
- Scenario 2: passing a stopped vehicle with oncoming traffic
- Scenario 3: waiting behind a vehicle at an intersection

In the first traffic scenario, the autonomous vehicle approached a stopped vehicle. Safe passing was possible, and the oncoming traffic lane was free of any obstacles. In this first scenario, the autonomous vehicle immediately started the passing maneuver when it approached the stopped vehicle.

In the second traffic scenario (Figure 6), a manually driven vehicle was oncoming, making safe passing impossible. The autonomous vehicle stopped and waited behind the stopped car, and began passing the stopped vehicle when the oncoming lane was free.

In the third traffic scenario, a manually driven vehicle was stopped at an intersection. The autonomous vehicle waited behind the stopped vehicle until it crossed the intersection. Then the autonomous vehicle continued driving, stopped at the imaginary stop sign and then continued crossing the intersection.



Fig. 6. Decision making experiment using the World Model as input. The autonomous CyCab (right) waits for the human-driven CyCab (left) to clear the oncoming lane before it passes the stopped car (black Citroen C3).

Although there were a number of remaining problems, such as the unreliable execution of driving maneuvers, the decision making subsystem was always able to avoid collisions with other vehicles and make appropriate driving decisions in real-time.

Therefore, since the real-time decision making subsystem received its input from the World Model, the World Model was able to provide sufficient information in real-time, in order to enable decision making and safe driving at the tested speed.

V. CONCLUSION

The presented world model for autonomous city vehicles fulfills its non-functional requirements as follows:

- World Model real-time performance measurements have been carried out, which show that the execution time for basic data management operations (i.e. inserting and retrieving of data) fulfills real-time requirements.
- Interoperability with various types of sensor components and communication subsystems is achieved through the definition of the World Model information in a generic, sensor-independent way.
- The object-oriented design allows to model the relationships between entities in an intuitive way, reflecting their relationship in the real traffic environment (e.g. obstacles on a road). Consequently, the object-oriented design simplifies the development process, while the application of software design patterns makes the World Model modular, thus easier to overview and understand.
- Portability to other autonomous vehicle control system architectures is achieved through a clearly defined API. Along with its flexible World Model Event interface,

the World Model, or even the entire Perception Subsystem can be easily integrated into other vehicle control system architectures, as long as they have a modular structure.

- Reusability of the World Model is possible in other applications, which have similar information requirements as those of a control system for autonomous city vehicles.

The main motivation of this work is to narrow the gap between the computer science and engineering disciplines in the research area of autonomous city vehicles. Being placed at their crossing point, the developed World Model focuses on software engineering methods, and enables the integration of sensor and perception technology, which is usually seen as an engineering task.

REFERENCES

- [1] J. Kolodko and L. Vlacic, "Cooperative autonomous driving at the Intelligent Control Systems Laboratory," *IEEE Intelligent Systems*, vol. 18, no. 4, pp. 8–11, 2003.
- [2] U. Zimmer and E. v. Puttkamer, "Comparing world-modelling strategies for autonomous mobile robots," in *IWK '94*, Ilmenau, Germany, 1994.
- [3] T. Hong, M. Abrams, T. Chang, and M. Shneier, "An intelligent world model for autonomous off-road driving," *Computer Vision and Image Understanding*, 2000.
- [4] J. S. Albus and A. J. Barbera, "Rcs: A cognitive architecture for intelligent multi-agent systems," in *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, IAV 2004*, 2004.
- [5] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Integrating perception and planning for autonomous navigation of urban vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, 2006.
- [6] SAFESPOT, "Safespot integrated project," 2010. [Online]. Available: <http://www.safespot-eu.org> [Last accessed: 13.05.2010]
- [7] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, 2008.
- [8] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, and D. Haehnel, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, 2008.
- [9] DARPA, "Sample route network definition file (rndf)," 29.07.2008 2007. [Online]. Available: [http://www.darpa.mil/GRANDCHALLENGE/docs/Sample_RNDF\(v1.5\).txt](http://www.darpa.mil/GRANDCHALLENGE/docs/Sample_RNDF(v1.5).txt)
- [10] B. J. Patz, Y. Papeis, R. Pillat, G. Stein, and D. Harper, "A practical approach to robotic design for the darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, 2008.
- [11] J. Leonard, J. How, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.
- [12] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, 2nd ed. Springer, 2008.
- [13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: elements of reusable object-oriented software*. Addison-Wesley, 1994.
- [14] S. Boisse, R. Benenson, L. Bouraoui, M. Parent, and L. Vlacic, "Cybernetic transportation systems design and development: Simulation software," in *IEEE International Conference on Robotics and Automation - ICRA'2007, Roma, Italy*, 2007.
- [15] G. Baille, P. Garnier, H. Mathieu, and P.-G. R., "Le cycab de l'inria rhone-alpes," Intitut National de Recherche en Informatique et en Automatique (INRIA), France, Tech. Rep. 0229, 1999.
- [16] A. Furda and L. Vlacic, "Towards increased road safety: Real-time decision making for driverless city vehicles," in *2009 IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, TX, USA, 2009.
- [17] —, "Multiple criteria-based real-time decision making by autonomous city vehicles," in *7th IFAC Symposium on Intelligent Autonomous Vehicles (IAV 2010)*, Lecce, Italy, 2010, accepted for publication.