

1st International Workshop on Autonomous Systems Design

ASD 2019, March 29, 2019, Florence, Italy

Edited by

Selma Saidi

Rolf Ernst

Dirk Ziegenbein



Editors

Selma Saidi

Hamburg University of Technology, Germany
selma.saidi@tuhh.de

Rolf Ernst

Technical University of Braunschweig, Germany
ernst@ida.ing.tu-bs.de

Dirk Ziegenbein

Robert Bosch GmbH, Germany
dirk.ziegenbein@de.bosch.com

ACM Classification 2012

Hardware → Analysis and design of emerging devices and systems; Computer systems organization → Robotic autonomy; Software and its engineering → Software safety; Computer systems organization → Dependable and fault-tolerant systems and networks

ISBN 978-3-95977-102-3

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-102-3>.

Publication date

March, 2019

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

License

This work is licensed under a Creative Commons Attribution 3.0 Unported license (CC-BY 3.0):
<https://creativecommons.org/licenses/by/3.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/OASlcs.ASD.2019.0

ISBN 978-3-95977-102-3

ISSN 1868-8969

<https://www.dagstuhl.de/oasics>

OASlcs – OpenAccess Series in Informatics

OASlcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASlcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Daniel Cremers (TU München, Germany)
- Barbara Hammer (Universität Bielefeld, Germany)
- Marc Langheinrich (Università della Svizzera Italiana – Lugano, Switzerland)
- Dorothea Wagner (*Editor-in-Chief*, Karlsruher Institut für Technologie, Germany)

ISSN 1868-8969

<https://www.dagstuhl.de/oasics>

Contents

Preface

<i>Selma Saidi, Rolf Ernst and Dirk Ziegenbein</i>	0:vii
--	-------

Regular Papers

Fault-Tolerance by Graceful Degradation for Car Platoons <i>M. Baha E. Zarrouki, Verena Klös, Markus Grabowski, and Sabine Glesner</i>	1:1–1:15
Feasibility Study and Benchmarking of Embedded MPC for Vehicle Platoons <i>Iñaki Martín Soroa, Amr Ibrahim, Dip Goswami, and Hong Li</i>	2:1–2:15
IDF-Autaware: Integrated Development Framework for ROS-Based Self-Driving Systems Using MATLAB/Simulink <i>Shota Tokunaga, Yuki Horita, Yasuhiro Oda, and Takuya Azumi</i>	3:1–3:9
Controlling Concurrent Change – A Multiview Approach Toward Updatable Vehicle Automation Systems <i>Mischa Möstl, Marcus Nolte, Johannes Schlatow, and Rolf Ernst</i>	4:1–4:15
Safety and Security Analysis of AEB for L4 Autonomous Vehicle Using STPA <i>Shefali Sharma, Adan Flores, Chris Hobbs, Jeff Stafford, and Sebastian Fischmeister</i>	5:1–5:13
Towards a Formal Model of Recursive Self-Reflection <i>Axel Jantsch</i>	6:1–6:15

Interactive Presentations

A Dependable Detection Mechanism for Intersection Management of Connected Autonomous Vehicles <i>Rachel Dedinsky, Mohammad Khayatian, Mohammadreza Mehrabian, and Aviral Shrivastava</i>	7:1–7:13
TrueView: A LIDAR Only Perception System for Autonomous Vehicle <i>Mohammed Yazid Lachachi, Mohamed Ouslim, Smail Niar, and Abdelmalik Taleb-Ahmed</i>	8:1–8:10
Generation of a Reconfigurable Probabilistic Decision-Making Engine based on Decision Networks: UAV Case Study <i>Sara Zermani and Catherine Dezan</i>	9:1–9:14

Preface

This volume contains the proceedings of the 1st International Workshop on Autonomous Systems Design (ASD 2019). The workshop is held in Florence, Italy on March 29, 2019, and is co-located with the 22nd Design, Automation and Test in Europe Conference (DATE 2019). ASD 2019 aims at exploring recent industrial and academic trends, methods and methodologies in autonomous systems design. The workshop is organized into regular sessions with peer-reviewed research papers selected from an open call, complemented by 4 invited talks and two distinguished keynotes. The presented contributions addressed different topics on robotics, automated driving and frameworks for autonomous systems like Robot Operating System (ROS) and AUTOSAR Adaptive.

Selected papers are included in this volume and are categorized into 6 long papers and 3 interactive presentations. The presented papers discuss recent development approaches for autonomous systems involving the integration of ROS-based self-driving system (Autoware) using MATLAB/Simulink, advanced implementations of model predictive control systems and multi-view model-based design and verification approaches. Another important discussed topic is related to dependable autonomous systems design based on degradation cascades for sensor and communication failures in autonomous car platoons, applying STPA-based (System Theoretic Process Analysis) hazard analysis technique for the design of robust autonomous emergency braking systems under safety and security requirements, and the incorporation of self-awareness in the design of autonomous systems using dynamic formal data flow semantics.

The first invited talk will focus on the next generation of ROS frameworks developed to address the main challenge of seamless integration of deeply embedded devices considering resource-constrained computing platforms, non-ideal networks and real-time requirements. The second invited talk will address the dependability challenge by providing reliable control solutions in cloud computing provided under formal guarantees. The two last invited talks, are dedicated to present recent research activities and derived findings of research and industrial clusters in the field of autonomous driving. The activities of two large projects in the field will be presented, namely the UNICARagil project to demonstrate disruptive modular architectures for agile automated vehicle concept, and the CCC (Controlling Concurrent Change) project to investigate automated integration of critical applications using self-adaptation with self-protection based on contracting and self-awareness.

The workshop will host two distinguished industrial keynotes highlighting important challenges and recent trends in the fields of autonomous design. In his keynote "Challenges of Automated and Connected Driving", Thomas Form, Head of Electronics and Vehicle Research at Volkswagen AG, will talk about the challenges in automated driving regarding sensor technologies, redundancies as well as verification and validation questions. Masaki Gondo, CTO at eSOL, the company that provides POSIX/AUTOSAR/TRON RTOS will talk about AUTOSAR Adaptive as a standardized software platform specification for the highly automated and autonomous driving and emphasize the role of OS architectures in coping with recent challenges in the field.

This volume will present a short summary of the considered keynotes and invited talks in addition to the selected long and interactive presentation papers.

■ List of Contributions

Keynote 1: Challenges of Automated and Connected Driving (Thomas Form)

Keynote 2: AUTOSAR Adaptive - Challenging the Impossible (Masaki Gondo)

Invited Talk 1: Bringing the Next Generation Robot Operating System on Deeply Embedded Autonomous Platforms (Ralph Lange)

Invited Talk 2: Autonomous Data Center - Feedback Control for Predictable Cloud Computing (Martina Maggio)

Invited Talk 3: An Approach to Automotive Service-oriented Software Architectures in a Multi-partner Research Project (Stefan Kowalewski)

Invited Talk 4: Controlling Concurrent Change- Design Automation for Critical Systems Integration (Rolf Ernst)

Regular Paper 1: IDF-Autaware: Integrated Development Framework for ROS-based Self-driving Systems Using MATLAB/Simulink (Shota Tokunaga et al.)

Regular Paper 2: Feasibility Study and Benchmarking of Embedded MPC for Vehicle Platoons (Inaki Martin Soroa et al.)

Regular Paper 3: A Multiview Approach Toward Updatable Vehicle Automation Systems (Marcus Nolte et al.)

Regular Paper 4: Fault-Tolerance by Graceful Degradation for Car Platoons (Mohammed Baha E. Zarrouki et al.)

Regular Paper 5: Safety and Security Analysis of AEB for L4 Autonomous Vehicle using STPA (Shefali Sharma et al.)

Regular Paper 6: Towards a Formal Model of Recursive Self-Reflection (Axel Jantsch)

Interactive Presentation 1: A Dependable Detection Mechanism for Intersection Management of Connected Autonomous Vehicles (Rachel Dedinsky et al.)

Interactive Presentation 2: A LIDAR Only Perception System for Autonomous Vehicle (Mohamed Yazid Lachachi et al.)

Interactive Presentation 3: Generation of a Reconfigurable Probabilistic Decision Making Engine based on Decision Networks: UAV Case Study (Sara Zermani et al.)

■ Keynotes & Invited Talks

Keynote 1: Challenges of Automated and Connected Driving

Speaker: Thomas Form, Head of Electronics and Vehicle Research, Volkswagen AG, Germany

In recent years, various publications and presentations from a lot of companies have shown the improvements in the sector of automated driving. The vehicle- and mobility-concept SEDRIC is a current example from the Volkswagen AG. However, for a release of these technologies there are several unresolved issues regarding sensor technologies, redundancies as well as verification and validation questions. Regarding sensors, the main objectives are miniaturization and reduction of system costs. Advantages and disadvantages of existing solutions have to be evaluated. In addition to economic aspects, ensuring the redundancy of the system is absolute necessary. Is, for example, Artificial Intelligence able to provide an independent second or third function path? Regarding verification and validation concepts, current discussions are focused on which scenarios have to be tested and how, in order to enable regulatory authorities to approve the release of automated driving functions? It is conceivable, that this is an automotive industry wide task that can only be solved in cooperation with all stakeholders.

Keynote 2: AUTOSAR Adaptive - Challenging the Impossible

Speaker: Masaki Gondo, Software CTO at eSOL Co., Ltd., Japan

The vast researches related to autonomous driving seem steadily progressing - it no longer makes news to just have some research vehicle drive autonomously. However, bringing this technology to the market, with all the associated legal, societal, and ethical responsibilities, with justifiable cost efficiency, is hard at its best, and impossible at its worst. Furthermore, the automotive industry is facing drastic challenges in electric vehicles, connected services, which also heavily impact the whole vehicle architecture. AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of automotive interested parties. One of its latest challenges is to develop the software platform specification for the highly automated and autonomous driving, named AUTOSAR Adaptive Platform. This talk gives an overview of the challenges of such a platform, followed by the solution approach of AUTOSAR reflecting the industrial needs, and the overall architecture of AUTOSAR Adaptive. It also introduces a new multi-kernel OS technology the author develops, describing why such OS architecture is essential for coping with the challenge in the long run.

Invited Talk 1: Bringing the Next Generation Robot Operating System on Deeply Embedded Autonomous Platforms

Speaker: Ralph Lange, Robert Bosch GmbH, DE

In the last decade, the Robot Operating System (ROS) has become the primary framework and middleware for robotics research and an important building block for the autonomous systems engineering in general. The Next Generation Robot Operating System (ROS 2) aims at strengthening this position by new mechanisms for resource-constrained computing platforms, non-ideal networks, real-time requirements and further fundamental needs from series development of autonomous systems. A particular challenge is the seamless integration of deeply embedded devices with ROS 2. In this talk, an overview to ROS 2 will be provided, followed by an analysis of basic issues for such seamless integration. As a solution, the micro-ROS stack will be presented in the second part of this talk. This includes an introduction to the up-coming DDS-XRCE middleware standard, a novel concept of system runtime configuration for ROS2 and micro-ROS, and early results on an extended API for predictable scheduling.

Invited Talk 2: Autonomous Data Center - Feedback Control for Predictable Cloud Computing

Speaker: Martina Maggio, University of Lund, SE,

Cloud computing gives the illusion of infinite computational capacity and allows for on-demand resource provisioning. As a result, over the last few years, the cloud computing model has experienced widespread industrial adoption and companies like Netflix offloaded their entire infrastructure to the cloud. However, with even the largest datacenter being of a finite size, cloud infrastructures have experienced overload due to overbooking or transient failures. In essence, this is an excellent opportunity for the design of control solutions, that tackle the problem of mitigating overload peaks, using feedback from the computing infrastructure. Exploiting control-theoretical principles and taking advantage of the knowledge and the analysis capabilities of control tools, it is possible to provide formal guarantees on the predictability of the cloud platform. This talk introduces recent research advances on feedback control in the cloud computing domain. This talk discusses control solutions and future research for both cloud application development, and infrastructure management. In particular, it covers application brownout, control-based load-balancing, and autoscaling.

Invited Talk 3: An Approach to Automotive Service-oriented Software Architectures in a Multi-partner Research Project

Speaker: Stefan Kowalewski, RWTH Aachen, DE

Novel software architectures will become necessary to cope with the short lifetime and innovation cycles of the technologies underpinning self-driving vehicles. In the UNICARagil project, seven German universities and six industrial partners join forces to research and demonstrate disruptive modular architectures for agile, automated vehicle concepts. As today's prevailing automotive electric, electronic and software architectures are mostly function-oriented and design-time integrated, they often are unsuitable for infield updates or system reconfiguration. In contrast, service-oriented software architectures are based on runtime integrated service and are a promising way forward. We present the lean and simple concept for service-orientation, that serves as the basis for the implementation of all vehicle functions in the UNICARagil vehicles.

Invited Talk 4: Controlling Concurrent Change- Design Automation for Critical Systems Integration

Speaker: Rolf Ernst, TU Braunschweig, DE

Embedded systems for safety critical and high availability applications have moved from isolated components to highly integrated mixed criticality networked systems with numerous shared resources. The resulting function interference challenges the design process, in particular in autonomous systems which shall independently manage software updates and hardware reconfigurations. With support from the German DFG, a group of 8 PIs has investigated automated integration of critical applications using self-adaptation with self-protection based on contracting and self-awareness. Applications were driving automation and space robotics. The talk will review the results of the six year project and outline the demonstrations which will be exhibited at the workshop.

Support and Acknowledgement

We would like to thank our distinguished keynote speakers as well as invited speakers for their valuable participation to the workshop. We would also like to acknowledge the contributions of authors and the help of program committee members in the review process. We extend our appreciation to the DATE organizing committee and K.I.T Group GmbH Dresden for their help and support in the logistics arrangements. ASD 2019 is partially supported by the IEEE Council on Electronic Design Automation CEDA and the Technical University of Braunschweig, we would therefore like to acknowledge their financial contribution. Lastly, we would like to thank the editorial board of the OASICs proceedings.

Organizers

Workshop Organizers

Rolf Ernst, Technical University of Braunschweig, Germany
Selma Saidi, Hamburg University of Technology, Germany
Dirk Ziegenbein, Robert Bosch GmbH, Germany

Publicity Chair

Sebastian Steinhorst, Technical University of Munich, Germany

Steering Committee

Marko Bertogna, University of Modena, Italy
Naehyuck Chang, Korea Advanced Institute of Science and Technology, Korea
Klaus Dietmayer, Ulm University, Germany
Karl Henrik Johansson, KTH Royal Institute of Technology, Sweden
Matthias Traub, BMW, Germany

Technical Program Committee

Assad Al Alam, Scania CV AB, Sweden
Mohammad Al Faruque, University of California, Irvine, USA
Marko Bertogna, University of Modena, Italy
Bart Besselink, University of Groningen, Netherlands
Samarjit Chakraborty, Technical University of Munich, Germany
Jyotirmoy Vinay Deshmukh, University of Southern California, USA
Marco Di Natale, Scuola Superiore Sant'Anna, Italy
Sebastian Fischmeister, University of Waterloo, Canada
Arne Hamann, Robert Bosch GmbH, Germany
Stefan Kowalewski, RWTH Aachen University, Germany
Martina Maggio, Lund University, Sweden
Rahul Mangharam, University of Pennsylvania, USA
Markus Maurer, TU Braunschweig, Germany
Karl Meinke, KTH Royal Institute of Technology, Sweden
Gabor Orosz, University of Michigan, USA
Dongwha Shin, Yeungnam University, Korea
Aviral Shrivastava, Arizona State University, USA
Lin Xue, Northeastern University, USA



■ List of Authors

Takuya Azumi, Graduate School of Science and Engineering, Saitama University, Japan
Rachel Dedinsky, Arizona State University, USA
Catherine Dezan, Lab-STICC, France
Rolf Ernst, Technische Universität Braunschweig, Germany
Sebastian Fischmeister, University of Waterloo, CA, Canada
Adan Flores, University of Waterloo, Canada
Sabine Glesner, Technische Universität Berlin, Germany
Dip Goswami, Eindhoven University of Technology, Netherlands
Markus Grabowski, Assystem Germany GmbH, Germany
Chris Hobbs, QNX Software Systems Limited, Canada
Yuki Horita, Hitachi Automotive Systems Ltd, Japan
Amr Inbrahim, Eindhoven University of Technology, Netherlands
Axel Jantsch, Technische Universität Wien, Austria
Mohammad Khayatian, Arizona State University, USA
Verena Klös, Technische Universität Berlin, Germany
Mohammed Yazid Lachachi, University of Sciences and Technology, Oran, Algeria
Hong Li, NXP Semiconductor, Netherlands
Mohammadreza Mehrabian, Arizona State University, USA
Mischa Möstl, Technische Universität Braunschweig, Germany
Smail Niar, Université Polytechnique Hauts-de-France, France
Marcus Nolte, Technische Universität Braunschweig, Germany
Yasuhiro Oda, Hitachi Automotive Systems Ltd, Japan
Mohamed Ouslim, University of Sciences and Technology, Oran, Algeria
Johannes Schlatow, Technische Universität Braunschweig, Germany
Shefali Sharma, University of Waterloo, Canada
Aviral Shrivastava, Arizona State University, USA
Inaki Martin Soroa, Eindhoven University of Technology, Netherlands
Jeff Stafford, Renesas Electronics America Inc., Canada
Abdelmalik Taleb-Ahmed, Université Polytechnique Hauts-de-France, France
Shota Tokunaga, Graduate School of Engineering Science, Osaka University, Japan
Mohammed Baha E. Zarrouki, Technische Universität Berlin, Germany
Sara Zermani, Lab-STICC, France

Fault-Tolerance by Graceful Degradation for Car Platoons

M. Baha E. Zarrouki

TU Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
baha_zarrouki@mail.tu-berlin.de

Verena Klös

TU Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
verena.kloes@tu-berlin.de

Markus Grabowski

Assystem Germany GmbH, Gutenbergstraße 15, 10587 Berlin, Germany
mgrabowski@assystemtechnologies.com

Sabine Glesner

TU Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
sabine.glesner@tu-berlin.de

Abstract

The key advantage of autonomous car platoons are their short inter-vehicle distances that increase traffic flow and reduce fuel consumption. However, this is challenging for operational and functional safety. If a failure occurs, the affected vehicles cannot suddenly stop driving but instead should continue their operation with reduced performance until a safe state can be reached or, in the case of temporal failures, full functionality can be guaranteed again. To achieve this degradation, platoon members have to be able to compensate sensor and communication failures and have to adjust their inter-vehicle distances to ensure safety. In this work, we describe a systematic design of degradation cascades for sensor and communication failures in autonomous car platoons using the example of an autonomous model car. We describe our systematic design method, the resulting degradation modes, and formulate contracts for each degradation level. We model and test our resulting degradation controller in Simulink/Stateflow.

2012 ACM Subject Classification Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Availability; Software and its engineering → Software design engineering

Keywords and phrases fault-tolerance, degradation, car platoons, autonomous driving, contracts

Digital Object Identifier 10.4230/OASICS.ASD.2019.1

1 Introduction

In autonomous car platoons vehicles drive with short inter-vehicle distances to increase traffic flow and reduce fuel consumption by travelling in the slipstream. The short distances can be achieved by exploiting real-time knowledge about the driving behaviour of preceding vehicles in the platoon. This knowledge is achieved by combining onboard sensors and wireless communication with platoon members. If a sensor or communication failure occurs, the required knowledge becomes unavailable and driving within a short distance is not safe anymore. In contrast to fail-safe systems, where a shut-down of actuators leads to a safe system state, autonomous vehicles have to be fail-operational, i.e. a shut-down of the vehicle during operation on a highway is not acceptable. Thus, failures have to be compensated and adherence to safety restrictions has to be guaranteed even under failure occurrence. For autonomous car platoons, this means that the inter-vehicle distance always has to be large enough to allow for an autonomous reaction to a sudden braking of preceding vehicles



© M. Baha E. Zarrouki, Verena Kös, Markus Grabowski, and Sabine Glesner;
licensed under Creative Commons License CC-BY

Workshop on Autonomous Systems Design (ASD 2019).

Editors: Selma Saidi, Rolf Ernst, and Dirk Ziegenbein; Article No. 1; pp. 1:1–1:15

OpenAccess Series in Informatics

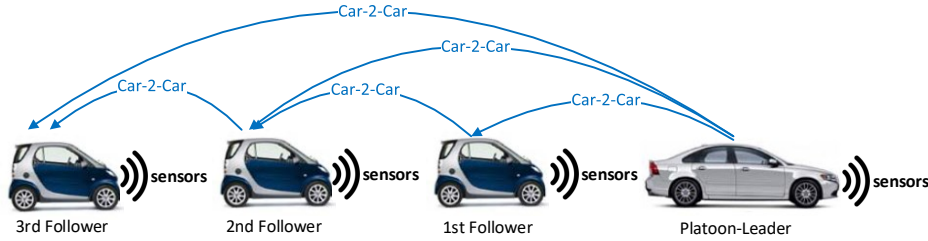


OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

without risking a collision. The required reaction time depends on the quality and speed of information about the behaviour of preceding vehicles, i.e. available sensors and inter-vehicle communication, and has to be reflected in the distance. One possibility to specify safe and fault-tolerant driving behaviour for autonomous platoons is to rely on graceful degradation. This means to systematically define a partial-order of less and less acceptable operation modes and to select the best achievable mode in presence of failures. Thus, the system maintains its operation as far as possible. As an example, an autonomous vehicle can be in the platoon-mode or in a mode of cooperative adaptive cruise control (CACC), where it only communicates with the vehicle directly in front of it, or in the mode of adaptive cruise control (ACC), where it relies on onboard sensors only. An operating mode in which the system is not operated at full functionality due to failures is called *degradation mode* and a sequence of less and less acceptable operation modes is called *degradation cascade*.

With the increasing complexity of vehicles, e.g. due to increasing functionality required for autonomous driving, the design of degradation cascades becomes challenging, i.e. managing the resulting amount of failure combinations. In [4], a systematic approach for design and verification of degradation cascades for embedded systems was presented. Following a systematic process can help to cope with the increasing complexity. Inspired by this work, we describe a systematic design of degradation cascades for sensor and communication failures in autonomous car platoons using the example of the autonomous model car “Velox” which serves as a case study in the AMASS research project¹. Note that although this paper focuses on safety, the proposed solutions can also be used to cope with security, i.e. attacks on sensors, actuators and communication channels, as long as these attacks are identified by some anomaly detection mechanisms (see [1] for an overview). We describe our systematic design method, the resulting degradation modes, and formulate contracts² for each degradation level. We model and test our resulting degradation controller in Simulink/Stateflow.

Car Platoons



■ **Figure 1** A platoon-drive.

In this paper, we focus on autonomous car platoons with at least two vehicles that drive with a small inter-vehicle distance on one lane of the highway. They synchronize their speed and sensor data based on onboard sensors and Vehicle-to-Vehicle (V2V) communication, as depicted in Figure 1. The platoon members can have different vehicle types (cars, trucks) from different manufacturers. The vehicle at the front of a platoon is the platoon leader.

¹ <https://www.amass-ecsel.eu/>

² Contracts are pairs of assumptions and guarantees that define the behaviour of individual system components to lower the overall complexity of large systems.

Based on user-inputs, it determines an efficient velocity and a desired inter-vehicle distance for the whole platoon. As the safety of inter-vehicle distances depends on individual vehicle characteristics (i.e. maximum braking deceleration), we assume that each platoon member adjusts its distance control setpoint such that it is always larger or equal to the individual safe inter-vehicle distance, and never smaller than the desired distance proposed by the platoon leader. Furthermore, the leader keeps a safe distance to vehicles or platoons in front of it. Each following vehicle is a follower (or platoon member) and autonomously has to maintain the inter-vehicle distance determined by the platoon leader. This is achieved by longitudinal control, which regulates the speed of the vehicle. The vehicles drive fully autonomously with a driver who is not prepared to take control, i.e. as if no driver were present.

Outline

This paper is structured as follows: In Section 2, we discuss related work on degradation concepts for car platoons. In Section 3, we systematically analyze the system architecture, identify relevant failures for platooning and discuss how to adapt the inter-vehicle distance to capture slower response times due to less precise information. The results are used to infer degradation modes for sensor and communication failures. Based on these reactions, we propose a compact degradation controller that we model in Simulink/Stateflow in Section 4. To ensure the safety of autonomous platoons that use graceful degradation, we define contracts and discuss how they can be analyzed and tested with our Stateflow model in Section 5. We conclude the paper in Section 6 and outline future work.

2 Related Work

In this section, we review related work on fault-tolerant designs and degradation strategies for CACC and platooning.

The work in [7] proposes a diagnostic system that monitors the sensors of the longitudinal and lateral controllers in autonomous vehicles that operate in a platoon. However, it does not detect communication failures and the authors do not define a reaction to the detected failures. In our work, we assume that sensor and communication failures are successfully detected and focus on appropriate and safe reactions to detected faults.

The work in [6] presents a graceful degradation technique for CACC in case of communication failures. The main idea is to estimate the acceleration of the preceding vehicle based on distance measurement information provided by onboard sensors. This strategy shows better performance than using ACC as a fallback option. This paper only considers one possible failure of CACC-mode which is the communication with the preceding vehicle and does not handle other failures i.e. distance measurement sensor failures that affect the estimation. In our work, we cover all possible sensor and communication failures affecting the longitudinal guidance in platoon-mode. Moreover, we define reactions and design a global state-machine-model that guides the vehicle completely-autonomously in a running platoon-drive. In our approach, we rely on a similar distance measurement as described in [6] in case of communication failures with the preceding vehicle.

Our work is similar to and based on the work presented in [9], which introduces a structured design of degradation cascades for car platoons and a contract-based design approach to ensure safety. The presented degradation cascade only switches between Platoon, CACC, ACC and manual driving. In our work, there is no possibility for a fallback to manual

driving as we handle fully automated vehicles (SAE Level 5³). Moreover, we introduce new degradation modes that allow the vehicle to handle more failure combinations and to deliver a better performance in the platoon.

Our work is inspired by the systematic approach to define, design, implement and verify degradation cascades for embedded systems presented in [4]. The approach proceeds systematically from the analysis of the system and its safety over defining degradation cascades and its requirements down to modelling and generating code from Simulink for real-time testing. The systematic approach is illustrated with a DC drive example. In our work, we apply a similar approach on autonomous vehicles in a platoon-drive.

3 Systematic Analysis of Possible Failures and Alternative Information Sources

In the regular industrial design process, the system design is followed by a safety analysis to check whether the system adheres to safety requirements. For safe-operational systems that rely on graceful degradation, possible failures already have to be considered during the design of degradation modes. Thus, we follow [4], and already perform a systematic safety analysis before the system design. With this minor change of the usual design flow, the systematic design of degradation cascades can be easily integrated into industrial practice.

Our systematic design approach for degradation cascades consists of a systematic analysis of the system architecture to identify a) failure sources that are relevant for platooning, and b) fallback alternatives for faulty system components. To capture the performance loss of fallback alternatives, we define individual failure-specific constants that we add to the reaction time of degraded vehicles. The reaction time describes how long it takes to detect and react to a sudden change in the behaviour of the preceding vehicle. It is used to define the safe inter-vehicle distance $x_{rd,i}$ between the i -th vehicle and the preceding vehicle:

$$x_{rd,i}(t) = x_r + t_{r,i} * v_i \quad (1)$$

The parameters of the equation are the remaining distance at standstill x_r , the reaction time $t_{r,i}$, and the current velocity v_i of vehicle i . This distance law assumes similar vehicle velocities and accelerations in a platoon drive.

Our failure-specific constants describe the additional time that is needed to detect sudden breaks in the presence of specific failures. They are added to the reaction time $t_{r,i}$ in equation 1. Based on our analysis results, we define degradation cascades that describe several degradation steps as a response to sequences of failures. We combine these cascades into a single and compact degradation controller, which we model in Simulink/Stateflow to enable simulation, testing and later controller synthesis.

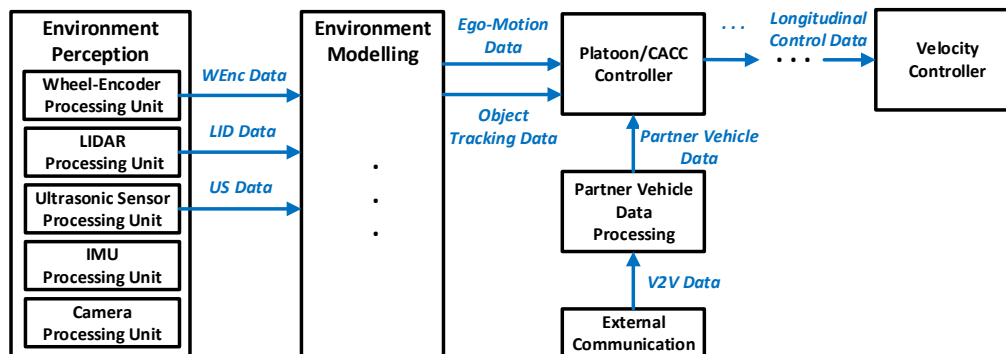
In this section, we first describe relevant parts of the system architecture of the “Velox” car in Section 3.1. To identify degradation modes for sensor and communication failures, we identify possible sources for required information about the environment, e.g. the distance to the vehicle in front, and about other platoon members, e.g. the velocity of the vehicle in front, based on available sensors and communication partners, and evaluate the influence of alternative information sources on the reaction time of the autonomous vehicle in Section 3.2.

³ The SAE Norm [8] defines six levels of autonomy for motor vehicle automation ranging from no automation (level 0) to full automation (level 5)

3.1 System Architecture

Within the scope of research projects, experimental cars on a scale of 1:8, the “Velox Cars”, were developed by “Assystem Germany GmbH”. The prototypes were designed for training and research purposes as well as for the development of advanced functions of highly automated systems (HAS) with a focus on autonomous driving. Various driving assistance functions such as a lane departure warning system, the ACC/CACC/Platoon or the Traffic Sign Assistant, have been developed. For this purpose, the model cars are equipped with a valuable range of sensors. In this work, the analysis is based abstractly on the architecture of the Velox car, which is generally similar to other vehicles.

In this paper, we assume a vehicle architecture as depicted in Figure 2. Each vehicle is equipped with ultrasonic sensors and a LIDAR (*Light Detection And Ranging*) radar to measure the distance to the vehicle in front. A sensor fusion of the ultrasonic sensor data and the LIDAR data is performed in order to detect measurement errors. In addition, the vehicles are equipped with an odometry unit to determine their own driving behaviour (acceleration, velocity, distance covered and position). A camera and an inertial measurement unit (IMU) are implemented on the vehicle. However, we assume that they are not used by the longitudinal controller for the normal Platoon/CACC function. Note that we here assume a similar architecture and equipment of all platoon members. However, the approach could also include vehicles that are not fully equipped with sensors or communication devices by handling these vehicles as vehicles with corresponding sensor or communication failures.



■ **Figure 2** A simplified vehicle architecture for platooning.

The signal flow in Figure 2 goes from the input on the left to the output on the right. The input corresponds to the sensors, and the output corresponds to the actuator control. All sensor data is processed in the corresponding Processing Unit in the **Environment Perception** layer. This layer contains function blocks such as the **Wheel-Encoder Processing Unit**, which acquires the odometer data (steering angle and wheel encoder data) and preprocesses the odometer data. The IMU is only used if the wheel encoder fails. In the subsequent **Environment Modelling** layer, the environment is modelled with the help of different algorithms. This layer contains function blocks such as a Vehicle Detection block. The **Platoon/CACC Controller** decides between platoon and CACC mode and calculates a target velocity to maintain the desired distance between the ego-vehicle and the vehicle in front. These decisions are based on the partner’s and the ego-vehicle data. The partner vehicle data are received by the **External Communication** interface for the communication between the own vehicle and external systems (Vehicle-to-Vehicle (V2V))

communication, GPS etc.). The received V2V data is then filtered by **Partner Vehicle Data Processing**. Furthermore, the Platoon/CACC Controller broadcasts the Ego vehicle data. The **Velocity Controller** regulates the speed of the vehicle and controls the motor based on the provided target velocity. We design our degradation concept for the Velox Car. In practice, it can be assumed that vehicles of different manufacturers participate in a platoon. Thus, the platoon can be regarded as a heterogeneous system of systems of different manufacturers. In order to realize the functionality of platoon driving, all vehicles have to be equipped with relative distance and speed sensors as well as V2V communication. Our results can easily be transferred to similar architectures.

3.2 Failures, Consequences, and Fallback Strategies

For the platoon operation, a Velox car needs the following capabilities: it can determine the distance to the vehicle in front, it can measure its own (ego) motion data (acceleration, speed, distance covered and position), and it can receive the motion of the vehicle in front and of the first vehicle in the platoon, i.e. the platoon leader. In normal platoon operation, the distance to the vehicle in front is determined by LIDAR and ultrasonic (US) sensor fusion, the calculation of the ego-motion is based on the wheel encoder, information about the front-vehicle motion is available via vehicle-2-front-vehicle communication, and the motion of the platoon leader is available via vehicle-2-leader communication.

■ **Table 1** Default and Fallback Information Sources.

Kind of Information	Default Source	1st Fallback Source	2nd Fallback Source
Distance to vehicle in front	LIDAR and Ultrasonic sensor fusion	LIDAR only, Ultrasonic only	GPS and Wheel-Encoder (for communication packet identification only)
Ego-Motion	Wheel-Encoder	Inertial-Measurement-Unit (IMU)	Motor-model for speed estimation
vehicle in front motion	vehicle-2-Front vehicle (V2F)	vehicle-2-Leader (V2L)	based on distance measurement
Motion of Platoon Leader	V2L	V2F	-

In the case of sensor and communication failures, the required information has to be obtained from alternative sources. In Table 1, we summarize available information sources in the Velox car and introduce new sources for the fallback scenario. If, for example, the LIDAR fails, ultrasonic data can be used without sensor fusion. However, the obtained information is less precise. If this sensor also fails, the system can still rely on the second fallback option: a combination of GPS and wheel-encoder that is precise enough to identify communication packets from the vehicle in front, but not for ACC-mode.

In the following, we describe the identified fallback possibilities for sensor and communication failures and their influence on the reaction time to changes in the behaviour of preceding vehicles. The resulting delays are expressed in terms of failure-specific constants. These constants have to over-approximate the actual delays as precisely as possible to ensure that the distance to the vehicle in front is as small as safely possible. The actual values depend on the characteristics of the sensors, actuators and the efficiency of the algorithms (fusion of data, recognition etc.) and can be determined with simulations and real-time tests.

Failure in Distance Measurement

In case of a failure of the LIDAR (LID_F) or the US sensor (US_F), it is no longer possible to use sensor fusion to increase precision, but we can still rely on the remaining sensor. As a result, the reaction time increases due to less precise sensor data. We capture this by adding a constant failure-specific factor, e.g. tr_{lid1} for LIDAR failures and tr_{us1} for US failures, that captures the necessary increase of the reaction time, to the reaction time in the normal platoon operation tr_{pl} . Thus, we get $tr_{pl}+tr_{lid1}$ and $tr_{pl}+tr_{us1}$, respectively. It applies $tr_{lid1} > tr_{us1}$, since the LIDAR is more accurate than the ultrasonic sensor.

If both distance sensors fail, the system can still rely on a combination of GPS and wheel-encoder to identify communication packets from the vehicle in front. In this case, tr_{lid_us} is added to the reaction time. However, this is not precise enough for CACC-mode.

Failure in estimating the Ego-motion

An error in the wheel encoder (WEnc_F) affects the acquisition of the covered distance, the own acceleration and speed (Ego-Motion-Data). If a wheel encoder error occurs, the system can still obtain 3-axis acceleration data from the motion sensor (inertial measuring unit (IMU)). However, this information is less precise. Accordingly, the reaction time increases to $tr_{pl}+tr_{wenc}$. If the IMU fails (IMU_F), it is still possible to roughly estimate the own velocity based on electrical values by using a motor model as discussed in [4]. The reaction time has to be increased to $tr_{pl}+tr_{wenc_imu}$.

Communication Failures

In the platoon operation, each vehicle communicates with the platoon leader and the vehicle in front of it. If the communication with the vehicle in front fails (V2F_F), the vehicle in front motion can only be estimated using the slower distance measurement, which means that the reaction times, and, thus, the safety distance, have to be increased when switching to the corresponding ACC mode. However, if communication to the platoon leader is still available, the platoon leader could be requested to forward messages from the vehicle in front. To this end, an extended packet filtering algorithm has to be implemented in the *Partner Vehicle Data Processing*.

If an error in the communication with the platoon leader (V2L_F) occurs (e.g. error during a sending procedure or error with the reception), we lose any information about the leader's driving behaviour. As a consequence, we would need to switch to the CACC mode and increase the inter-vehicle distance accordingly. However, if the vehicle in front is still able to communicate with the platoon leader and the ego vehicle has a stable communication connection to the vehicle in front, this vehicle can forward messages from the platoon leader. As message forwarding introduces some communication delay, reactions to sudden changes in the behaviour of the vehicle in front / the leader will also be delayed. Thus, the reaction time increases to $tr_{pl} + tr_{front}$ and $tr_{pl} + tr_{lead}$, respectively.

Based on the results of a systematic evaluation of available fallback possibilities for relevant sensor and communication failures, we define corresponding degradation modes for each combination of evaluated failures in the next section.

4 Degradation Controller

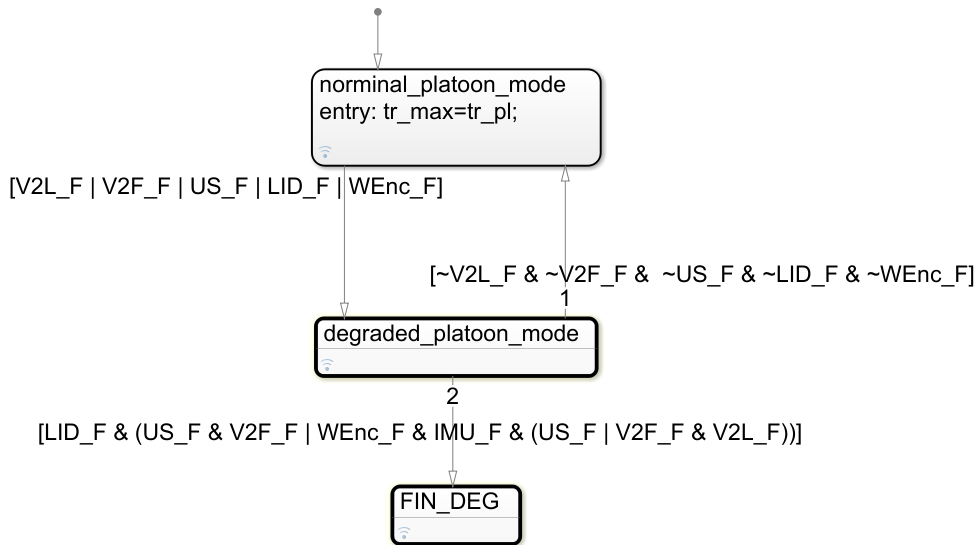
With our systematic analysis, we have identified fallback alternatives to compensate for sensor and communication failures within a running platoon-drive. In this section, we use the analysis results to infer degradation modes that describe less and less acceptable operation modes for platoon members. We propose a controller, modelled in Simulink/Stateflow, that safely guides the graceful degradation. Note that we assume the existence of a reliable and sufficiently fast fault detection method that we can rely on. This, of course, is not trivial, but out of the scope of this work. Our controller receives fault detection events from the fault detector and selects the best mode based on these events. The Stateflow model is used for systematic testing in the next section and can be further refined for controller code synthesis.

4.1 Degradation Cascades

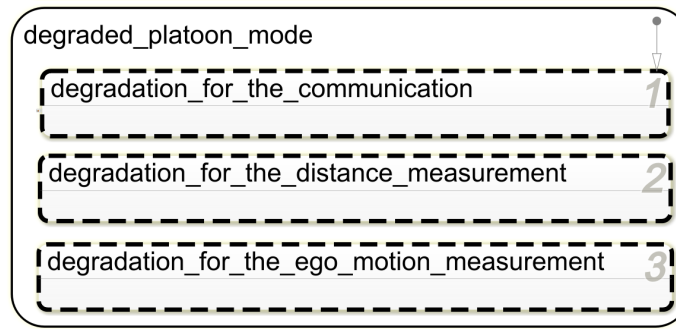
In case of failure combinations of different information types, our fallback strategies from the previous section can be combined step-by-step as long as information alternatives are available. If only knowledge about the platoon leader is missing, the vehicle can still switch to CACC and rely on vehicle-2-front-vehicle communication. If this communication also fails, only onboard sensor-based ACC may be possible. However, degradation is only possible as long as any acceptable operation mode can be executed. Thus, the last degradation step leads to a final degradation mode FIN_DEG, which describes a safe exit from the platoon. At this point, the system cannot rely on important sensor information and may not be able to communicate with other vehicles. Thus, it should come to a standstill as far to the right as possible, brake and warn other road users.

4.2 Controller Model

The Stateflow model, as depicted in Figure 3, consists of three hierarchical modes. The system starts in the nominal platoon mode. Failures cause a transition to the degraded platoon mode. If failures are resolved, it returns to nominal platoon mode. If no further acceptable degradation is possible, the last step leads to the final degradation mode (FIN_DEG).

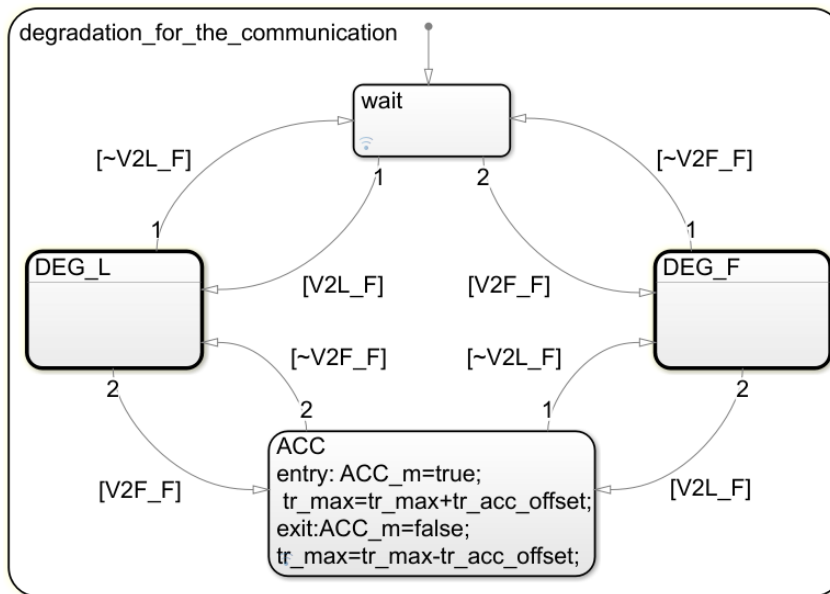


■ **Figure 3** Main Statechart.



■ **Figure 4** Statechart for the Degraded Platoon-Mode.

The degraded platoon mode, as depicted in Figure 4, consists of three parallel states, which in turn contain further state machines: *Degradation for the communication* (Figure 5), *Degradation for the distance measurement* (Figure 7) and *Degradation for the ego motion measurement* (Figure 8) according to the failure categories: communication failures, failures in distance measurement and failures in measurement of the ego-motion. In each parallel state, the reaction time is adjusted as described in the previous section. The overall reaction time is the sum of the normal reaction time tr_{pl} and all added constants. If, for example, an error occurs in the LIDAR, in the communication with the vehicle in front and in the wheel encoder, the reaction time to be added is the sum of tr_{lid} , tr_{front} and tr_{wenc} .

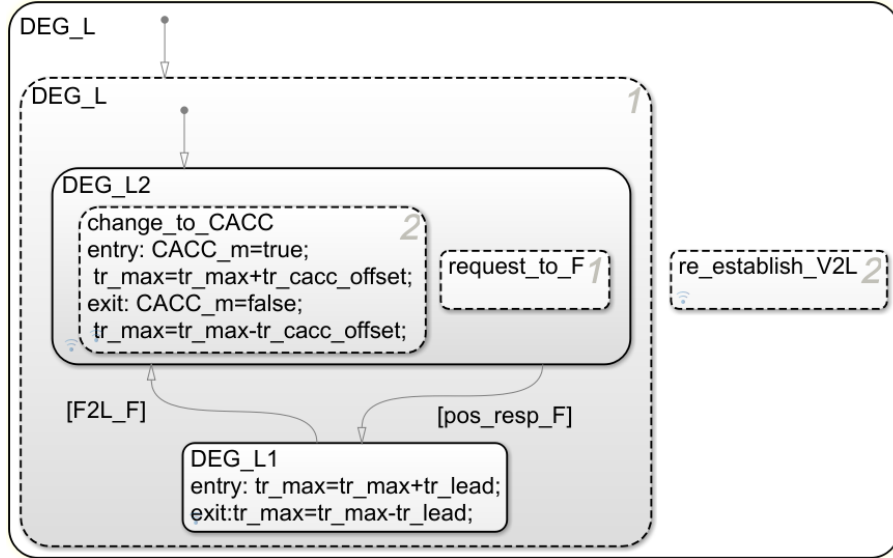


■ **Figure 5** Degradation for Communication Failures.

The state machine for *degradation in case of communication failures* is shown in Figure 5. The initial state “wait” corresponds to states without any communication errors. This state is active if the degraded platoon-mode is activated without communication errors, but with an error in the distance measurement or in the measurement of own motion. There are three different variants of degradation for communication: the failure is due to communication with the platoon leader ($V2L_F$), with the vehicle in front ($V2F_F$), or with both. When a failure

in the communication with the platoon leader (V2L_F) occurs, we switch to a degradation mode for communication with the leader. If the failure is resolved, the system returns to “wait” again. Failures within the communication with the vehicle in front (V2F_F) are handled analogously. If the communication with both, the leader and the preceding vehicle, is corrupted, the ACC mode is activated (ACC_m=true) and the ACC-specific constant tr_acc_offset is added to the reaction time. When this state is left, the offset is subtracted again. The same holds for a switch to the CACC mode, where tr_cacc_offset is added.

It contains a sub-state machine for corrupted communication with the platoon leader (DEG_L) as shown in Figure 6 (DEG_F is similar and, thus, omitted here). The vehicle switches to CACC mode (CACC_m=true) and tr_cacc_offset is added to the reaction time. At the same time, a request for the data of the platoon leader is sent to the preceding vehicle. When this data is received (pos_resp_F), the vehicle changes to a degraded state (DEG_L1) and the reaction time is increased by tr_lead . If the preceding vehicle loses connection to the platoon leader (F2L_F), the system returns to CACC mode. During this process, the system continuously tries to re-establish the communication with the platoon leader in order to switch back to an operating mode with a better performance.



■ **Figure 6** Degradation for Communication Failures with the Leader.

The state machine for *degradation in case of distance measurement failures* is depicted in Figure 7. The LIDAR and the ultrasonic sensor are mainly used for the identification of the received communication packets or for precise distance measurement in ACC mode. Starting from a wait state (wait), the system switches to a degraded state (DEG_LID1) and relies on the ultrasonic sensor for identification of communication packets if a LIDAR error occurs (LID_F). tr_lid1 is added to the reaction time. However, when ACC mode is active (ACC_m), it switches to a degraded state (DEG_LID2) and relies on the ultrasonic sensor for precise distance measurement. tr_lid2 is added to the reaction time. The same applies if an error occurs in the ultrasonic sensor, i.e. tr_us1 or tr_us2 are added to the reaction time and only the LIDAR is used. The following applies: $tr_lid2 > tr_lid1$ and $tr_us2 > tr_us1$. For failure-free communication, an estimate is sufficient to identify the communication packets. If the communication is faulty, an exact distance measurement is needed. Thus, the reaction times tr_lid2 and tr_us2 are higher.

If the data of both sensors are faulty, the system switches to a degradation state in which it relies on the GPS and the wheel encoder to identify communication packets. Compared to the positions of platoon partners that were received via V2V communication, communication-packets of the vehicle in front and the platoon leader can then be filtered. *tr_lid_us* is added to the reaction time in this case. This degradation is only possible if communication is available. Otherwise, the vehicle passes to the final degradation state (FIN_DEG).

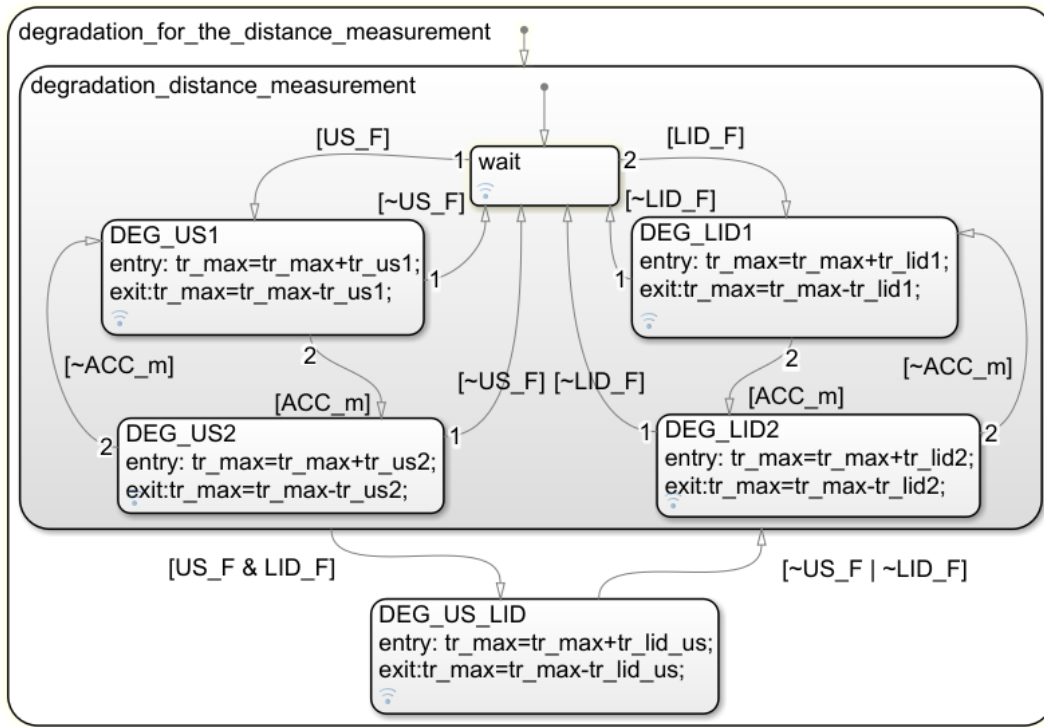


Figure 7 Degradation for Distance Measurement Failures.

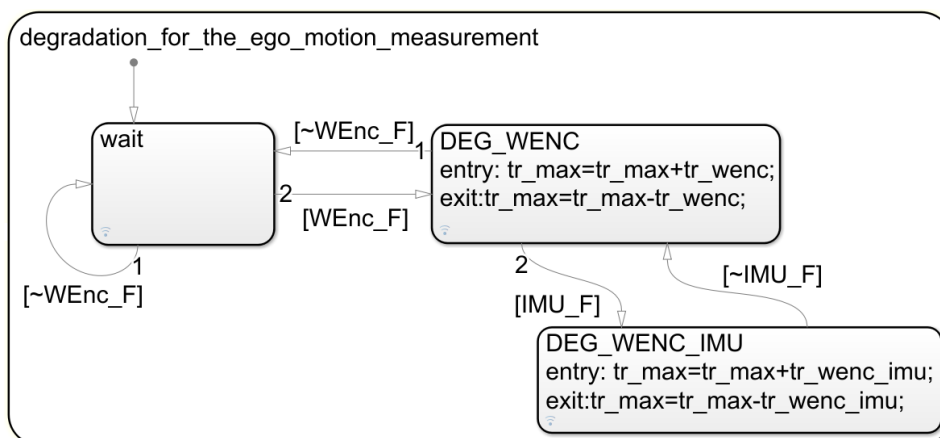


Figure 8 Degradation for the Measurement of Own Motion.

In Figure 8, we depict the state machine for degradation in case of *measurement failures of the ego-motion*. Starting from a wait state (wait), the system switches to a degraded state (DEG_WENC) and relies on the inertial measurement unit to determine its own acceleration if an error occurs in the wheel encoder (WEnc_F). tr_wenc is added to the reaction time.

If the inertial measurement unit is faulty too, we switch to a degraded state (DEG_WENC_IMU) and add tr_wenc_imu to the reaction time. In this mode, a motor model is introduced which estimates the motor speed from electrical values without using the physical speed sensor. The introduction of the motor model was discussed in [4]. Although the speed is not estimated very well, an estimation is better than a complete loss of the measurement of the ego-vehicle motion.

The proposed fallback concept can be further refined by changing reactions to failures or adding new actions. For example, in case of a distance measurement failure, the cameras can be used to estimate the distances. Furthermore, the logical expression that leads to the final degradation state can also be adjusted. This can be useful, for example, if it is determined during simulation or testing that a safe distance cannot be maintained for a certain combination of failures. The presented controller model contains degradation steps for all failure combinations that were considered during our systematic system analysis. To ensure that the controller is complete and that the reaction time estimate is safe, i.e. leading to a safe distance to the preceding vehicle, we systematically tested our controller, as described in the next section.

5 Assurance by Mode-Specific Contracts

To ensure a safe operation of vehicles, standards like ISO26262 [3] define concepts and procedures which need to be considered during the development of safety-critical functions. These standards explicitly recommend a formal system description for heavily safety-critical systems like our car platoon. Based on the formal system description, methods like model checking allow a partly or fully automated way of verifying and validating the system early during development. A promising approach to this is Contract-based Design[5]. By specifying pairs of assumptions and guarantees the behaviour of each system component can be defined on its own and thus lowering the overall complexity of large systems. The composition of components and their contracts can then be evaluated.

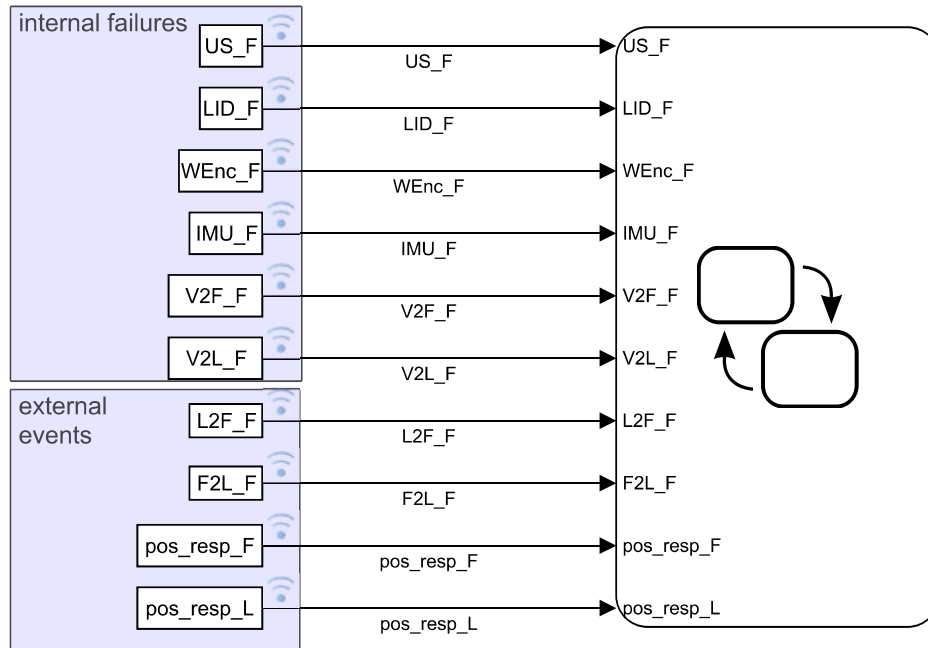
To validate our degradation controller we specify contracts in SSPL (*System Specification Pattern Language*)[2], which enables a formal and verifiable description of behaviour while also maintaining a readable appearance. We came up with 11 contracts which describe the degradation of the platooning function. As an example, the contract defining the minimum distance to the vehicle in front while platooning is active and no errors occur (nominal platoon-mode) looks as follows:

Assumption	all of the following conditions hold: - platooning is <i>ACTIVE</i> - com_error does not occur - distance_meas_error does not occur.
Guarantee 1	distance_to_front is always greater than d_min_pl
Guarantee 2	Whenever break_maneuver_front occurs then in response distance_to_front is never less than $distance_at_standstill$ starting immediately.

As a second example, we show a contract specifying a switch to ACC-mode ($\text{drv_mode} = \text{ACC}$) when the communication to the platoon head and the preceding vehicle is lost:

Assumption	No specific assumption. The contract's guarantees shall always hold.
Guarantee 1	Whenever $\mathbf{V2L_F}$ changes to <i>true</i> while all of the following conditions hold: - $\mathbf{V2F_F}$ is <i>true</i> - drv_mode is not <i>FIN_DEG</i> then in response drv_mode changes to <i>ACC</i> eventually.
Guarantee 2	Whenever $\mathbf{V2F_F}$ changes to <i>true</i> while all of the following conditions hold: - $\mathbf{V2L_F}$ is <i>true</i> - drv_mode is not <i>FIN_DEG</i> then in response drv_mode changes to <i>ACC</i> eventually.

As a first step towards a verified degradation controller, we have implemented and automated tests based on these contracts in Simulink to successfully check the state machines against our specification. To this end, we have simulated our controller in the presence of specific failure combinations and checked whether our guarantees hold. Figure 9 gives an overview of internal failures and external events that we have used for systematic testing of failure combinations. Expressing our system behaviour with semi-formal contracts in SSPL, has eased writing proper test cases for our state machines.



■ **Figure 9** Simulink/Stateflow model used to verify against contract specification.

6 Conclusion & Future Work

We have described a systematic design of degradation cascades for sensor and communication failures in autonomous car platoons using the autonomous model car “Velox” as example. We have modelled our resulting degradation controller in Simulink/Stateflow. For safety assurance, we have formulated contracts for each identified degradation level and used them for systematic testing. The methodology of the present work and its systematic approach is

inspired by [4] and has been adapted for the platooning function. In our work, new ideas for degradation and reactions to errors have been developed and presented at a conceptual level. It can serve as a basis for further improvement, development and implementation in future work. The systematic approach presented in this paper is not restricted to autonomous platoon driving but can also be applied to other problems and different systems in general.

In contrast to traditional, non-distributive systems, cooperative systems should not only deal with local failures, but also with failures of the other participants. Failures can, if possible, be transmitted via communication. However, if this is not possible due to communication failures, participants should also be able to deal with this situations and reach a functional and safe state. In future work, the degradation concept could be extended to feature additional modes allowing platoon members to autonomously determine which members are affected by malfunctions. For example, if a platoon member can only estimate its own acceleration with poor accuracy due to sensor failures, it will communicate its degradation mode to the other platoon participants. Another approach would be to define reactions beyond the consideration of individual vehicles on a platoon level. For example, if the platoon leader cannot be reached by several followers, the platoon leader could be changed, e.g. by choosing the next achievable follower as a new platoon leader. This kind of platoon management has not been systematically coped with so far and would definitely be a mandatory step for a robust and applicable platooning functionality.

As a first step, we have tested our concept based on formal contract specifications. By expressing our system behaviour with semi-formal contracts in SSPL, it has already been much easier to write proper test cases for our state machines. In future work, we aim at automatically verifying our contract specification against the static system architecture using model checking. A promising candidate is the model checking tool OCRA⁴ as it already supports this type of contract verification. We are currently developing a translation from SSPL to Othello, the contract specification language supported by OCRA. With an automatic translation, the engineers will not have to cope with difficult expressions in temporal logic but can rather use our template approach to specify and verify their systems.

References

- 1 Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. A Survey of Physics-Based Attack Detection in Cyber-Physical Systems. *ACM Computing Surveys (CSUR)*, 51(4):76, 2018.
- 2 Markus Grabowski, Bernhard Kaiser, and Yu Bai. Systematic Refinement of CPS Requirements using SysML, Template Language and Contracts. In Ina Schaefer, Dimitris Karagiannis, Andreas Vogelsang, Daniel Méndez, and Christoph Seidl, editors, *Modellierung 2018*, pages 245–260, Bonn, 2018. Gesellschaft für Informatik e.V.
- 3 ISO. ISO 26262 Road vehicles - Functional safety. Standard, International Organization for Standardization, 2011. In several parts: 1: Vocabulary, 2: Management of functional safety, 3: Concept phase, 4: Product development at the system level, 5: Product development at the hardware level, 6: Product development at the software level, 7: Production and operation, 8: Supporting processes, 9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses, 10: Guideline on ISO 26262.

⁴ Othello Contracts Refinement Analysis, <https://ocra.fbk.eu/>

- 4 B Kaiser, B Monajemi Nejad, D Kusche, and H Schulte. Systematic design and validation of degradation cascades for safety-relevant systems. In *Annual European Safety and Reliability Conference ESREL*, 2017.
- 5 Bernhard Kaiser, Raphael Weber, Markus Oertel, Eckard Böde, Behrang Monajemi Nejad, and Justyna Zander. Contract-based design of embedded systems integrating nominal behavior and safety. *Complex Systems Informatics and Modeling Quarterly*, 4:66–91, 2015.
- 6 Jeroen Ploeg, Elham Semsar-Kazerooni, Guido Lijster, Nathan van de Wouw, and Henk Nijmeijer. Graceful degradation of CACC performance subject to unreliable wireless communication. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1210–1216. IEEE, 2013.
- 7 Rajesh Rajamani, Adam S Howell, Chieh Chen, J Karl Hedrick, and Masayoshi Tomizuka. A complete fault diagnostic system for automated vehicles operating in a platoon. *IEEE Transactions on Control Systems Technology*, 9(4):553–564, 2001.
- 8 SAE International. *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*, January 2014. doi:10.4271/J3016_201401.
- 9 Irfan Sljivo, Barbara Gallina, and Bernhard Kaiser. Assuring degradation cascades of car platoons via contracts. In *International Conference on Computer Safety, Reliability, and Security*, pages 317–329. Springer, 2017.

Feasibility Study and Benchmarking of Embedded MPC for Vehicle Platoons

Iñaki Martín Soroa

Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
i.martin.soroa@studenten.tue.nl

Amr Ibrahim

Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
a.ibrahim@tue.nl

Dip Goswami

Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
d.goswami@tue.nl

Hong Li

Car Infotainment and Driving Assistance, NXP Semiconductor, Eindhoven, The Netherlands
Hong.r.li@nxp.nl

Abstract

This paper performs a feasibility analysis of deploying Model Predictive Control (MPC) for vehicle platooning on an On-Board Unit (OBU) and performance benchmarking considering interference from other (system) tasks running on an OBU. MPC is a control strategy that solves an implicit (on-line) or explicit (off-line) optimisation problem for computing the control input in every sample. OBUs have limited computational resources. The challenge is to implement an MPC algorithm on such automotive Electronic Control Units (ECUs) with an acceptable timing behavior. Moreover, we should be able to stop the execution if necessary at the cost of performance.

We measured the computational capability of a unit developed by Cohda Wireless and NXP under the influence of its Operating System (OS). Next, we analysed the computational requirements of different state-of-the-art MPC algorithms by estimating their execution times. We use off-the-shelf and free automatic code generators for MPC to run a number of relevant MPC algorithms on the platform. From the results, we conclude that it is feasible to implement MPC on automotive ECUs for vehicle platooning and we further benchmark their performance in terms of MPC parameters such as prediction horizon and system dimension.

2012 ACM Subject Classification Applied computing → Command and control; Computer systems organization → Embedded systems

Keywords and phrases Model predictive control, vehicle platoon, embedded implementation, code generation

Digital Object Identifier 10.4230/OASICS.ASD.2019.2

Funding This work is partially supported by the H2020 projects I-MECH (GA no. 737453) and oCPS (GA no. 674875).

1 Introduction

Vehicle platooning is an application based on Cooperative Adaptive Cruise Control (CACC) technology, which is an extension of Adaptive Cruise Control (ACC). In ACC, the vehicle senses the position of the preceding vehicle and adapts the speed to avoid a collision. CACC introduces V2V messages between different vehicles. These messages have much richer information including position, speed, acceleration or road intersection status among others. The richer information allows the vehicles to react faster to sudden changes in the preceding



© Iñaki Martín Soroa, Amr Ibrahim, Dip Goswami, and Hong Li;
licensed under Creative Commons License CC-BY

Workshop on Autonomous Systems Design (ASD 2019).

Editors: Selma Saidi, Rolf Ernst, and Dirk Ziegenbein; Article No. 2; pp. 2:1–2:15

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

vehicles and therefore, the distance between the vehicles can be reduced which enables to achieve better fuel efficiency and road capacity [21].

Model Predictive Control (MPC) is an optimal control strategy capable of satisfying constraints on the states of the system (plant) and the control input. The main challenge of MPC is its high computational requirements since it requires to solve an optimisation problem at every time step (sample) [7]. The MPC technology is extensively used in the chemical industry where the dynamics is generally slower. With the advent of powerful computing abilities of modern processors, MPC is making its way into other sectors such as the automotive industry [10]. One of the applications of MPC in the automotive industry is vehicle platooning. MPC has already been applied to vehicle platooning without explicitly considering constraints on the computational resources and the V2V communication time that is present in a real implementation [6]. In a real implementation, the ECU of a vehicle, which is an embedded device with limited resources, needs to solve the MPC optimization fast enough to meet the timing requirements imposed by the V2V communication.

MPC has already been implemented on embedded platforms successfully for different applications. In [31] the authors use a simple embedded device with an ARM processor running at 48MHz with 64kB of RAM memory. They control a system consisting of 8 states, 2 inputs and a control horizon of length 20 achieving a sampling period of 4ms. In [31], Fast Gradient Method algorithm (FGM) was used with fixed point operations and a tuned level of sub-optimality specific for the plant. When using floating point operations and decreasing the control horizon length to 15, it achieves a sampling time of 8ms. In [9], the authors achieved a sampling frequency in the kHz range using a processor with a clock frequency of 1GHz with a dedicated floating point unit. When controlling a larger system they manage to reach a sampling time of 13ms.

A number of works approached the embedded MPC problem using hardware accelerators [5] [28] [25] [17] [14] [22] usually on a Field Programmable Gate Array (FPGA). These works attempt to achieve sampling rates in the kHz range or control very large systems, while the vehicle platooning problem does not require very short sampling times nor large predictive models.

In order to solve the MPC optimisation problem an algorithm needs to be used. There are mainly two categories of MPC algorithms – explicit and implicit. In explicit MPC the solution is computed off-line and given to the controller as a look up table which usually requires large memory capacity. In implicit MPC the solution is computed on-line at each sampling period [1]. In this paper we focus on implicit MPC. Implicit MPC is the most commonly used method and there are a number well-developed state-of-the-art algorithms. Almost all of them can be classified in one of the following categories – Inner Point Method (IPM) [16], Active Set Method (ASM) [8], and (Fast) Gradient Method ((F)GM) [17] [3]. We analysed the feasibility of these algorithms with a special focus on FGM.

In order to determine if it is feasible to implement MPC for vehicle platooning on an embedded device, timing constraints must be met. The MPC algorithm needs to be able to compute the solution fast enough for a problem with similar dimensions and constraints as in vehicle platooning, described below (Section 4.2). The time available for the execution of the MPC task depends on the message rate (or sampling rate) supported by the V2V communication and the execution time of the other tasks running on the device. Ideally the execution time would be deterministic or bounded, which can be achieved for some of the state-of-the-art algorithms.

We also analyse the trade-off that needs to be made to balance the control performance and the execution time of the MPC task. We investigate the impact of the length of the control horizon (used in the MPC optimization) on the execution time, the effect of the algorithm choice and provide a number of guidelines for choosing the processor.

The rest of the paper is organised as follows. We describe the problem of vehicle platooning in Section 2. In Section 3, we analyse the characteristics and the performance of a platform suitable for being used for vehicle platooning. Next, we describe a possible implementation for vehicle platooning and we measure the overhead introduced by the other tasks that need to run on the selected platform in Section 4. We investigate a number of automatic C code generation tools available for MPC in Section 5. We analyse the computational requirements of different MPC algorithms in Section 6. Using the computational requirements of different algorithms and the performance of the platform, we provide an analytical estimation of the execution time on the selected platform in Section 7.1. We use the code generation tools to run a number of template MPC algorithms on the V2V wireless node in Section 7.2. Using the experimental and the analytical execution times we estimate the possible delays and sampling periods that can be achieved using MPC and the trade-offs that can be made, in Section 7.3. Finally we conclude in Section 8.

2 Vehicle platoons

2.1 V2V Communication and topology

The vehicle-to-vehicle communication (V2V) is performed following the standards of each country, most notably the standard of the EU, ETSI-ITS, and the standard of the USA, 1609 WAVE. Both standards are based on the IEEE 802.11p protocol stack. Under IEEE 802.11p, we can reach up to 10Hz message rate when the network usage is below 70%. The message rate can get as low as 1Hz under heavy traffic of vehicles with V2V communication devices [13].

In this paper we consider the Predecessor-Follower (PF) topology, where each vehicle receives messages from its predecessor (Fig. 1). Other topologies exist, such as Two-Predecessor-Follower (TPF) [30] and Leader-Predecessor-Follower (LPF) [30].

In Fig. 1, m_i is the message from the vehicle i including its speed, position and acceleration, and Δd_i is the error in distance (desired gap - actual gap) between the vehicles i and $i - 1$.

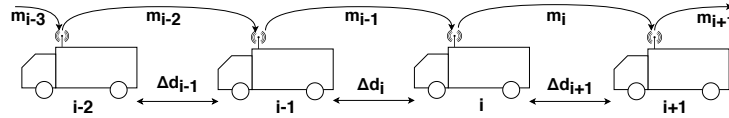


Figure 1 Predecessor-Follower topology.

2.2 Platoon model

The platoon model is distributed, each vehicle has a model of itself (vehicle model) and its relation with its predecessor (inter-vehicle dynamics).

The model of the vehicle i is obtained combining a simplified model of the longitudinal dynamics of the vehicle with the dynamics of a DC motor [27, 26], and it is given by:

$$\dot{x}_v^i = A_v^i x_v^i + B_v^i u_v^i \quad (1)$$

where A_v^i and B_v^i are the state and input matrices respectively, u_v^i is the duty cycle of the input of the motor and $x_v^i = [a^i \ \dot{a}^i]^T$ is the state vector. Where a^i and \dot{a}^i are the acceleration and the rate of change of the acceleration of the vehicle i , respectively. Moreover,

the state matrix A_v^i and the input vector B_v^i of vehicle i are defined as:

$$A_v^i = \begin{pmatrix} 0 & 1 \\ \frac{-1}{\tau^i \tau_a^i} & \frac{-(\tau^i + \tau_a^i)}{\tau^i \tau_a^i} \end{pmatrix} \in \mathbb{R}^{2 \times 2}, B_v^i = \begin{pmatrix} 0 \\ \frac{K^i K_a^i}{\tau^i \tau_a^i} \end{pmatrix} \in \mathbb{R}^{2 \times 1}.$$

where τ^i , τ_a^i , K^i , K_a^i are model parameters of the vehicle i .

To obtain the platoon model under the PF topology, the inter-vehicle dynamics relate the vehicle i to the vehicle $i - 1$. This is done by adding two new states, Δv^i and Δd^i , which represent the speed difference and the gap error between the vehicles, respectively. They are defined as $\Delta d^i = d^i - d_{des}^i$ and $\Delta v^i = v^{i-1} - v^i$, where Δd^i is the error between the actual gap (d^i) and the desired inter-vehicle gap (d_{des}^i) between the vehicle i and the vehicle $i - 1$. Δv^i is the velocity error between the vehicle i and the vehicle $i - 1$, where v^i denote for the velocity of the vehicle i . d^i and d_{des}^i are defined as $d_{des}^i = d_0 + \tau_h v^i$ and $d^i = q^{i-1} - q^i - L^i$, where d_0 is the gap between vehicles at standstill, τ_h is the constant headway time (the time the vehicle i needs to reach the position of the vehicle $i - 1$ when $d_0 = 0$). L^i , q^i are the length and position of the vehicle i , respectively.

Combining the vehicle model with the inter-vehicle dynamics we obtain the platoon model:

$$\dot{x}_p^i = A_p^i x_p^i + B_p^i u_p^i + G_p^i a^{i-1} \quad (2)$$

where $x_p^i = [a^i \quad \dot{a}^i \quad \Delta d^i \quad \Delta v^i]$ is the state vector, a^{i-1} is the acceleration of the preceding vehicle and A_p^i is the state matrix. The predictive model used for MPC will be obtained based on the platoon model in Eq.(2) (see Section 4.2).

3 Embedded platform: Cohda Wireless MK5 OBU

The Cohda Wireless MK5 is a platform developed by Cohda Wireless in partnership with NXP. It has been developed as a prototyping platform for V2V applications, such as CACC, and other Vehicle to Everything (V2X) applications.

3.1 Hardware

The platform has one main processor, NXP i.MX6 Dual Lite @ 800MHz (dual-core processor), paired with a communications co-processor, NXP MARS. It is equipped with 1GB of volatile memory. With a large volatile memory, memory is not a bottleneck and we are interested only in the computational power.

The platform has several ports and connectivity options. It can be connected to two 5.9GHz antennas, a GNSS antenna, μ SD card, Ethernet port, CAN bus port and audio jack. On top of that it has a DC power connection.

3.2 Software

The platform uses an Ubuntu distribution of Linux as its Operating System (OS). It is not a Real-Time OS (RTOS). There are system applications available on the platform. The most relevant are the communication stacks of the EU and the USA standards.

We also use the evaluation platform reported in [30] and available in [29], which allows to quickly measure the string stability of the platoon and takes care of all the tasks required to execute CACC in a modular approach. The structure of this evaluation platform is further detailed in Section 4.

3.3 Performance evaluation

The performance is measured in millions of floating point operations per second (Mflop/s) and millions of fixed point operations per second (Mop/s). We evaluate the average performance and its distribution in percentiles.

The type of operations measured are fixed point and floating point additions, and fixed point and floating point multiplications. Most MPC algorithms use only these operations. In Table 1, the performance of the platform is shown.

■ **Table 1** Performance of the Cohda Wireless MK5 platform.

	Fixed point addition	Fixed point multiplication	Floating point addition	Floating point multiplication
Fastest	230.1 Mop/s	214.0 Mop/s	113.0 Mflop/s	88.2 Mflop/s
95th percentile*	214.0 Mop/s	200.0 Mop/s	113.0 Mflop/s	88.2 Mflop/s
Average	174.0 Mop/s	156.0 Mop/s	112.0 Mflop/s	60.0 Mflop/s
5th percentile*	200.0 Mop/s	150.0 Mop/s	103.0 Mflop/s	78.9 Mflop/s
1st percentile*	107.0 Mop/s	88.2 Mop/s	50.8 Mflop/s	47.6 Mflop/s

*The k th percentile is the number larger than $k\%$ of the measurements.

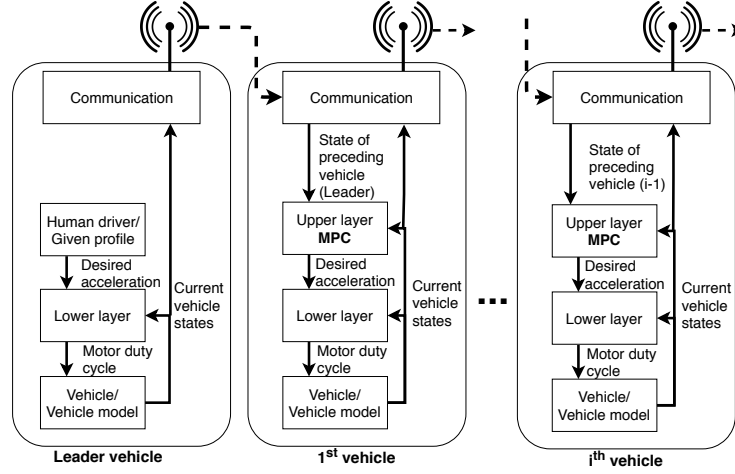
We use an internal timer for the measurement which runs at a frequency of 1MHz, while the processor runs at 800MHz. Therefore the accuracy of our measurement is within 800 clock cycles. The first test uses a large number of operations in a loop. We used 8×10^9 , 4×10^9 and 2×10^9 operations for each of the measured types. The different number of operations allows us to confirm that the execution time is linear to the number of operations. These measurements give us a notion of the average performance of the system.

As our system does not use a RTOS, during the time that the test executes (over 4 minutes in some cases) there are other tasks preempting the test. In order to measure the variability of the performance we designed a second test. In this test we measure the time needed to perform 3000 operations, and the measurement is repeated for 1 million times. We search for the fastest iteration and the 1st, 5th and 95th percentile. If performance requirement is higher, the device can be overclocked to reach 1GHz, and some secondary OS tasks such as Bluetooth can be shut down to remove their influence.

4 Overall architecture

In the Fig. 2 we can see a diagram showing how the system works. We use the platform in [30], with the MPC design in [15]. As we will use the PF strategy for the communication, the leader sends its state to the first vehicle, and the vehicle i sends its state to the vehicle $i + 1$. In a real environment, the leader vehicle would be driven by a human driver, and the commands control the real vehicle. In a simulation we create a profile for the acceleration commands and actuate the model of the vehicle.

The vehicle i receives the state from the vehicle $i - 1$ and uses a^{i-1} and x_p^i as inputs for the MPC controller (the upper layer), which computes the desired acceleration. The desired acceleration is used by the lower layer as reference value and outputs the duty cycle of the input to the motor, which controls the vehicle. The state of the vehicle (sensed or simulated) is given as an input to the two controllers and it is also sent to the next vehicle.



■ **Figure 2** Overall architecture of the system.

4.1 Lower layer controller

The lower layer controller is a state-feedback controller and it runs at a faster rate than the upper layer with a sampling rate of 2ms [15]. The output of this controller is the motor duty cycle which controls the vehicle.

4.2 MPC: Upper layer controller

In MPC we solve an optimisation problem by defining the following quadratic cost function subject to specific constraints on inputs and states:

$$J = x_{N+k|k}^i T P x_{N+k|k}^i + \sum_{j=0}^{N-1} (x_{j+k|k}^i T Q x_{j+k|k}^i + u_{j+k|k}^i T R u_{j+k|k}^i)$$

$$\text{subject to } x_{j+k+1|k}^i = \Phi^i x_{j+k|k}^i + \Gamma^i u_{j+k|k}^i + \Psi^i a_{j+k|k}^{i-1}, \quad j = 0, \dots, N-1$$

$$x_{min} \leq x_{j+k|k}^i \leq x_{max}, \quad j = 1, \dots, N$$

$$u_{min} \leq u_{j+k|k}^i \leq u_{max}, \quad j = 1, \dots, N \quad (3)$$

where J is the cost function, N is the length of the control horizon, $x_{j+k|k}^i$ is the predicted state vector of vehicle i after j steps computed at time k , where $x_{k|k}^i$ is the sensed state of vehicle i . $u_{j+k|k}^i$ is the computed input vector for the vehicle i for the j step, and Q , R and P are the weight matrices. It should be noted that a quadratic cost function is chosen so that the problem is convex and a global minimum can be found.

In order to use MPC we must discretize the platoon model in Eq. (2) using Zero-Order Hold (ZOH). After the discretization, the predictive model for vehicle i becomes:

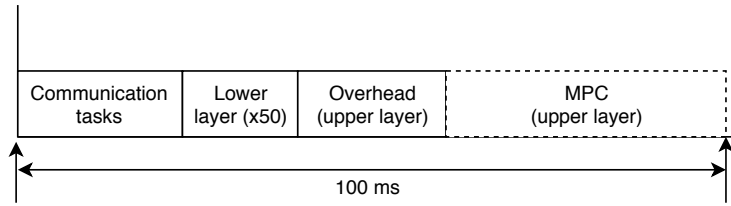
$$x_{j+k+1|k}^i = \Phi^i x_{j+k|k}^i + \Gamma^i u_{j+k|k}^i + \Psi^i a_{j+k|k}^{i-1}, \quad j = 0, \dots, N-1 \quad (4)$$

where $x_{j+k|k}^i = [a_{j+k|k}^i \quad \delta a_{j+k|k}^i \quad \Delta d_{j+k|k}^i \quad \Delta v_{j+k|k}^i]$ represent the predicted states. $u_{j+k|k}^i$ is the desired acceleration (the optimal control inputs that must be computed). $a_{j+k|k}^{i-1}$ is the predicted acceleration of the preceding vehicle. We consider that the future evolution of the acceleration of the preceding vehicle is constant. Therefore, it does not affect the optimisation process. The predictive model has 4 states and 1 input variable. Each of the states and the desired acceleration (u^i) have an upper and a lower bound. Therefore we have 10 constraints.

The upper layer uses MPC. It receives the current state of the vehicle and the state of the preceding vehicle, and gives the lower layer a new acceleration reference. The upper layer runs with a sampling time of 100ms since the maximum message rate is 10Hz. At every sample in which a new message has been received, the MPC controller computes an optimal series of future N inputs (N is the horizon length). When the message rate is lower than the sampling rate, the MPC controller automatically updates the desired acceleration using the next value of the optimal series of inputs that were computed in the last sample in which a message had been received. As the message rate can drop to 1Hz, we need the length of the control horizon to be at least 10 while a higher N could improve the quality of the control. We consider $10 \leq N \leq 20$.

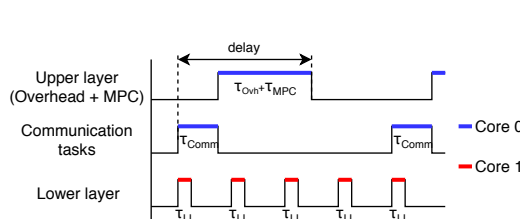
4.3 Execution time budget for MPC

In this section, we compute the maximum available execution time for the MPC algorithm considering a message rate of 10Hz. That is, the time available to execute the MPC algorithm after performing all the other system tasks – see Fig. 3. The platform needs to send and receive messages (communication task), and compute the result of the lower layer and upper layer controllers. The upper layer also has some overhead besides the MPC algorithm such as updating the value of some pointers and variables like the desired acceleration. Fig. 5 shows the tasks performed by each piece of hardware. The MARS co-processor sends and receives packages. The main processor creates the packages that need to be sent, processes the received packages, and executes the upper and lower layer controllers.

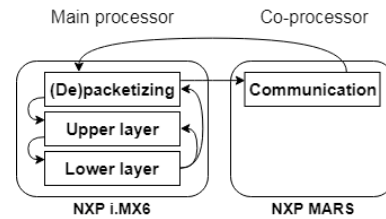


■ **Figure 3** Execution time requirement when running on a single core.

In order to measure the contribution of each task to the total execution time, we use the platform developed in [29] which uses a PID controller for the upper layer and a state-feedback controller for the lower layer. We removed the logging functions and the PID controller, so that we get a minimal version of the platform, and added time stamps to analyse the latency of each task. Furthermore, we have modified the platform so that the tasks in the main processor are scheduled using POSIX threads with a Fixed-Priority Preemptive Scheduler (FPPS). For tasks with equal priority it follows a First In, First Out (FIFO) schedule. The



■ **Figure 4** Typical execution without OS tasks.



■ **Figure 5** Task distribution and data flow on hardware.

OS has the highest priority, the lower layer and the (de)packetizing tasks are given an equal medium priority, while the upper layer has the lowest priority. The processor has two cores, making it possible to process two different tasks (threads) simultaneously. The kernel distributes the different tasks (including the OS tasks) between the 2 cores dynamically.

In Fig. 4 we can observe the expected execution of the tasks in the absence of OS tasks. The lower layer is represented at a lower frequency than the frequency implemented.

The results are shown in Table 2. We are mainly concerned about the tasks shown in Fig. 3. We present the measured maximum and average latencies after 100 runs and the execution frequency, i.e. how often does that task have to execute.

■ **Table 2** Latency of other tasks that run on the platform.

	Maximum latency	Average latency	Execution frequency
Lower layer controller	0.045ms	0.0151ms	500Hz
Upper layer controller overhead	0.059ms	0.0126ms	10Hz
Communication tasks	1.063ms	0.3611ms	$\leq 10\text{Hz}$

With the results in Table 2 we can obtain the execution time budget for the MPC task. In 100ms we need to perform the lower layer controller task 50 times ($\frac{500\text{Hz}}{10\text{Hz}}$), and the communication and the upper layer controller tasks only once. We used the worst case latencies to compute the execution time available for the MPC algorithm in the worst case. The worst case latencies will occur when the OS tasks are running on both cores of the processor, therefore for the worst case analysis we assume that there is only one core available for all the tasks. e_{MPC} , e_{Comm} , e_{Ovh} and e_{LL} denote the maximum latency of the MPC task, the communications tasks, the overhead of the high level controller and the low level controller respectively.

$$e_{MPC} \leq 100\text{ms} - (e_{Comm} + e_{Ovh} + 50e_{LL}) = 96.628\text{ms} \quad (5)$$

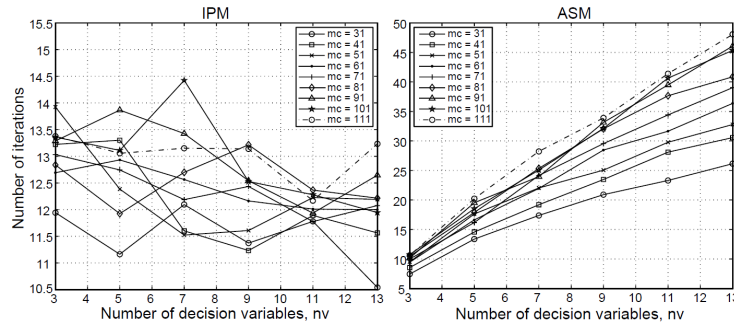
From the above experiments, we conclude that the effects of other tasks are almost negligible and we obtained an upper bound for the execution time. How to respect this requirement is analysed in Section 6. The quality of the control is affected by the sensor-to-actuator delay. Therefore, the execution time should be as short as possible.

5 Automatic C code generation for MPC

In order to facilitate implementing MPC algorithms on embedded platforms, automatic C code generators are developed. These tools take a description of the desired MPC problem and generate the necessary C code to solve it with a given optimisation algorithm.

Code generators are used academically and in the industry [4]. There are many tools available. Some of them are commercial (paid) tools, e.g., ODYS [24] or FORCES [11] while others (e.g., $\mu\text{AO-MPC}$ [32], CVXGEN [20], FiOrdOs [12], jMPC [2]) are free. We used CVXGEN, $\mu\text{AO-MPC}$ and FiOrdOs in this paper since they are free and they allow to stop executions providing a sub-optimal solution.

- **CVXGEN** has been developed in the University of Stanford. It allows to describe an optimisation problem in general terms, the problem description includes the dimensions of the different matrices and vectors, and some properties such as being positively definite, or diagonal. It does not need the exact values of each entry of the matrices. The algorithm used is based on CVX, a solver for MATLAB. The tool is online based and free for academic use [20].



■ **Figure 6** Number of iterations as a function of the number of variables (n_v) and constraints (m_c), reproduced from [19].

- **μAO-MPC** has been developed in the Otto von Guericke University of Magdeburg. This tool is a very similar to CVXGEN in its usage and flexibility. It uses a FGM algorithm for obtaining the solution. The tool can be downloaded for free and it works using Python, which is also a free tool [32].
- **FiOrdOs** has been developed in ETH Zurich. This tool requires a full description of the problem, with all the entries of the matrices before it can generate the code. It uses a FGM algorithm. The tool is a free toolbox for MATLAB [12].

6 MPC algorithms and computational requirements

In order to estimate the execution time of different algorithms, we need to know their complexity. The number of computations per iteration is deterministic in most algorithms, but the number of iterations depends on the convergence speed of the problem and the initial conditions making the total execution time unpredictable.¹

- **IPM** reaches the solution in steps towards solving the Karush-Kuhn-Tucker equations, making few but computationally heavy iterations [19]. For IPM we used the estimate of the number of flops shown in [19] which is also shown in Table 3. We use the Gauss-Jordan elimination with pivoting method for solving the linear systems using the estimate given in [19]. We assumed that division operations are equivalent to 10 multiplications. For the number of iterations, we took an approximate value based on Fig. 6, reproduced from [19], with 13 iterations.
- **ASM** tries to guess the constraints that are active in the solution (Active Set) and does it by adding the constraints one by one on every iteration [19]. For ASM we use the estimate of the number of flops found in [19]. We made the same choices as for IPM. For the number of iterations, we can find a direct linear relationship between the number of decision variables, n_v , and the number of iterations when looking at Fig. 6, see Table 3.
- **GM** computes the gradient of the cost function in the current point and next, it moves one step in that direction. It repeats the process until it finds the minimum. In the fast variants, FGM, a sub-optimal solution is accepted as a trade-off for a faster computation time. An important advantage of FGM is that it can give an output at any point in time, making it possible to bound the execution time. These methods require the cost function to be quadratic [17] [3].

¹ In [23] an upper bound for the number of iterations of some algorithms is found. However, it requires knowledge of the exact values of the predictive model and the bound is significantly larger than the observed number of iterations [31].

For FGM we use the estimations provided in [18], which analyses several different FGM algorithms. It analyses Bemporad's and Richter's algorithms, each of them with 2 alternative formulations, which give different computational requirements. The equations used can be seen in Table 3. These estimates don't specify the type of operations. We assume that 50 iterations are needed, based on the experiments performed in [18], but those values are based on a different problem than the one used in [19], therefore they might not be comparable.

■ **Table 3** Formulas used to compute the number of flops [8] [19] [18].

Algorithm	Flops per iteration	Number of iterations
IPM	$2n_v^2(n_c + 1) + n_v(7n_c + 2) + 14n_c + 1 + Ma(n_v) + Mm(n_v) + 10(3n_c + 1 + Md(n_v))$	13
ASM	$2n_v^2 + 2n_v(2n_c + 1) - n_c + Ma(n_v + 0.5n_c) + Mm(0.5n_c + n_v) + 10(n_c + Md(n_v + 0.5n_c))$	$2.5 \times n_v$
Bemporad's FGM u-formulation	$N^2 n_u(2n_u + 3n_y + 3n_c)$	50
Bemporad's FGM xu-formulation	$N(4n_x^2 + 6n_x n_u) + 6N(N_c + n_y)(n_x + n_u) + 4N n_u(n_x + n_u)$	50
Richter's FGM uy-formulation	$2N^2(n_y^2 + n_u n_y)$	50
Richter's FGM xuy-formulation	$2N(n_y + n_x)(5n_x + 2n_u + n_y)$	50

In Table 3 the new variables have the following meaning:

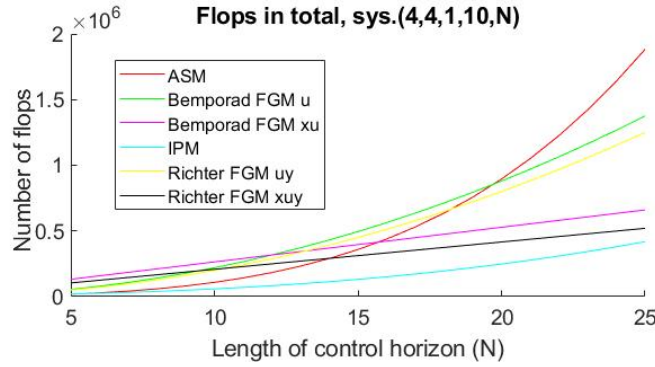
- n_x : Number of states of the plant model, in this case 4.
- n_y : Number of outputs of the plant model, in this case 1.
- n_u : Number of inputs of the plant model, in this case 1.
- n_c : Number of inequality constraints, in this case 10.
- n_t : Parameter computed as $n_t = n_u + n_y + n_x$
- n_v : Parameter computed as $n_v = N \cdot n_u$
- $Ma(x)$: Number of additions needed to solve the linear system of equations, computed as $Ma(x) = 0.5(x - 1)x(x + 1)$ when using Gauss-Jordan elimination.
- $Mm(x)$: Number of multiplications needed to solve the linear system of equations, computed as $Mm(x) = 0.5x^2(x + 1)$ when using Gauss-Jordan elimination.
- $Md(x)$: Number of divisions needed to solve the linear system of equations, computed as $Md(x) = x$ when using Gauss-Jordan elimination.

The effect of varying the control horizon length for a system with 4 states, 1 input, 1 output and 10 inequality constraints can be seen in Fig. 7. We can observe that the complexity of some algorithms grows exponentially while in others it grows linearly. Depending on the control horizon chosen for the application, different algorithms are recommendable.

7 Performance analysis

7.1 Estimated execution time

Using the specifications of the predictive model and the constraints in Section 4.2, taking $N = 15$, and the formulas given in Table 3, we can obtain the number of operations needed to solve MPC for vehicle platooning. Combining them with the performance of the device (Table 1) we can estimate the execution time for each algorithm.



■ **Figure 7** Effect of varying the length of the control horizon-Total number of flops

■ **Table 4** Execution time estimation for different algorithms for the model described in Section 4.2 and $N = 15$.

	Per iteration			Total		
	Flops	Time [ms] (multiplications ¹)	Time [ms] (mixed ²)	Flops	Time [ms] (multiplications ¹)	Time [ms] (mixed ²)
ASM	9.56×10^3	0.1593	0.1112	35.85×10^4	5.9750	4.1686
Bemporad's FGM u-formulation	9.9×10^3	0.1650	0.1151	49.5×10^4	8.2500	5.7558
Bemporad's FGM xu-formulation	7.92×10^3	0.1320	0.0921	39.6×10^4	6.6000	4.6047
IPM	10.11×10^3	0.1685	0.1176	13.14×10^4	2.1907	1.5284
Richter's FGM uy-formulation	9×10^3	0.1500	0.1047	45×10^4	7.5000	5.2326
Richter's FGM xuy-formulation	6.24×10^3	0.1040	0.0726	31.2×10^4	5.2000	3.6279

¹ This measurement assumes that all the operations are multiplications.

² This measurement assumes that half of the operations are additions and the other half multiplications.

In Table 4 we show the number of operations and execution time in total and per iteration for all the considered algorithms. Two different execution times are given, one under the assumption that all the operations are multiplications and the other assuming mixed operations, i.e. half of the operations are additions and the other half are multiplications.

7.2 Code generation experiments

The theoretical estimations can be too optimistic, as they assume that the data is always available, which is equivalent to having a infinitely fast memory. Using the code generation tools described in Section 5 we run an experiment (Appendix A) on the Cohda platform, obtaining a real execution time. The three algorithms are considered. We use approaches based on several iterations and sub-optimality levels. Each algorithm converges to the solution at a different speed. Therefore they need a different number of iterations. We determined the number of iterations as the minimum necessary to reach a value within 0.001 units of the solution for a very large number of iterations, which is assumed to be the optimal solution. This is equivalent to an error smaller than 1% in the problem used (Appendix A).

For CVXGEN the number of iterations varied with the control horizon length, being 63 for $N=10$, 56 for $N=15$ and 59 for $N=20$. μ AO-MPC and FiOrdOs use a dual approach, with an inner and outer loop. For μ AO-MPC we needed 30 iterations for the inner loop and 30 for the outer loop. For FiOrdOs the number of iterations of the inner loop is 1, and 125 for

the outer loop for all the horizons. The time displayed in Table 5 is the average between 100 solutions. We expect that the number of iterations needed to solve the vehicle platooning problem will be comparable to the number of iterations used in this problem.

■ **Table 5** Execution time of different automatic code generation tools for different control horizon lengths.

Horizon length	CVXGEN	μ AO-MPC	FiOrdOs
$N=10$	15.52 ms	106.150 ms	32.286 ms
$N=15$	26.02 ms	211.526 ms	73.632 ms
$N=20$	45.35 ms	358.030 ms	125.968 ms

7.3 Feasibility analysis

Following the execution diagram in Fig. 4, the delay from the sampling instant until the new control input is computed, depends on the execution times of the upper layer and the communication tasks. The execution time is variable due to the OS, making it impossible to obtain an exact value. We approximate it as:

$$delay \approx \tau_{MPC} + \tau_{Comm} + \tau_{Ovh} \quad (6)$$

where τ_{MPC} , τ_{Comm} and τ_{Ovh} are the average execution time of the MPC task, the communication task and the overhead of the upper layer, respectively.

Under the selected communication protocol, there is no use on having a faster sampling rate than 10Hz, but other communication protocols could be used. Therefore we will compute the maximum achievable sampling rate as the inverse of the delay. We consider that it is feasible to use an algorithm if its execution time is below the budget computed in Section 4.3.

Looking at the theoretical estimates, the execution time of IPM is the shortest, while the shortest FGM algorithm is Richter's algorithm using the xuy-formulation. For the FGM algorithms we don't know the type of operations. Therefore, we use the estimate in the case that all the operations are multiplications. For IPM the number of multiplications and additions are very similar [19] and we use the mixed estimation. When considering the experimental execution times, the best results are for CVXGEN for all the tested horizon lengths. In Table 6 we present the results for the selected algorithms allowing us to make the following observations. First, there is a significant difference in the execution time when using different algorithms or different control horizon lengths. Second, the delay is almost equal to the execution time of the MPC algorithm. Finally, all the selected algorithms are feasible to be used for this problem.

7.4 Trade-off analysis

From the complexity of the different algorithms (Section 6) we observe that the size of the predictive model and the length of the control horizon have a big impact on the complexity of the algorithm. Generally a longer control horizon length and a more accurate predictive model (which usually results in a larger model) give a better control performance, but the improvement might not be sufficient to overcome the negative effects of increasing the sensor-to-actuator delay and reducing the sampling rate.

When choosing hardware for MPC applications, the variability in the execution time must be taken into account. In all the implicit MPC algorithms the number of iterations varies depending on the initial conditions (making the total execution time vary), therefore it is not

■ **Table 6** Feasibility analysis, execution time, delay and the maximum sampling period for the selected algorithms.

	Execution time (ms)	Delay (ms)	Maximum sampling period (Hz)	Feasible
IPM $N = 15$	1.5284	1.9021	526	Yes
Richter's FGM xuy-formulation $N = 15$	5.2000	5.5737	179	Yes
CVXGEN $N = 10$	15.5200	15.8937	63	Yes
CVXGEN $N = 15$	26.0200	26.6368	38	Yes
CVXGEN $N = 20$	45.3500	46.1027	22	Yes

enough to select a processor capable of meeting the timing constraints for the average case. The processor depends on the requirements of the application, i.e. the MPC algorithm must be guaranteed to execute within the timing constraints 90% of the times. To provide such guarantees, it is required to perform multiple experiments under different initial conditions and obtain a probabilistic distribution of the execution time.

Finally, the MPC task can be parallelized when running on a multi-core processor. This can improve the execution time of MPC but it must be done ensuring that the task in charge of receiving new messages is able to run. Every received message needs to be processed and it must be recalled that every vehicle broadcasts several messages per second, making it possible to receive several hundreds of messages per second when there is traffic.

8 Conclusion

In this paper we analysed the feasibility of employing embedded MPC for vehicle platooning and provided an overview of the trade-offs that can be done. We obtained a bound for the maximum execution time admissible when taking into consideration the other system tasks that run on the platform. We have shown that it is feasible in two different ways. First, we analysed the computational complexity of different MPC algorithms and compared it to the performance of the device, obtaining a theoretical execution time. Second, we used automatic C code generation tools to measure the real execution time of MPC algorithms for different control horizon lengths. We compared the execution times to the execution time bounds, showing that it is feasible to use embedded MPC for vehicle platooning. In this process, we have benchmarked the performance of various MPC algorithms with respect to parameters such as the horizon length and the number of the states.

References

- 1 Alessandro Alessio and Alberto Bemporad. A Survey on Explicit Model Predictive Control. In *Nonlinear Model Predictive Control, LNCIS 384*, pages 345–369. Springer-Verlag, Berlin Heidelberg, 2009.
- 2 Auckland University of Technology. jMPC Toolbox. URL: <http://www.i2c2.aut.ac.nz/Resources/Software/jMPCToolbox.html>.
- 3 Alberto Bemporad and Panagiotis Patrinos. Simple and Certifiable Quadratic Programming Algorithms for Embedded Linear Model Predictive Control. *IFAC Proceedings Volumes*, 45(17):14–20, 2012.
- 4 Daniele Bernardini. ODYS and GM bring online MPC to production! | ODYS. URL: <http://www.odys.it/odys-and-gm-bring-online-mpc-to-production/>.

- 5 L.G. Bleris, P.D. Vouzis, M.G. Arnold, and M.V. Kothare. A co-processor FPGA platform for the implementation of real-time model predictive control. In *American Control Conference (ACC)*, pages 1912–1917, 2006.
- 6 Catalin Braescu, Razvan C. Rafaila, Alexandru Tiganasu, Anca Maxim, and Constantin F. Caruntu. Distributed model predictive control algorithm for vehicle platooning. *International Conference on System Theory, Control and Computing (ICSTCC)*, pages 657–662, 2016.
- 7 Eduardo F. Camacho and Carlos Bordons Alba. *Model Predictive Control*. Springer-Verlag London, London, 2007.
- 8 Gionata Cimini and Alberto Bemporad. Exact Complexity Certification of Active-Set Methods for Quadratic Programming. *IEEE Transactions on Automatic Control*, 62(12):6094–6109, 2017.
- 9 J Currie, A Prince-Pike, and D I Wilson. Auto-code generation for fast embedded Model Predictive Controllers. *International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 116–122, 2012.
- 10 Alexander Domahidi, Hans Joachim Ferreau, Stefan Almer, Helfried Peyrl, and Juan Luis Jerez. Survey of industrial applications of embedded model predictive control. *European Control Conference (ECC)*, pages 601–601, 2016.
- 11 Embotech. FORCES Pro code generator. URL: <https://www.embotech.com/forces-pro>.
- 12 ETH Zurich. FiOrdOs - Code Generation for First-Order Methods. URL: <http://fiordos.ethz.ch/dokuwiki/doku.php>.
- 13 European Telecommunications Standards Institute. Intelligent Transport Systems (ITS); Harmonized Channel Specifications for Intelligent Transport Systems operating in the 5 GHz frequency band, 2012.
- 14 S. Gopi, V. M. Vaidyan, and M. V. Vaidyan. Implementation of FPGA based model predictive control for MIMO systems. In *IEEE Conference on Systems, Process Control (ICSPC)*, pages 21–24, 2013.
- 15 Amr Ibrahim, Chetan Belagal Math, Dip Goswami, Twan Basten, and Hong Li. Co-simulation Framework for Control, Communication and Traffic for Vehicle Platoons. *Euromicro Conference on Digital System Design (DSD)*, pages 352–356, August 2018.
- 16 Stephen J. Wright. Applying new optimization algorithms to model predictive control. *International Conference on Chemical Process Control – CPC V*, pages 147–155, 1996.
- 17 Juan L Jerez, Paul J Goulart, Stefan Richter, George A Constantinides, Eric C Kerrigan, and Manfred Morari. Embedded Predictive Control on an FPGA using the Fast Gradient Method. *European Control Conference (ECC)*, pages 3614–3620, 2013.
- 18 Dimitris Kouzoupis. Complexity of First-Order Methods for Fast Embedded Model Predictive Control. Master’s thesis, ETH Zurich, 2014.
- 19 Mark S. K. Lau, S. P. Yue, K. V. Ling, and J. M. Maciejkowski. A comparison of interior point and active set methods for FPGA implementation of Model Predictive Control. *Proceedings of the European Control Conference*, pages 3–8, 2009.
- 20 Jacob Mattingley and Stephen Boyd. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012. doi:10.1007/s11081-011-9176-9.
- 21 V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura. Cooperative Adaptive Cruise Control in Real Traffic Situations. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):296–305, 2014.
- 22 Khalil Mohamed, Ahmed El Mahdy, and Mohamed Refai. Model Predictive Control Using FPGA. *International Journal of Control Theory and Computer Modeling*, 5(2), 2015.
- 23 Yurii Nesterov. *Introductory Lectures on Convex Optimization*, volume 87 of *Applied Optimization*. Springer US, 2004.
- 24 ODYS. ODYS QP solver, 2018. URL: <http://www.odys.it/qp/>.
- 25 Yasser Shoukry, M. Watheq El-Kharashi, and Sherif Hammad. MPC-On-chip: An embedded GPC coprocessor for automotive active suspension systems. *IEEE Embedded Systems Letters*, 2(2):31–34, 2010.

- 26 M Tsujii, H Takeuchi, K Oda, and M Ohba. Application of self-tuning to automotive cruise control. In *American Control Conference, 1990*, pages 1843–1848. IEEE, 1990.
- 27 A Galip Ulsoy, Huei Peng, and Melih Çakmakci. *Automotive control systems*. Cambridge University Press, 2012.
- 28 Adrian G Wills, Geoff Knagge, and Brett Ninness. Fast Linear Model Predictive Control Via Custom Integrated Circuit Architecture. *IEEE Transactions on control systems technology*, 20(1):59–71, 2012.
- 29 Sijie Zhu. NXP Platoon Control Algorithm Evaluation Platform. URL: <https://github.com/sijiezhuzhu/NXP-Platoon-Control-Algorithm-Evaluation-Platform>.
- 30 Sijie Zhu. Model-in-the-loop Experiments and Analysis of Platoon Control Algorithms. Master's thesis, Technische Universiteit Eindhoven, 2018. URL: <https://research.tue.nl/en/studentTheses/model-in-the-loop-experiments-and-analysis-of-platoon-control-alg>.
- 31 P. Zometa, M. Kogel, T. Faulwasser, and R. Findeisen. On Time versus Space and Related Problems. In *American Control Conference (ACC)*, pages 57–64, 2012.
- 32 P Zometa, M Kögel, and R Findeisen. μ AO-MPC Documentation. Technical report, Otto von Guericke Universität Magdeburg, 2016. URL: <http://ifatwww.et.uni-magdeburg.de/syst/research/muAO-MPC/doc/muaompc-1.0.pdf>.

A

 Parameters used in experimental analysis

$$\begin{aligned}
 Q &= \begin{bmatrix} 1.89 & 0 & 0 & 0 \\ 0 & 1.90 & 0 & 0 \\ 0 & 0 & 1.13 & 0 \\ 0 & 0 & 0 & 1.21 \end{bmatrix} \quad \Gamma = \begin{bmatrix} 1.75 \\ 1.90 \\ 0.69 \\ 1.61 \end{bmatrix} \quad x_0 = \begin{bmatrix} 0.20 \\ 0.83 \\ -0.84 \\ 0.04 \end{bmatrix} \quad x_{max} = \begin{bmatrix} 0.61 \\ 0.23 \\ -0.55 \\ -1.10 \end{bmatrix} \\
 P &= \begin{bmatrix} 1.44 & 0 & 0 & 0 \\ 0 & 1.03 & 0 & 0 \\ 0 & 0 & 1.46 & 0 \\ 0 & 0 & 0 & 1.65 \end{bmatrix} \quad \Phi = \begin{bmatrix} -0.88 & 0.71 & 0.36 & -1.90 \\ 0.24 & -0.96 & -0.34 & -0.87 \\ 0.77 & -0.24 & -1.37 & 0.18 \\ 1.12 & -0.77 & -1.11 & -0.45 \end{bmatrix} \quad x_{min} = \begin{bmatrix} -1.63 \\ -100 \\ -100 \\ -100 \end{bmatrix} \\
 R &= 1.05 \quad u_{max} = 1.38 \quad u_{min} = -0.49 \quad \Psi = [0 \ 0 \ 0 \ 0]
 \end{aligned}$$

IDF-Autoware: Integrated Development Framework for ROS-Based Self-Driving Systems Using MATLAB/Simulink

Shota Tokunaga

Graduate School of Engineering Science, Osaka University, Osaka, Japan

Yuki Horita

Hitachi, Ltd., Yokohama, Japan

Yasuhiro Oda

Hitachi Automotive Systems, Ltd., Hitachinaka, Japan

Takuya Azumi

Graduate School of Science and Engineering, Saitama University, Saitama, Japan

Abstract

This paper proposes an integrated development framework that enables co-simulation and operation of a Robot Operating System (ROS)-based self-driving system using MATLAB/Simulink (IDF-Autoware). The management of self-driving systems is becoming more complex as the development of self-driving technology progresses. One approach to the development of self-driving systems is the use of ROS; however, the system used in the automotive industry is typically designed using MATLAB/Simulink, which can simulate and evaluate the models used for self-driving. These models are incompatible with ROS-based systems. To allow the two to be used in tandem, it is necessary to rewrite the C++ code and incorporate them into the ROS-based system, which makes development inefficient. Therefore, the proposed framework allows models created using MATLAB/Simulink to be used in a ROS-based self-driving system, thereby improving development efficiency. Furthermore, our evaluations of the proposed framework demonstrated its practical potential.

2012 ACM Subject Classification Information systems → Open source software

Keywords and phrases self-driving systems, framework, robot operating system (ROS), MATLAB/Simulink

Digital Object Identifier 10.4230/OASICS.ASD.2019.3

Funding This work was partially supported by Mr. Tohru Kikawada and JST PRESTO, Japan (grant No. JPMJPR1751).

Yasuhiro Oda: Presently with Hitachi Industry & Control Solutions, Ltd.

1 Introduction

Self-driving systems continuously increase in complexity along with the increasing number of required functionalities. One approach to the development of complicated systems is the use of Robot Operating System (ROS) [5] [12] [13]. ROS characteristics, such as abstracting hardware and improving code reusability, make the development of such systems more efficient. A ROS-based self-driving system is Autoware [1]. Autoware is open-source software for autonomous vehicles and can be used in embedded systems, such as NVIDIA DRIVE PX2 [10] and Kalray MPPA-256 [11].



© Shota Tokunaga, Yuki Horita, Yasuhiro Oda, and Takuya Azumi;
licensed under Creative Commons License CC-BY

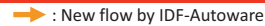
Workshop on Autonomous Systems Design (ASD 2019).

Editors: Selma Saidi, Rolf Ernst, and Dirk Ziegenbein; Article No. 3; pp. 3:1–3:9

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** System model of IDF-Autaware.

However, in the automotive industry, the design of self-driving subsystems, such as detection, planning, and control have often used MATLAB®/Simulink® [3]. The models designed using MATLAB/Simulink can not be directly linked to Autoware in the currently adopted development framework. To integrate such models into Autoware, it is necessary to generate and incorporate the associated C++ code. Although MATLAB/Simulink has a C++ code generation functionality, code corresponding to Autoware (i.e., ROS) can not be generated, thereby deteriorating development efficiency. Moreover, it is possible that a model ported to Autoware will not perform as designed because the MATLAB/Simulink environment differs from that of Autoware. To address these limitations, we propose a framework called IDF-Autoware [2] (Figure. 1) that manages models designed using MATLAB/Simulink as *nodes* that represent individual processes in ROS. This enables data exchange between Autoware and MATLAB/Simulink, thereby allowing the models to be used without incorporation into Autoware.

To the best of our knowledge, this is the first work that co-simulation and operation of a real vehicle using MATLAB/Simulink for self-driving systems. The main contributions of this study are as follows:

- We confirmed the practicality of the method by comparing the data transfer time and processing capacity of ROS and MATLAB/Simulink (Section 3.1), as well as that the *nodes* designed using MATLAB/Simulink could be applied to the co-simulation and operation of an autonomous vehicle;
- We improved the design efficiency in MATLAB/Simulink based on IDF-Autoware generating MATLAB template scripts and Simulink template models (Section 3.2), which help a developer design *nodes* for Autoware using MATLAB/Simulink;
- We improved usability by extending Runtime Manager, which is a graphical user interface (GUI) tool for Autoware, to enable operations for MATLAB/Simulink (Section 3.3), as well as making available the other functionalities provided by IDF-Autoware (e.g., template generation).

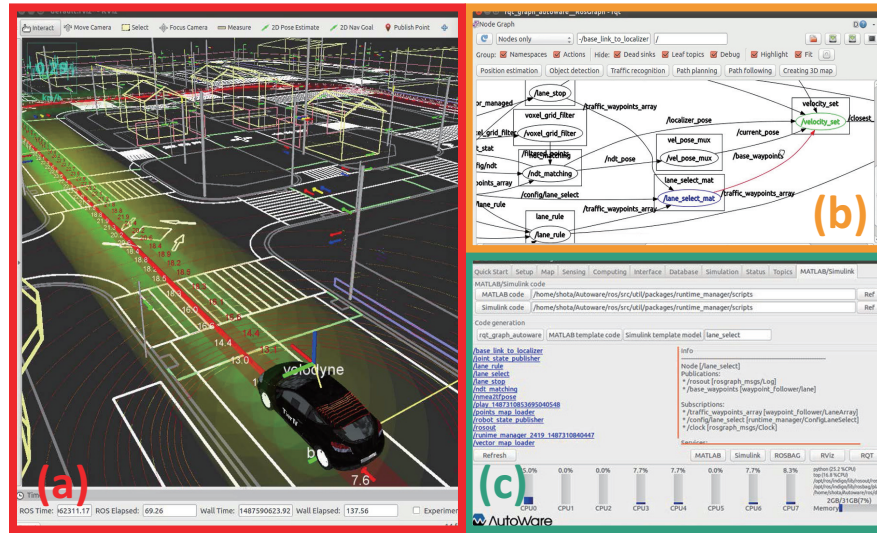


Figure 2 Screenshot of co-simulation using IDF-Autoware: (a)RViz displaying Autoware status, (b) the `rqt_graph_autoware`, and (c) the Runtime Manager for IDF-Autoware.

2 Design and Implementation

The functionalities provided by IDF-Autoware facilitate the integrated development of Autoware and MATLAB/Simulink. The key functionalities are as follows (Figure. 1):

- They generate MATLAB template scripts and Simulink template models, and provide visualization tools to aid template generation (Section 2.1);
- They enable MATLAB/Simulink to operate on Runtime Manager, to display *node* information, and to make use of the other provided functionalities (Section 2.2)

In this section, we discuss the design and implementation of each of these functionalities, and use cases of the proposed framework are shown.

2.1 Template Generation

When MATLAB/Simulink is used to design *nodes* for Autoware, the *nodes* must contain essential information, such as a *node* name, the *topics* to publish/subscribe, and the *message* type of each *topic*. This information can be obtained by analyzing the source code of Autoware and executing ROS commands. However, the need for such analyses places a burden on developers, especially on those who are unfamiliar with ROS. Therefore, we provided functionalities that allow the generation of MATLAB template scripts and Simulink template models that include this necessary information, as the templates help developers design *nodes* in MATLAB/Simulink. Additionally, we made two visualization tools to aid the template generation. One is the `rqt_graph_autoware` plugin (Figure. 2 (b)). In addition to the functionalities of `rqt_graph` [7], `rqt_graph_autoware` can render *node* dependency, such as sensing, perception, decision, and planning, for Autoware applications. The other tool displays a list of the running *nodes* and provides information on any *node* selected from the list.

As noted, before the template of a desired *node* is generated, it is necessary to obtain node information; therefore, a .yaml file containing information pertaining to all Autoware *nodes* was created. Based on this information, templates are created using functions provided by Robotics System Toolbox™ [4], which provides the interface between ROS and MATLAB/Simulink. Developers can create *nodes* for Autoware in MATLAB/Simulink using the generated template.

To implement the `rqt_graph_autoware` plugin, we created .dot files that render *node* dependency graphs for each Autoware's application. Moreover, to create the GUI for `rqt_graph_autoware`, we added buttons to `rqt_graph` using Qt designer, which is a Qt tool for designing a GUI. The buttons were configured to open each .dot file, and clicking on these buttons cause a graph to be drawn. This allows developers visualization of the *nodes* included in each Autoware's application.

To display *node* information, we used a `roscmd` command-line tool [6] that includes commands that fetch *node* information, including `roscmd list` and `roscmd info node_name`. The `roscmd list` command displays a list of running *nodes*, whereas `roscmd info node_name` displays information about the *topics* to be published/subscribed by the *node*. Displaying the results of these commands in Runtime Manager renders the *node* information easily comprehensible. Section 2.2 describes the method for displaying these results in Runtime Manager.

2.2 Runtime Manager for IDF-Autoware

Autoware and MATLAB/Simulink are operated with different GUI tools; thus, this is troublesome for users who want to use the two simultaneously. Therefore, we added GUIs to the Autoware's GUI tool (i.e., Runtime Manager) to allow use of MATLAB/Simulink and the functionalities provided in IDF-Autoware (Figure. 2 (c)). These GUIs enabled the following functionalities:

- Starting MATLAB, Simulink, and `rqt_graph_autoware`;
- Executing MATLAB scripts and Simulink models;
- Generating MATLAB template scripts and Simulink template models;
- Displaying *node* information.

This unification of operation method simplifies the MATLAB/Simulink operation and the utilization of the provided functionalities.

Runtime Manager was designed using the wxPython toolkit [9]. Therefore, we designed the GUIs for the added functionalities using wxGlade [8], and outputted its designs as wxPython. The GUIs involve buttons and panels that execute each functionality.

We next modified the Runtime Manager execution code to configure them for GUI functionalities. The execution code imports modules, including the code generated by wxGlade, and loads the .yaml files. In the execution code, loading .yaml files initiates functions that align simple operations to specified buttons. Therefore, by creating a yaml file for MATLAB/Simulink, we configured the initiation of MATLAB, Simulink, and `rqt_graph_autoware` to each button.

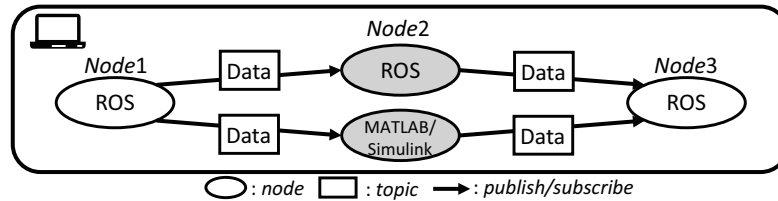
To allow the execution of MATLAB scripts and Simulink models from Runtime Manager, we created multiple GUIs with the following configurations:

- A button to open a dialog for file selection;
- A panel displaying the absolute path of the selected file; and
- A button to execute the file displayed on the panel.

This execution button was designed to run if the selected file was a MATLAB/Simulink file (i.e., a .m or .slx file).

■ **Table 1** Evaluation environment.

CPU	Model number	Intel Core i7-6700K
	Cores	4
	Threads	8
	Frequency	4.00 GHz
Memory		32 GB
ROS		Indigo
MATLAB/Simulink		R2016b
OS		Ubuntu 14.04.5 LTS



■ **Figure 3** Measurement of transfer time.

To generate MATLAB template scripts and Simulink template models, we designed the following GUIs: a panel to input the *node* name and buttons to run the execution code that generates the template of the input *node*.

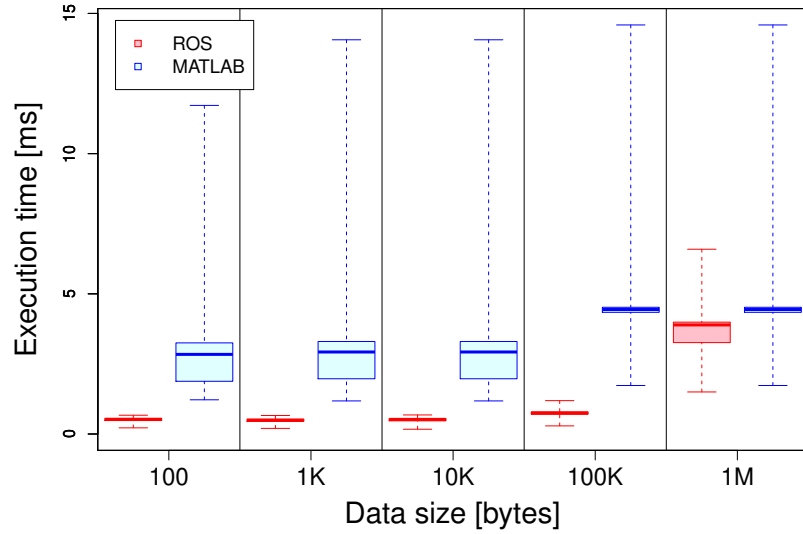
For the *node* information display, we designed two panels, with the first displaying the output of the executing *rostopic list*. When a *node* is selected from the list, the second panel displays the output of *rostopic info the_selected_node_name*, which eliminates the need to enter the *rostopic* command.

2.3 Use Case

IDF-Autoware allows co-simulation of Autoware and MATLAB/Simulink. The demonstration video can be viewed at the following hyperlink: <https://youtu.be/X4d9VbXnPeg> (Figure. 2). In this video, one of the *nodes* necessary for planning is executed by MATLAB/Simulink. This simulation facilitates an operational check of MATLAB/Simulink *nodes*. Moreover, it can also be used for experiments using an autonomous vehicle. The demonstration video showing operating of the autonomous vehicle using IDF-Autoware can be seen at the following hyperlink: <https://youtu.be/wusCU2VPGGQ>.

3 Evaluations

The main goal of this study was to improve development efficiency. To demonstrate this improvement, the practicality of IDF-Autoware, efficiency, and usability were evaluated. To evaluate the practicality, we compared the communication times among *nodes* within ROS and between ROS and MATLAB/Simulink. Additionally, we performed a co-simulation and operation of an autonomous vehicle to show the practicality of the proposed framework. We investigated the design efficiency by measuring the generated MATLAB/Simulink template. To evaluate the usability, we compared the development environments with Autoware, Robotics System Toolbox, and IDF-Autoware. These evaluations demonstrated that IDF-Autoware improved the development efficiency. Table 1 summarizes the software and hardware environments used in the experiments.



■ **Figure 4** The average transfer time according to the size of the *message* data.

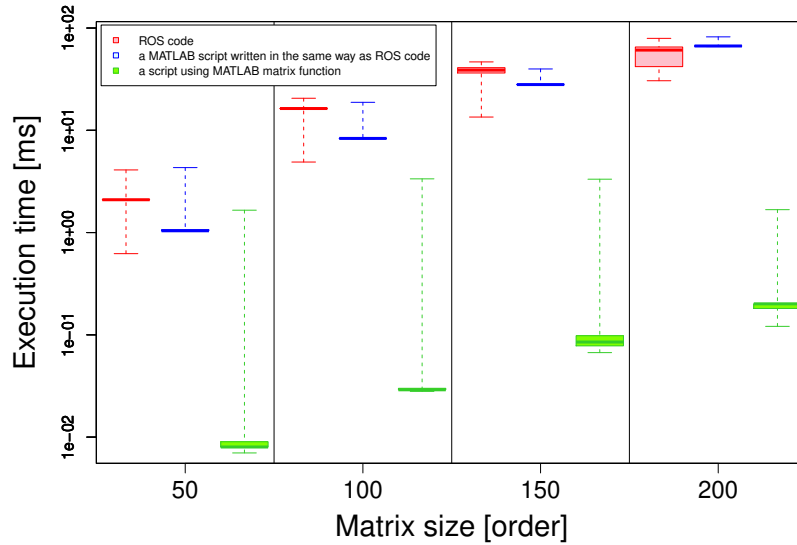
3.1 Practicality

IDF-Autoware enabled the communication of nodes designed using MATLAB/Simulink with Autoware *nodes* to improve the development efficiency. However, it was necessary to consider the effect of using Autoware with only ROS and together with MATLAB/Simulink together. Therefore, to evaluate practicality, ROS and MATLAB/Simulink were compared as follows:

1. According to the relationship between the transfer time and the data size when a *message* is sent via ROS and via MATLAB/Simulink, respectively; and
2. According to the processing capacity when the same type of method was used.

As shown in Figure. 3, the transfer time was defined as the elapsed time when *Node 1* published the *message* to *Node 3*, which subscribed the *message* via *Node 2*. The processing capacity was compared with the processing time over 1,000 iterations and using the same machine (*Node 1* published the *message* at 10 Hz).

We measured the ROS and MATLAB/Simulink transfer time when the *message* data size on each *topic* was set to 100, 1 K, 10 K, 100 K, and 1 M bytes. Figure. 4 shows the transfer times via ROS and MATLAB/Simulink plotted against each data size. Both the ROS and MATLAB/Simulink transfer times increased along with data size, although the data transfer by MATLAB/Simulink had an overhead exceeding that of ROS. However, the MATLAB/Simulink transfer time did not exceed the Autoware maximum of 32 Hz.



■ **Figure 5** The average processing time according to each matrix size.

■ **Table 2** Task reduction using MATLAB template scripts.

	MATLAB template scripts
Generated lines	$(1) + \alpha(2) + \beta((3) + 2(4))$
(1): Defining <i>node</i>	
(2): Defining <i>publisher</i>	
(3): Defining <i>subscriber</i>	
(4): Defining callback function	
α : The number of <i>publishers</i>	
β : The number of <i>subscribers</i>	

To evaluate the processing capacity, we measured the processing times of ROS and MATLAB/Simulink when multiplying square matrices on the order of 50, 100, 150, and 200, which served as easy points of reference to enable comparison of ROS with MATLAB/Simulink rather than as a requirement for self-driving. The evaluation measured the time required to process the time complexity at each matrix size and assessed the performance of the functions provided by MATLAB/Simulink. Therefore, the MATLAB/Simulink processing time was measured using two MATLAB scripts: one written in the same way as the ROS code, and the other using MATLAB matrix functions. Figure. 5 shows the processing times at each matrix size. When using the MATLAB script written in the same way as the ROS code, the processing times of ROS and MATLAB/Simulink were approximately the same. By contrast, when the MATLAB script used matrix functions, its processing time was significantly shorter than that of the other two methods, because processing was executed on multiple cores with multiple threads, even when this was unspecified. Comparison of the processing times with the transfer times revealed that the script using matrix functions was again significantly faster, thereby confirming that application of the functions provided by MATLAB/Simulink code enabled the handling of processes with large time complexity (e.g., image processing), even when accounting for the transfer time. Therefore, as shown the videos in Section 2.3, the practicality of IDF-Autoware is demonstrated.

■ **Table 3** Task reduction using Simulink template models.

	Simulink template models
Simulink blocks	$\alpha((1) + (2) + (3)) + \beta((4) + (5) + (6))$
Settings	$(i) + (\alpha + \beta)((ii) + (iii) + (iv) + 2(v))$
(1): Placing <i>Publisher</i>	(i): Defining model name
(2): Placing <i>Message</i>	(ii): setting <i>message</i> name
(3): Placing Bus Assignment	(iii): setting <i>topic</i> name
(4): Placing <i>Subscriber</i>	(iv): Configuring <i>topic</i> source
(5): Placing Bus Selector	(v): Connecting blocks
(6): Placing Terminal	α : The number of <i>publishers</i>
	β : The number of <i>subscribers</i>

■ **Table 4** Functionalities available with Autoware, Robotics System Toolbox, and IDF-Autoware.

	Autoware [1]	Robotics System Toolbox [4]	IDF-Autoware [2]
Operating Autoware	✓		✓
Operating MATLAB/Simulink		✓	✓
Communicating between Autoware and MATLAB/Simulink		✓	✓
Drawing node dependency	✓		✓
Generating MATLAB/Simulink templates			✓
Displaying node information			✓

3.2 Efficiency

To improve the design efficiency, a functionality to generate both MATLAB template scripts and Simulink template models was provided. These templates help developers design *nodes* for Autoware in MATLAB/Simulink.

Table 2 shows the amount of the template generated by a MATLAB template script. The MATLAB template script defines the essential information, as mentioned in Section 2.1, and creates callback functions utilized when a *topic* is subscribed. For example, the *lane_stop* *node* required for planning has one *publisher* and five *subscribers*. One line is generated to define a *node*, a *subscriber*, and a *publisher*, and two lines are generated to define the callback function. Therefore, in total, 17 lines are generated for the MATLAB template script for the *lane_stop* *node*.

When creating a Simulink model, it is necessary to place and configure the Simulink blocks, to define the model name, and to connect the blocks. Table 3 summarizes the number of Simulink blocks placed and the settings created by a Simulink template model. The Simulink template model defines the model name and places the essential Simulink blocks, thereby creating a model for Autoware. Additionally, the Simulink blocks are configured and connected together. For example, when the Simulink template model of *lane_stop* *node* is generated, 18 Simulink blocks are placed and 31 settings are configured in total.

If the functionality allowing MATLAB/Simulink templates to be generated is not provided, the developer must examine the *node* information and define it in a MATLAB script or a Simulink model. By contrast, when the templates are used, this becomes unnecessary; therefore, this improves design efficiency.

3.3 Usability

IDF-Autoware enables the operation of MATLAB/Simulink in Autoware and provides functionalities to improve the usability. Here, we compared the available functionalities between Autoware, Robotics System Toolbox, and IDF-Autoware, as summarized in Table 4.

Autoware cannot operate MATLAB/Simulink, and Robotics System Toolbox cannot operate Autoware. IDF-Autoware provides functionalities required to operate MATLAB/Simulink in Runtime Manager for IDF-Autoware, such as starting MATLAB/Simulink or executing MATLAB scripts and Simulink models. Therefore, IDF-Autoware can operate both systems. Communication between Autoware and MATLAB/Simulink is possible in Robotics System Toolbox and IDF-Autoware. Moreover, IDF-Autoware provides a drawing to visualize *node* dependency using the `rqt_graph_autoware` plugin created by extending `rqt_graph` available in Autoware. In addition to these features, IDF-Autoware can generate MATLAB/Simulink templates and display *node* information. Because this increases the number of available functionalities, the usability is also enhanced, which in turn improves development efficiency.

4 Conclusion

In this paper, we described the development of an integrated development framework for Autoware with MATLAB/Simulink (IDF-Autoware) that enabled communication between Autoware and MATLAB/Simulink. We evaluated the data transfer time and processing capacity of MATLAB/Simulink, and confirmed the practicality of the method by using both co-simulations and experiments using an autonomous vehicle. IDF-Autoware facilitated the generation of MATLAB/Simulink templates that can help developers create models using MATLAB/Simulink for Autoware, thereby improving the design efficiency. Furthermore, the functionalities added to IDF-Autoware allow Runtime Manager to operate MATLAB/Simulink and various functionalities, further improving usability. Our findings confirmed that IDF-Autoware improved the development efficiency.

References

- 1 Autoware/github.com. URL: <http://github.com/CPFL/Autoware>.
- 2 IDF-Autoware/github.com. URL: <https://github.com/T-Shota/IDF-Autoware>.
- 3 MATLAB/Simulink. URL: <http://www.mathworks.com>.
- 4 Robotics System Toolbox. URL: <https://mathworks.com/products/robotics.html>.
- 5 ROS.org. URL: <http://www.ros.org>.
- 6 ROS.org/rosnode. URL: <http://wiki.ros.org/rosnode>.
- 7 ROS.org/rqt_graph. URL: http://wiki.ros.org/rqt_graph.
- 8 wxglade.sourceforge.net. URL: <http://wxglade.sourceforge.net>.
- 9 wxpython.org. URL: <http://wxpython.org>.
- 10 S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi. Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems. In *Proc. of ICCPS*, 2018.
- 11 Y. Maruyama, S. Kato, and T. Azumi. Exploring Scalable Data Allocation and Parallel Computing on NoC-Based Embedded Many Cores. In *Proc. of ICCD*, 2017.
- 12 M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *Proc. of ICRA, Open-Source Software Workshop*, 2009.
- 13 Y. Saito, T. Azumi, S. Kato, and N. Nishio. Priority and Synchronization Support for ROS. In *Proc. of CPSNA*, 2016.

Controlling Concurrent Change – A Multiview Approach Toward Updatable Vehicle Automation Systems

Mischa Möstl 

Technische Universität Braunschweig, Institute of Computer and Network Engineering
Hans-Sommer-Str. 66, 38106 Braunschweig, Germany
moestl@ida.ing.tu-bs.de

Marcus Nolte

Technische Universität Braunschweig, Institute of Computer and Network Engineering
Hans-Sommer-Str. 66, 38106 Braunschweig, Germany
nolte@ifr.ing.tu-bs.de

Johannes Schlatow 

Technische Universität Braunschweig, Institute of Computer and Network Engineering
Hans-Sommer-Str. 66, 38106 Braunschweig, Germany
schlatow@ida.ing.tu-bs.de

Rolf Ernst

Technische Universität Braunschweig, Institute of Computer and Network Engineering
Hans-Sommer-Str. 66, 38106 Braunschweig, Germany
ernst@ida.ing.tu-bs.de

Abstract

The development of SAE Level 3+ vehicles [24] poses new challenges not only for the functional development, but also for design and development processes. Such systems consist of a growing number of interconnected functional, as well as hardware and software components, making safety design increasingly difficult. In order to cope with emergent behavior at the vehicle level, thorough systems engineering becomes a key requirement, which enables traceability between different design viewpoints. Ensuring traceability is a key factor towards an efficient validation and verification of such systems. Formal models can in turn assist in keeping track of how the different viewpoints relate to each other and how the interplay of components affects the overall system behavior. Based on experience from the project Controlling Concurrent Change, this paper presents an approach towards model-based integration and verification of a cause effect chain for a component-based vehicle automation system. It reasons on a cross-layer model of the resulting system, which covers necessary aspects of a design in individual architectural views, e.g. safety and timing. In the synthesis stage of integration, our approach is capable of inserting enforcement mechanisms into the design to ensure adherence to the model. We present a use case description for an environment perception system, starting with a functional architecture, which is the basis for componentization of the cause effect chain. By tying the vehicle architecture to the cross-layer integration model, we are able to map the reasoning done during verification to vehicle behavior.

2012 ACM Subject Classification Hardware → Safety critical systems

Keywords and phrases safety, behavior, functional, architecture, multi-view, automated driving

Digital Object Identifier 10.4230/OASICS.ASD.2019.4

Funding This work was funded by the Deutsche Forschungsgemeinschaft (DFG) under the grant FOR1800 Controlling Concurrent Change (CCC).

Acknowledgements The authors thank the other members of the CCC research group for many fruitful discussions and their support.



© Mischa Möstl, Marcus Nolte, Johannes Schlatow, and Rolf Ernst;
licensed under Creative Commons License CC-BY
Workshop on Autonomous Systems Design (ASD 2019).

Editors: Selma Saidi, Rolf Ernst, and Dirk Ziegenbein; Article No. 4; pp. 4:1–4:15



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In recent years, huge progress has been generated toward the commercialization of automated vehicles systems. The focus of the industry has shifted from advanced driver assistance systems (ADAS), corresponding to SAE level 1 and 2 [24] to automated vehicle systems of SAE Levels 3+. However, while impressive results are achieved regarding environment perception algorithms, also due to the introduction of machine learning technology, verification and validation of Level 3+ systems becomes increasingly difficult. This is especially true, if it must be considered that software intense systems will most likely require frequent after-market updates for deploying bugfixes, and/or updates of the vehicle’s functionality.

Challenges for safety verification are on the other hand caused by increased complexity of the perception systems required to generate a representation of the vehicle’s environment which is sufficiently detailed to make decisions in complex traffic scenes (cf. [13]). On the other hand, replacing the driver is equivalent to replacing vast parts of the safety system of SAE Level 1 and 2 systems. Established safety design processes must thus be rethought and extended in order to suit the newly arising challenges when removing the driver from the control loop. Safety strategies which only assure that the driver can control system failures by being able to physically overrule system commands to the drive train or steering system do not apply anymore.

For the automotive industry, the safety standard ISO 26262 [10] provides guidelines for designing functionally safe systems. This subsumes hazards caused by malfunctioning behavior of E/E components and ensures the correct implementation of functional (safety) requirements. One frequently formulated drawback of the ISO regarding the applicability to Level3+ systems is that it does not consider nominal behavior of the overall E/E system (cf. [19, 4, 13, 8]) and thus does not provide guidelines on how to define the functional requirements for the system. However, this formulation of safe nominal behavior (or *external behavior* as defined in [20], according to [3]) and the boundaries of safe nominal behavior is crucial when it comes to ensuring safety of driverless vehicles, as the system must not pose a threat to its passengers and/or other traffic participants. For this publication, we adopt the terminology as defined by Waymo in their 2017 safety report [32], referring to the process of defining safe nominal behavior as *behavioral safety* (cf. [4]). The upcoming ISO PAS 21448 “Road Vehicles – Safety of the Intended Functionality” is partially addressing this problem, however the scope of the current draft standard is intentionally limited to SAE Level 1 and 2 systems [12], while the defined concepts might also apply to levels of higher automation.

While there is a number of recent publications on how to extend the concept phase of ISO 26262 toward the definition of safe behavior [19, 4, 8], e.g. based on a scenario-driven concept phase, we would like to elaborate on the consequences of behavioral safety considerations from a systems engineering point of view. As we have argued in [4], the design of safe automated vehicle should follow a *safety by design* paradigm as a cross-domain effort over different disciplines. For this purpose we have proposed an architecture framework in accordance with ISO 42010 [11], featuring safety as a cross-cutting viewpoint and formulating a *functional*, a *capability*, *software* and *hardware* viewpoint and attributing *behavioral safety* to the former two and *functional safety* to the latter two viewpoints. Correspondences and correspondence rules, as defined in ISO 42010, are represented in example mappings between components in the respective viewpoints. While we formulate the need for formal methods to represent and instantiate the different viewpoints in the architecture framework, the actual instantiation was not part of the initial contribution.

Behavioral and functional safety of vehicle automation systems is one of the grand challenges for future automotive systems. Reducing the necessary testing efforts to ship updates for vehicular systems, especially of models that are already in production and in the field is the second grand challenge. The later is particularly interesting to reduce costs. In this paper, we want to show that concepts for safety related systems engineering ([4]) can be combined with automated integration mechanisms and tools as investigated in the project *Controlling Concurrent Change (CCC)*¹. As a result of such a design and integration flow, we envision systems where software updates and upgrades can be easily deployed at a minimum of cost for integration testing and safety validation through testing.

We illustrate this idea based on an update scenario for the automation system of a research vehicle. Therefore, we first introduce the architecture framework we use to assess behavioral safety in section 2. We maintain traceability from the functional viewpoint up until integration in this architecture (cf. Figure 2), by using Traceability in this architecture is maintained in two ways: For behavioral safety the process is still manual, first ideas to further automate this are also presented in section 2. The example showcase is then presented in section 3, while section 4 then presents the key ideas how we automate the integration and verification based on the presented architecture framework. This section also includes a description of the resulting cross-layer system model. Finally, section 5 concludes the paper.

2 Behavioral Safety in Systems Engineering

As stated in Section 1, the concept of behavioral safety is a potential missing link to extend the concept phase of established ISO-26262-compliant processes toward the application for SAE Level 3+ vehicle automation systems. In this section, we summarize the architecture framework described in [4] and discuss the implications of behavioral safety on traceability requirements for system properties in the design phase and at runtime.

Considering behavioral safety as an integral part of the safety concept creates the problem of defining appropriate behavior in different scenarios [4]. An example scenario is displayed in Figure 1 with the vehicle approaching a pedestrian crosswalk.

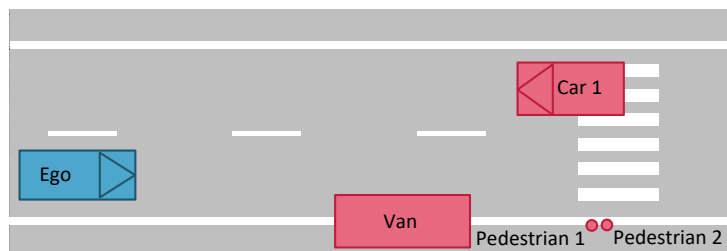


Figure 1 Example scenario: Automated vehicle approaching a pedestrian crosswalk occluded by a parked vehicle with oncoming traffic and pedestrians who are likely to cross.

At the scenario level, abstract safety goals can be formulated, e.g. by stating that the automated vehicle must not enter oncoming traffic. A process of how these abstract safety goals can be decomposed into (functional) safety requirements and actual technical requirements has been formulated in [4]. A short summary of the described process following an (iterative) Item Definition can be stated as follows:

¹ <https://ccc-project.org>

1. Conduct Hazard Analysis and Risk Assessment by possible accidents in the defined scenarios.
2. Define safety goals.
3. Define a risk minimal state for the scenario at hand.
4. Define functional safety concept (safety requirements and hazard mitigation strategies) for fulfilling the safety goals.
5. Combine with functional architecture and required system capabilities derived during the item definition to derive technical requirements.

However, the consequences of formulating behavioral safety requirements for systems development reaches further than defining requirements at the beginning of the development and validation and test before market release in a classic V-Model-like development process. In addition, the adherence to safety requirements must be monitored at runtime. This is required to initiate emergency strategies for reaching a risk minimal state in case safety requirements are violated.

For monitoring system behavior at runtime, we have proposed the application of ability and skill graphs [21] and their integration into a development process [20]. They represent functional dependencies in the system, formulating the required capabilities to fulfill the vehicle's mission. They explicitly model external system behavior as well as dependencies for performance assessment at a functional level, and provide guidance for the decomposition of functional requirements into technical requirements.

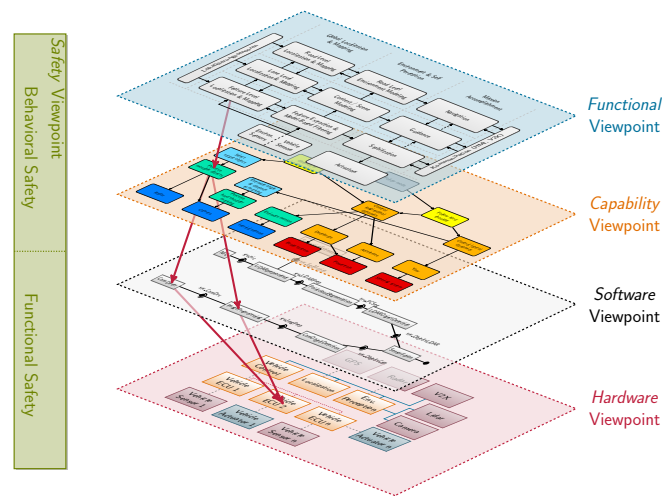
A core question which needs to be addressed in this context is, how the technical implementation, which is subject to functional safety requirements can cause hazards at the behavioral level. This is where traceability aspects come into play: Assuming that a *functional* system architecture and a capability representation are available after the concept phase of the development process, technical architectures in terms of hardware and software architectures are developed during system implementation. As defined in the ISO 42010, a sound architecture framework requires the formulation of correspondences and correspondence rules between different architectural views. In our formulated architecture framework (cf. Figure 2), this means that we need mapping relations between *functional*, *capability*, *hardware* and *software* components (depicted in Figure 2 as red arrows).

However, while informal formulations of those correspondences can assist during system development, informal notations are not suitable to support system monitoring at runtime. For this purpose a formal system model is required which can relate formalized requirements to the current system configuration, e.g. including component mappings or interface and task dependencies.

To demonstrate this, we performed a manual ability and skill graph based assesment for an automated driving function of a research vehicle, and used its results as the input for a model-based integration flow. The vehicle and the automation function is explained in the following section. How this function is integrated into a vehicle system in a correct-by-construction fashion is subsequently explained in section 4, which will explain our cross-layer model instantiating the multi-view architecture.

3 Concurrent Change Use-Case

The CCC approach combines a conventional lab-based design of individual functions with an automated integration process which ensures that updates are applied to an already deployed system only if the system can still adhere to the required safety and security constraints.



■ **Figure 2** Architecture framework presented in [4]: Showing safety as a cross-cutting viewpoint orthogonal to the functional, capability, hardware and software viewpoint. Note that depicted architectures are examples, such as a component architecture in the software viewpoint. Red arrows depict example mapping relations (correspondences) between components in different viewpoints.

This becomes particularly challenging as the target platform is shared by multiple functions with different criticality. All side effects must therefore be anticipated and either be bounded or mitigated in order to ensure safe operation of critical functions at all times. For this purpose, all requirements and constraints must be explicitly specified in the input models. Another challenge consists in finding (and specifying) appropriate abstractions that guide the decisions which must be made during such a model-based integration process, as these are usually based on experience and expert knowledge, which is only implicitly available.

We demonstrate the applicability of the approach on an environment perception and motion planing showcase that we will introduce in the following:

3.1 Research Vehicle MOBILE

For showing the applicability of the approaches developed in the CCC project in an automotive context, the research vehicle MOBILE [5] built at the Institute of Control Engineering at TU Braunschweig serves as a demonstrator platform.

MOBILE was originally built as a demonstrator for the development of vehicle dynamics control algorithms and vehicle systems engineering applications. It features of four close-to-wheel electric drives (4×100 kW), as well as individually steerable wheels, and electro-mechanic brakes [5]. The vehicle features a FlexRay backbone for inter-ECU-communication and additional CAN bus interfaces, which are used for communication with sensors and actuators for vehicle control. The ECUs for vehicle control are programmed in a customized MATLAB/Simulink tool chain. Combined with detailed vehicle-dynamics models, the tool chain serves as a means to establish a rapid-prototyping process for vehicle control algorithms.

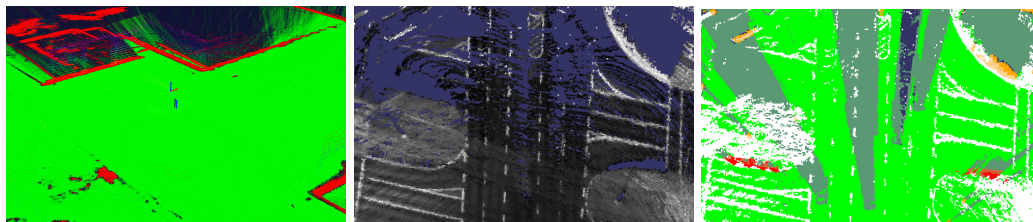
The basic idea in the project scope is to demonstrate how the CCC architecture can contribute to a state-of-the-art environment perception system in an automated vehicle. For this purpose, the research vehicle MOBILE has been equipped with three roof-mounted LiDAR sensors (cf. Figure 4a), as well as a highly accurate localization platform. Additional hardware platforms were installed in the vehicle to run environment perception and motion planning algorithms in the CCC middleware. The ECUs and sensors of the CCC subsystem

are interconnected by Ethernet and connected to the legacy vehicle control through a CAN interface. In addition, the algorithms can be run on a legacy platform as it is used in the *Stadtpilot* [33] project for comparison.

3.2 Environment Perception & Trajectory Planning System

The sensors provide a 360° representation of the vehicles' environment and enable it to navigate its path around obstacles in its vicinity. For the CCC project, we have ported selected algorithms from the *Stadtpilot* project, focusing on the representation of the static vehicle environment. For this purpose, incoming sensor raw data from the LiDAR sensors is combined into a point cloud. Each measurement contains position and reflectivity information. Thus, apart from the information about obstacle positions, reflectivity information can be used to create a monochrome image, making it possible to e.g. detect lane markings in the LiDAR data.

In several steps, measurements are annotated with measurement classes (e.g. ground measurements, valid measurements on actual objects, clutter, etc.). The resulting annotated point cloud is then fed into an occupancy grid [6] (cf. Figure 3a), which accumulates measurements over time. The grid framework is based on a multi-layer approach to represent environment features in distinct layers. Examples of three layers are depicted in Figure 3. Figure 3a shows an example of occupancy information in terms of free (green), occupied (red) and unknown (dark blue) space. In addition, the mentioned reflectance information (Figure 3b) for ground-labeled points is represented in a separate layer. For a more detailed description of the processing chain, please refer to [14], [23]. At the end of the sensor-data



(a) Occupancy grid: discretized map displaying free (green) and occupied areas (red) around the vehicle. (b) Reflectance grid: reflectance values allow detecting lane markings (white) [23]. (c) Fused grid layer: each color map indicates a different represented feature [23].

■ **Figure 3** Three layers of the grid framework to represent environment features on an intersection.

processing chain for the static environment, the different layers are fused into a consistent representation of the static vehicle environment.

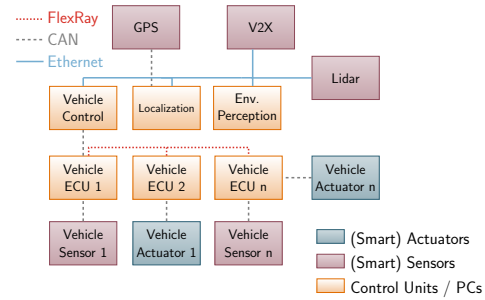
The grid representation is always kept in a local coordinate frame, which moves with the vehicle. The vehicle's position is acquired from an accurate tightly-coupled GNSS/INS platform (global position is obtained via GPS and fused with accelerations & angular rates).

(A) fused local occupancy grid(s) provide the basis to perform trajectory planning for automated driving. For this purpose, the system generates a target pose in a reachable area of the vehicle's environment and the trajectory is planned from the current position to the target pose in the vehicle coordinate frame. Trajectory planning is performed in a model-based fashion, using front- and rear-axle steering. The underlying trajectory control algorithms use the available actuators (4× steering, 4× brakes & drives) to control the vehicle to the planned trajectory. For details on and architectural considerations for trajectory planning, refer to [19]. Aspects of the applied control algorithms are presented in [30].

By representing the available actuators, trajectory planning considers the vehicle's current abilities. By monitoring e.g. sensor quality, actuator performance and control quality, the system will be able to react to failures in the system. A simple example here is the presence of a steering actuator failure, which can be compensated at the control level, as well as by adapting the trajectory planning algorithm. Monitoring of non-functional properties, such as timing is performed directly in the middleware.



(a) Roof-mounted lidar sensors.

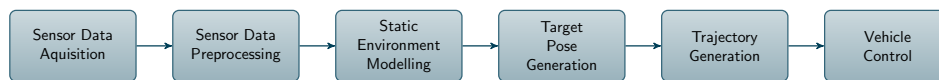


(b) Hardware architecture used in MOBILE.

■ **Figure 4** Research vehicle MOBILE.

The algorithms required to demonstrate the use case will run in a distributed system, as shown in Figure 4b. The platform can be separated into two parts: While the lower part of the displayed ECUs is responsible for controlling the individual actuators of the vehicle, the upper part performs environment perception and trajectory planning tasks. As the CCC middleware only runs in the context of the environment perception system, the system model must support transitions between legacy-parts of the system, running without the project middleware and those parts, which are fully controllable by the Multi Change Controller (cf. section 4).

A coarse grained functional architecture of the use-case is depicted in Figure 5.

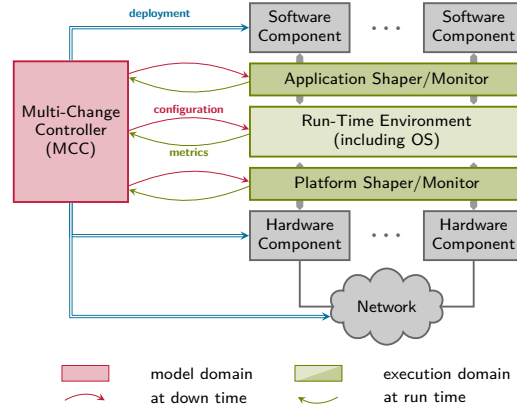


■ **Figure 5** Coarse functional system architecture.

4 CCC's integration and verification system

For the model-based integration approach pursued here, the system is composed of two segregated domains: the model domain and the execution domain.

Figure 6 shows the conceptual setup of the system. A Multi-Change Controller (MCC) (red) hosts the model domain, consisting of the cross-layer model, as well as configuration generation and verification. We aim for component-based models – including software as well as hardware components – as they reduce dependencies in the architecture to the explicitly modeled interfaces. The components are generic building blocks of the system that is composed of these components such that they implement the desired functionality and fit to the particular target platform. Each change to the system must be coherently representable in this system-wide model for analyzing any potential cross-layer dependencies, as well as for other analyses to ensure freedom from interference for the individual functions that a set of (software) components create.



■ **Figure 6** CCC architecture comprising a model domain (red), an execution domain (green) as well as changing software/hardware components (gray).

Similar to the conventional V-model development process, the MCC gradually refines the model representation of the new system configuration during the integration process. This is done based on a cross-layer model that captures relevant viewpoints of the system. The process generates new configurations and subsequently checks them for requirements satisfaction. If a new configuration satisfies all requirements and is rated as an improvement to the current one, it can be deployed into the execution domain.

Verification is separated due to the fact that not all requirements can be systematically considered during configuration generation. E.g. software response times are hard to optimize if arbitrary activation patterns are assumed. Consequently, an autonomous configuration and verification goes beyond a multi-dimensional optimization of requirement satisfaction.

Our execution domain, is based on the open-source *Genode OS Framework* [7]. This framework follows the microkernel approach and employs a strict decomposition of the system on the application level, resulting in a service-oriented architecture in which separate components implement and provide services for other components. While decomposition can already deal with liveliness issues [1] that arise in mixed-critical systems, dependencies on the execution time or response time of other components remain. Note that, however, the methods developed in the model domain are not restricted to these semantics but can be adapted to different implementation models.

4.1 The MCC's cross-layer model

The core concept of the MCC's model domain is that a) the system is represented on different layers of abstraction, and b) that models describing different viewpoints of the architecture are connected through mappings. Consequently, the described mappings between model artifacts are the implementation of correspondences from the abstract architecture framework.

To perform the integration task in the MCC we define three architecture layers, where each layer is treated as a graph. The top layer is a *function model*, that captures functional aspects implementation independent.

► **Definition 1.** A *function model* is a graph $\mathcal{FG} = (F, \hookrightarrow)$ where the nodes in F describe the functions, and \hookrightarrow is the set of edges that describe functional interactions.

For instance the function chain depicted in Figure 5 fulfills this definition.

A further necessity of such a layer lies in the fact that safety requirements are derived from implementation independent functional descriptions of a system [10, Part1] (cf. section 2). In order to implement functions, they are decomposed into components. Since during implementation, mappings of software components to hardware components might already be fixed, e.g. because code of one software component requires certain peripherals of a hardware component, they are already part of the component model. In our employed Run Time Environment (RTE), data exchange from one component to one or more others is performed through read-only memory (ROM) components. ROMs implement synchronous bulk transfer of data based on remote procedure calls (RPCs) [7]. If a reader on a remote resource requires contents of a ROM, proxy ROM components on both ends of the communication are inserted that provide the required data on the remote side via a network connection. Formally we define:

► **Definition 2.** *A component model is a graph $\mathcal{CG} = (C \cup \mathcal{C}_{Rom.s} \cup \mathcal{R}^{abs}, \xrightarrow{w} \cup \xrightarrow{r} \cup \xrightarrow{m})$ where the nodes are the unified set consisting of C that describes the set of software components implementing functions, $\mathcal{C}_{Rom.s}$ the set of ROM components, and \mathcal{R}^{abs} the set of abstract resources of the system. The edges either describe a read (\xrightarrow{r}) or write (\xrightarrow{w}) operation between software components and ROMs, or a mapping (\xrightarrow{m}) of a software component to a resource (\mathcal{R}^{abs}).*

In the course of generating configuration candidates the MCC applies pattern based transformations on \mathcal{FG} to produce a component model instance \mathcal{CG} . For the example use-case the function chain from Figure 5 is mapped to components in Figure 7 (second layer from the top). The transformation is based on selecting components that implement a function from a component repository. The repository is populated through formal xml-based descriptions of components. A more detailed account of this transformation is provided in [28].

In a subsequent step, the MCC's configuration generation refines the component model to an instance model, which only contains instantiated components. This process also allows refining components $c \in C$ into sub-components, which again can be linked by ROM components. The semantics of the resulting instance model are similar to \mathcal{CG} , however it only contains the minimal number of component instantiations under cardinality constraints, i.e. the maximum number of instantiations of a component on a particular CPU. This also results in a mapping of components to particular hardware components, i.e. from abstract resources to individual CPUs. The instance model of the use-case is depicted as the third layer from the top in Figure 7. Yet note, that some components are shown as composites (light blue) due to space limitations.

The knowledge of the concrete instance model together with the knowledge about the communication mechanisms allows the MCC to derive and map additional layers that model certain aspects of the system in order to represent particular viewpoints such as safety, availability or security. The requirements for these viewpoints – e.g. a safety-level requirement or a real-time constraint – are collected for each component in a so-called contracting language, which serves as an input to the MCC. Viewpoint-specific analyses are implemented as separate entities in the MCC, e.g. in order to resolve run-time dependencies between software components as presented in [27].

4.2 Analysis and Verification by the MCC

For this paper we restrict the scope to outlining how timing and safety requirements are verified by the MCC. W.r.t. safety we further limit ourselves to freedom from timing interference. For the external behavior of the vehicle, timing properties are crucial when it

comes to vehicle control. As unaccounted delays can cause degraded control performance or even instable controllers, the adherence to timing constraints in the timing domain must be ensured.

In order to reason about end-to-end function timing, a model describing the timing behaviour is necessary. The transformation of the component-based software structure of the Genode OS Framework together with the RPC semantic used by the ROM components to a timing model is described in detail in [26]. The transformation result explicitly expresses effects such as blocking and priority inheritance, while preserving the event chains. [26] also describes how response-time bounds can be computed over a chain of components. Possible alternatives to compute response-time bounds is e.g. MAST [2].

However, if hardware resources are shared with components from other cause effect chains, possibly even components with a different criticality than the chain under analysis, only verifying response-time bounds is insufficient. In the use-case depicted in Figure 7 this is the case for the shared vehicle network. Following a conservative design strategy, a designer would have to assume that by sharing the resource the components are mutually dependent and that any dependency leads to interference, i.e. failures causing malfunctioning behavior. Consequently, absence or strict bounds on the dependencies have to be proven in order to argue freedom from interference.

A timing model for the Ethernet network can be derived from the knowledge of: (i) how traffic is routed through the network, (ii) which components inject Ethernet frames into the network at which rate, and (iii) what the maximum payload per frame is.

How traffic is routed is known, as this is under control of the MCC which also deploys the network configuration. Similarly, the components which inject frames are already known in the component model. The rate at which they emit a frame i into the network can be abstracted by standard event models, δ_i^+, δ_i^- . These are event model abstractions of concrete execution traces that capture the maximum/minimum time interval between n consecutive activation events. δ_i^+ and δ_i^- for a frame i can be derived from the results of the timing analysis of the component chains on the computation resources with the analysis described in [26]. Only the maximum payload per frame must be extracted from contracting information, which must be fed into the MCC. Based on this information a timing model for the Ethernet network as e.g. described in [31] can be derived. It is formally based on Compositional Performance Analysis (CPA) [22]. In this model each task τ_i represents a frame that is competing for arbitration on a switch port, i.e. the switch ports are the resources. The payload of each frame is captured by bounds on its worst-case execution time (WCET) C_i^+ /best-case execution time (BCET) C_i^- on the wire including all protocol overhead. Chains of dependent tasks on different resources, i.e. Ethernet switch ports, then model a data stream. This model provides the basis to derive the timing-dependency graph (TDG) for the network and the components injecting the traffic as e.g. described by [15].

► **Definition 3.** A *Timing Dependence Graph* is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of nodes $v_i, v_j \in \mathcal{V}$ and edges $e_k \in \mathcal{E}$ where each edge $e_k = (v_i, v_j)$ describes that v_j is dependent on v_i . Each node v_i either describes a task parameter $p \in P = \{C^+, C^-, \delta_{in}^+, \delta_{in}^-\}$ or an (intermediate) timing analysis result $r \in R = \{w^+, w^-, \delta_{out}^+, \delta_{out}^-, R^+, R^-, q_{max}\}$.

To transform the timing model's parameter and results into a TDG, two conversion functions are necessary to populate the edge set of the TDG \mathcal{G} .

► **Definition 4.** The *parameter conversion function* is a function

$$\vartheta_p : \mathcal{T} \times \{C^+, C^-, \delta_{in}^+, \delta_{in}^-\} \mapsto \mathcal{V} \quad (1)$$

that maps each input parameter type $p \in P$ for a task $\tau_i \in \mathcal{T}$ to a node $v = \vartheta_p(\tau_i, p)$ with $v \in \mathcal{V}$ in the TDG, and the result conversion function:

$$\vartheta_r : \mathcal{T} \times \{w^+, w^-, \delta_{out}^+, \delta_{out}^-, R^+, R^-, q_{max}\} \mapsto \mathcal{V} \quad (2)$$

that maps each result type $r \in R$ of a task $\tau_i \in \mathcal{T}$ to a node $v = \vartheta_r(\tau_i, r)$ with $v \in \mathcal{V}$ in the TDG.

This conversion function is analysis-specific, i.e. how CPA's busy-window (w^+/w^-) and output event models ($\delta_{out}^-/\delta_{out}^+$) are computed. In general, a TDG is constructed in four steps: First, for each task in the task graph, the timing dependency graph is populated with the nodes describing its parameters. In the second step, all explicit dependencies between tasks on different resources are added as edges in the graph. This happens for two tasks τ_a and τ_b by inserting two edges $e_k = (v_i, v_j)$ and $e_l = (v_m, v_n)$ into the dependency graph in order to capture the dependency between their output and input event model ($\delta_{a,out}^-/\delta_{a,out}^+$ and $\delta_{b,in}^-/\delta_{b,in}^+$). More precisely, $v_i = \vartheta_r(\tau_a, \delta_{out}^-)$ and $v_j = \vartheta_p(\tau_b, \delta_{in}^-)$ as well as $v_m = \vartheta_r(\tau_a, \delta_{out}^+)$ and $v_n = \vartheta_p(\tau_b, \delta_{in}^+)$. The third step then deals with the dependencies on each resource. It adds dependency edges according to the construction of the busy window (w^+/w^-), and the computation of response times (R^+/R^-). This implies that, for each scheduler, a specific transformation is necessary. Consequently, the third step must be carried out for each resource individually, respecting its scheduling analysis. Dependent tasks on a resource can either be treated as in step two following the generalized CPA theory, or be treated through the local resource analysis step, as e.g. done in [25] who considers task chains under static-priority preemptive (SPP) scheduling. The fourth step then deals with capturing the dependencies that influence the computation of the output event model, based on the resource-analysis results and the applied propagation strategy to bound them. W.l.g. we assume busy-window propagation as described by Theorems 1–3 in [29].

Dependencies are consequently expressed as edges between timing model parameters in the TDG. The TDG allows identifying timing dependencies that data which is transmitted over the network experiences.

Since the functional model \mathcal{FG} has a correspondence rule with the safety viewpoint (cf. Figure 2), we can trace safety requirements from there over \mathcal{FG} to individual task chains and thus to the timing model and the TDG. [16] treats safety requirements on timing requirements as so called *confidence requirements*. The confidence requirement expresses how well all timing parameters to compute a timing bound must be known, in order to utilize the computed bound as proof that the timing requirement and consequently the safety requirement is fulfilled.

The input description of components on the other hand supplies information how accurate timing parameters like WCET/BCET, e.g. payload sizes, are known. By propagating confidence values through the TDG of the system in a flow like manner where the lowest possible confidence is assigned to a TDG node, also every timing requirement in the TDG receives a confidence value. In cases where a mismatch between the assigned confidence and the confidence requirement exists, the MCC either must reject such a configuration or instantiate enforcement mechanisms to guarantee the expected model behaviour at run-time.

4.3 Monitoring and Enforcement

If the timely transmission of this data is safety relevant and dependent on parameters with lower confidence than its requirement, the MCC must take actions to bound these dependencies.

Several authors, e.g. [17, 9, 18] have proposed monitoring and enforcement techniques to conform run-time inputs to model behavior. These techniques can also be used to shape the injected traffic into the network. The MCC can deploy such mechanisms into the execution domain (cf. Figure 6). They render a dependency innocuous since they increase the confidence into a parameter to the confidence of the enforcement mechanism, which is typically high or the highest in the system. This is due to the fact that the monitors are reliable middleware components. In order to prevent overly excessive monitoring and enforcement the MCC coordinates the model enforcement strategy. Two possible strategies for efficient placement of monitors that perform enforcement are described by [16], a greedy input monitor placement and a min-cut strategy. For the MCC the greedy input placement is more suitable as it avoids complex network management where monitors would have to be implemented in the switches of the Ethernet network.

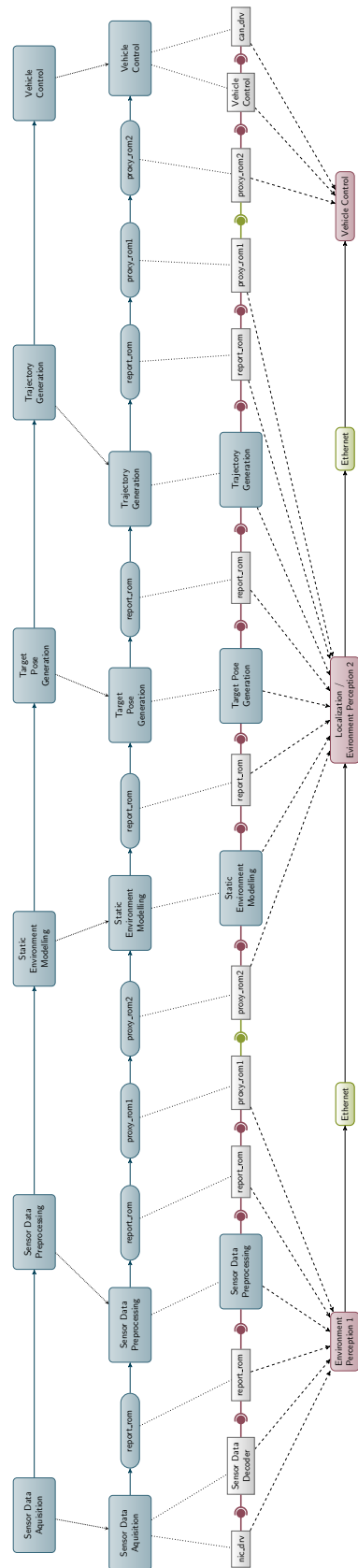
Through this enforcement, the network is guaranteed to operate within the bounds of the timing analysis. A reevaluation of the confidence values after placing enforcing monitors shows that confidence requirements are now fulfilled. Together with the timing analysis this is a sufficient proof of freedom from timing interference ([10, clause 3.75,part1]). In the case study depicted in Figure 7 the MCC performs this for the data that is transferred over the shared Ethernet network, i.e. between the Sensor Data Preprocessing and the Static Environment Modelling components, as well as for the reference trajectory sent to the vehicle control component which interfaces with the legacy control subsystem of the vehicle.

5 Conclusion

In this paper we have presented a design and integration flow that respects safety aspects of SAE level 3+ vehicle functions. We argued that the system emergent property of safety requires traceability in a design. To ensure this traceability during integration, we presented the MCC based integration flow in section 4, where traceability is inherent due to the automated model-based integration flow. This is mainly achievable due to the cross-layer model as an implementation of multiviewpoint modelling and the dependency analysis that is performed based on the cross-layer model. In section 4 we have particularly shown how this is handled for complex timing dependencies. However, the derivation and formulation of functional safety requirements for the MCC are still manual. It is our vision, to further automatize the coupling between behavioral and functional safety (cf. Figure 2), i.e. integrating this aspect in future versions of the MCC, as it is currently a manual process.

References

- 1 Genode OS Framework release notes 16.11, 17.08, 18.02 and 18.08. URL: <https://genode.org/documentation/release-notes/index>.
- 2 MAST: Modeling and Analysis Suite for Real-Time Applications. URL: <http://mast.unican.es/>.
- 3 A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.
- 4 G. Bagschik, M. Nolte, S. Ernst, and M. Maurer. A System’s Perspective Towards an Architecture Framework for Safe Automated Vehicles. In *2018 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2438–2445, 2018.
- 5 Peter Johannes Bergmiller. *Towards functional safety in drive-by-wire vehicles*. Springer, 2015.
- 6 A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. doi:10.1109/2.30720.



■ **Figure 7** Visualization of how the obstacle avoidance function graph from Figure 5 is mapped to a component graph (upper half). The component graph is transformed into the component instance graph which also contains the mapping to the hardware resources (lower half). Red interface symbols indicate Genode interfaces, while green represent a data transfer over Ethernet.

- 7 Norman Feske. Genode OS Framework Foundations 18.05, May 2018. URL: <http://genode.org/documentation/genode-foundations-18-05.pdf>.
- 8 Patrik Feth, Rasmus Adler, Takeshi Fukuda, Tasuku Ishigooka, Satoshi Otsuka, Daniel Schneider, Denis Uecker, and Kentaro Yoshimura. Multi-aspect Safety Engineering for Highly Automated Driving: Looking Beyond Functional Safety and Established Standards and Methodologies. In Barbara Gallina, Amund Skavhaug, and Friedemann Bitsch, editors, *Computer Safety, Reliability, and Security. SAFECOMP 2018. Lecture Notes in Computer Science*, volume 11093, pages 59–72. Springer International Publishing, 2018.
- 9 Kai Huang, Gang Chen, C. Buckl, and A. Knoll. Conforming the runtime inputs for hard real-time embedded systems. In *2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 430–436, June 2012.
- 10 Intern. Organization for Standardization - ISO. *ISO 26262 - Road vehicles - Functional safety*, 2 edition, April 2011.
- 11 Intern. Organization for Standardization - ISO. *ISO/IEC 42010 - Systems and software engineering – Architecture description*, 2011.
- 12 International Standard Office. *ISO/PRF PAS 21448: Road vehicles : Safety of the intended functionality*. ISO, 2018.
- 13 Helmut Martin, Kurt Tschabuschnig, Olof Bridal, and Daniel Watzenig. Functional Safety of Automated Driving Systems: Does ISO 26262 Meet the Challenges? In *Computer Safety, Reliability, and Security. SAFECOMP 2017. Lecture Notes in Computer Science*, pages 387–416. Springer, Cham, 2017.
- 14 Richard Matthaei, Gerrit Bagschik, Jens Rieken, and Markus Maurer. Stationary Urban Environment Modeling Using Multi-Layer-Grids. In *17th International Conference on Information Fusion*, 2014.
- 15 Mischa Möstl and Rolf Ernst. Cross-Layer Dependency Analysis with Timing Dependence Graphs. In *Proceedings of the 55th Design Automation Conference (DAC)*, 2018.
- 16 Mischa Möstl, Johannes Schlatow, and Rolf Ernst. Synthesis of Monitors for Networked Systems With Heterogeneous Safety Requirements. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2824–2834, November 2018. doi:10.1109/TCAD.2018.2862458.
- 17 M. Neukirchner, T. Michaels, P. Axer, S. Quinton, and R. Ernst. Monitoring Arbitrary Activation Patterns in Real-Time Systems. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, pages 293–302, 2012. doi:10.1109/RTSS.2012.80.
- 18 Moritz Neukirchner, Philip Axer, Tobias Michaels, and Rolf Ernst. Monitoring of Workload Arrival Functions for Mixed-Criticality Systems. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 88–96. IEEE, 2013. doi:10.1109/RTSS.2013.17.
- 19 M. Nolte, M. Rose, T. Stolte, and M. Maurer. Model predictive control based trajectory generation for autonomous vehicles — an architectural approach. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 798–805, 2017.
- 20 Marcus Nolte, Gerrit Bagschik, Inga Jatzkowski, Torben Stolte, Andreas Reschka, and Markus Maurer. Towards a Skill- And Ability-Based Development Process for Self-Aware Automated Road Vehicles. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Yokohama, Japan, 2017.
- 21 Andreas Reschka, Gerrit Bagschik, Simon Ulbrich, Marcus Nolte, and Markus Maurer. Ability and Skill Graphs for System Modeling, Online Monitoring, and Decision Support for Vehicle Guidance Systems. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 933–939, Seoul, South Korea, 2015.
- 22 Kai Richter. *Compositional Scheduling Analysis Using Standard Event Models*. Institut für Datentechnik, 2005. URL: <http://www.digibib.tu-bs.de/?docid=00001765>.
- 23 J. Rieken, R. Matthaei, and M. Maurer. Toward Perception-Driven Urban Environment Modeling for Automated Road Vehicles. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 731–738, 2015.

- 24 SAE. J3016: Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems. *Vehicle Electrification Subscription*, 2014.
- 25 Johannes Schlatow and Rolf Ernst. Response-Time Analysis for Task Chains in Communicating Threads. In *Real-Time Embedded Technology & Applications Symposium (RTAS)*, Vienna, Austria, April 2016.
- 26 Johannes Schlatow and Rolf Ernst. Response-Time Analysis for Task Chains with Complex Precedence and Blocking Relations. *International Conference on Embedded Software (EMSOFT)*, *ACM Transactions on Embedded Computing Systems ESWEEK Special Issue*, 16(5s):172:1–172:19, September 2017.
- 27 Johannes Schlatow, Mischa Möstl, and Rolf Ernst. An extensible autonomous reconfiguration framework for complex component-based embedded systems. In *12th International Conference on Autonomic Computing (ICAC)*, pages 239–242, Grenoble, France, July 2015.
- 28 Johannes Schlatow, Marcus Nolte, Mischa Möstl, Inga Jatzkowski, Rolf Ernst, and Markus Maurer. Towards model-based integration of component-based automotive software systems. In *Annual Conference of the IEEE Industrial Electronics Society (IECON17)*, Beijing, China, October 2017. doi:10.24355/dbbs.084-201803221525.
- 29 Simon Schliecker, Jonas Rox, Matthias Ivers, and Rolf Ernst. Providing accurate event models for the analysis of heterogeneous multiprocessor systems. In *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis*, CODES+ISSS '08, pages 185–190, New York, NY, USA, 2008. ACM.
- 30 T. Solte, Tianyu Liao, Matthias Nee, Marcus Nolte, and Markus Maurer. Investigating Cross-Domain Redundancies in the Context of Vehicle Automation - A Trajectory Tracking Perspective. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2398–2405, 2018.
- 31 Daniel Thiele, Philip Axer, and Rolf Ernst. Improving Formal Timing Analysis of Switched Ethernet by Exploiting FIFO Scheduling. In *52nd Annual Design Automation Conference, DAC '15*, pages 41:1–41:6, New York, NY, USA, 2015. ACM.
- 32 Waymo. Waymo Safety Report: On the Road to Fully Self-Driving, 2017.
- 33 J. M. Wille, F. Saust, and M. Maurer. Stadtpilot: Driving autonomously on Braunschweig's inner ring road. In *2010 IEEE Intelligent Vehicles Symposium*, pages 506–511, June 2010.

Safety and Security Analysis of AEB for L4 Autonomous Vehicle Using STPA

Shefali Sharma 

Electrical and Computer Eng., University of Waterloo, Waterloo, Canada
s335shar@uwaterloo.ca

Adan Flores

Electrical and Computer Eng., University of Waterloo, Waterloo, Canada
aaflores@uwaterloo.ca

Chris Hobbs

QNX Software Systems Limited, Kanata, Canada
chobbs@qnx.com

Jeff Stafford

Renesas Electronics America Inc., Farmington Hills, USA
jeff.stafford@renesas.com

Sebastian Fischmeister

Electrical and Computer Eng., University of Waterloo, Waterloo, Canada
sfischme@uwaterloo.ca

Abstract

Autonomous vehicles (AVs) are coming to our streets. Due to the presence of highly complex software systems in AVs, there is a need for a new hazard analysis technique to meet stringent safety standards. System Theoretic Process Analysis (STPA), based on Systems Theoretic Accident Modeling and Processes (STAMP), is a powerful tool that can identify, define, analyze and mitigate hazards from the earliest conceptual stage deployment to the operation of a system. Applying STPA to autonomous vehicles demonstrates STPA's applicability to preliminary hazard analysis, alternative available, developmental tests, organizational design, and functional design of each unique safety operation.

This paper describes the STPA process used to generate system design requirements for an Autonomous Emergency Braking (AEB) system using a top-down analysis approach to system safety. The paper makes the following contributions to practicing STPA for safety and security:

1. It describes the incorporation of safety and security analysis in one process and discusses the benefits of this;
2. It provides an improved, structural approach for scenario analysis, concentrating on safety and security;
3. It demonstrates the utility of STPA for gap analysis of existing designs in the automotive domain;
4. It provides lessons learned throughout the process of applying STPA and STPA-Sec ¹.

2012 ACM Subject Classification Hardware → Safety critical systems; Networks → Cyber-physical networks

Keywords and phrases Functional Safety, Security, STAMP, STPA, STPA-Sec, ISO 26262, AEB, Advanced Driver Assistance Systems (ADAS), Automated Vehicles, SoC (System-On-Chip)

Digital Object Identifier 10.4230/OASICS.ASD.2019.5

¹ STPA-Sec identifies and frames the security problems [10]



1 Introduction

AV functionality is rapidly adopted through ADAS technology today. Combining this rate of adoption with the complexity of the autonomous vehicle's system architecture and its use of complex SoC, it is essential for Tier-1² and semiconductor suppliers to be diligent in their collaborative effort to design for functional safety and for the mitigation of cybersecurity threats impacting functional safety.

STPA is a new hazard analysis technique and a new model of accident causation, based on systems theory rather than reliability theory [4]. STPA has the same goals as any other hazard analysis technique, that is, to recognize scenarios leading to identified hazards so that they can be eliminated or controlled. STPA, however, has an innovative theoretical basis or accident causality model. STPA is designed to address increasingly common component interaction accidents, along with component failure accidents, which can result from design flaws or unsafe interactions among non-failing (operational) components [3]. In fact, the causes identified using STPA are a superset of those identified by other techniques [4].

This paper provides an example of applying STPA to an AEB system primarily designed for functional safety as well as to mitigate risks associated with cybersecurity vulnerabilities. In this, we have combined functional safety analysis with safety-relevant security analysis.

A methodology is defined to analyze functional safety and cybersecurity, first for the AEB system, and then for the interactions, searching specifically for security vulnerabilities that might contribute to safety hazards.

The next step in the analysis is the identification of accidents and unacceptable losses along with accident hazards and unacceptable loss hazards. We define accident hazards and unacceptable loss hazards, keeping in mind that the implementation of the AEB system is on an L4 AV. Because of the level of autonomy of the vehicle, it is safe to assume there is no driver interaction for the control of the vehicle or the AEB system. In this analysis, the system hazards lead to high-level system constraints and further refinement in STPA Steps 1 and 2.

As we move forward in the analysis, while applying the STPA process, additional dependencies are going to be identified. Knowing this, we can define a basic initial high-level control structure³ which will be updated in later steps of the analysis. The final control diagram⁴ captures the dependencies from both a safety and cybersecurity perspective.

From the high-level control diagram, the next step is to identify CAs (Control Actions). Evaluation of potential hazardous sources is shown in the refined control diagram, considering all of the diagram's inputs and outputs. We also considered component failure, but the analysis is not limited to this. Instead, it presents all aspects of the system's performance, including cybersecurity features negatively impacting functional safety. From this analysis, we are defining a set of causal factors and causal accident scenarios.

The novelty of this paper lies in the addition of a more systematic approach to the conventional STPA approach. Identifying the scenarios by analyzing the components associated with the control flow, and the causal factors corresponding to each scenario, constitutes the next step. From the causal factors, we are refining the safety constraints so that they can produce technical safety requirements (TSRs). Comparison of the TSRs against an existing

² Companies which supply components directly to the original equipment manufacturer (OEM), that set up the chain.

³ See Figure 3.

⁴ See Figure 4.

autonomous vehicle design (the autonomy vehicles designed as ASIL-D L4 fail-operational systems) is carried out to identify design gaps for future improvement. This gap analysis on an existing system demonstrates how to make safety and security design changes part of a continuous improvement that must be at the heart of every safety culture.

2 How the analysis started

We started by reviewing an existing autonomous vehicle in need of formal safety analysis. The initial plan was to use a conventional Hazards Analysis and Risk Assessment (HARA) analysis because the group already had experience using this method. But then we learned about STPA and decided to assess its suitability for a system of this scale. We had read reports of its application to much larger systems [3] and wanted to determine whether it would scale to a single, embedded system. Using this approach, we can generate high-level safety constraints in the early stages of development. These constraints can then be tailored to generate detailed safety requirements on individual components of the analyzed system[8].

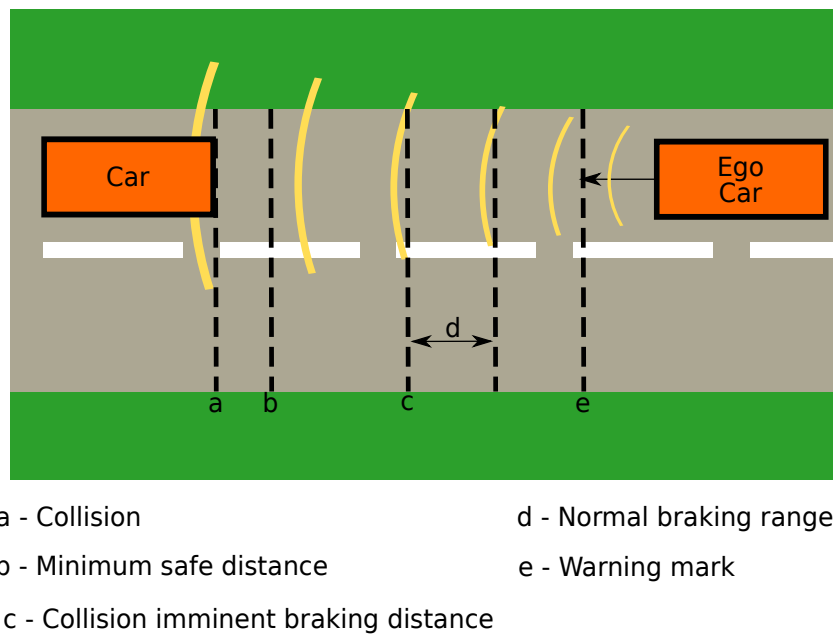
To avoid biasing our results, we established that the safety analysis should be as general as possible without being directly involved with the current implementation. Thus the result of this analysis was a list of technical safety requirements which we could use to perform an analysis on the current physical architecture and find possible security and safety issues.

We needed to select vehicle functionalities that played an important role in vehicle and occupant safety. The vehicle component also had to be a part of a well-contained function to complete the analysis in the time span available. For these reasons, we selected the L4 AEB function for our analysis.

2.1 The AEB subsystem

An AEB system of L4 AV aid in avoiding accidents by identifying potential collisions with the help of a perception system (LIDAR, RADAR, stereo vision, etc.), computing localization, path planning and determining object trajectory. If a collision is unavoidable, these systems prepare the vehicle to minimize the impact by lowering its speed. It is important to note that the AEB itself is independent of the normal braking system of the vehicle. Once the AEB has identified a potential threat, it takes control of the braking system to mitigate the threat. This functionality has a significant effect on the safety of the vehicle and its occupants, making it an excellent vehicle subsystem for our analysis.

When looking at the distances between the vehicles as shown in Figure 1, we can establish safety thresholds. The first threshold is the warning distance that notifies the AV when the proximity between ego vehicle and the vehicle in front is becoming dangerous; it is recommended for the ego vehicle to start slowing down and increasing the distance between the vehicles. At this distance, the probability of a collision is low. The next threshold is the normal braking limit. At this distance, the normal braking system of the vehicle starts slowing down the vehicle. If the braking system is unable to slow down the vehicle and increase distance, the vehicle will reach the Collision Imminent Braking distance (CIBd) and will activate the AEB system. At this point the collision probability is high, and the AEB needs to take immediate action. The AEB's objective is to stop or slow down the vehicle before it reaches the Minimum Safe Distance (MSD). The MSD threshold is the only fixed value amongst all the thresholds. The rest of the values are dependent on the road conditions (weather and road surface) and the speed of the vehicle.



■ **Figure 1** Threshold distances for the braking system.

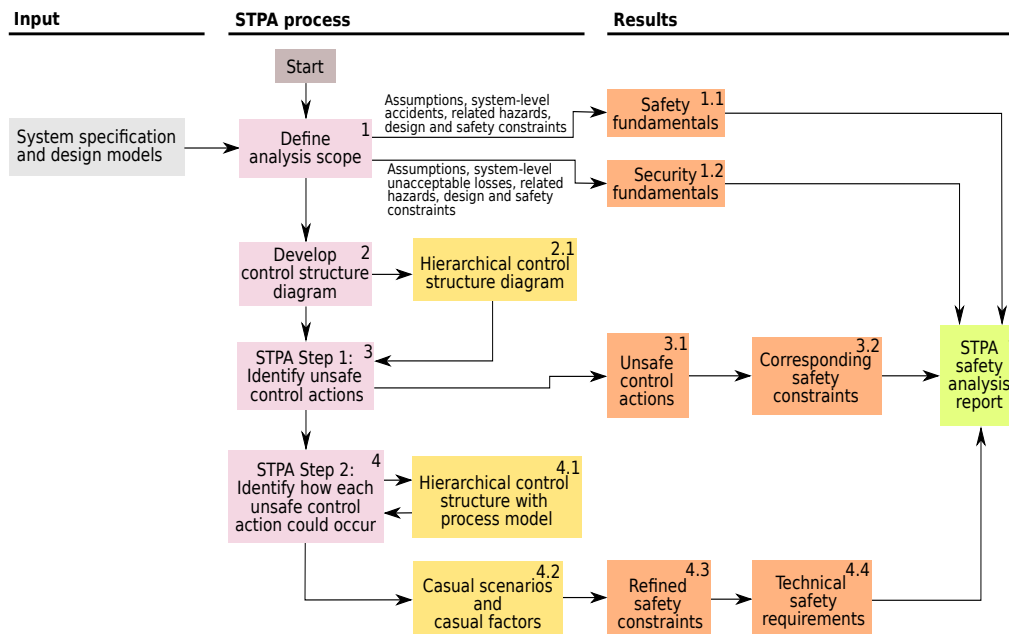
3 Methodology

The methodology used in the current approach combines safety and security analysis. This approach considers the functional safety and the security-affecting safety. Figure 2 presents the methodology we are using for the STPA analysis[2]:

1. Define analysis scope
 - a. Accidents
 - b. Hazards
 - c. High-level constraints
2. Develop control structure diagram
3. Identify unsafe control actions
 - a. Unsafe control actions
 - b. Corresponding safety constraints
4. Identify the occurrence of unsafe control actions
 - a. Hierarchical control structure with the process model
 - b. Causal factors, scenarios, and refined safety constraints
 - c. Technical safety constraints

The elements 1(c), 3(b), and 4(c) constitute the STPA analysis report which defines the safety constraints for a safer and more secure system.

The analysis considers a detailed analysis of various blocks of Figure 2. The constituents of the multiple blocks are referred with an identifier as the various parts of each block, to serve as a starting ground for the next block.



■ **Figure 2** STPA methodology.

3.1 Scope

The methodology begins by defining the scope of the analysis. For the system under consideration, the scope is as follows: “*The analysis presents functional safety analysis for AEB for an AV using vehicle state and environmental data analysis to contribute to the safety of the passengers and environment.*”

3.1.1 Assumptions

After defining the scope, the next step is to define certain conditions that serve as the basis for analysis development. Thus, the analysis considers certain assumptions related to the working conditions. These conditions are also helpful in setting the limits to the analysis. Although, the authors recognize that it would be beneficial to further analyze the assumptions from the perspective of an expanded scope. Here are a couple of examples:⁵:

Assumption 1: AEB functions for collisions from all angles, not just traditional forward-collisions (no lateral maneuvering or acceleration commanded, considering only the brake actuation).

Assumption 5: Path prediction of surrounding mobile objects is available to the AEB system.

There are certain logical conditions behind including these assumptions in the analysis. General cases assume collision primarily from the front. This analysis, however, also examines projected paths of side objects relative to the AV projected path. Hence, the Assumption 1.

⁵ These are some of the assumptions we are referring here from the analysis. To have consistency with the report [6], we are using the same identifiers.

The analysis considers an assumption about the availability of data from the surroundings, such as for calculating the collision imminent braking distance and path prediction from the surrounding mobile objects. Hence, Assumption 5.

Some of the assumptions also consider certain conditions outside the scope of the analysis. For example:

- The variation in braking performance based on the mechanical condition of AV tires,
- The sensor performance can be negatively impacted by maintenance or improper care,
- No manufacturing defects and
- All the components are correctly working as they are quality checked and properly maintained.

3.1.2 Accidents

An accident is an undesired or unplanned event that results in the loss of a human life, human injury, property damage, etc. The accidents considered in the analysis are:

- A1:** The AV collides with a mobile object.
- A2:** The AV collides with an immobile object.
- A3:** The AV passengers injured without collision.

In defining the accidents, we first discussed various scenarios that the AV can encounter on the road. Next, we grouped the elements of the scenarios into different categories: vehicles, pedestrians, cyclists, stationary objects, etc. As the analysis was evolving, these subsets posed certain problems; for example, a dustbin could start off as a stationary object, but due to the wind, could start rolling on the road and become non-stationary. We decided that instead of defining it by its current state of activity, we can describe it with its innate ability. So after refinement we devised two subsets: mobile and immobile. For example: if a mailbox were on an HD map, it would be an immobile object. If that same mailbox were blown from its bolts by high wind and became non-stationary, it would be a mobile object requiring identification of the AD sensor system because it is no longer in its original position as shown in the HD map. Here, “mobile” is anything that can move, irrespective of the external influence. Thus A1 and A2 are considered as two potential accidents for the analysis. Also, as in the definition of accident, anything that causes harm to human occupants needs to be considered and is stated as A3. While sitting inside the AV, under certain circumstances such as sudden braking (braking deceleration exceeds the safety physics to passengers) can harm the occupants even when there is no collision.

The next step in the analysis was to define system-level hazards. These are the system states or set of conditions, which together with a particular set of worst-case environmental conditions, would probably lead to an accident.

3.1.3 System level hazards

System level hazards can lead to accidents considered in the analysis. Some of the hazards are listed below:

- AH1:** AV does not maintain Minimum Safe Distance (MSD) from a Forward Mobile Object (FMO).
- AH2:** AV does not maintain MSD from Prohibited Area (PA).
- AH3:** AV occupants exposed to unhealthy g-forces in vehicle exceeding the safety threshold of AV.

Maintaining a safe distance from a vehicle in front is a necessary condition for AEB. If the vehicle is unable to keep MSD from a forward mobile object, then this could be the potential cause of an accident and thus become a hazard that could lead to an accident. The condition for the MSD from an FMO is a prerequisite for the safety of the AV. There are certain areas which have restricted access to traffic. The AVs should ensure that they do not enter such areas and this has been considered – in the analysis as AH2. PA can mean any area – military field, recent accident site, landslide site, etc., – AV’s design is not suitable for L4 functionality in a PA. The thresholds predefined in the system related to BFC (Braking Force Command) shall always be complied with because they have the potential to harm the occupants if they exceed a certain threshold level and thus constitute a hazard for the analysis (AH3).

After the identification of hazards, the next step was to describe high-level constraints. These prevent the accident from occurring. Thus, HLCs (High-Level Constraints) provide the set of requirements with which the system shall comply to be functionally safe. These are defined consistently to have traceability to the corresponding hazards. Using a consistent structure can be helpful for the automation of the process. Although this analysis doesn’t automate the process, consistency in the structure helped in having a symmetric structure.

During this analysis, we were struggling with the question of whether we should generate two different reports relating to safety and security or whether they should be merged into one. We realized that safety and security are closely interlinked and therefore merged them into one single analysis.

For example: If the AV speed sensor information is spoofed (security threat) then it can lead to a hazardous scenario ultimately leading to an accident (safety threat).

If, due to delayed EPS sensor information (safety threat), BFC fails to set the braking force = 0% even after the removal of earlier hazard, this situation could lead to an unnecessary halt, and thus personal identifiable information of occupants could be inferred (security threat).

3.1.4 High-level safety constraints

High-level safety constraints define the initial set of safety requirements for the system.

3.2 STPA Step 1

The identification of unsafe control actions and the corresponding safety constraints are discussed in this section.

3.2.1 Safety control structure

The control structure is a preliminary process model for the system. It is a functional decomposition of the system. While working on the control structure, we faced certain challenges such as level of detail to be considered. For the sake of a systematic and structured approach, a control structure is the most crucial thing for the safety analysis. We should only consider the blocks responsible for significant functionality such as controller, actuator, process, and feedback. The structure is only a generic one and does not consider the level of granularity. It gives us an overview of how the execution of instruction is taking place without considering the complete internal functionality of the various components involved. Here follows the description of various blocks within the analysis:

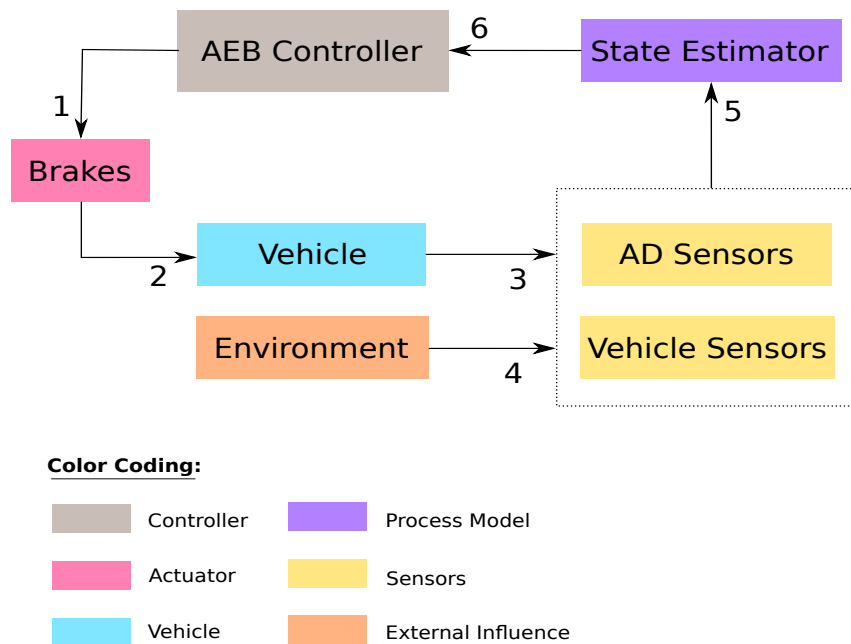
Controller: In the system under analysis, the AEB controller is responsible for generating and controlling the BFC.

Actuator: In this system, brakes are the actuator responsible for implementing the BFCs.

Controlled process: The AEB controls the braking of the vehicle.

Feedback: The feedback from the vehicle state and the surrounding environment through the sensors is collected in the state estimator, and thus constitutes the feedback network.

The control structure for the system under consideration is as shown in the Figure 3.



■ **Figure 3** Control loop structure.

By following the STPA process diligently, through detailed use of refined control diagrams, we have a reference to verify that the hazards identification is adequate, and through continued refinement, a benchmark for the design to support continuous improvement over the life of the item. During the analysis, we struggled with the level of detail to be present in the control loop diagram. After creating several revisions of the control loop, we concluded that it should be generic in form and that a further level of detail would not add value to the analysis. For the Control Loop, it shall be in basic generic form and the later stages shall consider the details.

3.2.2 Unsafe control actions

This step performs the identification of the unsafe control actions each component can create which helps in refining the safety requirements and constraints of the system. It will determine the causes of these unsafe control actions. The UCAs are defined using the control actions that can lead to accidents. So, this analysis is considering two control actions based on the control diagram. Here we have taken the BFC (Braking Force Command) coming from the controller; it is only the command and not the force. Two states considered in the analysis are: BFC disengaged (0%), and BFC engaged (modulated engagement ranging from 0% – 100%). After the identification of control actions, the next step is to identify the potential causes of unsafe control.

The four ways a controller can provide unsafe control are the following [3]:

1. A control action required for safety is not provided.
2. An unsafe control action is provided.
3. A potentially safe control action is provided too late or too early (at the wrong time) or in the wrong sequence.
4. A control action required for safety is stopped too soon or applied for too long.

We considered these four categories as a basis for identification of the control table entries. Some of the unsafe control actions considered are listed here:

UCA 1: AEB does not provide BFC when AV is at a closer distance than the CIBd.

UCA 3: AEB does not provide required braking force value when AV is at a closer distance than the CIBd.

If BFC is not applied even when the AV is within the CIBd from an object, then this can be a potential unsafe control action, which could lead to an accident. Hence, UCA 1 belongs to the category of “control action required, but not provided.” Another UCA is when the BFC is applied, but the braking force $<$ RDR (Required Deceleration Rate) can also lead to an accident, and is therefore an unsafe control action. Similarly, other UCAs are considered, based upon the time of application of BFC and the total time span of BFC application. Thus, the UCA table is formed.

3.2.3 Safety constraints

The UCAs help to find reasons behind unsafe actions and guide design engineers to eliminate or control them. We referred to table 1 for UCAs, and SCs sets the requirements for the systems. The refined safety constraints are defined in a consistent language as follows:

SC 1: AEB shall provide BFC when AV is at a closer distance than the CIBd.

SC 3: AEB shall provide required braking force value when AV is at a closer distance than the CIBd.

3.3 STPA Step 2

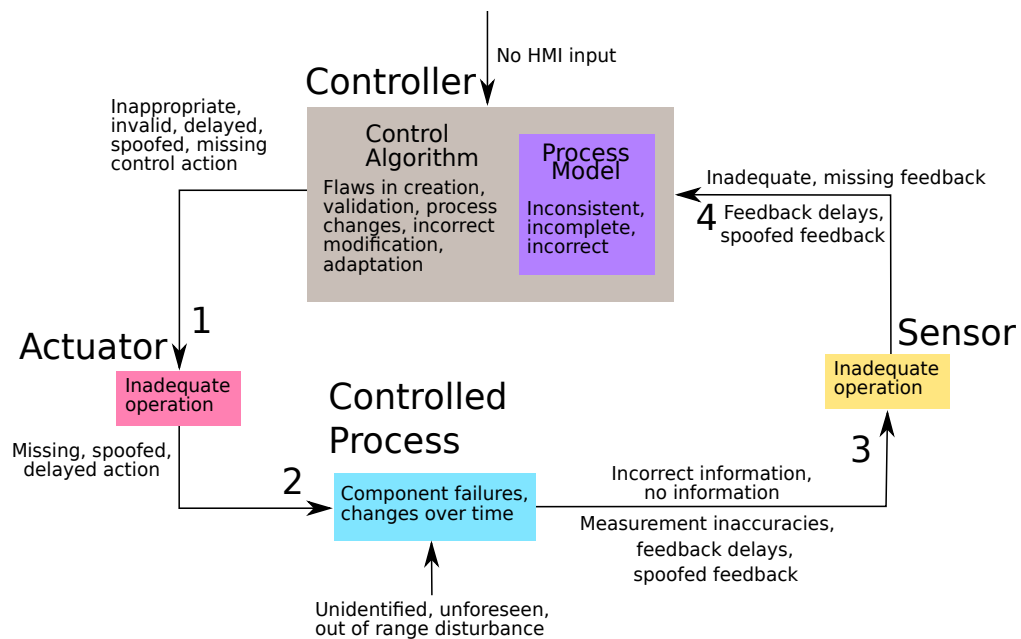
This section identifies the reasons behind the unsafe control actions.

3.3.1 Causal factors and causal accident scenarios

After the identification of the unsafe control actions, we followed STPA Step 2 (Figure 2) to identify the potential causes of unsafe control actions, to understand their presence and how to prevent their occurrence [9]. However, accidents can still occur even without unsafe control actions if, for example, correct and safe control actions are provided, but not executed by other components in the system. The identification of the causal factors can identify a violation of safety constraints despite safe control actions; this is important.

To study the scenarios and causal factors corresponding to each UCA, we made a structured approach:

1. Identify scenarios using UCAs.
2. Identify causal factors corresponding to each scenario by analysing the components associated with the control flow diagram.



■ **Figure 4** Control loop structure.

■ **Table 1** Table for identification of causal factors.

Blocks	Actions	Reasons
Sensors, Controller, Actuator, Controlled process	Missing information	Due to spoofing, component failure, electrical requirements not met, communication failure
	Inadequate information	
	Incorrect information	
	Delayed information	

We studied the STPA problem as a whole and took parts from the methods available from various researchers in the field [4], [9]. Then we created a hybrid to perform the required analysis.

For example: For the scenario table we tried to use the conventional STPA approach, but found that for our analysis the basic scenarios are sufficient and other detailed scenarios (scenarios arising from feedback issues, etc.) merely lead to redundant scenarios. We created twelve scenarios, but when we started defining the causal factors after three scenarios, they began to repeat and became redundant for our purposes. So we removed the detailed scenarios and analyzed only basic, generic scenarios.

Table 1 provides a systematic and structured approach to analyzing the causal factors. For each of the four blocks in the control structure, we considered four actions: the information is missing, inadequate, incorrect and delayed. The reasons behind these unsafe actions could be spoofing, component failure, electrical requirements not met or communication failure. Considering these actions and the reasons behind them, the causal factors can be identified.

3.3.2 Rationale table

The analysis uses a supporting table for the causal factor entries. It verifies the table entries and explains the thought process behind the causal factors. It can serve as a reference table for refined safety constraints and technical safety requirement tables.

Rationale for CF1.1a (Causal Factors). If the OPP (Object Predicted Path) is calculated incorrectly, there is the potential for the actual object path to be closer to the AV path than calculated. In this case, the controller will not send the BFC command, even though the autonomous vehicle's predicted path has reached the minimum safe distance from the object's predicted path. An image processing performance fault could prevent the correct calculation required for the identification of an object that is within the MSD of AV.

It was recognized and accepted that some rationale repeated itself. When this occurred, we reviewed the causal factor table for correctness and appropriateness, and if it still provided a distinctly different CF (Causal Factor), then the repeated rationale conditions were accepted. The repeated nature is suitable for automation and desirable, as long as it is applied to each unique and new CF.

3.3.3 Refined safety constraints and Technical safety requirements

After the identification of reasons behind the UCAs, the constraints on the system were redefined to eliminate or avoid the causes behind the UCAs. These new safety constraints created from the causal factors contained the rationale tables.

Technical safety requirements: This step is responsible for the implementation of refined safety constraints on the system. These represent the technical requirements for a safe system. We used these TSRs to make the gap analysis for the already existing architecture and modified the design of the system.

4 Results: Lessons learned

This section discusses the contribution of this work to making an adhoc STPA more systematic. During the analysis we learned lessons, which will be useful in structuring future analysis systematically. The lessons are summarised below:

► **Lesson Learned 1.** We realized that certain factors could act as a basis for the analysis development which could have an impact on the definition of the safety fundamentals. The first priority is to define boundaries which are defined as the assumptions for the analysis.

► **Lesson Learned 2.** We realized that the identification of accidents and hazards lacks a systematic approach. SOTIF (Safety Of The Intended Function) details from current PAS (Public Available Specification) can be useful for better structuring. We tried to make the identification of accidents and hazards systematic by considering the various scenarios in a symmetric way. The purpose of such a systematic approach is to get rid of the current brainstorming process and in its place, to establish a concrete, automatic method of scenario identification for the analysis.

► **Lesson Learned 3.** From our analysis we realized that the control diagram must represent the basic blocks with generic functionalities and terms. The control diagram is essential and must represent a complete overview of the function under consideration. During the analysis, the control loop serves a reference block and the representation of the control structure keeps the analysis streamlined.

► **Lesson Learned 4.** We have created one single report considering safety and security hazards that threaten safety. Because the safety and security issues are often interlinked, one such report, addressing both problems, is an efficient way to analyze them.

► **Lesson Learned 5.** The novelty of our current work is the systematic analysis of causal factors. The approach presented in Table 1 avoids unnecessary mental exercise. Here we predefined certain actions and the possible reasons for those actions. By correlating actions and reasons, using permutation and combination, the causal factors are devised. Since one of our motives is to automate this process using this constructive approach, we can automate the causal factor generation as well.

► **Lesson Learned 6.** Making a rationale table for each causal factor table is undoubtedly useful as it lists the logic behind the causal factors and serves as a reference for further steps. The cause-effect relationship of the unsafe actions is exploited in the rationale table. The use of rationale tables helps to identify flaws in the original causal factors and thus works as a checkpoint for those factors.

► **Lesson Learned 7.** While using this analysis for finding the gaps in the existing architecture we realized that any architecture could make use of it. We performed the analysis independently of the current design and later compared the technical safety requirements with the existing design. By using a generic rather than the specific approach we found that more extensive applications are possible. The analysis can be used for evaluating any existing AEB system. The gaps provided us with the list of changes that the current architecture might incorporate in order to be safer and more secure.

► **Lesson Learned 8.** Another important lesson learned is about the residual risk inherent in any system. Residual risk refers to some risks which are present but acceptable, in our system. The assumptions made in the analysis are part of the residual risk. The integration of the outcome of this analysis with ISO standard is also an area where we should consider the presence of residual risk which is an integral part of the safety analysis and should be taken into account while doing the analysis.

5 Related work

STPA proved to be a more powerful and useful technique for evaluating safety-critical systems in the automotive domain by identifying the potential accident scenarios that include the entire accident process, including design errors, software flaws, component interaction accidents and human decision-making errors contributing to accidents [1].

Both ISO 26262 and STPA are based on a systems engineering framework in which a system is considered to be more than merely the sum of its parts [5]. The development and top-down analysis are common to both. While ISO 26262 [7] emphasizes the importance of considering the context of a system in achieving safety (including the role of safety management and safety culture), there seems to be no consensus on whether ISO 26262 considers the context to be a part of the hazard analysis of an item. On the other hand, STPA includes all relevant aspects of the system's environment, including the driver.

6 Conclusions

In the safety analysis, the STPA process has been used to generate system design requirements for an Automatic Emergency Brake (AEB) using a top-down analysis approach to system safety. The STPA analysis provides an improved structured approach for scenario analysis,

concentrating on safety and security. We learned lessons while applying the STPA process. The STPA has benefits, but needs to be integrated with the ISO to produce more efficient results. Doing Functional safety analysis and cyber security analysis in parallel is efficient and effective, but tool support is required. STPA is a structured and systematic approach that reduces mental exercise.

7 Future work

The next step can be a comparative study, comparing the analysis with standard ISO. Further, the analysis can potentially be expanded beyond the AEB module to cover the complete functionality of AVs.

References

- 1 Asim Abdulkhaleq and Stefan Wagner. Experiences with applying STPA to software-intensive systems in the automotive domain. *Stuttgart*, 2013.
- 2 Asim Abdulkhaleq, Stefan Wagner, Daniel Lammering, Hagen Boehmert, and Pierre Blueher. Using STPA in Compliance with ISO 26262 for Developing a Safe Architecture for Fully Automated Vehicles. *arXiv preprint*, 2017. [arXiv:1703.03657](#).
- 3 N Leveson. An STPA Primer, Version 1. *Massachusetts Institute of Technology*, pages 22–65, 2013.
- 4 Nancy Leveson. *Engineering a safer world: Systems thinking applied to safety*. MIT press, 2011.
- 5 Archana Mallya, Vera Pantelic, Morayo Adedjouma, Mark Lawford, and Alan Wassyng. Using STPA in an ISO 26262 Compliant Process. In *International Conference on Computer Safety, Reliability, and Security*, pages 117–129. Springer, 2016.
- 6 Shefali Sharma Adan Flores Chris Hobbs Jeff Stafford and Sebastian Fischmeister. Functional Safety and Cybersecurity Assessment of L4 Autonomous Emergency Braking System. *University of Waterloo*, 2018.
- 7 Standard. ISO 26262 Road vehicles—Functional Safety. *ISO*, 2011.
- 8 John Thomas. Systems Theoretic Process Analysis (STPA) Tutorial, 2013.
- 9 John P Thomas IV. *Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis*. PhD thesis, Massachusetts Institute of Technology, 2013.
- 10 W Young. STPA-SEC for cyber security mission assurance. *Eng Syst. Div. Syst. Eng. Res. Lab*, 2014.

Towards a Formal Model of Recursive Self-Reflection

Axel Jantsch

TU Wien, Vienna, Austria

axel.jantsch@tuwien.ac.at

Abstract

Self-awareness holds the promise of better decision making based on a comprehensive assessment of a system's own situation. Therefore it has been studied for more than ten years in a range of settings and applications. However, in the literature the term has been used in a variety of meanings and today there is no consensus on what features and properties it should include. In fact, researchers disagree on the relative benefits of a self-aware system compared to one that is very similar but lacks self-awareness.

We sketch a formal model, and thus a formal definition, of self-awareness. The model is based on dynamic dataflow semantics and includes self-assessment, a simulation and an abstraction as facilitating techniques, which are modeled by spawning new dataflow actors in the system. Most importantly, it has a method to focus on any of its parts to make it a subject of analysis by applying abstraction, self-assessment and simulation. In particular, it can apply this process to itself, which we call recursive self-reflection. There is no arbitrary limit to this self-scrutiny except resource constraints.

2012 ACM Subject Classification Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Self-organizing autonomic computing

Keywords and phrases Cyber-physical systems, self-aware systems, self-reflection, self-assessment

Digital Object Identifier 10.4230/OASICS.ASD.2019.6

1 Introduction

When the autonomous system itself and its environment are exceedingly complex, dynamic and unpredictable, a comprehensive and correct assessment of the system's situation is a prerequisite for good decisions. This insight has led to a proliferation of research that approach the challenges from various angles and run under names like autonomic computing [29, 32] and organic computing [23]. Self-awareness has become associated with many self-* properties including self-monitoring and self-adaptation and it has been identified as key element for designing complex computer systems [1] and cyber-physical systems [3]. The challenge has been picked up by funding organizations such as DARPA [25] and the European Commission [4] who have allocated significant funds for this research. These efforts have resulted in many conference papers, journal articles and four books [11, 18, 26, 32]. Several surveys have systematically reviewed the research landscape [9, 16, 19, 27].

While the term self-awareness is used in the literature in different ways and various definitions have been provided, researchers at a 2015 Dagstuhl Seminar have proposed a comprehensive working definition, as summarized by Kounev et al. [13], which is worth quoting in full:

Self-awareness, in this context, is defined by the combination of three properties that IT systems and services should possess:

1. *Self-reflective*: i) aware of their software architecture, execution environment and the hardware infrastructure on which they are running, ii) aware of their operational goals in terms of QoS requirements, service-level agreements (SLAs) and cost-



© Axel Jantsch;

licensed under Creative Commons License CC-BY

Workshop on Autonomous Systems Design (ASD 2019).

Editors: Selma Saidi, Rolf Ernst, and Dirk Ziegenbein; Article No. 6; pp. 6:1–6:15

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and energy-efficiency targets, iii) aware of dynamic changes in the above during operation,

2. *Self-predictive*: able to predict the effect of dynamic changes (e.g., changing service workloads or QoS requirements) as well as predict the effect of possible adaptation actions (e.g., changing service deployment and/or resource allocations),
3. *Self-adaptive*: proactively adapting as the environment evolves in order to ensure that their QoS requirements and respective SLAs are continuously satisfied while at the same time operating costs and energy-efficiency are optimized.

Two reference architectures have been developed where these principles are at least partially implemented, the EPiCS architecture [17,18] and the Learn-Reason-Act loop [12].

Although these and similar definitions are useful, they are still vague and imprecise. For instance the definition above repeatedly uses the term “aware” in defining self-awareness and thus does not explain what is meant by *awareness*. What would be the difference between being “aware of operational goals in terms of QoS requirements” and storing a list of QoS requirements and using them during operation? Does “aware of dynamic changes” mean that some variables and models are updated and then the system continues to use the new values, or does it mean that the system realizes that a change has happened and ponders its cause and its implications?

Not least because much of the research on self-awareness is inspired by psychology (e.g. see [16]) the term “self-awareness” seems to suggest more than a set of variables and models that represent some features of the system, that it can access during operation. In particular, the definition above, and all other definitions presented in the computing literature, leave it open if the self-models are self-created based on self-observations or if the self-models are provided by the designer. If the latter is the case, would the self-model keep track if the reality changes? Also, should the system be aware of its self-awareness? And should this awareness be recursive without bounds? Should the self-awareness be self-adaptive as the environment, the system, and the self-model changes, as tasks become more or less urgent, as resources become available or are withdrawn?

Different answers to these questions can lead to technically useful solutions and there seems to be a spectrum between the point where everything is defined at design time and the point where everything is self-constructed at run-time. Self-models, self-adaptation and self-awareness push towards run-time, but how far should we go and how do we determine the trade-offs?

Addressing these questions will require to be precise with terminology and to define and model the involved concepts explicitly and in stringent formal terms. The following is an attempt of a formal model of self-awareness but it should not be taken as the final solution but rather as a first step. At several points we are less precise and less complete than we would like to be, partially due to limited space but mostly because of an incomplete understanding of what would be the best choices that lead to a sound basis for modeling, design, exploration and verification. The hope is that a precise formalism will eventually facilitate a design methodology and effective exploration of the design choices in the space of self-aware systems.

2 Notation

We use dynamic dataflow based on static dataflow process network models such as [7,8,14]. But we generalize these models to allow for dynamic changes in the network structure which results in dynamic dataflow not unlike the dynamic model proposed by Grosu and Stølen [5,6].

The processes in the network are called actors and they communicate with each other through signals.

2.1 Signals

Actors communicate with each other by writing to and reading from signals. Signals may be produced by sensors or may be the control inputs for actuators. The environment of an actor can also be modeled as an actor; hence, the actors communicate with each other and with the environment by means of signals.

Given is a set of values V , which represents the data communicated over the signals. *Events*, which are the basic elements of signals, are or contain values. Signals are sequences of events. Sequences are ordered and we use subscripts as in e_i to denote the i^{th} event in a signal. E.g. a signal may be written as $\langle e_0, e_1, e_2 \rangle$. In general signals can be finite or infinite sequences of events and S is the set of all signals.

We assume an untimed model of computation [8, 15] and signals encode only a partially ordered time, meaning that events within one signal represent a relative ordering in time but events in different signals are not directly related in time. I.e. an event e appearing before another event e' in the same signal occurs before e' ; but we do not know which of two events in different signals occur earlier or later.

We use angle brackets, “ \langle ” and “ \rangle ”, to denote ordered sets or sequences of events, but also for sequences of signals if we impose an order on a set of signals. $\#s$ gives the length of signal s . Infinite signals have infinite length and $\#\langle \rangle = 0$.

We use the notation $\text{Signal}(V)$ to denote a type of signal that consists of elements of the set V . E.g. $\text{Signal}(\mathbb{R})$ would denote signals with real numbers, $\text{Signal}(\mathbb{N})$ would denote signals with natural numbers and $\text{Signal}(\{T, F\})$ would denote signals that contain the two types of elements T and F .

Signals are point-to-point connections between actors, and there can only be one producer and one consumer for each signal. If events of a signal should be used by more than one actor, we need a copy actor that copies the input signal to two or more output signals. If two or more actors should contribute to one signal, we need a merge actor that defines how the events from the producing actors are merged. In the figures of this article we sometimes omit the copy and merge actors for convenience and clarity, but the model always requires them.

2.2 Signal Partitioning

We use the partitioning of signals into sub-sequences to define the portions of a signal that are consumed or emitted by an actor in each activation cycle.

A *partition* $\pi(\nu, s)$ of a signal s defines an ordered set of signals, $\langle r_i \rangle$, which, when concatenated together, form the original signal s . The function $\nu : \mathfrak{S} \rightarrow \mathbb{N}$ defines the lengths of all elements in the partition, where \mathfrak{S} is the set of states of the partitioning process. For example, if we have a signal $s = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle$, the partitioning process runs through the sequence of states $\langle q_0, q_1, q_2, \dots \rangle$, and $\nu(q_0) = \nu(q_1) = 3, \nu(q_2) = 4$, then we get the partition $\pi(\nu, s) = \langle \langle 1, 2, 3 \rangle, \langle 4, 5, 6 \rangle, \langle 7, 8, 9, 10 \rangle \rangle$.

Note, that there is nothing static about this partitioning, because the size of the next partition can be determined by the actor during each activation. Signal partitioning only captures the notion of activation cycles of actors which repeatedly consume part of the input and produce more and more of the output.

2.3 Actors

An Actor $A \in \mathcal{A}$ maps a set of input signals to a set of output signals. Actors repeatedly evolve through activation cycles, and in each cycle part of the input signals are consumed and part of the output signals are generated. Also, an actor may have an internal state which is also drawn from the set of all subsets of values V .

\mathcal{A} denotes the set of actors, and $\mathfrak{S} = \mathcal{P}(V)$, the power set of the set V , denotes the set of states, with $\epsilon \in \mathfrak{S}$ being the empty set, i.e. the state that has no values. To capture the notion of activation cycle and to model the behavior of actors, we introduce the state, the next state function g , the output encoding function f , and the partitioning function ν of an actor. Thus an actor with n input and m output signals is an eight tuple $A = \langle \mathfrak{T}, I, O, z_0, f, g, \nu, \vec{m} \rangle$ as follows:

$\mathfrak{T} \subseteq \mathfrak{S}$... set of states
$I \subseteq \mathcal{P}(S)$... set of input signals
$O \subseteq \mathcal{P}(S)$... set of output signals
$z_0 \in \mathfrak{T}$... the initial state
$\nu : \mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$... input partitioning function
$f : \mathcal{P}(S) \times \mathfrak{S} \rightarrow \mathcal{P}(S)$... output encoding function
$g : \mathcal{P}(S) \times \mathfrak{S} \rightarrow \mathfrak{S}$... next state function
$\vec{m} : \mathfrak{S} \rightarrow \text{Action}$... a meta operator

Note, that the sets of input and output signals can dynamically change during the operation, and, consequently, the functions ν, f and g may have to deal with different numbers of signals at different time. Events from an input signal not consumed by an actor during an activation cycle are left in the signal for later consumption, and if no events are generated for a particular output signal, the signal is unchanged. Thus, an actor is free to ignore input and output signals in which case they are never modified.

The meta operator \vec{m} can invoke any of the following actions, which modify the global dataflow network:

Action	
addsig (s)	add signal s
connectsig (s, A)	add signal s to the set of input signals in actor A .
connectosig (s, A)	add signal s to the set of output signals in actor A .
delsig (s)	delete input signal s and remove it from the input and output signals of the connected actors.
addactor ($\mathfrak{T}', I', O', z'_0, f', g', \nu', \vec{m}'$)	create a new actor $A' = \langle \mathfrak{T}', I', O', z'_0, f', g', \nu', \vec{m}' \rangle$
delactor (A')	delete actor A'
nop	do nothing

Since the **addactor** action assumes all involved signals exist, unconnected signals have to be created first with **addsig** and used in **addactor** calls or attached to existing actors with **connectsig** and **connectosig** actions.

An actor A can be applied to a set of input signals to generate events on output signals. It does so by repeatedly consuming values from the input signals and producing values for the output signals. Each such activity is called *activation* or *activation cycle*. The number of input values consumed in each activation cycle is determined by the partitioning function ν . g computes the sequence of states $\langle z_1, z_2, z_3, \dots \rangle$ and f gradually produces the values for the output signals. For actor A we write $A(\langle s_1, s_2 \rangle) = \langle s_3, s_4 \rangle$ to denote an actor that consumes two input signals s_1 and s_2 and generates two output signals s_3 and s_4 .

Let A and B be two actors with input and output signals I_A, I_B, O_A and O_B , respectively. If a subset O'_A of A 's output signals has the same type as a subset of B 's input signals I'_B , they can be connected such that the signals $O'_A = I'_B$. This results in a *compound actor* C with input signals $I_C = I_A \cup (I_B \setminus I'_B)$ and output signals $O_C = (O_A \setminus O'_A) \cup O_B$. The semantics of such actor networks (or process networks) are developed in [7], together with an analysis of loops and deadlocks.

3 Abstraction

Abstraction is a prerequisite for self-modeling because the model that an actor entertains of itself, must be simpler, hence more abstract, than itself. Since we try to capture the notion of unlimited recursive self-modeling, we need to make sure, that the self-model at one level is more abstract than the self-model of the previous level. Here we do not show what a “good” abstraction is or how to derive it, but we only show that certain signal abstractions, that we use in later sections, have reduced information content.

3.1 Signal Abstraction

Given two signals $s_1 : \text{Signal}(V_1)$ and $s_2 : \text{Signal}(V_2)$, an abstraction of s_1 is a mapping $B_\alpha : \text{Signal}(V_1) \rightarrow \text{Signal}(V_2)$ with an abstraction function $\alpha : \langle V_1 \rangle \rightarrow V_2$ that maps sequences of s_1 onto individual values of s_2 .

For instance, if a thermometer measures the sequence of temperature values as $s_1 = \langle 36.7, 36.8, 36.7, 36.8, 36.9, 36.9, 37.0, 37.0, 37.1, 37.2, 37.3, 37.2, 37.3, 37.3, 37.4, 37.5, 37.6, 36.6 \rangle$, then the abstraction B_α with

$$\alpha(\langle t_1, t_2, t_3 \rangle) = \begin{cases} \text{l} & \text{if } (t_1 + t_2 + t_3)/3 < 35.5 \\ \text{n} & \text{if } 35.5 \leq (t_1 + t_2 + t_3)/3 < 37.5 \\ \text{c} & \text{if } 37.5 \leq (t_1 + t_2 + t_3)/3 < 38.5 \\ \text{h} & \text{if } 38.5 \leq (t_1 + t_2 + t_3)/3 \end{cases}$$

the abstraction B_α would map three consecutive temperate measurements onto one symbol, i.e. $B_\alpha(s_1) = \langle \text{n}, \text{n}, \text{n}, \text{n}, \text{n}, \text{c} \rangle$.

Many signal processing functions can be considered abstractions. E.g. an ECG signal can be abstracted into a sequence of pulse periods, or into a sequence of P, Q, R, S, T symbols to indicate the main components of the ECG signal. The important points are that the abstracted signal represents less information and thus can be encoded with fewer bits, and that it reflects regularities and repetitive patterns. If a sequence of values appears many times in the input signal, this sequence can be abstracted into one abstract symbol. (GrammarViz [28] and unsupervised symbolization [20] are examples for general methods for signal abstraction.)

3.2 Information Reduction by Abstraction

The *Shannon Entropy* [2] provides a formalism for measuring the information content of a signal.¹ Let $V = \{v_1, v_2, \dots, v_N\}$ be the set of symbols that appear on the signal s and let

¹ The Shannon Entropy assumes independent, identically distributed random variables, which in fact cannot be assumed in our case. In the following we use the Shannon Entropy as an estimate but recognize the need for a more appropriate model.

the probability of v_i be p_i , then the Shannon Entropy H of signal s is

$$H(s) = - \sum_{i=1}^N p_i \log p_i.$$

$H(s)$ gives the average amount of information per symbol. For a signal of length m , $s = \langle e_1, e_2, \dots, e_m \rangle$ the information content is

$$I(s) = - \sum_{j=1}^m \mathbf{E}[\log p(e_j)] = -m \sum_{i=1}^N p_i \log p_i$$

where $p(e)$ is the probability of event e and N is the number of distinct symbols.

3.2.1 Value Abstraction

We consider two types of abstraction, time and value abstraction. Let the value abstraction function α_v be

$$\alpha_v(\langle x \rangle) = \begin{cases} \mathfrak{A} & \text{if } x = \mathfrak{a} \text{ or } x = \mathfrak{b} \\ x & \text{otherwise} \end{cases}$$

which maps two symbols \mathfrak{a} and \mathfrak{b} onto the same symbol \mathfrak{A} and leaves all other symbols unmodified. The abstraction B_{α_v} leaves the lengths of signals unchanged but reduces the number of different symbols by one. As a consequence, the information content of the abstracted signal is reduced as well which can be expressed by way of the Shannon Entropy.

Let $V = \{v_1, v_2, \dots, v_N\}$ be a set of symbols with $v_1 = \mathfrak{a}$ and $v_2 = \mathfrak{b}$, let $V_A = \{\mathfrak{A}, v_3, \dots, v_N\}$ be another set of symbols with $N - 1$ elements, let p_i be the probability of occurrence of v_i with $p_1 = p_{\mathfrak{a}}$ and $p_2 = p_{\mathfrak{b}}$, and $p_{\mathfrak{A}}$. Further, let s be a signal of length m and let $s_A = B_{\alpha_v}(s)$ be the abstracted signal of equal length. The Shannon entropy of these two signals is

$$\begin{aligned} H(s) &= - \sum_{i=1}^N p_i \log p_i = -p_{\mathfrak{a}} \log p_{\mathfrak{a}} - p_{\mathfrak{b}} \log p_{\mathfrak{b}} - \sum_{i=3}^N p_i \log p_i \\ H(s_A) &= -p_{\mathfrak{A}} \log p_{\mathfrak{A}} - \sum_{i=3}^N p_i \log p_i \end{aligned}$$

Since the last sum is identical in both expressions and since $p_{\mathfrak{A}} = p_{\mathfrak{a}} + p_{\mathfrak{b}}$ we have as entropy difference of these two signals

$$H_{\delta} = H(s) - H(s_A) = p_{\mathfrak{a}} \log \frac{p_{\mathfrak{a}} + p_{\mathfrak{b}}}{p_{\mathfrak{a}}} + p_{\mathfrak{b}} \log \frac{p_{\mathfrak{a}} + p_{\mathfrak{b}}}{p_{\mathfrak{b}}}$$

The information content decreases on average by H_{δ} per symbol and it depends only on the probabilities of the two abstracted symbols \mathfrak{a} and \mathfrak{b} . For the special case $p_{\mathfrak{a}} = p_{\mathfrak{b}} = p$ and the base 2 logarithm we have $H_{\delta} = 2p$.

3.2.2 Time Abstraction

Let the time abstraction function α_t be

$$\alpha_t(\langle x_1, x_2 \rangle) = \begin{cases} \mathfrak{A} & \text{if } x_1 = \mathfrak{a} \text{ and } x_2 = \mathfrak{a} \\ \langle x_1, x_2 \rangle & \text{otherwise} \end{cases}$$

B_{α_t} maps two consecutive occurrences of \mathfrak{a} onto \mathfrak{A} and leaves all other symbols unchanged. We assume that all symbols \mathfrak{a} appear in pairs, thus all \mathfrak{a} 's are replaced by \mathfrak{A} 's. This is not a simplification since we can pick an arbitrary pair of symbols, say $\langle \mathfrak{b}, \mathfrak{c} \rangle$ and first apply a value abstraction to transform it into $\langle \mathfrak{a}, \mathfrak{a} \rangle$ pairs, and then apply the time abstraction function α_t . The key point is that α_t shortens the signal by replacing all pairs of symbols $\langle \mathfrak{a}, \mathfrak{a} \rangle$ by a new symbol \mathfrak{A} .

α_t reduces the signal length but not the number of symbols and not necessarily the information per symbol. Thus, the reduction of information content comes from the decreasing signal length. While the general case is quite involved, we can illustrate the trend with a special case. Assume an abstraction function

$$\alpha_t(\langle x_1, x_2 \rangle) = \begin{cases} \mathfrak{A} & \text{if } x_1 = \mathfrak{a} \text{ and } x_2 = \mathfrak{a} \\ \mathfrak{B} & \text{if } x_1 = \mathfrak{b} \text{ and } x_2 = \mathfrak{b} \\ \mathfrak{C} & \text{if } x_1 = \mathfrak{c} \text{ and } x_2 = \mathfrak{c} \\ \dots & \end{cases}$$

Assume further that s consists only of symbol pairs like $s = \langle \mathfrak{a}, \mathfrak{a}, \mathfrak{c}, \mathfrak{c}, \mathfrak{a}, \mathfrak{a}, \mathfrak{b}, \mathfrak{b}, \mathfrak{a}, \mathfrak{a}, \mathfrak{d}, \mathfrak{d}, \mathfrak{b}, \mathfrak{b}, \dots \rangle$. The abstraction $B_{\alpha_t}(s)$ will then half the length of s but probabilities will be maintained like $p_{\mathfrak{a}} = p_{\mathfrak{A}}$, $p_{\mathfrak{b}} = p_{\mathfrak{B}}$, $p_{\mathfrak{c}} = p_{\mathfrak{C}}$, etc. Thus, the Shannon Entropy for s and $s_A = B_{\alpha_t}(s)$ is

$$\begin{aligned} H(s) &= - \sum_{i \in \{\mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \dots\}} p_i \log p_i \\ H(s_A) &= - \sum_{i \in \{\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \dots\}} p_i \log p_i = H(s) \end{aligned}$$

Hence, the Shannon Entropy denotes the average information content per symbol, which is unchanged. However, the information content of the entire signal is as follows.

$$\begin{aligned} I(s) &= mH(s) \\ I(s_A) &= \frac{m}{2}H(s_A) = \frac{I(s)}{2} \end{aligned}$$

if the length of s is m and the lengths of s_A is $m/2$ as a result of the abstraction.

Time abstraction has its name because signals encode timing information. This means that merging two consecutive symbols into one decreases the number of symbols per time. Hence, timing abstraction reduces the information content per time unit.

Reducing the amount of information is a necessary condition for an abstraction but it is not sufficient for a useful abstraction. A useful abstraction will reduce the information that is less relevant and keep the important information, thus increasing its prominence. Much could be said about finding good abstractions, see for instance [30] for effective abstraction techniques. Also note, that what constitutes a useful abstraction depends on the actor's goals and condition.

3.3 Abstractions as Actors

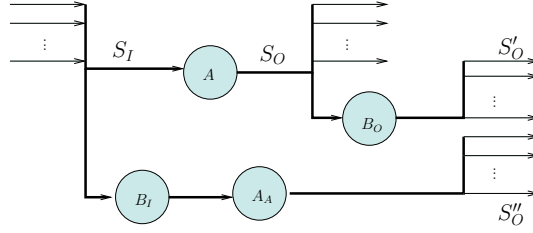
Signal abstraction is modeled as an actor that maps one or more input signals onto output signals. Let $B = \langle \mathfrak{T}, I, O, z_0, \nu, f, g, \vec{m}_0 \rangle$ with $\mathfrak{T} = \{\epsilon\}$ (the actor is stateless), $I = \{\text{Signal}(V_1), \text{Signal}(V_2), \dots\}$, $O = \{\text{Signal}(V')\}$ (one or more input and one output signal), $z_0 = \epsilon$, (No initial state), $\nu(\cdot) = \langle c_1, c_2, \dots \rangle$ (the actor consumes a constant number of values from each input signal), $g(\cdot, \cdot) = \epsilon$ (no states), $\vec{m}_0(\cdot) = \text{nop}$ (no meta actions).

B maps one or more input signals consisting of symbols from V_1, V_2, \dots onto one output signal with symbols from V' , and if it applies a combination of value and time abstractions, we call it a signal abstraction. We can of course conceive more complex abstractions with internal states, but this kind of abstraction will suffice to illustrate the approach.

Given three actors A, B_I, B_O an **actor abstraction** $\text{ActAbstraction}(A, B_I, B_O) = A_A$ denotes an abstraction of actor A , if

$$B_O(A(S_I)) = A_A(B_I(S_I))$$

for all set of input signals S_I that can be consumed by A .



■ **Figure 1** A_A is an abstraction of A if $S'_O = S''_O$.

This situation is depicted in Figure 1. A_A operates on an abstraction of the input signals S_I , abstracted by the actor B_I . If the output signals S''_O generated by A_A are identical to the signals S'_O , which are abstractions of S_O , then actor A_A is an abstraction of actor A . This definition is not constructive and does not tell us how to derive A_A , or B_I or B_O ; nor does it tell us what a useful abstraction is. Intuitively A_A should be significantly simpler than A but should faithfully reflect relevant properties of A .

4 Self-Model

An actor with a self-model has an abstract model of its own behavior, an abstract model of the environment it interacts with, and the capability to simulate these abstract models together.

Let A be an atomic or compound actor (as defined in section 2.3) arbitrarily complex actor, let B_I and B_O be abstractors of the input and output of A , respectively, and let $\text{ActAbstraction}(A, B_I, B_O) = A_A$, just as discussed in section 3.3. Further, let E be the environment the actor interacts with through the signal sets S_I and S_O , and let E_A be an abstraction of E , such that we have $\text{ActAbstraction}(E, B_O, B_I) = E_A$.

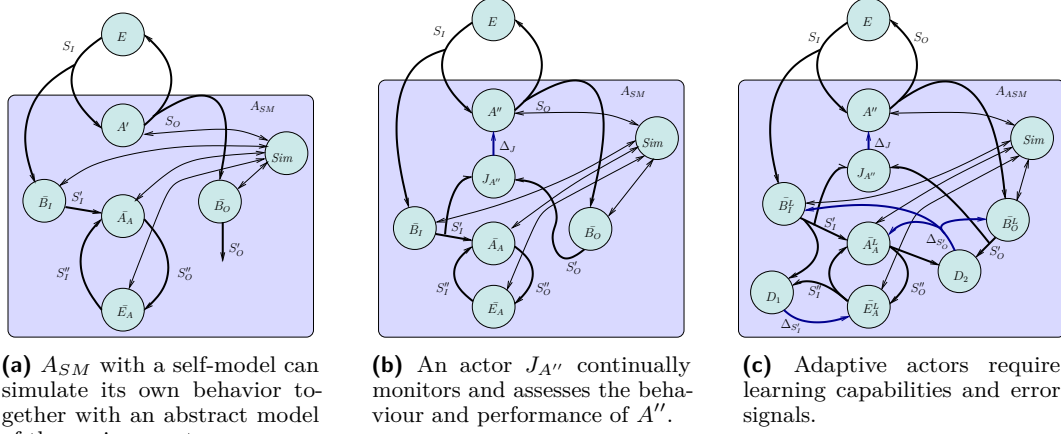
Moreover, let \bar{A} be a *simulatable actor* derived from A which behaves like A with the following additions:

- It has an additional input signal denoted as *control signal*.
- It can be stopped and resumed at will through the control signal.
- For each input signal of A it has two input signals of the same type; hence it has two sets of input signals with identical types. The control signal selects one of the two sets for input in each activation cycle.
- It has an additional output signal, denoted as *status signal* that reports its internal status under control of the control signal.

The whole situation is illustrated in Figure 2a. In addition we see in the figure a *Sim* actor, which controls the models \bar{A}_A , \bar{E}_A , \bar{B}_I and \bar{B}_O to simulate them. Also, instead of actor A we have a modified actor A' which acts just like A but can at appropriate times invoke the

simulator, learn about simulation results, which then can support its decision process. Thus, given appropriate actors $A, \bar{A}_A, \bar{E}_A, \bar{B}_I, \bar{B}_O$ and Sim , $ActorSelfModel(A, \bar{A}_A, \bar{E}_A, \bar{B}_I, \bar{B}_O) = A_{SM}$ is a resulting actor corresponding to the one shown in Figure 2a.

4.1 Self-Assessment



■ **Figure 2** A self modeling actor A_{SM} .

To allow for self-assessment the actor requires a model of the specification and requirements of itself. Such a model can be an elaborate functional model, or it can be a list of properties that at all times have to be fulfilled. A large body of literature has studied this problem under terms such as run-time monitoring, fault tolerance, and reliability. Thus, we assume solutions readily exist and a dedicated actor, named $J_{A''}$, continually monitors the input and output signals of the actor under observation and detects functional and performance aberrations. We could connect $J_{A''}$ to the actor inputs S_I and outputs S_O , however, it is more likely that $J_{A''}$ operates on abstractions of those signals like the one provided by \bar{B}_I and \bar{B}_O . The output of actor $J_{A''}$ in Figure 2b is denoted as Δ_J and signifies the difference between expected and observed behavior. It is fed back to actor A'' to allow for the use of this information and improve its performance. Hence, we have a variation of the actor without that facility, which was named A' .

If $J_{A''}$ also maintains a history of the assessment, it facilitates a holistic lifetime self-assessment as a basis for hindsight analysis, self-explanation and self-improvement.

4.2 Adaptive Self-Model

To model adaptive actors we need to capture the notion of learning.² A *Learning Actor* A^L is an actor that takes an error signal as input, in addition to the other signals it needs for its operation, and modifies its behavior with the goal to minimize the error in the error signal. Hence, let A be an actor with input signals I_A and output signals O_A , the learning actor

² The meaning of terms “learning”, “adaptation” and “optimization” overlap. Here we use the term “learning” as a basic capability of an actor to modify its own behavior based on an error signal. Depending on how this capability is used, the actor may be *self-optimizing*, when the behavior improves within the same environment, or *adaptive*, when its behavior appropriately changes as response to a changing environment, or both.

A^L has input signals $I_{A^L} = I_A \cup \{\sigma_\epsilon\}$ and output signals $O_{A^L} = O_A$, where σ_ϵ is an error signal that reflects the quality of A 's performance in some way. It could be a simple numeric signal or it could be structured to detail which parts and aspects of A 's behavior exhibits which quality.

Considering Figure 2b, there are several parts that we would like to see continually improved, in particular the abstractions \bar{B}_I and \bar{B}_O , and the abstract models \bar{A}_A and \bar{E}_A . If we want to make them learning actors, we have to identify the source of the error information. While actors could use application specific information, obvious generic sources are the differences between signals S'_O and S''_O , and between signals S'_I and S''_I .

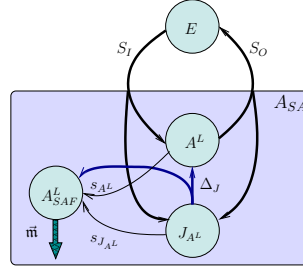
Consequently, we introduce actors that analyze the differences in two sets of signals to generate Δ signals that inform other actors about observed differences. Figure 2c shows two actors, D_1 and D_2 that analyze and compare signals S'_I , S''_I and signals S'_O , S''_O , respectively, to generate the signals $\Delta_{S'_I}$ and $\Delta_{S'_O}$. These Δ signals are then used by the learning actors \bar{B}_I^L , \bar{B}_O^L , \bar{A}_A^L and \bar{E}_A^L to improve their models and their behavior. Figure 2c shows one possible scenario but many other strategies are conceivable and other information sources can be utilized to improve learning actors. We imagine that the learning actors in Figure 2c start with an initial, relatively crude model or behavior which then is continuously improved with the expectation that this continuous improvement eventually leads to far better models and behaviors for \bar{B}_I^L , \bar{B}_O^L , \bar{A}_A^L and \bar{E}_A^L than could possibly be accomplished with careful engineering at design time.

5 Recursive Self-Reflection

Our actor has been extended quite significantly as illustrated in figure 2c. Before moving on, let's step back and consider what we have done. We have added functionality to our original actor A twice, as indicated by the two ticks. We have added assessment and simulation facilities together with abstract models of the actor itself and the environment. This allows for improved behavior of the actor by using self-assessment information from $J_{A''}$ and by using predictions from Sim . In addition we have introduced learning capabilities for the abstractions. Thus, A'' is continually improving by three different means: self-assessment, simulation based prediction, improving abstract models.

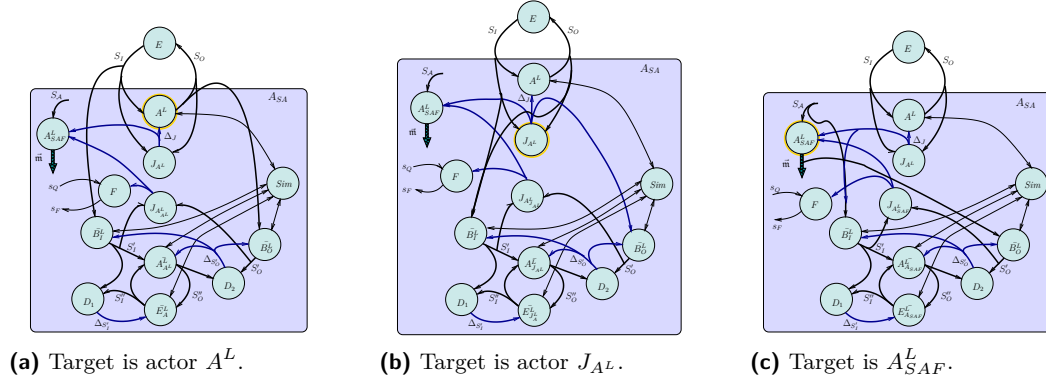
As a result we have obtained the actor A_{ASM} , an *adaptive, self-modeling actor*. Is it self-aware? The abstract self-model, the simulation engine, the self-assessment and the learning capabilities are all ingredients of self-awareness but they are not self-awareness, just like flour, sugar, raisins, yeast are ingredients for a cake, but they are not yet the cake. In fact, A_{ASM} can be considered the cake, but we are not looking for the cake, we are looking for the *process* of baking. So far we have used the mechanisms abstraction, simulation, assessment, and learning deliberately to construct something which resembles self-awareness, but the result is not self-awareness because self-awareness is the process, not the result. We need a general method that uses those mechanisms and can be applied to any actor, not just A . In particular, it must also be applicable to itself.

Consider Figure 3, where a learning actor A^L interacts with the environment and is continuously monitored by J_{A^L} . Imagine the monitor J_{A^L} is more complex than checking properties. It keeps track of a set of goals that may be hierarchically organized and in part mutually contradictory. The goals could be to perform some useful function, to keep the battery loaded, avoid harming people, avoid damage to itself and to its environment, etc. The Δ_J signal informs to which extent these goals are satisfied at any time during operation.



■ **Figure 3** A self-awareness facilitating actor.

In addition we have an actor A_{SAF}^L that facilitates self-awareness. It is informed by J_{A^L} about the actor's performance and, through the signals s_{A^L} and $s_{J_{A^L}}$ it keeps track of which actors are in the system. If it deems necessary, for instance when it is unhappy about the actor's performance, it can trigger an investigation. At its disposal it has simulation, abstraction, learning and other facilities. Picking an actor, for instance A_L , it can spawn a monitoring and assessment setup as illustrated in Figure 4a. It does all this through meta actions through the \bar{m} output in the figure. The self-awareness facilitator still keeps track of many, but not necessarily all, actors in the system, which is indicated by the S_A input signal. It can spawn a new investigation into any of the newly created actors if deemed useful and if the available resources suffice.



■ **Figure 4** A self-awareness facilitating actor A_{SAF}^L targeting other actors for study.

In addition, we propose to also provide an explanation actor F that, through a question and explanation interface (signals s_Q and s_F), provides a mechanism to explain what has happened, which decisions have been taken, what observations have been made. We expect this actor to be useful in the interaction with other systems. In particular in the interaction with humans it will convey to which extent the A_{SA} actor is self-aware and at what level it understands what it is doing.

For now, let's assume A_{SAF}^L deletes the newly created actors and returns to the state shown in figure 3, and then picks actor J_{A^L} as a next target for investigation, the result of which is shown in figure 4b. Moreover, it may target itself, if unhappy with its own performance or if just curious, and thus create a situation as shown in figure 4c.

In its simplest form the proposed self-reflection mechanism picks an actor, atomic or compound, abstracts this actor and assesses the behavior of the abstracted actor by comparing it to the actor's stated goals. Hence, a prerequisite for this operation is the accessibility of

stated goals, which may be part of the actor under study or may come from somewhere else. From that it follows, that A_{SAF}^L can study any other actor for which it has access to its inputs, outputs and goals. This may in principle be the case for any of the actors visible in Figures 4a – 4c. But note, that not every interior detail of an actor is necessarily subject to this mechanism, since it is limited to behavior visible from the outside.

To warrant the name “recursive” the mechanism must be applicable to itself and to thus recursively derived actors, without principle limits. Consequently, any actor created by A_{SAF}^L in Figure 4 could have its inputs, outputs and goals again be accessible to A_{SAF}^L . Without working out the details here this is plausible for all created actors because A_{SAF}^L generates the inputs and outputs itself and “knows” what it is supposed to accomplish.

In summary, we define self-awareness as the capability to pick any actor in the system, it may be a simple or compound actor or the entire system itself, and apply abstraction, assessment, prediction, and learning techniques, as outlined in this article, in order to analyze, assess and possibly improve its performance.

6 Related Work

As alluded to in the introduction a substantial amount of papers have been published on the topic of self-awareness. Here we only compare our proposal to definitions of self-awareness that have similar scope and ambition.

In 2009 Agarwal et al. [1] argue that self-aware subjects should be “*introspective*” (they can observe and optimise their own behaviour), “*adaptive*”, “*self-healing*”, (they monitor themselves for faults and take corrective actions), “*goal oriented*”, and “*approximate*”, (they use the least amount of precision to accomplish a given task).

In 2011 Lewis et al. [19] base their concepts on work in psychology, in particular on Morin’s definition of self-awareness as “*the capacity to become the object of one’s own attention*” [22] and Neisser’s five-level model [24] which includes the “*ecological self*”, the “*interpersonal self*”, the “*extended self*”, the “*private self*” and the “*conceptual self*”, the last being “*the most advanced form of self-awareness, representing that the organism is capable of constructing and reasoning about an abstract symbolic representation of itself*” [19].

In 2014 Jantsch et al. [10] give seven properties that constitute awareness and define a subject to be aware at level 0 to 5, depending on which of these properties are exhibited by the subject. For instance level 4 requires that the subject assesses its own performance over the history of its lifetime, and can simulate future actions for prediction and planning purposes. The highest level 5 defines group awareness which requires subjects to be aware of its peers in a group.

We have cited the 2017 definition by Kounev et al. [11] in the introduction and repeat here only that it requires a subject to be self-reflective, self-predictive and self-adaptive.

All these definitions have some concepts in common, like goal orientation, adaptation, and introspection, but also differ in whether they include self-healing, approximation, learning, or prediction. But note that a definition that does not include an aspect such as learning probably does not mean to exclude it either. What is mentioned explicitly may only reflect the prominence given to some of the aspects, while others are less emphasized. These ambiguities and imprecision are a consequence of the informal style used to describe rather than define the key concepts of self-awareness.

Hence the first main difference to the work cited above is our attempt to provide a formal semantic for the involved concepts thus avoiding ambiguities and imprecision. We admit, that this attempt in giving a formal semantic is not complete but we argue it is a first step that shows the contours of such a semantic and that suggests it can be given.

The work by Vassev and Hinchey [31] is a formal approach to model self-awareness based on knowledge representation. It captures knowledge the system has about itself that includes information, rules, constraints and methods. The formal model has the benefit of clarity and unambiguity which makes clear that awareness is reduced to knowledge representation. In the described case study this self-knowledge is used by robots in a swarm to make situation dependent decisions that sensibly contribute to an overall swarm behavior. However, a mechanism to observe, assess and reason about its own usage of self-knowledge is missing.

Hence, the second main difference is the concept of recursive reflection. No other previous definition or model allows for applying self-awareness recursively onto its own activity. However, we contend that this unbounded recursion is the essence of self-awareness and it requires a formal model to demonstrate its feasibility and its utility.

7 Conclusions

The proposed formal model of self-awareness is based on a dynamic dataflow semantics. It captures the notion of signal abstraction, actor abstraction, adaptive actors, self-assessment, and recursive self-reflection. Even though many details of the formalism are still missing and the approach has not yet been demonstrated we are hopeful that it can be implemented and simulated in an appropriate framework.

A particular appealing aspect of recursive self-reflection is its promise, that any particular situation can be abstracted up to a level, where it is amenable to the assessment and planning capabilities of the system. Thus, there is no situation too complex that the self-aware actor is able to handle, provided it finds the appropriate sequence of abstractions. Since an abstraction step reduces the amount of information and since abstractions can be recursively applied, a given situation can be abstracted up to the level, where its information amount is within the limit of the system. The human mind seems to be doing something similar, because it manages to analyze, elaborate, and handle arbitrarily complex subjects even though the amount of conscious information processing is severely limited as has been established in Miller's seminal paper in 1956 on the magical number seven [21], and confirmed many times since then. If this analogy is correct, and if sufficiently effective and efficient abstraction techniques can be developed and employed, recursive self-reflection would turn out to be a wonderfully general tool for dealing with arbitrary situations where assessment and planning is crucial but an overwhelming diversity and complexity seems to render any general technique futile. These are big ifs and a number of questions arise.

Abstraction techniques. We need efficient techniques for automated abstraction. The definition of **ActAbstraction** is not constructive and there seems to be no good, general method to abstract an arbitrary actor. However, many abstraction methods exist but all of them have their strength and drawbacks. Thus, we need to identify good abstraction methods for our purpose and we need methods to select the most appropriate for a specific actor and for specific objectives.

Abstraction level. Related to the abstraction method is the question of the right abstraction level. A given set of data and a given abstractor can be abstracted more or less. It is not well understood what constitutes a good abstraction level in general, and how to identify a good abstraction level in a particular case.

Assessment techniques. We need good assessment techniques. Again, we do not have good general methods for assessment of an arbitrary actor.

Goal Management. Complex systems often have a complex goal structure, which may be hierarchical and dynamic with partially overlapping and partially mutually exclusive

goals. Handling these goals and assessing an actor's performance with respect to given goals is an interesting challenge.

Learning. Machine learning is an active research domain and many methods have been proposed and studied. The challenge for us is to identify appropriate and efficient learning methods streamlined for our purpose.

Simulation. Finally, general and efficient simulation methods will be instrumental to make self-awareness as proposed efficient. The key here is probably not the simulation method itself, but to find the right abstraction level in combination with efficient simulation methods.

With a precise, formal and operational model of self-awareness we can identify its challenges, address the open problems and study its benefits and drawbacks in the context of specific applications. As a result, self-awareness could be made into a powerful generic method that can be the foundation of truly autonomous systems.

References

- 1 A. Agarwal, J. Miller, J. Eastep, D. Wentzlauff, and H. Kasture. Self-aware Computing. *Final Technical Report AFRL-RI-RS-TR-2009-161*, page 81, 2009.
- 2 T. M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, Hoboken, NJ, USA, 2006.
- 3 Lukas Esterle and Radu Grosu. Cyber-physical systems: challenge of the 21st century. *e & i Elektrotechnik und Informationstechnik*, 133(7):299–303, November 2016.
- 4 European Commission. Self-Awareness in Autonomic Systems, January 2013.
- 5 Radu Grosu and Ketil Stølen. A denotational model for mobile point-to-point dataflow networks. Technical Report Technical Report SFB 342/14/95 A, Technische Universität München, 1995.
- 6 Radu Grosu and Ketil Stølen. A Model for Mobile Point-to-Point Data-flow Networks without Channel Sharing. In *In Proc. AMAST'96, LNCS*, pages 504–519, 1996.
- 7 Axel Jantsch. *Modeling Embedded Systems and SoCs - Concurrency and Time in Models of Computation*. Systems on Silicon. Morgan Kaufmann Publishers, June 2003.
- 8 Axel Jantsch. Models of Embedded Computation. In Richard Zurawski, editor, *Embedded Systems Handbook*. CRC Press, 2005. Invited contribution.
- 9 Axel Jantsch, Nikil Dutt, and Amir M. Rahmani. Self-Awareness in Systems on Chip – A Survey. *IEEE Design Test*, 34(6):1–19, December 2017.
- 10 Axel Jantsch and Kalle Tammemäe. A Framework of Awareness for Artificial Subjects. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, CODES '14, pages 20:1–20:3, New York, NY, USA, 2014. ACM.
- 11 S. Kounev, J.O. Kephart, A. Milenkoski, and X. Zhu, editors. *Self-Aware Computing Systems*. Springer, 2017.
- 12 Samuel Kounev, Peter Lewis, Kirstie Bellman, Nelly Bencomo, Javier Camara, Ada Diaconescu, Lukas Esterle, Kurt Geihs, Holger Giese, Sebastian Götz, Paola Inverardi, Jeffrey Kephart, and Andrea Zisman. The Notion of Self-Aware Computing. In Samuel Kounev, Jeffrey O. Kephart, Aleksandar Milenkoski, and Xiaoyun Zhu, editors, *Self-Aware Computing Systems*, pages 3–16. Springer, 2017.
- 13 Samuel Kounev, Xiaoyun Zhu, Jeffrey O. Kephart, and Marta Kwiatkowska. Model-driven Algorithms and Architectures for Self-Aware Computing Systems (Dagstuhl Seminar 15041). *Dagstuhl Reports*, 5(1):164–196, 2015.
- 14 Edward A. Lee. A Denotational Semantics for Dataflow with Firing. Technical Report UCB/ERL M97/3, Department of Electrical Engineering and Computer Science, University of California, Berkeley, January 1997.

- 15 Edward A. Lee and Alberto Sangiovanni-Vincentelli. A Framework for Comparing Models of Computation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1217–1229, December 1998.
- 16 Peter R. Lewis. Self-aware Computing Systems: From Psychology to Engineering. In *Proceedings of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1044–1049, 2017.
- 17 Peter R. Lewis, Arjun Chandra, Funmilade Faniyi, Kyrre Glette, Tao Chen, Rami Bahsoon, Jim Torresen, and Xin Yao. Architectural Aspects of Self-aware and Self-expressive Computing Systems. *IEEE Computer*, August 2015.
- 18 Peter R. Lewis, Marco Platzner, Bernhard Rinner, Jim Torresen, and Xin Yao, editors. *Self-Aware Computing Systems: An Engineering Approach*. Springer, 2016.
- 19 P.R. Lewis, A. Chandra, S. Parsons, E. Robinson, K. Glette, R. Bahsoon, J. Torresen, and Xin Yao. A Survey of Self-Awareness and Its Application in Computing Systems. In *Self-Adaptive and Self-Organizing Systems Workshops (SASO), 2011 Fifth IEEE Conference on*, pages 102–107, October 2011.
- 20 Yue Li and Asok Ray. Unsupervised symbolization of signal time series for extraction of the embedded information. *Entropy*, 19(4), January 2017.
- 21 George A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *Psychological Review*, 63:81–97, 1956.
- 22 Alain Morin. Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views. *Consciousness and Cognition*, 15(2):358–371, 2006.
- 23 Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors. *Organic Computing — A Paradigm Shift for Complex Systems*. Birkhauser, 2011.
- 24 Ulric Neisser. The roots of self-knowledge: Perceiving self, it, and thou. *Annals of the New York Academy of Sciences*, 818:19–33, 1997.
- 25 L.D. Paulson. DARPA creating self-aware computing. *IEEE Computer*, 36(3):24, March 2003.
- 26 Jeremy Pitt, editor. *The Computer after Me : Awareness and Self-Awareness in Autonomic Systems*. Imperial College Press, 2014.
- 27 J. Schaumeier, J. Jeremy Pitt, and G. Cabri. A Tripartite Analytic Framework for Characterising Awareness and Self-Awareness in Autonomic Systems Research. In *Proceedings of the Sixth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*, pages 157–162. IEEE Computer Society, 2012.
- 28 P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A.P. Boedihardjo, C. Chen, S. Frankenstein, and M. Lerner. GrammarViz 2.0: a tool for grammar-based pattern discovery in time series. In *ECML/PKDD*, 2014.
- 29 R. Sterritt and M. Hinchey. SPAACE IV: Self-properties for an autonomous and autonomic computing environment - part IV a newish hope. In *IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems (EASE)*, pages 119–125, 2010.
- 30 Joshua B. Tenenbaum, Charles Kemp, Thomas L. Griffiths, and Noah D. Goodman. How to Grow a Mind: Statistics, Structure, and Abstraction. *Science*, 331(6022):1279–1285, 2011.
- 31 E. Vassev and M. Hinchey. Knowledge Representation and Awareness in Autonomic Service-Component Ensembles - State of the Art. In *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 110–119, March 2011.
- 32 Martin Wirsing, Matthias Hölzl, Nora Koch, and Philip Mayer, editors. *Software Engineering for Collective Autonomic Systems: The ASCENS Approach*. Springer, 2015.

A Dependable Detection Mechanism for Intersection Management of Connected Autonomous Vehicles

Rachel Dedinsky

Arizona State University,
660 S Mill Ave, Tempe, AZ, US
rdedinsk@asu.edu

Mohammad Khayatian

Arizona State University,
660 S Mill Ave, Tempe, AZ, US

Mohammadreza Mehrabian

Arizona State University,
660 S Mill Ave, Tempe, AZ, US

Aviral Shrivastava

Arizona State University,
660 S Mill Ave, Tempe, AZ, US

Abstract

Traffic intersections will become automated in the near future with the advent of Connected Autonomous Vehicles (CAVs). Researchers have proposed intersection management approaches that use the position and velocity that are reported by vehicles to compute a schedule for vehicles to safely and efficiently traverse the intersection. However, a vehicle may fail to follow intersection manager (IM) scheduling commands due to erroneous sensor readings or unexpected incidents like engine failure, which can cause an accident if the failure happens inside the intersection. Additionally, rogue vehicles can take the advantage of the IM by providing false position and velocity data and cause traffic congestion. In this paper, we present a new technique and infrastructure to detect anomalies and inform the IM. We propose a vision system that can monitor the position of incoming vehicles and provide real-time data for the IM. The IM can use this data to verify the trajectories of CAVs and broadcast a warning when a vehicle fails to follow commands, making the IM more resilient against attacks and false data. We implemented our method by building infrastructure for an intersection with 1/10 scale model CAVs. Results show our method, when combined with an IM dataflows, is more dependable in the event of a failure compared to an IM without it.

2012 ACM Subject Classification Computer systems organization → Embedded and cyber-physical systems

Keywords and phrases Connected Autonomous Vehicles, Intersection Management, Dependable Systems

Digital Object Identifier 10.4230/OASICS.ASD.2019.7

Category Interactive Presentation

1 Introduction

CAVs are expected to shape the future of the automobile industry because they have the potential to minimize traffic, increase user satisfaction by enabling user autonomy, and, most importantly, increase safety for all interaction models. In order to truly be accepted by the community, it is important to have reliable infrastructure to support these vehicles.

A point of complexity in this fully autonomous system is managing traffic at an intersection. To manage traffic best, infrastructure is implemented to support intersection management.



© Rachel Dedinsky, Mohammad Khayatian, Mohammadreza Mehrabian, and Aviral Shrivastava;
licensed under Creative Commons License CC-BY

Workshop on Autonomous Systems Design (ASD 2019).

Editors: Selma Saidi, Rolf Ernst, and Dirk Ziegenbein; Article No. 7; pp. 7:1–7:13

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Automated Intersection Management or Intersection Managers (IM(s)) as they are referred to in this paper interact with vehicles as they approach an intersection. IMs have the potential to make intersections safer and more efficient compared to the involvement of human drivers in interactions, as these IMs instruct traffic as an unbiased external system specifically designed for both safety and efficiency. Humans are biased and prone to making a greedy decision in safety-critical situations which can decrease throughput and lead to accidents. Thus, many researches regarding intersection optimization have involved the use of IMs.

Some of the most notable existing researches using IMs are Autonomous Intersection Management(AIM)[7], Crossroads[2], and Robust Intersection Management(RIM)[11]. In AIM, vehicles query the IM with a specific velocity of arrival and trajectory of arrival in order to reserve a timeslot through the intersection, which is commonly called a query-based IM(QB-IM). In Crossroads, vehicles provide their current position and velocity to the IM and request a velocity from the IM to drive safely through the intersection; this technique is commonly referred to as a velocity-assignment Intersection Management(VA-IM). VA-IM is improved in RIM; typically a VA-IM assumes constant velocity through the intersection but RIM tracks position trajectory in order to account for external disturbances. From experimentation, we know that even RIM can't account for all external disturbances, because it has no physical monitoring system to ensure that the trajectory of arrival assigned in RIM is actually achieved.

A common technique in these IMs to achieve safe operation despite uncertainty in vehicle's trajectory is to consider a safety buffer around the vehicles to ensure safe scheduling of vehicles through the intersection. Though this achieves some level of reliability by considering worst case position uncertainty of vehicles, this can still be erroneous if too much trust is placed on one system. Consider the case where a vehicle has inaccurate positioning information which is being reported to the IM. If the error of positioning is greater than the threshold safety buffer, an accident is likely to occur in the intersection. For example in AIM to achieve a safe scheduling of vehicles through the intersection, the CAVs send the IM completion times; this way the IM knows which vehicles are about to enter the intersection, are in the intersection, or have completed their trajectories through the intersection. The completion time data which the IM uses to create a schedule are based on the data the car is reporting for its position, and in the case that the car is reporting positioning outside of the threshold safety buffer value the schedule will be inaccurately computed. True safety relies on redundancy of systems to maximize the safety of the IMs.

In addition to safety, these IMs must be more efficient to be accepted by the community. A study done by University of Michigan [5] showed that connected vehicles, while improving quality of life for humans, also opens the door for cyber-attacks. Their study attacked I-SIG sponsored by the United States Department of Transportation and their results showed how drastically throughput can decrease if traffic control is not secured. Those doing research on IMs use encryption to prevent outsiders from attacking the system. Additionally, IMs are external systems and therefore unbiased because they produce the schedule to proceed through the intersection instead of vehicles possibly making greedy decisions. However, if an attacker found a way to spoof into the system, there is no way to detect a sybil attack since there is no data validation.

We propose a Detection System as an additional way to advance safe operation. The Detection System will act as a supervisory system that can verify the behavior of vehicles before and inside the intersection. This external system will supplement IMs positioning data given by the CAVs with real-time, environmental sensing data. Additionally, this system will have the capability to validate if vehicles are communicating with the IM, if a vehicle is

actually entering the intersection (eg. sensing for a sybil attack), and ensuring the connected data aligns with the environmental data. In the event of an accident-prone situation, the IM can broadcast a warning to the CAVs interacting with the intersection. This solution can be applied to many intelligent intersection management schemes already implemented or in development, since it acts as an external agent whose primary focus is to identify positioning information (which can be extrapolated to velocity data using a simple distance versus time equation). This enables many IMs to have more fine-tuned positioning data and these management schemes can therefore make more informed decisions for scheduling vehicles through the intersections. The implementation of the Detection System was done by building intersection infrastructure and CAVs that are 1/10 scale.

2 Related Work

Many different intelligent autonomous intersection schemes have been proposed so far [3, 9, 12, 13]. We need to validate that the intersection management schemes are as robust, resilient, reliable, and redundant as possible.

A popular solution to this problem is known as the Autonomous Intersection Management (AIM)[7]. As vehicles approach the intersection at a constant speed, the vehicle sends a speed query to the IM and the IM replies with either a yes or a no to the request based on other requests it is managing for safe operation. If the answer from the IM is yes, the vehicle continues through the intersection at the requested constant speed; however if the answer is no the vehicle slows down and again queries the IM. Once the vehicle is through the intersection it would send a done signal to the intersection manager. This query-based approach worked well on both hybrid (both autonomously-driven and human-driven vehicles operating on the road) and completely autonomous intersections and it did not degrade the position uncertainty due to computational delay on the IM, however it increased network traffic and the amount of computation done on both the vehicle and the IM. In order to have safe operation, AIM relies on the accuracy of the data being communicated between the vehicles and the IM.

Another cutting-edge solution followed it, known as Crossroads [2], whose contribution was to have the IM calculate and assign the velocity of completely autonomous vehicles requesting to cross the intersection. This method when compared to AIM decreased network traffic and computational delay. Vehicles in this research were given a safety buffer which accounted for uncertainty in velocity and position. Network and computational delays are included in finding the round trip delay and this round trip delay is incorporated into the safety buffer. The main issue with this scheme was that it was prone to external disturbances and model mismatches (i.e. a bump in the road or wind). This is because Crossroads tracked a constant velocity through the intersection without accounting for the effect of position uncertainty resulting from these disturbances. The resulting position uncertainty leads to the possibility of inaccurate positioning data being communicated between the vehicles and the IM. This situation could eventually lead to accidents since the IM isn't aware of the vehicles' exact positioning.

The research which improved on Crossroads was Robust Intersection Management (RIM)[11], whose contribution was to track the trajectory of vehicles through the intersection. The premise was to assign a velocity of arrival (VOA) and trajectory of arrival (TOA) to vehicles who were about to enter the intersection. The tracked trajectory accounted for the time it would take the vehicle to speedup or slowdown its velocity to reach its respective VOA. This helped account for any position disturbances and therefore lead to a more accurate model for scheduling vehicles crossing the intersection. Though RIM has higher certainty of

7:4 A Dependable Detection Mechanism for IM of CAVs

vehicle positioning, it still relies entirely on accurate data transmission from the vehicles to the IM which can be error prone as seen in Figure 1.

Since all the researches above have one point of failure for obtaining positioning information, they are all prone to attacks which involve spoofing of location. Further, the system may fail if the communication between the vehicle and the IM fails. Whether a direct attack or lack of validation of connected data, any positioning inaccuracy can be detrimental to safe operation of the IM. To improve all these models, it is extremely beneficial to create redundant systems.

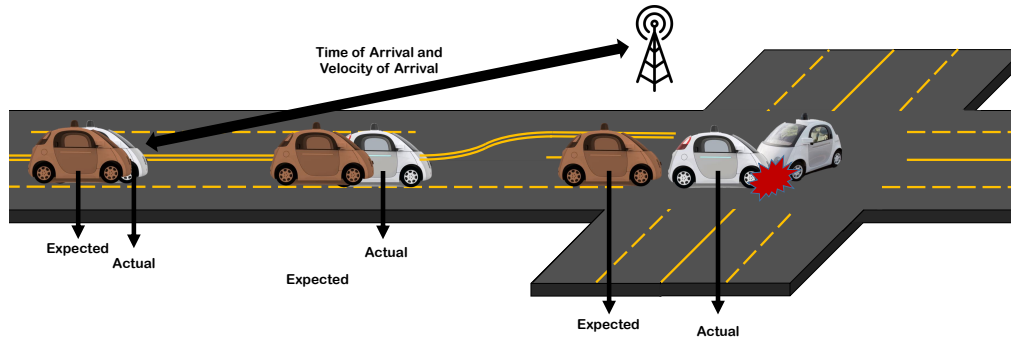


Figure 1 RIM Algorithm: A car will communicate with the IM to get a Time of Arrival (TOA) and Velocity of Arrival (VOA). However, it may not be able to meet the TOA and VOA and can cause an accident.

3 Proposed Method

The Detection System needs to be able to gain a robust understanding of the environment, and the solution implemented to find positioning data was image processing. It is important for this system to act independently from the IM to prevent interaction bias. To act independently, this system must do all processing externally from any of the IM researches, acting only as a reporting mechanism. The interaction model between the IM and Detection System uses I2C, where position determined by the Detection System is sent using I2C to the IM to be used for computation. Note that for security purposes, the Detection System is completely offline, so accessing the system can only be done physically. This adds a layer of data integrity which is dependent on the physical security to the Detection System and the cyber-security available to the IM. Because the Detection System contains four subsystems, there were four I2C communication points between the IM and the Detection System.

As described in Algorithm 1, the Detection System uses live video feed, parses the image using image processing, and alerts the IM using interrupts that a vehicle with a given identification is approaching the intersection. To do the image processing, the camera is calibrated, the image is normalized, and then the front bumper of the car is identified. Once the bumper is located in the image, the distance to the car can be found. This distance is transferred over I2C to the IM.

In Algorithm 2, the IM uses the calculated distance and identification provided by the Detection System to parse for specific packets being sent from the vehicles. Then the IM will process the environmental data and compare it to the connected vehicle data. During this comparison the IM can perceive four different scenarios: The first is that the image processing data doesn't match the connected data, the second is that a request is received but no vehicle is detected, the third is the Detection System recognizes a vehicle is approaching even though

Algorithm 1: Pseudocode for Camera platform.

```

1 for every  $\Delta t$  seconds do
2   Read Image;
3   Process Image ;
4   Send vehicles' data to IM;
5 end

```

the vehicle hasn't communicated with the IM, and the last is that the connected data is the same as the environmental sensing data. These scenarios are discussed in Section 4.

Algorithm 2: Pseudocode for IM.

```

1 Timer_ISR (every t seconds){
2   Get data from cameras;
3 }
4 if (A packet is received) then
5   Process the data;
6   if Scenario 1: Camera's data doesn't match the expected then
7     Send back a packet to the vehicle
8   end
9   if Scenario 2: A request is received but no vehicle is detected then
10    Change Policy e.g. MAC blocking
11  end
12  if Scenario 3: vehicle didn't communicate (A rouge driver) then
13    Broadcast an alert to other automated vehicles
14  end
15 end

```

4 Empirical Evaluation

The Detection System is beneficial to preserve high precision of position certainty for the IM. We implemented and tested with the four previously discussed scenarios in order to show the advantages of using the Detection System. Our testing has shown that in all four scenarios, the Detection System was able to observe and report the vehicle's location within 2 cm accuracy.

4.1 Testbed

4.1.1 The components

The Detection System created in this research involved four separate, equivalent subsystems to work together. Each subsystem's purpose was to monitor a given lane of traffic. Given a typical intersection, there are four lanes of traffic therefore four subsystems were needed for testing of the Detection System. Each subsystem was built using a Raspberry Pi 3 Model B as the processing agent, running operating system Raspbian Sketch 4.14. A Raspberry Pi V2 Camera was used for capturing the visual input to the subsystem. OpenCV 2.4.9.1 in

Python 3.5 was used for image detection. Note the interaction of all these components in Figure 2.

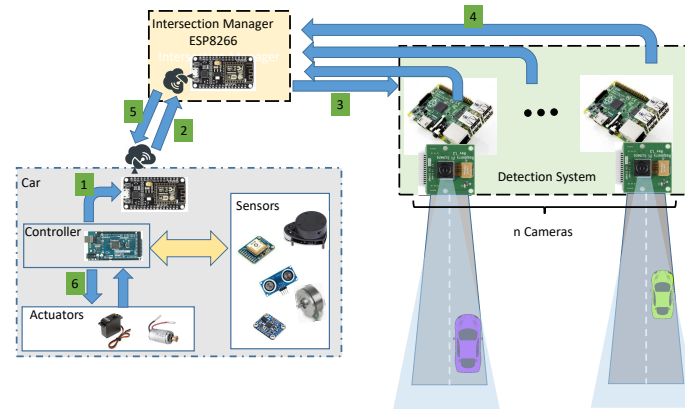


Figure 2 The IM, Detection System and vehicle interact with each other in five steps to verify the car data using the environmental sensing data.

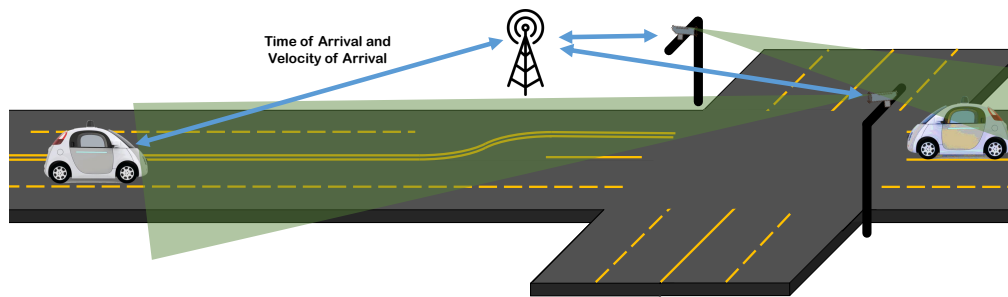
4.1.2 Chosen IM

In order to test the Detection System, a version of RIM was used for the IM. RIM iterated on both AIM and Crossroads, and has the most accurate model for positioning data. By using the model with the most accurate positioning, the potential of the Detection System can be seen more clearly.

Briefly, RIM divides the action of a vehicle approaching the intersection into four phases. Initially, the vehicles synchronize their clocks with the IM and pass a physical synchronization line. If the synchronization is not successful the vehicle tries to synchronize again. If the synchronization is successful, then when the vehicles pass the physical transmit line they send to the IM a packet containing position, velocity, acceleration, timestamp, outgoing lane, minimum and maximum acceleration, and identification. If this transmit is unsuccessful, the vehicle slows down and tries again. If this transmit is successful, the IM calculates a trajectory of arrival and velocity of arrival for the vehicle based on the packet sent by the vehicle and the respective scheduling algorithm being used to control the vehicles which are currently interacting with the intersection. Then the IM sends this packet back to the requesting vehicle, meanwhile the vehicle is keeping a constant velocity headed toward the intersection until it receives further instruction. Finally, the vehicle creates a reference trajectory from the received packet from the IM and follows it through the intersection[11].

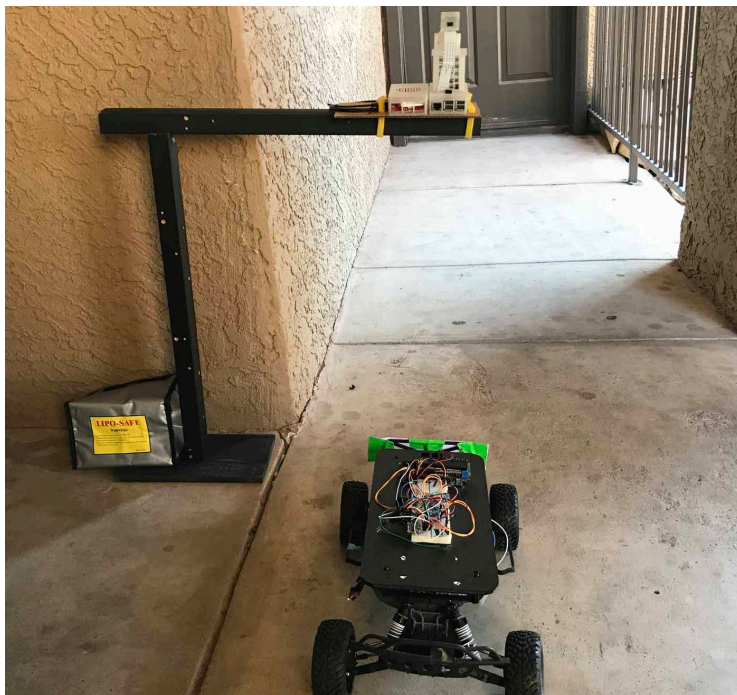
Using RIM described above as the basis for the IM, the Detection System was incorporated into a variant algorithm for RIM which for our purposes will be called RIM Robust. RIM Robust varies from RIM after phase two finishes. In phase three after the vehicle transmits its data to the IM, the IM verifies that the transmitted data is the same as the environmental sensing data from the Detection System.

To implement RIM Robust as seen in Figure 3, the first step was to build a model intersection with model CAVs. The vehicles were built using TRAXXAS RC Car with onboard ESP8266 boards for communication with the IM, which also used an ESP8266 board. All boards were designed using the Arduino software suite. A wooden arch was crafted for the infrastructure to support the Raspberry Pi and Camera above the 1/10 scale model of



■ **Figure 3** RIM Robust has a surveillance Detection system that monitors the position of vehicles and reports it to the IM.

intersection using TRAXXAS RC car. Given that the average traffic light is approximately 18 ft. to 22 ft. in height, our scaled, wooden infrastructure's height was 26 inches tall. The infrastructure of the model intersection can be seen in Figure 4.

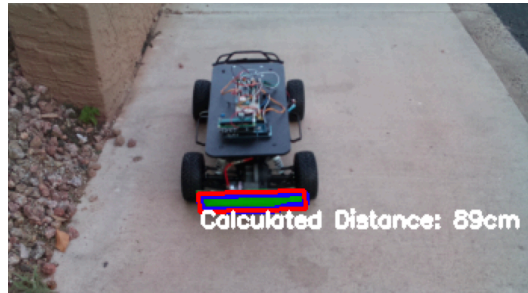


■ **Figure 4** At one corner of the intersection, you can see one of the four subsystems of the Detection System mounted on top of the wooden infrastructure with a TRAXXAS RC Car about to drive underneath.

4.1.3 Image Processing

For image processing, the camera first had to be calibrated. It is important to note the field of view (a restriction to what is visible in the frame[1]) because it is directly related to the focal length (the distance from the focal point to the vertex of the first optical surface[10]) which is needed to understand distances in a picture. Note that depending on the angle at

which the camera is pointed at the intersection, different focal lengths will result. Once the camera is calibrated and the focal length is found, the images read in could be normalized for image processing and the front bumper of the RC cars can be more easily detected. Once the bumper was detected, trackers were created for each RC car[8] which could be used to find distance by comparing with a base set of images. This base set of images defined the ratio of pixels to distance, so by comparing an identified pixel to the base set you can estimate the distance as shown in Figure 5.



■ **Figure 5** Detect the front bumper use the pixels to approximate actual distance from the camera to the vehicle.

4.1.4 Preliminary results

In Figure 6, the chart shows a comparison between the vehicle self-reporting error and the Detection System error. The Detection System has an error of approximately 2cm on average for true distance compared to actual distance. This uncertainty could be introduced by error in the perception algorithm itself. This error is small, though in a real-world scenario noisy perception data is likely and thus this methodology to obtain data will need to be tuned to any given environment. The results from testing show that safety of the intersection is increased because the likelihood of an accident is minimized, and the total throughput is increased because there are no accidents that result (where accidents cease throughput of cars through the intersection). Throughput here is a measure of managed vehicles divided by total wait time.

4.2 The Benefits of the Detection System

Analyzing RIM Robust, four scenarios may occur shown briefly in Algorithm 2. Note that the positioning data reported by the CAVs is prone to higher amounts of error compared to the Detection System, which has positioning data that is not prone to external disturbance/misreporting and therefore the Detection System's data overwrites the CAVs' data. In the first scenario, the environmental data is different from the reported data from the vehicle. This scenario requires the IM to go into recovery mode and update the vehicle's information packet to what is observed from the environmental data. The second scenario is that a request is received by the IM from a vehicle but the requesting vehicle is not detected by the Detection System. This implies an invalid request is coming from an attacker or an error occurred, therefore mark this user as an invalid user by implementing methods such as MAC blocking. This scenario requires that RIM Robust's policy will be updated so as to handle attackers who are trying to slow throughput or cause harm. The third scenario is that the Detection System finds a vehicle entering the intersection which didn't communicate.

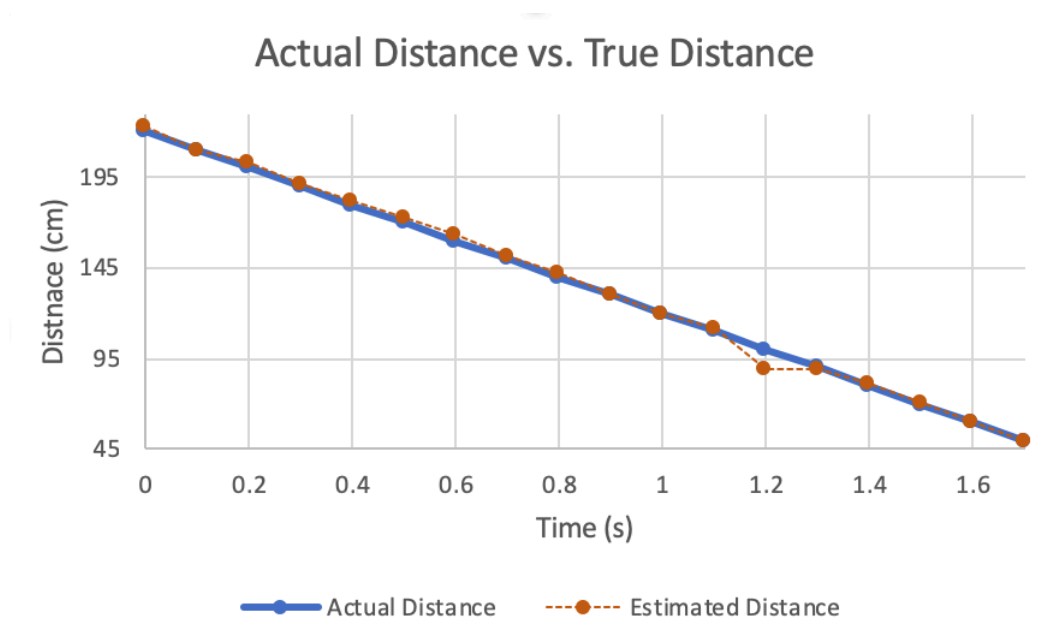


Figure 6 Observe the difference in position from the known distance to the calculated distance by the Detection System during one run of the program.

This scenario again requires RIM Robust's policy will be updated to handle a rouge driver attack, where a vehicle outside the system maliciously goes rouge. Note the interactions for the three error scenarios described here are depicted in Figure 7. Finally in the last scenario, the vehicle's data matches the Detection System's data. This requires no change to the operation of RIM and so the algorithm continues as described in RIM.

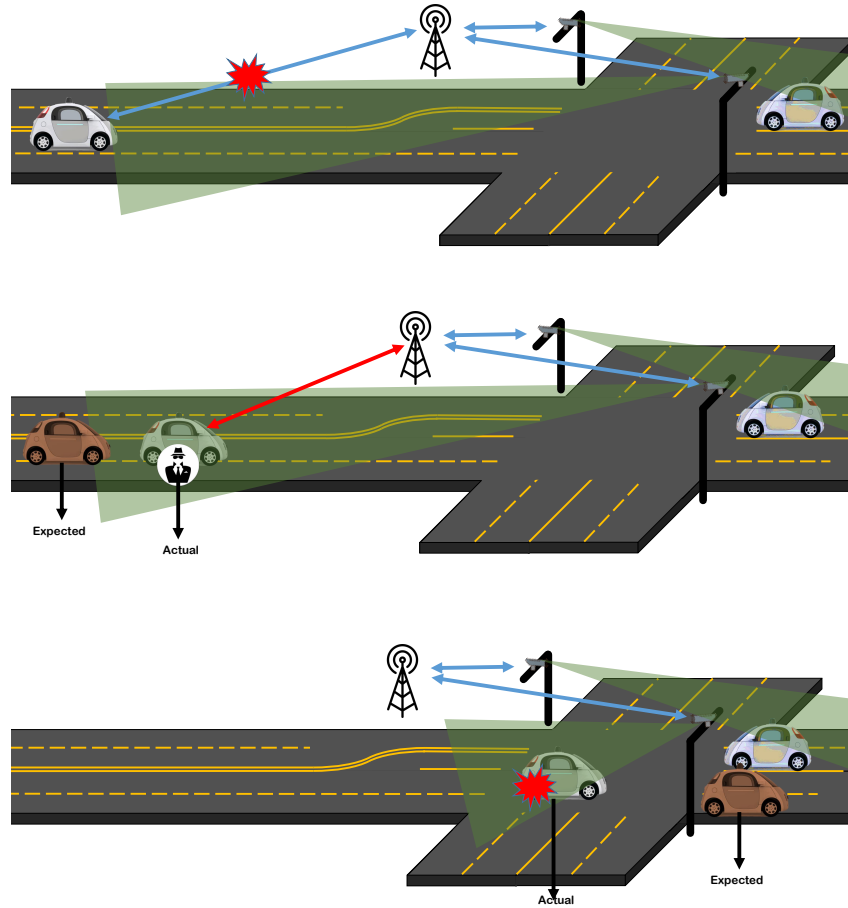
In order to test this system, RIM Robust was implemented using recovery mechanisms such as alerting CAVs of potential attacks, MAC blocking, and updating vehicle's information.

4.3 Sybil Attack case study

4.3.1 Theory

The Sybil attacker uses false data(eg. ID) to authenticate itself and sends a request to the IM for passage through the intersection[4]. The attack would cause the IM to schedule a vehicle which isn't actually present in the intersection with the intention of decreasing throughput through the intersection[6]. This leaves the intersection vulnerable in high traffic situations. To control such situations, the Detection System can monitor the traffic and determine whether or not the vehicle exists or not. If there is a vehicle detected using the environmental sensing data, the Detection System goes a step further and identifies the vehicle using the identification characteristics such as license plate. If the identification packet sent by the vehicle contains a matching license plate, the vehicle can continue communication with the intersection. If the vehicle does not match any existing, approaching vehicles in the intersection the vehicle is label as a Sybil attacker and is marked as an invalid user in the intersection using MAC blocking.

A large scale sybil attack could result in denial of service to valid vehicles entering the intersection because the sybil attacker is jamming the IM network with requests. To explain this in more detail, in a typical traffic situation there is a worst case time it takes a vehicle to go through an intersection; this would be in fast, high traffic situations. By simulating this traffic situation, the worst case downtime between requests can be approximated. The



■ **Figure 7** Three beneficial scenarios of the Detection System (from top to bottom): when the vehicle's communication systems are malfunctioning, when an attacker claims there is a vehicle nearing the intersection when there is not, and when there is a discrepancy between the connected data and environmental detection data.

worst case downtime is representative of the time interval between requests in the case where the maximum number of requests are being made in a short period of time, and therefore the worst case downtime is the smallest time interval that can occur between requests in high traffic situations. If the interval were any smaller than the determined worst case downtime it would mean that vehicles were physically overlapping which is impossible.

4.3.2 Implementation

In this experiment we used one ESP8266 microcontroller board for simulating the attacker. In the event that a sybil attack is sensed, based on the identification and information sent in the packet from the vehicle, the Detection System verifies that in the respective outgoing lane there is a vehicle that matches the identification information. If such a vehicle doesn't exist, the first step is to alert CAVs in the intersection to a potential threat and to proceed with caution. The next step is to get the MAC of the malicious requesting vehicle and use the MAC address to block it from interacting with the IM and other CAVs on the network.

4.3.3 Results

It was observed that as the number of these sybil attacks increases, the throughput of the intersection decreases drastically in RIM because RIM schedules all vehicles that make a request. However using RIM Robust, the vehicles are tracked and if a vehicle that is communicating with the IM isn't detected, it is blocked from the intersection; therefore, the only vehicles scheduled are those physically present at the intersection and there is no throughput degradation.

4.4 Vehicle failure

4.4.1 Theory

If vehicle fails to communicate with the IM, whether it be that the communication systems aren't functional or possibly the vehicle doesn't have enough power for communication, the intersection should be able to manage traffic. A distinction should be made here: a vehicle failure is not a latency issue in communication between IM and CAV. In the scenario of latency in communication and the vehicle is too close to the intersection, the vehicle would slow to a stop if necessary to initiate communication. However, in the situation where vehicles don't have the ability to communicate at all, the Detection System should be able to identify them once they are within bounds of entering the intersection.

4.4.2 Implementation

When the Detection System identifies a vehicle which has crossed the synchronization line and hasn't communicated with the IM yet, RIM Robust can plan its scheduling accordingly. Two cases may occur if the vehicle doesn't communicate with the intersection. The first is if the vehicle comes to a stop outside of the intersection. In this case vehicles already scheduled proceed normally and vehicles which are communicating to receive a schedule slow down. These vehicles wait for a notice from the IM, whose job is to observe the location of the vehicle. The second case is if the vehicle doesn't stop before reaching the intersection. In this case the car continues through the intersection without communicating to the Detection System. For this situation, the vehicles already scheduled proceed with caution and vehicles which are communicating to receive a schedule slow down until the car has passed through the intersection. Once the vehicle which is not communicating exits the intersection, normal operation proceeds. For this case study, TRAXXAS RC cars went through the intersection without communicating with the IM. This was done to observe if the Detection System recognized that cars were present in the intersection and prevent a potential accident.

4.4.3 Results

This is a necessary precaution for managing vehicles who cannot be scheduled. The broken-down vehicle situation eliminates a lane of traffic, which is extremely harmful to managing traffic properly. This case study didn't implement managing this situation beyond alerting CAVs to the situation. In the other scenario of this case study where a vehicle proceeds through the intersection without communication, it is assumed the vehicle which cannot communicate with the IM is either a potentially malicious vehicle or it is a vehicle whose communication is down. Both are a threat to safe operation, and due to the unpredictable nature of this vehicle it is safest to clear the intersection of all vehicles and let this vehicle make its decision (pass through the intersection) before normal operation begins. Comparing

throughput in this scenario, RIM may cease safe operation if a rogue car causes an accident but RIM Robust ensures no accident will occur so traffic will continue normally after the rogue car is handled.

4.5 Reporting error

4.5.1 Theory

The vehicle may fail to maintain the expected trajectory of arrival or velocity of arrival, and this failure can cause a position error greater than the threshold set for an acceptable safety buffer. This leaves the intersection vulnerable to collision, so to validate this situation the Detection System monitors the vehicles' positioning data and ensures scheduling doesn't violate any safety buffer boundaries.

4.5.2 Implementation

RIM Robust was tested for this scenario by hard-coding an error in the reporting mechanism of the TRAXXAS RC Car to be plus or minus one meter from a given position. The role of the Detection System is to monitor the actual position of the vehicle and report this to the IM. If the IM recognizes a discrepancy between the reported value from the vehicle and the observed value from the environmental sensing data the IM defaults to using the data provided by the Detection System and updates both the vehicle's information and the scheduling algorithm inputs.

4.5.3 Results

No safety buffers were violated in the RIM Robust. Violation of safety buffers may lead to accidents in RIM and halt operation of the intersection entirely, but this scenario won't occur using RIM Robust.

5 Conclusion

As processes for driving continue to become more automated, it becomes pertinent to ensure safe passage during traffic. Intersections become an area of concern for safe passage, as this is the most complex area of traffic to navigate given the infinite scenarios that can occur. IMs are typically a great way to increase efficiency of traffic, but ultimately IMs are unsuccessful if they cannot direct vehicles safely in all scenarios. Many researchers using IMs to conduct traffic have one source for position reporting and therefore lack redundancy. The advantage of using the Detection System is the high level of position certainty that can be obtained. Adding the Detection System to an IM makes the intersection more resilient to attack or inaccurate reporting because it increases the reliability of the system by duplicating positioning data sources. By relaying the information obtained by the Detection System back to the IM, and combining it with the connected data between vehicles and the IM, a more accurate portrayal of the intersection is obtained and vehicles can react safely to even more scenarios than was possible before.

References

- 1 Patricia L Alfano and George F Michel. Restricting the field of view: Perceptual and performance effects. *Perceptual and motor skills*, 70(1):35–45, 1990.

- 2 Edward Andert, Mohammad Khayatian, and Aviral Shrivastava. Crossroads - A Time-Sensitive Autonomous Intersection Management Technique. In *Proceedings of The 54th Annual Design Automation Conference (DAC)*, 2017.
- 3 Tsz-Chiu Au and Peter Stone. Motion Planning Algorithms for Autonomous Intersection Management. In *Bridging the Gap Between Task and Motion Planning*, 2010.
- 4 Karl C Bentjen. Mitigating the Effects of Cyber Attacks and Human Control in an Autonomous Intersection. Technical report, AIR FORCE INSTITUTE OF TECHNOLOGY WRIGHT-PATTERSON AFB OH WRIGHT-PATTERSON AFB United States, 2018.
- 5 Qi Alfred Chen, Yucheng Yin, Yiheng Feng, Zhuoqing Morley Mao, and Henry Xianghong Liu. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS'18)*, San Diego, CA, February 2018.
- 6 John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.
- 7 Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.
- 8 Mohsen Kheirandish Fard, Mehran Yazdi, and Mohammadali MasnadiShirazi. A block matching based method for moving object detection in active camera. In *Information and Knowledge Technology (IKT), 2013 5th Conference on*, pages 443–446. IEEE, 2013.
- 9 Chien-Liang Fok, Maykel Hanna, Seth Gee, Tsz-Chiu Au, Peter Stone, Christine Julien, and Sriram Vishwanath. A Platform for Evaluating Autonomous Intersection Management Policies. In *Proceedings of IEEE/ACM 3rd International Conference on Cyber-Physical Systems*. IEEE Computer Society, 2012.
- 10 John E Greivenkamp. *Field guide to geometrical optics*, volume 1. SPIE Press Bellingham, WA, 2004.
- 11 Mohammad Khayatian, *et. al.* RIM: Robust Intersection for Connected Autonomous Vehicles. In *RTSS*. IEEE, 2018.
- 12 Joyoung Lee and Byungkyu Park. Development and Evaluation of A Cooperative Vehicle Intersection Control Algorithm under The Connected Vehicles Environment. *IEEE ITS Transactions*, 2012.
- 13 Florent Perronnet, Abdeljalil Abbas-Turki, and Abdellah El-Moudni. Cooperative Intersection Management: Using Mini-Robots to Compare Sequenced-based Protocols. *Les Journées Nationales des Communications dans les Transports (JNCT)*, 2013.

TrueView: A LIDAR Only Perception System for Autonomous Vehicle

Mohammed Yazid Lachachi¹

University of sciences and technologies d'Oran - Mohamed-Boudiaf, LMSE, Algeria
yazid.lachachi@univ-usto.dz

Mohamed Ouslim

University of sciences and technologies d'Oran - Mohamed-Boudiaf, LMSE, Algeria
ouslim@yahoo.com

Smail Niar

Université Polytechnique Hauts-de-France, France
smail.niar@uphf.fr

Abdelmalik Taleb-Ahmed

Université Polytechnique Hauts-de-France, France
Abdelmalik.Taleb-Ahmed@uphf.fr

Abstract

Real time perception and understanding of the environment is essential for an autonomous vehicle. To obtain the most accurate perception, existing solutions propose to combine multiple sensors. However, a large number of embedded sensors in the vehicle implies to process a large amount of data thus increasing the system complexity and cost. In this work, we present a novel approach that uses only one LIDAR sensor. Our approach enables reducing the size and complexity of the used machine learning algorithm. A novel approach is proposed to generate multiple 2D representation from 3D points cloud using the LIDAR sensor. The obtained representation solves the sparsity and connectivity issues encountered with LIDAR-based solution.

2012 ACM Subject Classification Computing methodologies → Computer vision representations

Keywords and phrases Ranging Data, Computer Vision, Machine Learning

Digital Object Identifier 10.4230/OASICS.ASD.2019.8

Category Interactive Presentation

1 Introduction

Camera, Stereo-camera, RADAR, and LIDAR are the sensors that provide our view of the world. To enable machines to understand our world, multiple approaches have been developed. Camera-based systems are the most dominant [2, 19, 3]. Recently, approaches using depth information generated from stereo-camera have started to gain popularity, as they have proven to improve accuracy. Knowing that camera and stereo-camera are almost the same since the stereo-camera enable the recovery of depth information compared to the monocular camera. Similar Machine Learning (ML) algorithms for multi-class object detection and segmentation are used with minor modifications. On the other hand, the LIDAR has fallen behind performing mainly detection tasks [20, 17, 14] as the variance in the points coordinates ease the task.

The richness and variation of data fed to an ML determine its complexity and size. These two factors have a direct impact on network accuracy. Images are well structured and rich in information, as each pixel has a non-zero value and adjacent to another pixel on a 2D plane.

¹ Corresponding author



However, this richness adds more challenges like illumination conditions, shadows, colors, and textures. All these parameters require an ML architecture to have more layers to capture abstract aspects like car tires and window edges. A large amount of filters is also needed to accommodate the infinite changes that we can obtain like car shapes and the possible textures of an object. The LIDAR is an active sensor capable of operating in the majority of weather conditions and at any daytime. Unfortunately, due to its rotatory movement, the obtained points are unstructured, unordered and have no connectivity between them. These weaknesses make it difficult to use for multiple purposes when compared to the camera sensor.

In this work, we propose a new approach to generate multiple 2D representations using only a 3D LIDAR. These representations reflect the real world and are immune to camera limitations. These representations allow us to see the silhouette of objects without the added textures, colors and illumination condition. Some of the generated representations are an up-sampled version of the ranging data projected into an image, where the new values are filled using interpolation. Further, the obtained representations simplify the ML architecture and reduce its size thus gaining in processing time and getting closer to real time.

Our Contributions are as follow:

- Novel range data up-sampling process to enable camera-based approaches to be used with LIDAR data.
- Robust normal map estimation.
- We present a new approach to reducing the complexity of current machine learning algorithms and their computation overhead.

In Section 2, we present the related work to up-sampling the ranging data, then we introduce the concept in Section 3 and method to generate the representation in Section 4. We then conclude the paper by presenting the benefits of the proposed representations for machine learning algorithms.

2 Related Work

The first attempt to generate dense information from range data which corresponds to LIDAR points cloud in our case was made by Diebel et al [4]. The authors proposed a range of data and image fusion to generate a dense depth map from a low-resolution one. They fed the projected range data with the image to Markov Random Field regressor, then they generate the high-resolution depth map by iterating through the MRF. Their experiments showed that textured surfaces had a negative impact on the estimated values as color value changed.

In [1], Andreasson et al. proposed 5 interpolation methods as a replacement to the MRF proposed by [4]. The nearest range (NR) and multi-linear interpolation (MLI) are color free estimations. The color was considered in the two modified version NRC and LIC. These approaches relied on an empirically predefined constant and parameter-free version of LIC was further introduced as PLIC. Their results in a laboratory environment under controlled illumination has proved that color improved accuracy. However, experiments in real conditions showed that color had the opposite effect. This limitation is caused mainly by variant illumination conditions.

In [10], Yuhang et al proposed a new approach to generate depth, height and a normal map from the range data and image. They used different features to determine the value of the new pixels. The features were used to minimize the effects of textured surfaces and illumination conditions on the final result. However, their result accuracy varies according to the used window size. These representations have proven their effectiveness in the work of [9].

Further, the KITTI benchmark recently launched a competition to generate the depth map from the LIDAR data, when used solely or in conjunction with a camera. This competition saw the introduction of machine learning algorithms, where the best at the time of writing this paper is [15]. The authors considered the up-sampling task as a deep regression problem. They feed depth data and color sequences to the regression network.

Ku et al [13] proposed a simpler approach to up-sample the range data, by using a computational fast process like morphological dilatation, filling and blurring. They were able to generate the depth map in 11ms on a CPU and their results were comparable to results obtained by convolutional neural networks.

In [16], Schneider et al. propose to use the edge information to guide the up-sampling. The authors considered the up-sampling as a global energy minimization problem. Although the approach was developed for time-of-flight cameras, the authors were able to extend it to range data and compete in the KITTI depth completion.

Dimitrievski et al [5] proposed a morphological neural network, their approach approximates morphological operations using a novel Contraharmonic Mean Filter layers. The proposed network is modified U-net architecture with morphological layers.

Multiscale networks have produced very interesting results in the KITTI challenge. In [11], Huang et al proposed a hierarchical multi-scale network, they introduced three sparsity-invariant operations. These operations were used to create a sparsity-invariant multi-scale encoder-decoder network. The method was developed to deal with the sparsity problem in the range data generated by LIDAR.

Although these approaches are able to generate dense depth map representations and more representations in some of them, the following problems arise:

- A degraded image quality is obtained under unfavorable illumination conditions.
- A large number of points must be used for the estimation.
- The whole process must be repeated much time to obtain a different representation. This factor increases the processing time with the number of generated representations.

Existing approaches use a window to retrieve the close points for the estimation. A wide window utilization could augment greatly the number of used points, thus generating false estimations at high variance regions. Furthermore, the number of points cannot be reduced due to adapt the algorithm to the hardware processing constraints. In this work, we consider a newly generated pixels as part of a triangle surface, where the value is an interpolation between three closest points. This approach removes the need for a fixed window and allows the generation of the representations without the need to repeat the process.

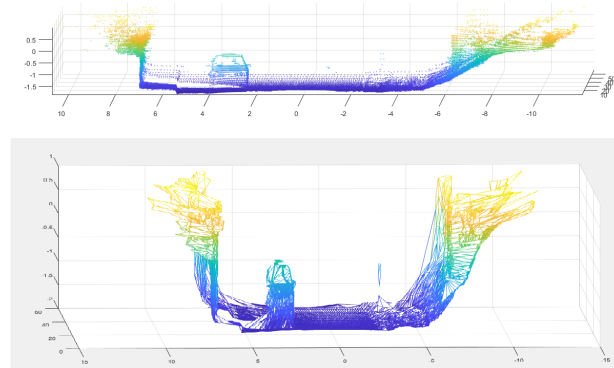
3 Optimized 3D representation for LIDAR data

Real objects are in general modeled as 3D model mesh consisting of 3D points and their connectivity list. In this work, we propose to generate a 3D mesh from the LIDAR ranging data. The high accuracy of this data enables the creation of a mesh that mimics the environment of the vehicle. Greater control over desired the desired output resolution is then possible. This obtained mesh enable the generation of a set of representations, which we formulate as follows:

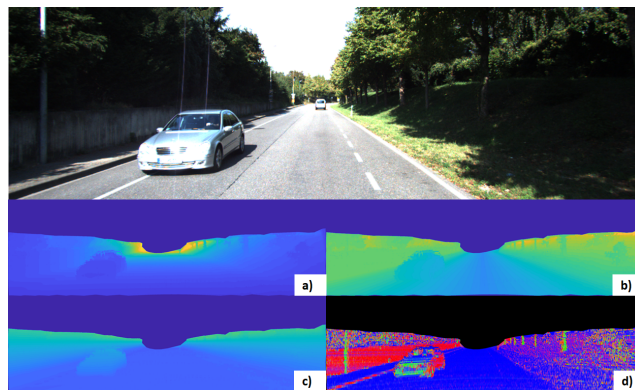
$$\mathbb{R}^3 \equiv \{\mathbb{R}^2, \mathbb{R}^2, \mathbb{R}^2, \dots\} \quad (1)$$

Figure 1 presents a 3D mesh obtained by our approach and Figure 2 shows the multiple obtainable representations. The Figures 2.b to 2.d representations show (respectively) the rendered X, Z, Y and the reflection of the surface. These representation using only the mesh

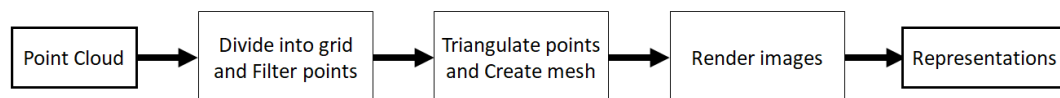
8:4 TrueView: A LIDAR Only Perception System for Autonomous Vehicle



■ **Figure 1** Reconstructed mesh from point cloud.



■ **Figure 2** Rendered representations (from top to bottom) a) Initial image, b) up-sampled x coordinates, c) Up-sampled z coordinates, d) Up-sampled y coordinates, e) Reflection data, f) Estimated normal map (final result).



■ **Figure 3** Overall Algorithm.

and points coordinates in the real world. The last representation is a 3-channel image, where each channel contains one of the normal parameter estimated from the three points making the surface. In the last representation (Figure 2.f), the coordinates of the points and their connectivity were used to generate the results.

Further results can be acquired, such as object contours, using the mesh connectivity list and the angle between the points. This last feature is very interesting as it allows to obtain the contours with a relatively reduced processing. In the next section, we present our method to obtain the connectivity list.

4 Method

A 3D mesh is a virtual construct composed of a 3D point set and their connectivity list. The LIDAR generates an accurate point cloud from its environment. These points can be considered as the first half of the mesh. However, the rotatory movement, the angular speed, the number of receptors and the used algorithm have a great impact on the number of points. These factors have an important variance in the possible results obtained.

The steps to generate the representation are shown In Figure 3. We bring the reader attention to the fact that all the steps can be done in parallel.

4.1 Division and Filtering

Point clouds are unordered and have no connectivity by nature. Attempting to generate a mesh in this format will be time-consuming. A simpler and effective approach is to process the points in 2D space. This allows to reduce the complexity of the algorithm and gives faster processing compared to 3D space representation.

The projected point number can be also be reduced to accelerate the processing. Our experiments show that reducing the number of points will not greatly reduce the quality of the results. This point will be discussed in the experiments section.

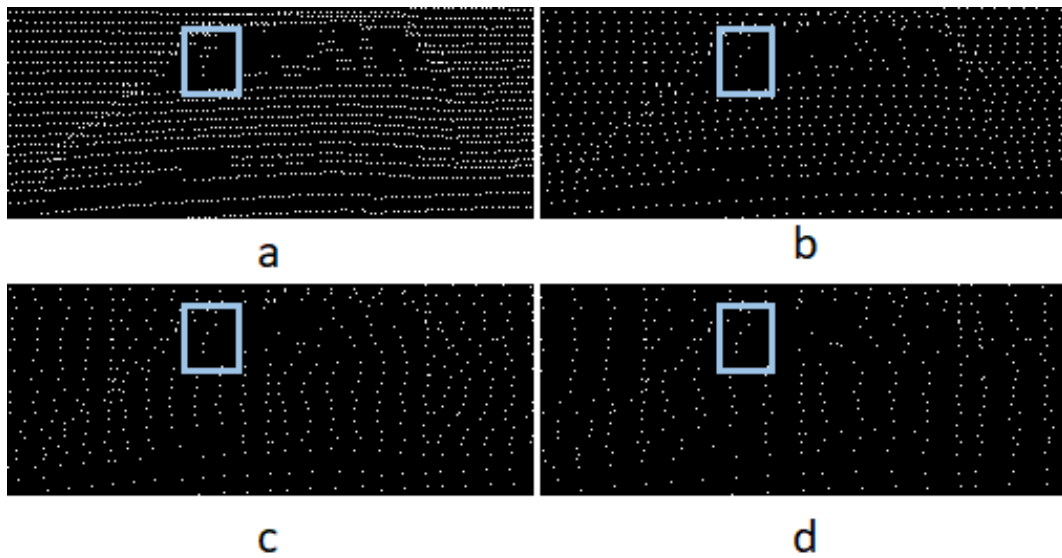
After the projection, the points are divided into a grid with equal dimensions. The number of points in each grid cell is then reduced based on distance criteria. This distance controls the number of points used to create the mesh. In this work, the Manhattan distance is used for its computation simplicity. In Figure 4, reduced points with different distances are shown. We attract the reader attention to the blue rectangle. As can be seen, the details of the scene are preserved even with a big distance.

4.2 Triangulation and mesh creation

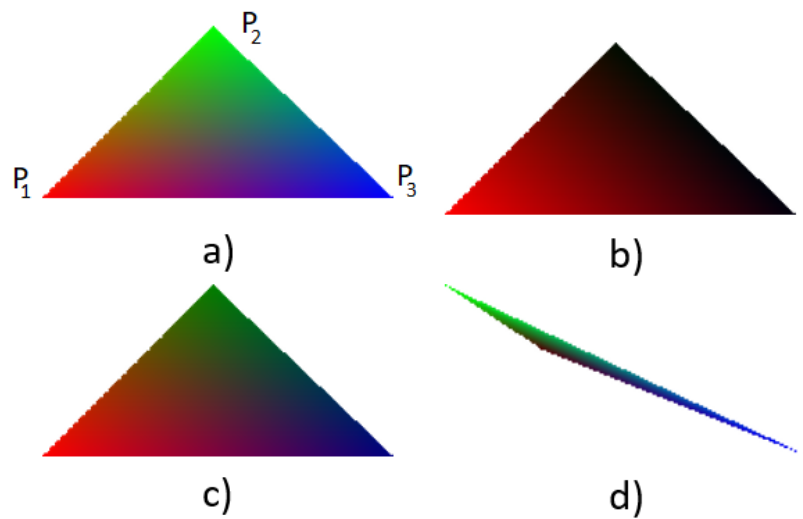
Delaunay triangulation is used to create the connectivity list between the points. Multiple variants of the algorithm exist from which we denote: Flip algorithms [12], Incremental [8, 7] and Divide and Conquer [6]. The Divide and conquer variant is chosen in our approach for two main reasons. First, generating these representations have to be as fast as possible. The work of Su et. Al [18] proved that this variant is the most performing approach. Secondly, the algorithm has been developed with GPUs in mind. The points in the grid cells can be processed separately and in parallel then merged at the end.

4.3 Rendering

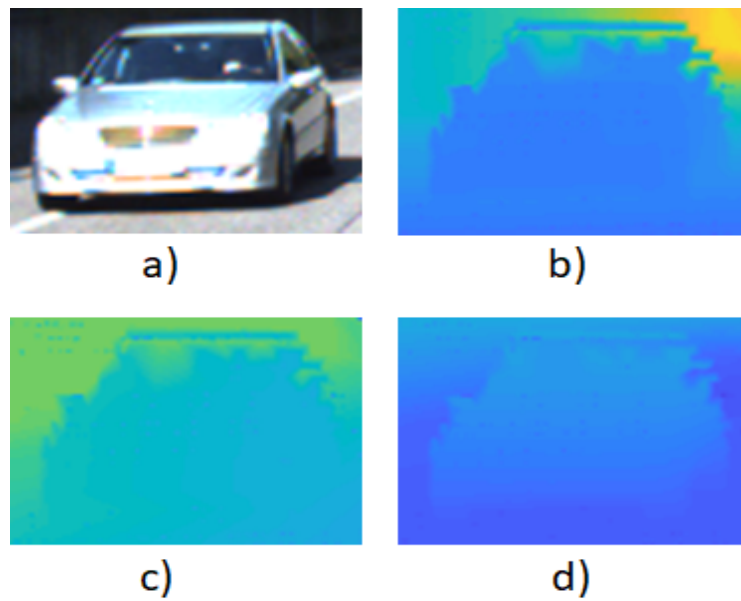
The rendering step is where the up-sampling is performed. For each triangle, the value and positions of the delimiting points are used to interpolate the new values at rendering time. Figure 5 presents four examples of rendered triangles.



■ **Figure 4** Ranging data projected after filtering with different distances: a) 1 pixel, b) 5 pixels, c) 10 pixels, d) 15 pixels.



■ **Figure 5** Samples of rendered triangles mimicking possible interpolations cases.



■ **Figure 6** Up-sampled ranging data: a) the original image, b ,c and the respective up-sampled x, y and z coordinates.

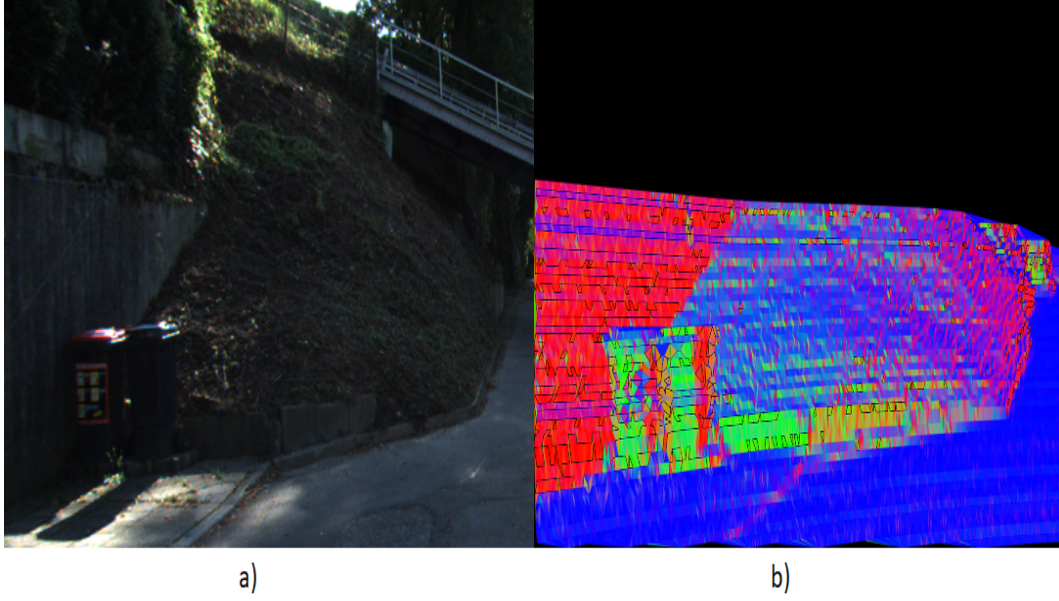
Each example in Figure 5 represents a possible case.

In this example P1, P2, and P3 are the delimiting points of the triangle, each point has a different color to mark its contribution to the interpolation. In 5.a), the three points have the same value to simulate points on the same surface. The three points contribute equally to the interpolation. In the case of 5.b), P2 and P3 are 0.05 the value of P1 to simulate the case of a distant point that is connected to close ones, it can be observed that P2 and P3 do not contribute. Case 5.c) simulates the case where the three points belong to the same object with a slight difference in the value. We bring the reader's attention to the differences between this case and case In 5.a) for comparison. Case In 5.d) present a randomly shaped triangle.

5 Improving ML

Machine Learning (ML) based algorithms is becoming popular where a large set of data must be processed in a reduced time period. The three most researched tasks in ML are classification, detection, and segmentation. With the proposed representations in this paper, detection and segmentation tasks are targeted for AV. We present two possible improvements to accelerate and train machine learning algorithms.

Detection and segmentation algorithms are used to extract object shape, delimiters, and position from the provided images. In contrast to existing classification algorithms that have to differentiate between two visually different instance of the same object. We propose to use the generated representations as a replacement to camera provided images. The idea is that object in a scene can be detected using their silhouettes instead of their texture. Figure 6 shows the image of a car and the corresponding x, y and z representations. The reduced visual complexity produced by textures, illumination, and shadows allows the reduction of filters number inside layers, thus compressing the size of the model and accelerating the processing time.



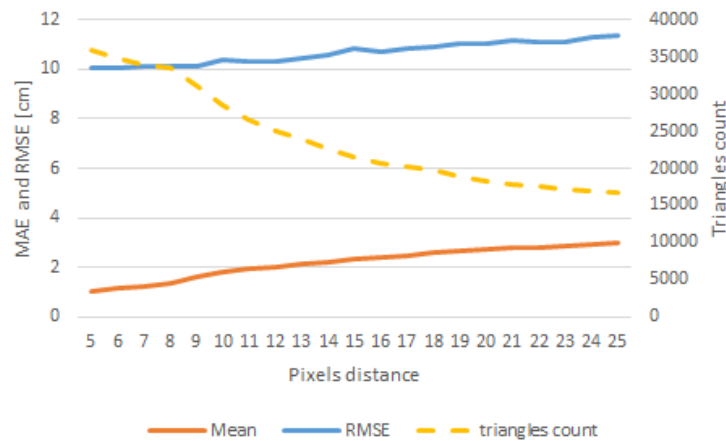
■ **Figure 7** Generated normal map for a complex environment: a) The original image b) The generated normal map.

Furthermore, the generated normal map enables the clustering of points into surfaces and ease the differentiation between the multiple objects in a scene. In Figure 7 we present a scene that contains the normal that can be found in a complex environment.

As a second improvement, we propose to train ML algorithms using synthesized representations using modeling software. In fact, the representations do not require any realism and can be used in real-world applications.

6 Experiments

To evaluate the accuracy of the generated meshes, a ground-truth mesh is needed. However, this information is unavailable, as an alternative, the evaluation was carried using the generated representations. A 100 scene was chosen at random from different sets, and for each scene we generated the X, Y and Z representations with filtering distances in the range [5 - 25] pixels between the points, we found through experiments that a distances less than 5 pixels will create small triangles that cannot be rendered in the flowing step. The results are compared using the Root of Mean Squared Error (RMSE), Mean of Absolute Error (MAE) to the ranging data, and the number of triangles. The RMSE will give an insight on the standard deviation of interpolation error, whereas the MAE will reflect the common error value. Finally, the number of triangles in the mesh present us with the impact of rendering on the hardware. In Figure 7, the MAE and RMSE metric is plotted for the height (Z) representation, the results are in relation to pixels distance between projected points, in addition to triangles count. The result shows that with a distance of 5 pixels introduces a 1 cm absolute error and a 10 cm deviation compared to 3 cm absolute error with 25 pixels distance, it can be observed that standard deviation is only about 2 cm more for 25 pixels distance for 45% of the triangles counts. Thus, enabling the possibility to choose a balance between the desired accuracy and the dedicated processing power or processing time.



■ **Figure 8** The different fusion architectures.

■ **Table 1** Interpolation error MAE and RMSE in meters.

Error Metric per coord		Filtering distance in pixel			
		5	11	19	25
X	MAE	0.160	0.320	0.443	0.498
	RMSE	1.857	1.902	2.053	2.119
Y	MAE	0.045	0.088	0.153	0.168
	RMSE	0.763	0.768	0.807	0.824
Z	MAE	0.009	0.019	0.026	0.029
	RMSE	0.100	0.102	0.109	0.113

In Table 1, the interpolation error is presented, in relation to the distance by which the number of points is reduced. From the table, it can be observed that the error is strongly connected to the range. For example, values for the X coordinates range from 0 to 60 meters, where the mean absolute error increase from 16 cm to about 50 cm the more the filtering distance increase, which is not the case for the Z coordinates that range from -2m to 1m relative to LIDAR position.

7 Conclusion and Future Works

This paper presents an ongoing work on generating multiple representations from LIDAR ranging data. Our aim is to introduce a novel approach to reduce the size of ML architectures and augmenting the training set. Currently, we are implementing the generation of the representation to run in parallel on a GPU. In the next steps, we will evaluate the impact of reducing the number of points on the processing time and results in accuracy. We will also validate our hypothesis of compressing ML architecture by testing it on a variety of detection and segmentation architectures.


References

- 1 Henrik Andreasson, Rudolph Triebel, and Achim Lilienthal. Non-iterative vision-based interpolation of 3D laser scans. In *Autonomous Robots and Agents*, pages 83–90. Springer, 2007.
- 2 Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, pages 2040–2049, 2017.
- 3 Zhe Chen and Zijing Chen. RBNet: A Deep Neural Network for Unified Road and Road Boundary Detection. In *International Conference on Neural Information Processing*, pages 677–687. Springer, 2017.
- 4 James Diebel and Sebastian Thrun. An application of markov random fields to range sensing. In *Advances in neural information processing systems*, pages 291–298, 2006.
- 5 Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. Learning Morphological Operators for Depth Completion. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 450–461. Springer, 2018.
- 6 Rex A Dwyer. A faster divide-and-conquer algorithm for constructing Delaunay triangulations. *Algorithmica*, 2(1-4):137–151, 1987.
- 7 Herbert Edelsbrunner and Nimish R Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15(3):223–241, 1996.
- 8 Leonidas J Guibas, Donald E Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(1-6):381–413, 1992.
- 9 Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.
- 10 Yuhang He, Long Chen, Jianda Chen, and Ming Li. A novel way to organize 3D LiDAR point cloud as 2D depth map height map and surface normal map. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 1383–1388. IEEE, 2015.
- 11 Zixuan Huang, Junming Fan, Shuai Yi, Xiaogang Wang, and Hongsheng Li. HMS-Net: Hierarchical Multi-scale Sparsity-invariant Network for Sparse Depth Completion. *arXiv preprint*, 2018. [arXiv:1808.08685](#).
- 12 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.
- 13 Jason Ku, Ali Harakeh, and Steven L Waslander. In Defense of Classical Image Processing: Fast Depth Completion on the CPU. *arXiv preprint*, 2018. [arXiv:1802.00036](#).
- 14 Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *arXiv preprint*, 2017. [arXiv:1712.02294](#).
- 15 Fangchang Ma, Guilherme Venturelli Cavaleiro, and Sertac Karaman. Self-supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera. *arXiv preprint*, 2018. [arXiv:1807.00275](#).
- 16 Nick Schneider, Lukas Schneider, Peter Pinggera, Uwe Franke, Marc Pollefeys, and Christoph Stiller. Semantically guided depth upsampling. In *German Conference on Pattern Recognition*, pages 37–48. Springer, 2016.
- 17 Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement. *arXiv preprint*, 2018. [arXiv:1811.03818](#).
- 18 Peter Su and Robert L Scot Drysdale. A comparison of sequential Delaunay triangulation algorithms. *Computational Geometry*, 7(5-6):361–385, 1997.
- 19 Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, 2018.
- 20 Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *arXiv preprint*, 2017. [arXiv:1711.06396](#).

Generation of a Reconfigurable Probabilistic Decision-Making Engine based on Decision Networks: UAV Case Study

Sara Zermani 

Université de Guyane, Espace-Dev, UMR 228, Cayenne, France
Sara.Zermani@gmail.com

Catherine Dezan 

Université de Brest, Lab-STICC, CNRS, UMR 6285, Brest, France
Catherine.Dezan@univ-brest.fr

Abstract

Making decisions under uncertainty is a common challenge in numerous application domains, such as autonomic robotics, finance and medicine. Decision Networks are probabilistic graphical models that propose an extension of Bayesian Networks and can address the problem of Decision-Making under uncertainty. For an embedded version of Decision-Making, the related implementation must be adapted to constraints on resources, performance and power consumption. In this paper, we introduce a high-level tool to design probabilistic Decision-Making engines based on Decision Networks tailored to embedded constraints in terms of performance and energy consumption. This tool integrates high-level transformations and optimizations and produces efficient implementation solutions on a reconfigurable support, with the generation of HLS-Compliant C code. The proposed approach is validated with a simple Decision-Making example for UAV mission planning implemented on the Zynq SoC platform.

2012 ACM Subject Classification Computer systems organization → Self-organizing autonomic computing; Computer systems organization → Embedded hardware

Keywords and phrases Decision networks, Bayesian networks, HLS, FPGA

Digital Object Identifier 10.4230/OASICS.ASD.2019.9

Category Interactive Presentation

Funding This work is supported by CNRS fundings through the PICS project SWARMS.

1 Introduction

Embedded Decision-Making is necessary in a number of contexts including medical applications and autonomous vehicles. Embedded solutions make it possible to adapt to the constraints of a real-time response that would be impossible with centralized off-board decision making due to the need for communication media access. For instance, in the case of unmanned aerial vehicles (UAVs) or intelligent vehicles, on-board decision making based on image recognition enables real-time responses to propose appropriate action (e.g., to continue tracking or to dismiss) in an autonomous manner. Many examples of embedded decisions have been recently tested [26] [20] [6].

Among the techniques available for Decision-Making, three approaches have recently emerged in the literature [3]: 1) Multicriteria Decision-Making techniques, 2) Mathematical Programming techniques and 3) Artificial Intelligence. Nevertheless, to deal with uncertainty of the environment and of the system (external or internal hazards), fuzzy techniques [2] or stochastic/probabilistic models such as Bayesian Networks [21] are used. In this paper, we focus on Decision Networks (also called Influence Diagrams), which are considered as an



© Sara Zermani and Catherine Dezan;
licensed under Creative Commons License CC-BY

Workshop on Autonomous Systems Design (ASD 2019).

Editors: Selma Saidi, Rolf Ernst, and Dirk Ziegenbein; Article No. 9; pp.9:1–9:14

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

extension to Bayesian Networks (BN), including a Decision-Making (DM) mechanism based on utility tables. Compared with a fuzzy model, a Decision Network (DN) is specified by a causality graph that encapsulates the expert knowledge of the system and provides a good comprehensive and efficient formulation for the designer.

For an embedded version of a Decision-Making mechanism, the following features are of major importance: 1) achieving (real-time) performance under the constraints of memory or computation related to the embedded system, 2) ensuring quality of service under power consumption constraints. For Bayesian Networks, literature on embedded implementation can be found for both software [22] and hardware versions, the latter with reconfigurable hardware [16] [18] [25] using Field-Programmable Gate Array (FPGA). In [25], the authors propose BN in reconfigurable hardware, but the decision mechanism that integrates temporal specification in temporal logic is operated by the embedded processor. For embedded DN, the ARPHA framework [22] [7] makes it possible to design specific failure scenarios from fault tree specifications. ARPHA generates an embedded software version of a DN, but does not propose any hardware alternative. To the best of our knowledge, no hardware implementation of DN on reconfigurable platforms have been proposed as alternative embedded solutions.

In this paper, we propose a design tool to generate a reconfigurable implementation of Decision Networks from high-level specifications of a Decision-Making engine. This is the first design tool that proposes automatic generation of Decision Networks on FPGA. The main contributions are: 1) the specific High Level Synthesis (HLS) transformations to produce a HLS-Compliant C Code, 2) the generation of adequate HLS directives for efficient implementation on an FPGA/SoC (System On Chip) platform.

The paper is organized as follows. Section 2 gives an introduction to Decision Networks. Section 3 introduces the design tool to generate the embedded Decision-Making engine from DN specifications, with a specific focus on the dedicated HLS transformations and HLS directives for an efficient implementation on the FPGA/SoC platform. Section 4 presents a case study that validates the flow up to implementation on a Zynq platform.

2 Background on Decision Networks and Probabilistic Beliefs

Decision Networks extend Bayesian Networks to provide a mechanism for making rational decisions by combining probability and utility theory. In addition to chance nodes defined by a BN, a DN also includes utility and decision nodes. Decision nodes represent the set of choices open to the decision maker, while utility nodes are used to express preferences among possible states of the world represented by the chance nodes and decision nodes.

2.1 Bayesian networks for probabilistic beliefs

BNs are probabilistic graphical models used to understand and control the behaviour of systems [8] by providing diagnoses. They are composed of nodes and oriented arcs between nodes representing the knowledge expertise of the system. In Fig. 1 a), we propose to evaluate the probability of a UAV increasing or decreasing its altitude (U_A) based on the information given by two sensors: the altimeter sensor (S_A) and the barometer sensor (S_B). The BN nodes of the networks represent random variables whose values can depend on specific states, and the arcs of the network indicate the conditional dependencies represented by the conditional probabilities defined with probability tables (CPTs). In the example, each node has two states, represented by the values *inc* and *dec*. The probabilities of the BN are also known as the parameters of the network.

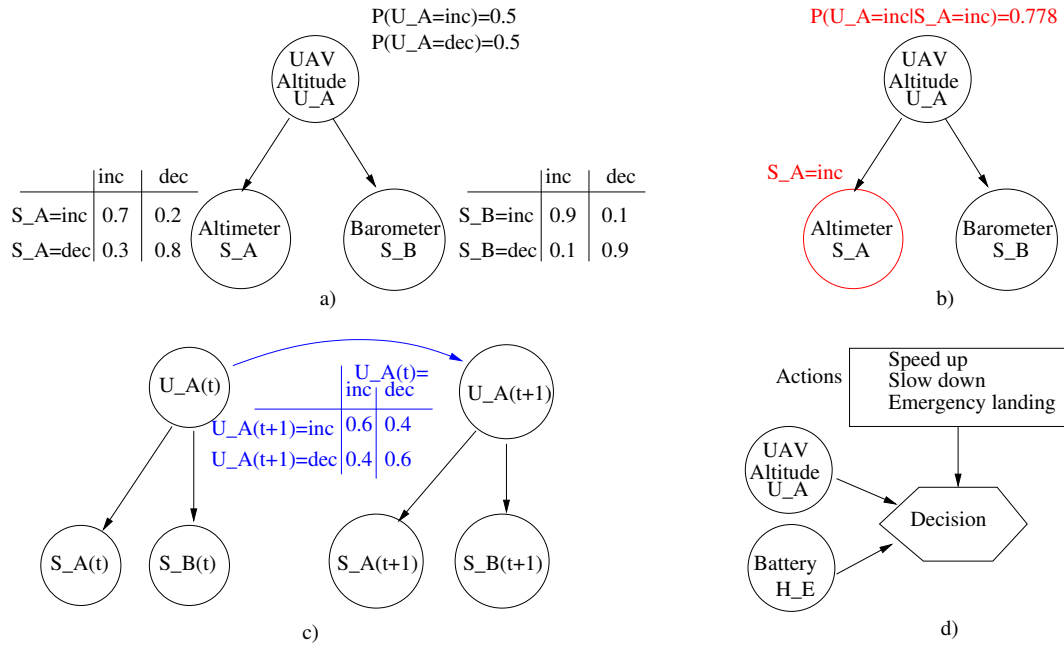


Figure 1 BN principle and DN example: a) BN example, b) Inference illustration, c) Dynamic BN, d) DN example.

To obtain the probability of one variable being in a specific state (A), we use an inference mechanism that takes into account some observations (evidence indicators) of the system in order to compute the posterior probabilities over A using Bayes' theorem (Eq. 1).

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A) \quad (1)$$

We use Bayes' theorem for our example (illustration in Fig. 1 b) to compute the probability of the variable U_A being in a state "increasing" by taking into account the value of the altimeter S_A as evidence.

This is done as follows:

$$\begin{aligned} P(U_A = Inc|S_A = Inc) &= \frac{P(S_A = Inc|U_A = Inc)P(U_A = Inc)}{P(S_A = Inc)} \\ &= \frac{0.7 * 0.5}{0.7 * 0.5 + 0.2 * 0.5} = 0.778 \end{aligned}$$

We can thus say that the probability of the altitude status being "increasing" is equal to 0.778. If, for example, we add observations from the barometer sensor, also giving it an increasing value, the probability $P(U_A = Inc|S_A = Inc, S_B = Inc)$ increases to 0.969.

The probabilities can change over time. So it could be more appropriate to use dynamic BN to model the change. In the case of a UAV, if we have a greater chance of maintaining an increasing altitude if the UAV is already in this situation, and can model this with time-dependent variables as proposed in Fig. 1 c).

As BN are dedicated to computing the probabilities of the states, we need an extra mechanism to express the decision making that takes into account these values for the choice of appropriate actions to safely continue a mission. We therefore use the Decision Networks for the Decision-Making process.

2.2 Example of a Decision Network

DN are directed acyclic graphs (DAG) with nodes belonging to three different categories: chance nodes (ellipses or circles in graphical notation), which represent (as in BN) discrete random variables; decision nodes (rectangles), which represent actions or decisions with a predefined set of states; and value nodes (diamonds), which represent utility (or cost) measures associated with random or decision variables. Edges represent direct (possibly causal) influence between connected objects. An example of Decision-Making (see Fig. 1 d)), linking the two BNs (representing the probabilities of the UAV Altitude state and of the well-functioning Battery state), and the possible actions (Speed Up action, Slow Down action and Emergency Landing action) to be chosen with the respect of a utility table. The utility table (detailed in the Table 1) gives a score for each action relative to the BNs, and the choice of the adequate action is given by computing a utility function (U_f) for each action.

■ **Table 1** Utility table for decision making with DN associated with the actions Speed Up (SU), Slow Down (SD) and Emergency Landing (EL).

UAV Alt. (U_A)	inc						dec					
Battery (H_E)	ok			bad			ok			bad		
Actions (A)	SU	SD	EL	SU	SD	EL	SU	SD	EL	SU	SD	EL
U	100	0	0	0	0	100	0	100	0	0	0	100

To compute the utility function we need the probabilities provided by the BNs. The utility function of each action is equal to the sum of the products of the action with the adequate probability concerning the UAV state and the battery state. The action to be chosen is the action that has the highest utility function.

Let us consider the example of Fig. 1 d):

$$\text{Action_to_activate} = \text{Max}(U_{f_{SU}}, U_{f_{SD}}, U_{f_{EL}})$$

where each $U_{f_k}(k = \{SU, SD, EL\})$ is equal to

$$U_{f_k} = \sum_i \sum_j U(A = k, U_A = i, H_E = j) * P(U_A = i) * P(H_E = j)$$

$$(i = \{inc, dec\} \text{ and } j = \{ok, bad\})$$

In this example, if we take the BN probabilities $P(U_A = inc) = 0.9$ and $P(H_E = ok) = 0.8$ and the utility table in Table 1, then $U_{f_{SU}} = 72$, $U_{f_{SD}} = 8$ and $U_{f_{EL}} = 20$, which means that the action chosen is *SU* ("Speed up") this case.

3 Design Tool Proposal

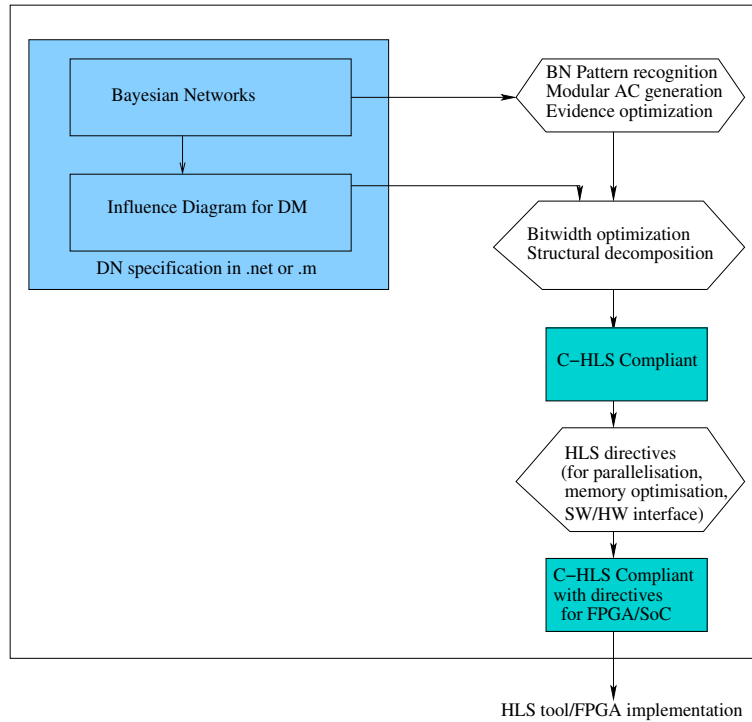
The proposed design tool (Fig. 2) incorporates the two following main layers:

1. In the first layer, the Bayesian core tool takes a DN as input. The DN specifications can be expressed in *.net* format or *.m* format to make them compatible with other tools such as Genie [14] or BNT [19] for Matlab. First, a series of dedicated high-level

transformations are proposed for the BN part: AC compilation based on model patterns and evidence optimization. Then, optimizations on the whole DN part are proposed, such as bitwidth optimization and a functional/structural decomposition based on the choice of the elementary function for a hierarchical decomposition before the generation of the C-code.

2. In the second layer, a refinement of C-code is proposed by introduction of HLS directives for code parallelization, memory and interface management. This latter C-HLS compliant code is tailored for complete FPGA implementation on ZedBoard.

The first layer proposes high-level transformations and provides parallel opportunities for the code, independent of the target platform. The second layer gives a more practical guide for parallel implementation on a FPGA/SoC platform. This second layer is platform dependent. In this presentation, we focus on specific high-level transformations for BN and on the generation of C-code in order to show the ability of the design tool to generate parallelism. We therefore do not detail the bitwidth adaptation and functional/structural decomposition. The probabilities are defined here in a floating-point representation in the different examples of Section 4, but they can easily be limited to fixed-point representation, thus saving some FPGA resources at the same time.



■ **Figure 2** Proposed HLS Design tool for Decision Networks on an FPGA support.

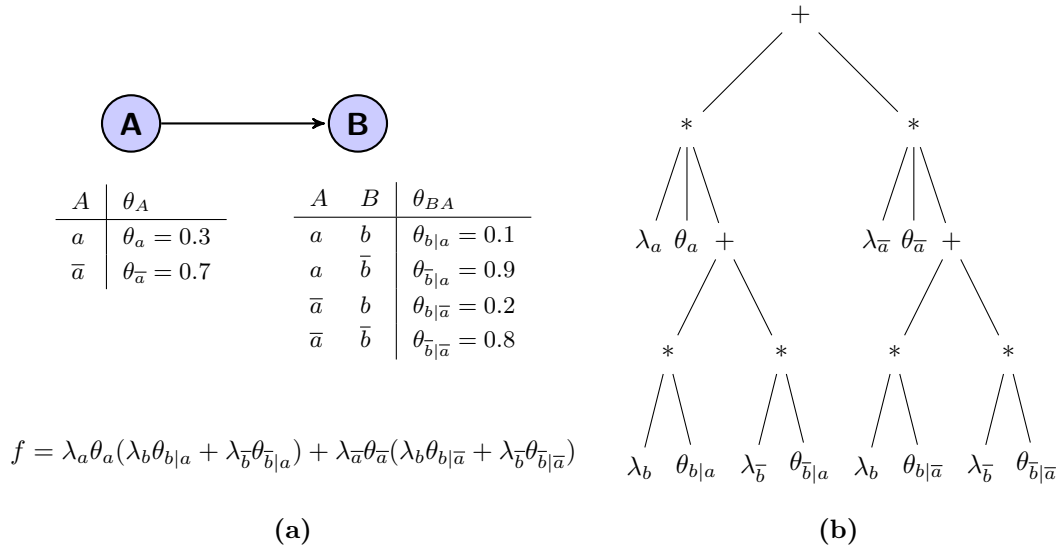
3.1 Bayesian core tool for DN: Dedicated high-level transformations for DN

In this section, we start with an introduction to the intermediate representation (arithmetic circuit compilation) used to synthesize the BN specifications. High-level transformations are then proposed to generate a synthesizable C-code for DN specifications.

3.1.1 Arithmetic circuit (AC) compilation for BN inference

BN inference algorithms are used to answer queries when computing posterior probabilities. Classical inference algorithms are based on propagation on the junction tree. A major problem for embedded systems is the complexity of the computation. Algorithms based on ACs are powerful and can produce predictable real-time performances [4] [10].

The AC representation of BN can be built from the multilinear function (MLF) f [11] associated with the marginal probabilities of the BN (Figure 3). The leaves of the arithmetic circuit are λ (evidence indicator) and θ (network parameter), and the inner nodes of the tree represent a multiplication (*) or an addition (+), alternately. To compute the posterior probability $P(x|e) = \frac{f(x,e)}{f(e)}$ (where x is a variable and e the evidence) for the diagnostic, two steps are required: the first to evaluate $f(e)$ and the second to compute the circuit derivatives to obtain $f(x,e) = \frac{\partial f}{\partial \lambda_x}$.



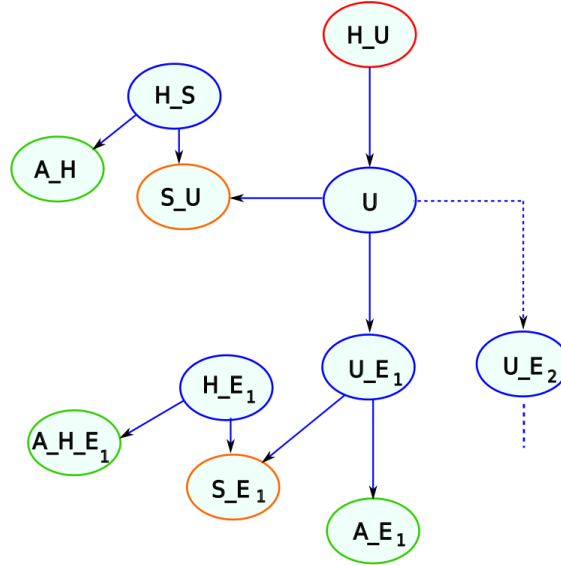
■ **Figure 3** (a) A Bayesian network with a multilinear function. (b) The corresponding arithmetic circuit.

3.1.2 Modular AC generation associated with model patterns

It is possible to simplify the generation of AC if the BN structures correspond to identified patterns. For instance, in the examples of Section 4, the structure of BN is clearly repetitive and matches the pattern we named the FMEA_HM pattern, described in Fig. 4. Other patterns can also be used, like the SWHM (software Health Management) model proposed in [24]. The BN of FMEA_HM pattern evaluates the probability of well functioning for a specific item in the system providing the diagnosis, also called *HM* (Health Management).

In such cases, the generation of the AC uses a modular approach of inference computation by taking advantage of the factorization of the MLF and the regular structure associated with the pattern. In the case of an FMEA_HM pattern, the AC MLF factorization is based on the relationship between child and parent nodes. The sub-MLFs for child nodes (if there is no conditional dependence between them) are represented by a + in the AC, and parent nodes combine them with a *. In this model, all error type nodes and all monitor nodes are children of the U node (unobservable status of the system). Since there is no conditional

dependence between these nodes, their MLF can be calculated separately and in parallel, similar to the sub-BNs of each error type, where the parent node is the error type and the child nodes are the monitor and appearance context nodes. This allows us to have a hierarchical and modular AC.



■ **Figure 4** FMEA_HM pattern for BN where the S_U , S_{Ei} nodes represent the sensor nodes of either a hardware or a software monitor, the nodes U , U_{Ei} are the internal states possibly affected by an error Ei , the H_U node represents the health of the system, H_S and H_{Ei} nodes represent the health of the sensor, and the A_H , $A_{H_{Ei}}$ are the appearance contexts.

This principle can be easily extended to Dynamic Bayesian Networks by considering the temporal variables.

3.1.3 BN optimization based on evidence

In an AC, we can see that the values $(\lambda_x, \lambda_{\bar{x}})$ of evidence of a variable X are equal to:

- $(\lambda_x, \lambda_{\bar{x}}) = (1, 1)$ when there is no observation on the variable X ,
- $(\lambda_x, \lambda_{\bar{x}}) = (1, 0)$ when the evidence is on the variable X and the observation is x ,
- $(\lambda_x, \lambda_{\bar{x}}) = (0, 1)$ when the evidence is on the variable X and the observation is \bar{x} .

Furthermore, in our examples in Section 4, two types of node (observable and unobservable) are known: C, A and S nodes are observable (evidence), so they take the values $(1, 0)$ or $(0, 1)$ for evidence. The other nodes are unobservable, so they take the values $(1, 1)$ for evidence. These observations and the symmetrical structure of the AC make it possible to reduce the AC as follows:

- Delete the left (or right) topmost branch containing a C (or an H) node, and in all sub-branches where C (or H) appears, replace the probability parameters by a choice between the right or left value. We can simplify here, because the value of $(\lambda_x, \lambda_{\bar{x}})$ of these nodes is never equal to 1 at the same time,
- Repeat this procedure for all C and H nodes,
- Fix all the λ_x values for the unobservable nodes at $(1, 1)$.

3.1.4 C-synthesizable code for DN

The decision-making approach is based on the utility function equation, taking the HM results from BNs, actions and utility table as input. The C-synthesizable code is generated for BN in two ways: a) in a hierarchical way, by choosing several kinds of elementary block, or b) in a flat way. In a hierarchical version, the elementary blocks are chosen considering the structure of the AC tree.

The C-synthesizable code of the DN is given by the following algorithm:

Algorithm 1 Decision Making.

Require: Proba from BNs HM_1, HM_2, \dots, HM_n , actions A_1, A_2, \dots, A_n and the utility table U

for all states i_1 of HM_1 **do**

...

for all states i_n of HM_n **do**

for all actions A_k **do**

// Compute utility function for each action

$$U_{f_A_k} = U_{f_A_k} + U(A_k, HM_1 = i_1, \dots, HM_n = i_n) \\ * P(HM_1 = i_1) * \dots * P(HM_n = i_n)$$

end for

end for

...

end for

return Maximum of $U_{f_A_k}$

3.2 Generation of HLS directives for a SoC implementation on ZedBoard

3.2.1 ZedBoard target for SoC/FPGA implementation

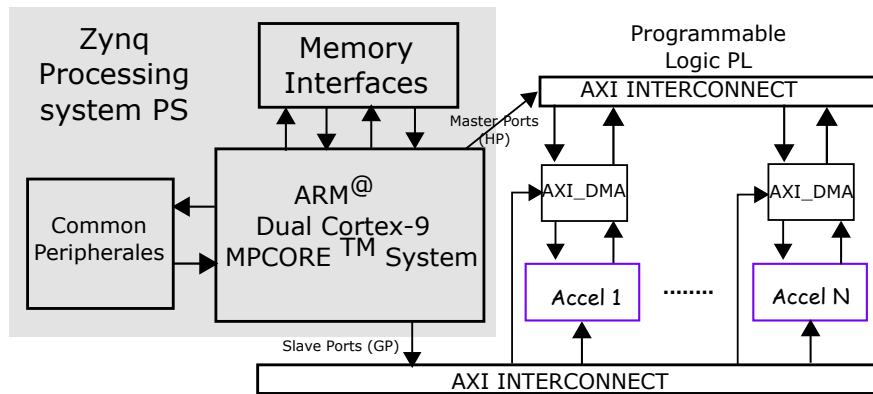
Our design approach is characterized by the separation of processing components from functional programmable components. The proposed design targets the ZedBoard incorporating a hybrid Zynq processor [9]. As shown in Figure 5, the architecture is built around the ARM Cortex-A9 processor (Zynq processing system PS). The processor communicates with dedicated HW accelerators using programmable logic through an Advanced eXtensible Interface (AXI) bus.

For the SoC implementation on ZedBoard, numerous HLS directives have been proposed [1]. Here, we list only the main ones used for this experimentation.

3.2.2 Parallelization directives at function calls and for loops

In order to increase the parallelism, the following main directives are chosen:

1. **INLINE:** Inlines this function call (does not create a separate level of RTL hierarchy) and allows resource sharing and optimization across hierarchy levels.
2. **UNROLL:** Duplicates computation inside the loop and increases computation resources, decreases number of iterations.
3. **PIPELINE:** Pipelines computation within the loop (or region) scope and increases throughput and computation resources.



■ **Figure 5** The hybrid architecture of the Zynq processor for SoC implementation.

Most of the time, the resources available for the IP are limited because other applications can share the resource. So the parallelization directives are used considering resource constraints.

3.2.3 Memory management directives

For the storage of the parameters of the Bayesian network, two main options are used for an embedded version inside the programmable logic component:

1. **BRAM:** with the directives *array_map* or *array_partition*
2. **LUT:** default option

The data organization should be addressed correctly taking into account the interface mechanism.

3.2.4 Interface management

The inputs of the SoC are the evidence of the networks that are data provided by sensors. These are stored in an external memory (DDR). As for the evidence of the networks, the parameters of the BN are inputs of the IP and can be stored either inside the SoC (BRAM, LUT or CACHE) or outside in an external memory. This choice depends on the designer's needs and on the possible changes of the network parameter values. To access the external memory, different interface options are possible:

1. **STREAM:** AXI stream
2. **DMA:** Master DMA

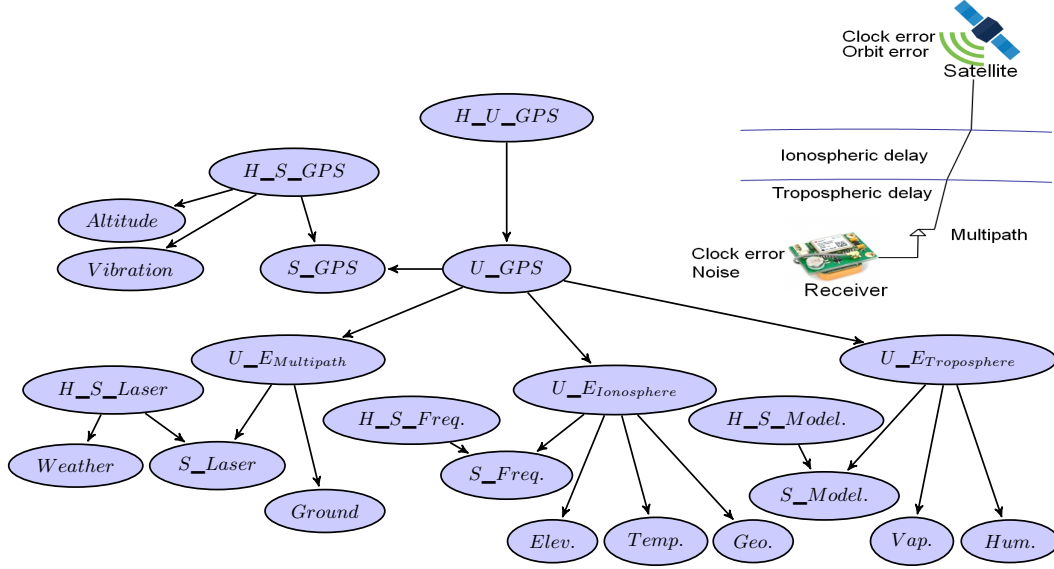
4 Case Study of a Simple UAV Mission Plan

In this section, we present a case study that validates the design flow up to implementation on the Zynq platform. In this section we consider a simple UAV mission that address two main failure scenarios; one related to GPS failure and the other to the battery failure. Both can be expressed by BN, considering the FMEA_HM pattern.

4.1 Bayesian networks for the health of the GPS receiver and of the battery

The accuracy and reliability of the position given by a GPS depend on contextual factors affecting the satellite signal during its propagation or its reception. The sources of error can be identified at the system level by means of additional bias in the computation of

pseudo-range measurements [17] [23]. These measurements can tune the GPS positioning accuracy from slightly imprecise to completely faulty. They can be improved by introducing observations [12] or real environments [13]. The main GPS localization errors are illustrated in Fig. 6 with a BN.



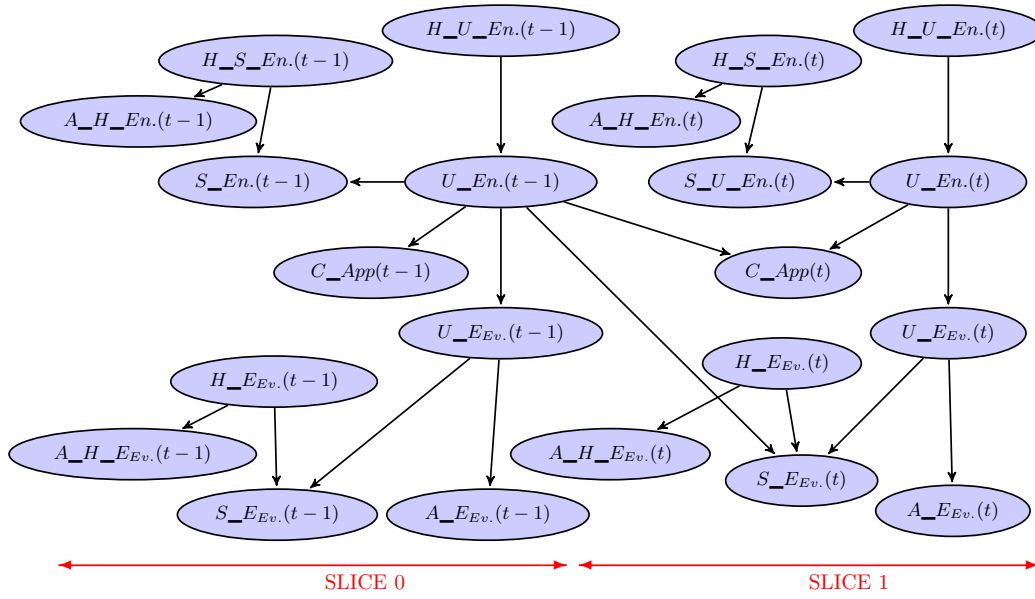
■ **Figure 6** GPS potential errors and BN description.

To model battery behaviour, we use a dynamic BN to represent the linear progression of the energy consumption over time.

This dynamic Bayesian network is described at two time steps (2TBN, cf Figure 7) to take into account the previous past value. External events such as strong wind can increase the energy consumption, as may any application/component used in a period of time. Each status of an application (U node) is associated with a command node (C) that represents an action to enable or disable the application. The battery level is given by a sensor that can fail; its health is reinforced by the appearance contexts (low temperature, for example).

4.2 Decision making with an influence diagram

Figure 8 shows the decision-making mechanism of the mission considering the two cases of failure (GPS failure (GPS HM), Battery failure (Energy HM)) and three actions (Nothing to report, Change localization method, Emergency Landing). This figure illustrates the monitoring of the GPS HM, the energy consumption HM and the DM at each time. Figure 8.(a) shows the interest of the context appearance nodes, which reinforces the confidence in error types and sensor health. For example, from time $t = 0$ until $t = 4$, we observe no problem in the GPS localization and, due to the evidence on appearance contexts, the probability of the health of the system grows from 0.835 to 0.915. At time $t = 4$, a problem in the GPS localization is observed. Without observation on the appearance context nodes, the probability of the health of the system decreases to 0.365, but according to the evidence on appearance contexts it decreases to 0.143. Figure 8.(b) shows monitoring in a nominal case for energy consumption. According to observation of sensors and appearance contexts, the energy consumption is healthy but decreases over time because the related Bayesian



■ **Figure 7** 2TBN for Health Management for the battery.

Network evolves dynamically. Figure 8.(c) shows the evolution of Decision Making over time based on the health probability of the GPS localization and energy consumption. From time $t = 0$ until $t = 4$, the maximum of the utility functions is “nothing to report”, which reflects the case of “no problem in the mission”. At time $t = 4$, the maximum of the utility functions is “change localization method”, which reflects a problem in the mission, i.e., the GPS failure scenario is detected.

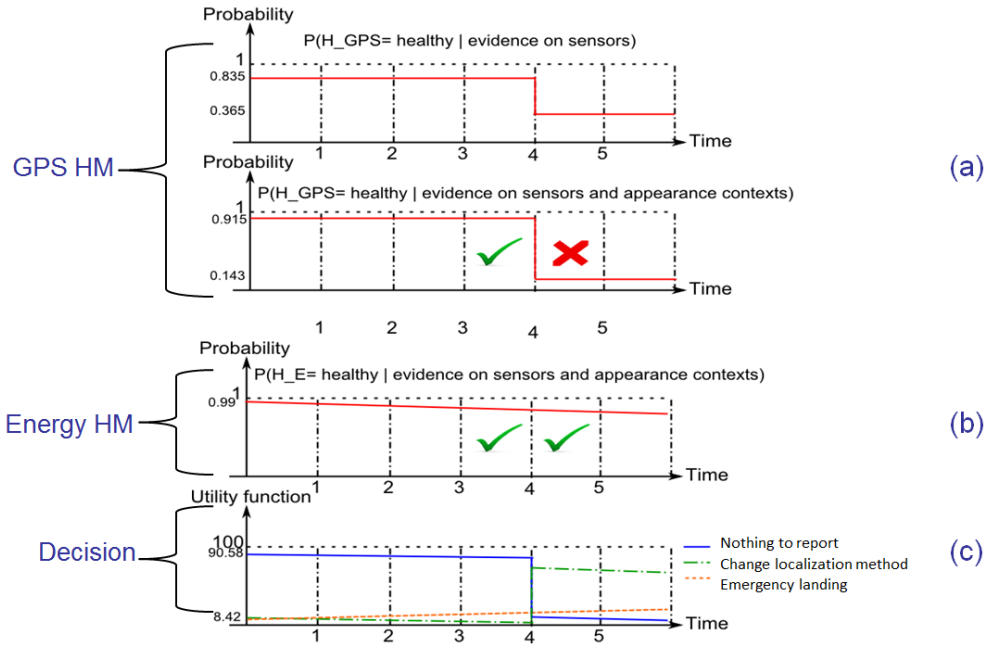
4.3 SoC implementation

For our case study, Table 2 shows the evaluation of resources used for our implementation in terms of Bloc RAM (memory BRAM), Digital Signal Processors (DSP), Look Up Tables (LUT) and Flip Flop registers (FF). The total number of DSPs on the ZedBoard equals 220, that of LUTs is 53200 and that of FFs is 106400. The results are given for hardware solutions maximizing parallelism. The parallelism is better exploited in the case of GPS because there is less conditional dependency between nodes. The complete DM model uses the same number of DSPs as in case of GPS HM, which is explained by resource sharing.

■ **Table 2** Resources used by the BN model for the programmable logic part.

	BRAM	DSP	LUT	FF
GPS localization HM	0%	34%	33 %	9%
Energy consumption HM	0%	29%	27%	7%
GPS HM+ Energy HM+ Decision	0%	34 %	51%	14%

Table 3 shows the performance for the HW/SW implementation. We observe a good HW speed-up in all cases because the SW implementation is sequential. A better speed-up is observed in the energy case, which is due to the computational complexity from the dynamic BN. The complete DM model has a good speed-up, which could be improved, but only at



■ **Figure 8** Results for the Decision Making of a simple UAV mission.

the expense of more resources. We also observe that although the communication time to send the network parameters is high, this can be eliminated or reduced if a local memory is used, which also means an increase in HW resources.

■ **Table 3** The HW/SW performance of the BN nodes and DM, where $\text{HW Speed-up} = \text{SW execution time} / \text{HW execution time}$.

	SW time (cycles)	HW time (cycles)	HW Speed-up
GPS localization HM	955	391	2.42
Energy consumption HM	1268	339	3.74
GPS HM+ Energy HM+ Decision	2190	698	3.13

■ **Table 4** Performance, Resource and Energy consumption of the Decision Making mechanism.

	Resource				Latency (cycles)	Speed-up	Energy cons. (μJ)
	BRAM	DSP	LUT	FF			
HM/DM in HW	14%	50%	34%	21%	419	6.28	9.81
HM in HW only	13%	44%	42%	20%	665	3.96	15.11

The results in Table 4 show the interest of a complete solution implemented in HW in terms of performance and energy consumption. Nevertheless with three actions, a good speed-up is already obtained for both versions of the DM, if HM are implemented in HW.

5 Summary and Conclusions

We present an original design tool for the implementation of Decision Networks on FPGA. This tool helps the designer to specify and implement the Decision-Making process under real-time constraints and energy constraints, making it suitable for embedded solutions on autonomous vehicles. We detail the design flow, giving the specific HLS optimizations and transformations available to generate implementation on an FPGA/SoC platform. We propose a validation example that demonstrates the interest of such a tool by providing efficient HW implementation for the Decision-Making engine.

Because of the complexity related to BN and DN definition and to their related classical inefficient implementation using junction trees [15], decision-making mechanisms based on DN were not really considered as a possible engine for embedded applications until now. Nevertheless, new compilation methods based on AC and on pattern identification for the BN description can help to achieve efficient implementation on both CPU and FPGA. These methods also help us to define a completely new embedded Decision-Network engine on the both supports. The present paper addresses this opportunity.

As perspectives for future work, we propose to integrate the runtime constraints of the sensors to fit the constraints of the mission in a more realistic manner. In this way, we could extend and couple the presented offline tool, which provides HW/SW implementation of mission planning, by adding an online engine that can choose the most appropriate version of the decision core implementation considering the CPU load, the system constraints in terms of energy and timing, and which can take into account service quality requirements. If the implementation of the decision process can be achieved on an FPGA/SoC support, the HW/SW alternatives can be chosen dynamically at runtime in the same way as this is possible for other applications, such as those for image processing [5].

References

- 1 Vivado-HLS. <https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html/>. Accessed: 2016-09-30.
- 2 Kai-Yuan Cai and Lei Zhang. Fuzzy reasoning as a control problem. *IEEE Transactions on Fuzzy Systems*, 16(3):600–614, 2008.
- 3 Junyi Chai, James NK Liu, and Eric WT Ngai. Application of decision-making techniques in supplier selection: A systematic review of literature. *Expert Systems with Applications*, 40(10):3872–3885, 2013.
- 4 Mark Chavira and Adnan Darwiche. Compiling Bayesian Networks with Local Structure. In Leslie Pack Kaelbling and Alessandro Saffioti, editors, *IJCAI*, pages 1306–1312. Professional Book Center, 2005. URL: <http://www.ijcai.org/papers/0931.pdf>.
- 5 Hanen Chenini, Dominique Heller, Catherine Dezan, Jean-Philippe Diguët, and Duncan Campbell. Embedded real-time localization of UAV based on an hybrid device. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1543–1547. IEEE, 2015.
- 6 Hsin-Han Chiang, Yen-Lin Chen, Bing-Fei Wu, and Tsu-Tian Lee. Embedded driver-assistance system using multiple sensors for safe overtaking maneuver. *IEEE Systems Journal*, 8(3):681–698, 2014.
- 7 Daniele Codetta-Raiteri and Luigi Portinale. Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 45(1):13–24, 2015.
- 8 Robert G Cowell. *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business, 2006.

- 9 L.H. Crockett, R. Elliot, and M. Enderwitz. *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc.* Strathclyde Academic Media, 2014. URL: <http://books.google.fr/books?id=9dfvoAEACAAJ>.
- 10 Adnan Darwiche. A differential approach to inference in Bayesian networks. *J. ACM*, 50(3):280–305, 2003. doi:10.1145/765568.765570.
- 11 Adnan Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM (JACM)*, 50(3):280–305, 2003.
- 12 Fabio Dovis, Bilal Muhammad, Ernestina Cianca, and Khurram Ali. A Run-Time Method Based on Observable Data for the Quality Assessment of GNSS Positioning Solutions. *Selected Areas in Communications, IEEE Journal on*, 33(11):2357–2365, 2015.
- 13 Vincent Drevelle and Philippe Bonnifait. Reliable positioning domain computation for urban navigation. *Intelligent Transportation Systems Magazine, IEEE*, 5(3):21–29, 2013.
- 14 Marek J Druzdzel. SMILE: Structural Modeling, Inference, and Learning Engine and GeNie: a development environment for graphical decision-theoretic models. In *Aaai/Iaai*, pages 902–903, 1999.
- 15 Frank Jensen, Finn V Jensen, and Søren L Dittmer. From influence diagrams to junction trees. In *Uncertainty Proceedings 1994*, pages 367–373. Elsevier, 1994.
- 16 Z. Kulesza and W. Tylman. Implementation Of Bayesian Network In FPGA Circuit. In *Proceedings of the International Conference on Mixed Design of Integrated Circuits and System*, pages 711–715, June 2006. doi:10.1109/MIXDES.2006.1706677.
- 17 Richard B Langley. The GPS error budget. *GPS world*, 8(3):51–56, 1997.
- 18 Mingjie Lin, Ilia Lebedev, and John Wawrzyniek. High-throughput Bayesian Computing Machine with Reconfigurable Hardware. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, pages 73–82, New York, NY, USA, 2010. ACM. doi:10.1145/1723112.1723127.
- 19 Kevin Murphy et al. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.
- 20 Kourosh Noori and Kouroush Jenab. Fuzzy reliability-based traction control model for intelligent transportation systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(1):229–234, 2013.
- 21 Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- 22 Luigi Portinale and Daniele Codetta-Raiteri. ARPHA: AN FDIR ARCHITECTURE FOR AUTONOMOUS SPACECRAFTS BASED ON DYNAMIC PROBABILISTIC GRAPHICAL MODELS. In *Proc. IJCAI workshop on AI on Space, Barcelona*, 2011.
- 23 Daniel Salós, Christophe Macabiau, Anaïs Martineau, Bernard Bonhoure, and Damien Kubrak. Nominal GNSS pseudorange measurement model for vehicular urban applications. In *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, pages 806–815. IEEE, 2010.
- 24 Johann Schumann, Timmy Mbaya, Ole Mengshoel, Knot Pipatsrisawat, Ashok Srivastava, Arthur Choi, and Adnan Darwiche. Software health management with Bayesian networks. *Innovations in Systems and Software Engineering*, 9(4):271–292, 2013.
- 25 Johann M Schumann, Kristin Y Rozier, Thomas Reinbacher, Ole J Mengshoel, Timmy Mbaya, and Corey Ippolito. Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems. *International Journal of Prognostics and Health Management*, 6, 2015.
- 26 Poorani Shivkumar. Intelligent controller for electric vehicle. In *2008 IEEE International Conference on Sustainable Energy Technologies*, pages 978–983. IEEE, 2008.