A Flexible Scheduling Architecture of Resource Distribution Proposal for Autonomous Driving Platforms

Hadi Askaripoor[a], Sina Shafaei[b] and Alois Knoll[c]
Informatics Department, Technical University of Munich, Boltzmannstr. 3, 85748 Garching bei München, Germany

Keywords: Scheduling, Resource Planner, Autonomous Driving, Platform Architecture, Decisions Making.

Abstract: Autonomous driving has attracted a significant amount of attentions over the last ten years. Providing a flexible platform to schedule the executions of the tasks under hard real-time constraints is also a crucial matter which needs to be taken into account by the integration of intelligent applications. In this work, we propose a resource planner, consisting of a monitoring mechanism, context manager, and decision unit which facilitates the timing requirements in the presence of AI-based applications for the autonomous vehicles.

1 INTRODUCTION

With the rise in the number of on-board sensor devices, rapid changes in urban landmarks and transportation facilities, and the complex road conditions of people and vehicles, the ability to safely respond to autonomous vehicles in real-time is constantly increasing. The computing resource capabilities of on-board hardware are limited, which present a serious challenge to self-driving vehicles in real-time. The resource requirements for information processing in the vehicle's automatic driving process are different according to the applications. In the normal driving scenario, the vehicle only needs to handle the corresponding road conditions smoothly. However, in an emergency scenario the processing time is as short as a second, including the calculation of the emergency measures and the time it takes to complete the emergency measures. The computing resources of self-driving cars are limited, and the transmission, the brakes, and the steering modules in the car all require a reaction time. Furthermore, autonomous vehicles employ plenty of safety-critical applications. Therefore, if the scheduling strategy is not capable of guaranteeing the intelligent system's complete reasoning operations and responses to physical abnormalities within a certain time period, ending up in hazardous situations is highly imminent— leading to catastrophic consequences.

[a] https://orcid.org/0000-0002-2570-5010
[b] https://orcid.org/0000-0002-9381-0197
[c] https://orcid.org/0000-0003-4840-076X

Currently, many useful and practical machine learning-based applications exist, which are already integrated in semi-autonomous vehicles and platforms. For example, road image recognition, behavior recognition, object recognition, decision making, trajectory decision, and down-scaling techniques are very popular applications among the developed solutions to ease the decision making process for the intelligent vehicle. This represents the need to have a coordinated scheduling mechanism for the resources and the sensors combined with the highest level of flexibility that can work with the divers range of the AI-based applications of the intelligent vehicle platform.

2 RELATED WORK

Machine learning is the core of the AI-based application in the intelligent vehicle and its applications are distributed from the sensory level to the decision making level. A very intuitive example of this process is found in the work of (Thammachantuek et al., 2018a), in which they propose support vector machines (SVMs) for recognition of road images. Four actions such as turn left, turn right, forward, and stop are used as an example to verify the accuracy of classification results. For more complex scenarios like having the pedestrian near the road, the authors of (Fang et al., 2018) provide a solution based on two sensors combining machine learning techniques to monitor the motion of people. Further-

Askaripoor, H., Shafaei, S. and Knoll, A.
A Flexible Scheduling Architecture of Resource Distribution Proposal for Autonomous Driving Platforms.
DOI: 10.5220/0010471605940599
In *Proceedings of the 7th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2021)*, pages 594-599
ISBN: 978-989-758-513-5

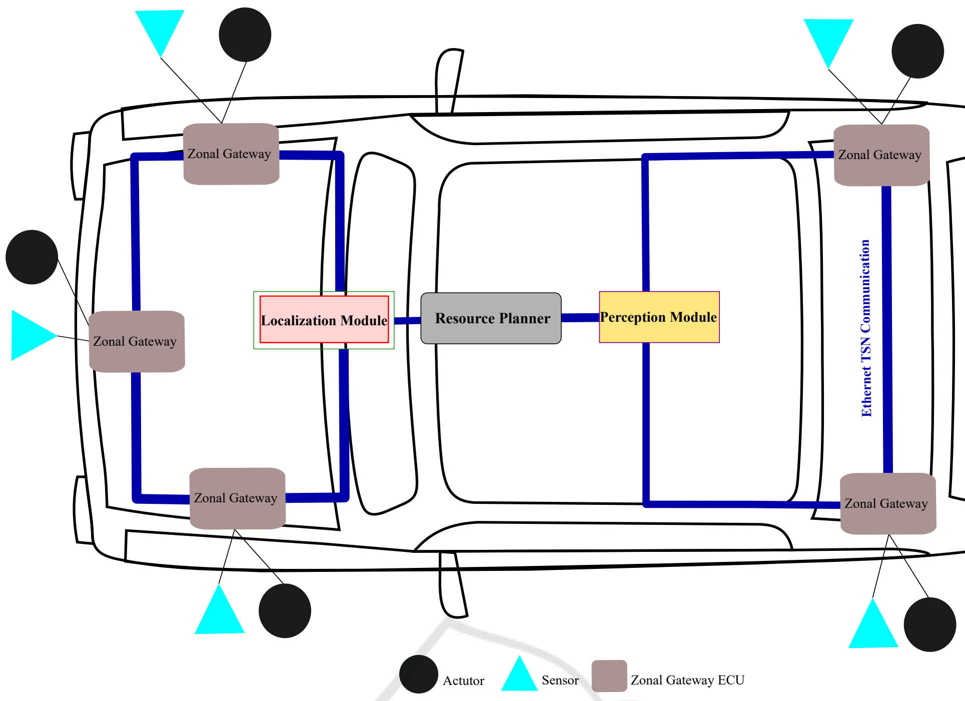

Figure 1: Integration of the proposed resource planner in the platform of E/E architectures.

more, Hayat et al. (Hayat et al., 2018) takes it one step further and proposes the use of multi-classes object recognition by using CNN for this aim. Thammachantuek et al. (Thammachantuek et al., 2018b) focus on decision-making and provide a performance comparison of well-known supervised learning algorithms such as SVM, MLP, CNN, DT, and RF. At the end, the outcome of this application will be consumed by other functions or applications like trajectory planning. Similar to the work of (Werling et al., 2010), a trajectory generator may be proposed to generate different trajectories; however, machine learning techniques will be employed on the fly to choose the optimal trajectory. This example of dataflow from the sensors to the applications demonstrates the potential integration between machine learning solutions and the importance of having a coordination mechanism to distribute the available resources to desired entities within the expected time frame.

Liu and Layland (Liu and Layland, 1973) proposed the earliest deadline first scheduling in their classic work, which assumes that a multitasking system consisting of several independent periodic tasks runs on a single processor. The earliest deadline first algorithm is the optimal algorithm on the non-mixed-critical system which can reach the maximum resource utilization limit, however in a mixed-critical system like autonomous vehicle it is unable to guarantee that emergency tasks will be executed first. In terms of fixed-priority scheduling, the earliest deadline first algorithm can ensure the high priority of safety-critical tasks, but the lower-priority tasks can wait an indefinite amount of time before being executed, leading to a higher task starvation and missing rate of low-priority tasks. Based on fixed-priority scheduling, an improved machine learning-based algorithm as Fair Emergency First (FEF) is designed for scheduling (Malik et al., 2019b). A scheduling algorithm requires a framework or platform to be fully integrated and also the onboard functions to be enabled. In this regard, FEF has been recently used as the core approach of the hybrid agent in the scheduler model of an autonomous vehicle (Malik et al., 2019a). Kong and Wu (Kong and Wu, 2005) have proposed a mechanism which gathers two inputs from the environment for processing more dynamic scenarios.

Similarly, Yingzi et al. (Yingzi et al., 2009) proposed a set of dispatching rules which are used to schedule the task in dynamic task lists. The lists convert the scheduling problem into a reinforcement learning problem first, followed by the construction of a dynamic programming process. Shulga et al. (Shulga et al., 2016) have aimed to extract the job in sub-tasks and assign them as multiple threads to computational units; therefore, a making-decision system is placed ahead of the scheduling unit.

3 METHODOLOGY AND PROPOSED ARCHITECTURE

In order to compute an architectural proposal based on predefined optimization goals by collecting system hardware/software properties and applications with a focus on artificial intelligence, we propose an approach in this work and refer to it as *resource planner* in the following sections. Our resource planner is proposed for state-of-the-art electronic and electrical (E/E) architecture of a vehicle as presented in Figure 1. The architecture of the resource planner consists of three major mechanisms, namely *Monitoring Mechanism*, *Context Manager*, and *Decision Unit*.

3.1 Monitoring Mechanism

To monitor application timing, we have designed a decentralized mechanism that focuses on identification of timing violations of each application deployed on the E/E platforms of the vehicle. Since the automotive applications are divided into the categories of safety-critical and non-safety critical, the real-time monitoring plays a pivotal role in precisely detecting timing violations in safety-critical applications. These mechanisms have access to all nodes of the car topology, where each node represents a specific application. In our considered E/E architecture, as depicted in Figure 1, the monitoring approach is developed in the perception and localization modules to monitor the timing of running applications for each one of those modules and the output is forwarded to the resource planner. As the input of the monitoring approach, timing requirements related to each application (e.g., starting times and deadlines) as well as the timing order of each application (i.e., execution priority or parallelism) are provided to the monitoring mechanism. As the output, in the case of any missing application deadline or timing violation, is exclusively implemented on perception and localization modules, the relevant timing violations are recognized and are forwarded to a warning segment located in the decision unit (See Fig. 2).

3.2 Context Manager

This mechanism aims to identify the driving context of the vehicle. For instance, driving a car into a tunnel, in rainy weather conditions, and traffic jam situation are interpreted as three driving situations. In order to distinguish the driving context, an interaction and data exchange must be done between the context manager mechanism and other modules. More specifically, the vehicle state (computed by the localization module) as well as the objects around the vehicle (calculated by the perception module) must be identified simultaneously and fused into context manager in order to compute the current driving context of the car. Eventually, the result of this mechanism is utilized by the decision unit. The labeling approach is used to identify the driving situation in the context manager with respect to localization and perception calculation. In other words, the vehicle state and the objects around it are labeled periodically in such a way that they make the situation of the vehicle distinguishable according to its surrounding environments. Therefore, in case of any updates in the labels, the resource planner is notified accordingly.

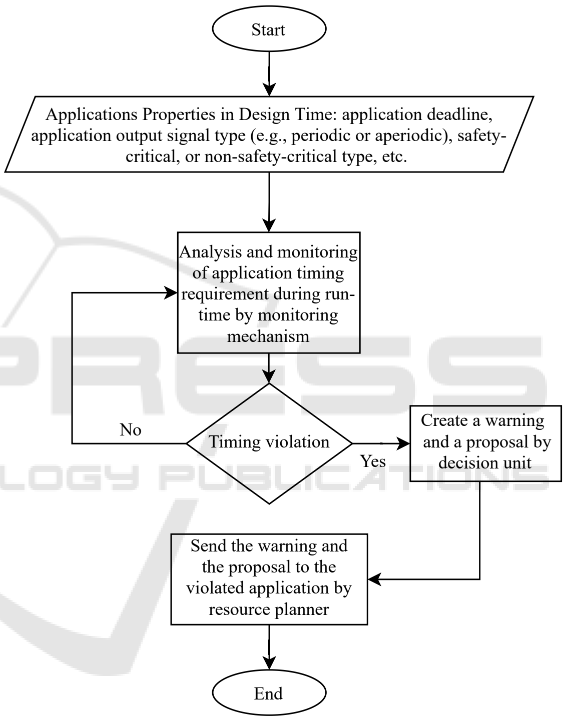
Start
Applications Properties in Design Time: application deadline, application output signal type (e.g., periodic or aperiodic), safety-critical, or non-safety-critical type, etc.
Analysis and monitoring of application timing requirement during run-time by monitoring mechanism
No
Timing violation
Yes
Create a warning and a proposal by decision unit
Send the warning and the proposal to the violated application by resource planner
End

Figure 2: The resource planner architecture to identify and warn the timing violations in the proposed architecture.

3.3 Decision Unit

After recognition of the timing violations by the monitoring mechanism, the result is sent to the warning segment. In this segment, the violated module which has missed the application deadline is identified (in our scenario, the localization or perception module). Accordingly, a warning regarding the relevant timing violations generated by the decision unit is transferred to the module by the resource planner. The module is then informed about its timing violation, which is relevant to a running application. Furthermore, a proposal, including a suggested solution to decrease the

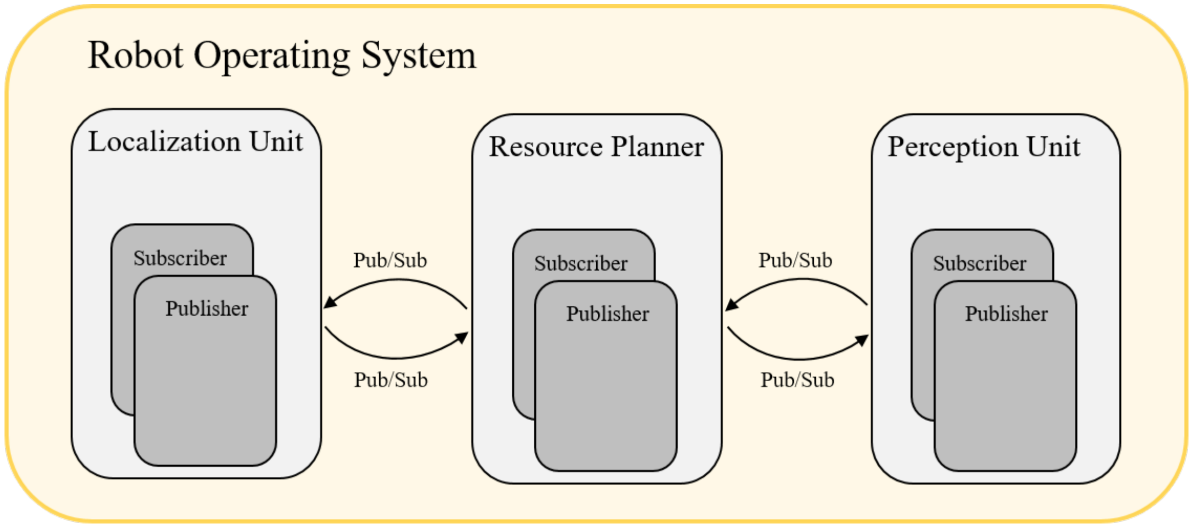

Figure 3: The proposed architecture in ROS2.

timing violations in run-time, e.g., killing the applications in an optimized and safe manner, is created by the decision unit. Accordingly, the module, which has violated the timing requirement of the application, can utilize the proposed solution to reduce the violation, while the functional safety point of view of the applications and other modules must be considered at run time (e.g., change the running order of the applications or killing the applications according to safety aspects). In contrast to our previous work in progress that the architecture synthesis was going to be performed in the design-time (Askaripoor et al., 2020), in this study the architecture synthesis is done in run time focused on timing violations, as shown in Figure 2.

As mentioned above, the result of the context manager is forwarded to the decision unit as well. In this unit, after labeling processing and understanding the current context of the vehicle in real-time, aim is to proceed with the reduction of the computational power of the system. For example, when a car drives into a tunnel, the Global Positioning System (GPS) will fail; accordingly, detailed and accurate mapping systems will help prevent accidents. Therefore, the allocated resources to analyze the GPS data can be assigned to other critical applications in the case of resource shortages. Another situation could be switching to lidar algorithms in rainy weather instead of the camera because of the inefficiency of the camera as the main source. This type of weather results in killing camera-related algorithms concerning context changes as well as functional safety aspects. Providing empty resources—in other words, decreasing the computational power based on the context changes as a proposal—is provided by the resource planner. The created proposal is sent to the relevant modules; whereas, its usage is only decided by them considering the functional safety aspects.

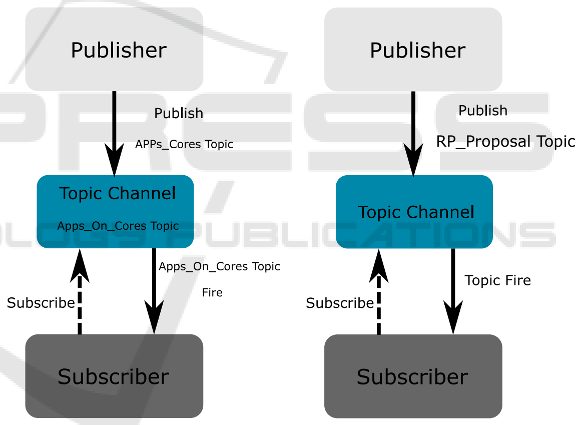

Figure 4: The publish-subscribe pattern to communicate with other modules.

We have utilized the publisher-subscriber pattern to transmit the data between the resource planner and the other two modules in our considered E/E architecture, as depicted in Figure 4. For instance, the resource planner can publish its proposal to the topic channel and the localization module can subscribe to this channel to access the proposal. Conversely, the core information, including which application is running on which core, can be published to the topic channel by a module (e.g., the perception module), and the resource planner can subscribe to it in order to acquire the status of the cores and calculate the proposal, which was explained in decision unit, based on the context manager and monitoring mechanism.

Furthermore, we have used an open-source *Robot*

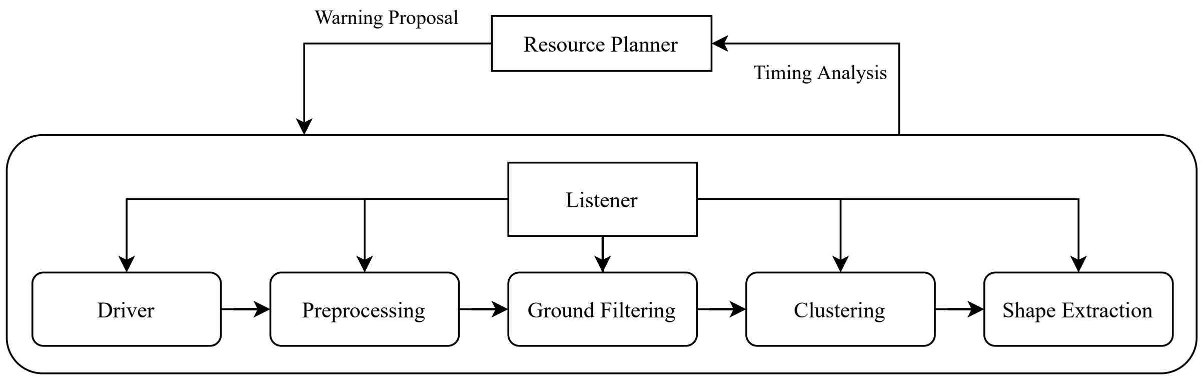

Figure 5: Identifying the timing violation of the Lidar Object Detection Stack developed in Autoware by resource planner.

Operating System 2 (ROS2)(Thomas et al., 2014) to implement our proposed publisher-subscriber pattern considering the fact that it also supports real-time communications. ROS2 employs *Data Distribution Service* (DDS) as the backbone of the communications between publishers and subscribers. DDS is suitable for real-time distributed embedded systems due to its different transport configurations (e.g., fault-tolerance and deadline) and scalability (Maruyama et al., 2016). The proposed architecture utilized in ROS2 is presented in Figure 3.

4 EVALUATION CRITERIA AND OUTLOOK

In order to evaluate our proposed approach, we have defined an automotive use case using *Autoware* (an open-source ROS 1/2-based software for self-driving vehicles)(Kato et al., 2018). Our developed approach in ROS2 will connect to Autoware in such a way that the data related to the localization and perception unit fit into the predefined modules. In the next step, the resource planner applies its mechanisms for the localization and the perception units.

The topic statistics feature of ROS2 provides the measurement of statistics for messages received by any subscription, including message period, message age, and data type value statistics (Thomas et al., 2014). By using this method, monitoring the subscription timing of each node related to a specific application is feasible. However, there are some limitations regarding this feature (e.g., enabling the topic statistics only via the subscriber) which must be solved by modifying the source code of the topic statistics (e.g., the topic statistics enabled by using ROS parameters).

Furthermore, using *high-resolution clock* in *C++ programming language* allows us to calculate a desired node latency as well as a system latency (Stroustrup, 2013). According to Figure 5, the node latency (e.g., clustering) can be calculated by listening to the time difference between its output and its input while utilizing *high-resolution clock*. Similarly, the system latency for the object detection stack can be computed by subtracting the Shape Extraction node output time and the Driver node input time utilizing the same clock. In the next step, the timing analysis regarding the node or the system is transferred to the resource planner. Subsequently, the timing violations are identified, and a warning proposal is sent to the nodes/system (here, object detection stack) which violate their timing requirements.

By following this proposed approach, we can identify the possible timing violations in the system as well as in the nodes based on our predefined timing requirements and notify the violated nodes or system.

5 CONCLUSION AND FUTURE STEPS

In this paper, we presented a resource planning proposal which not only identifies the timing violations that occurred in AI-based applications but also notifies the violated applications accordingly. Moreover, the resource planner can recognize the driving context of the vehicle by utilizing the pub/sub approach. The proposed resource planner includes three methods such as monitoring mechanism, context manager, and decision unit. In order to implement and evaluate this approach, ROS2 and Autoware were specified to be utilized respectively.

Our future work consists of the following steps, implementing monitoring, warning mechanisms using Autoware stack to monitor system and node latencies, measure message drops, and evaluate the accuracy of the proposed warning mechanism.

Secondly, integrating the resource distribution mechanism, in the case of any core fails in a multicore computational unit. In other words, the resource planner will recognize cores failure and then will assign the safety-critical application, which was running on the failed core, to the other non-safety-critical core, which has a non-safety-critical application running. Consequently, it will cause in distributing the consumption of the cores and minimizing the message drops, and timing latencies for safety-critical applications.

ACKNOWLEDGEMENTS

This work is part of the project ”KI-FLEX” (project number 16ES1027) which is funded by German Federal Ministry of Education and Research (BMBF) within the framework of the guidelines on promoting research initiatives in the field of “AI-based electronic solutions for safe autonomous driving.

REFERENCES

Askaripoor, H., Farzaneh, M. H., and Knoll, A. (2020). Considering safety requirements in design phase of future e/e architectures. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1165–1168. IEEE.

Fang, C.-C., Mou, T.-C., Sun, S.-W., and Chang, P.-C. (2018). Machine-learning based fitness behavior recognition from camera and sensor modalities. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 249–250. IEEE.

Hayat, S., Kun, S., Tengtao, Z., Yu, Y., Tu, T., and Du, Y. (2018). A deep learning framework using convolutional neural network for multi-class object recognition. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 194–198. IEEE.

Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S., Hirabayashi, M., Kitsukawa, Y., Monrroy, A., Ando, T., Fujii, Y., and Azumi, T. (2018). Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE.

Kong, L.-F. and Wu, J. (2005). Dynamic single machine scheduling using q-learning agent. In *2005 International Conference on Machine Learning and Cybernetics*, volume 5, pages 3237–3241. IEEE.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61.

Malik, S., Ahmad, S., Kim, B. W., Park, D. H., and Kim, D. (2019a). Hybrid inference based scheduling mechanism for efficient real time task and resource management in smart cars for safe driving. *Electronics*, 8(3):344.

Malik, S., Ahmad, S., Ullah, I., Park, D. H., and Kim, D. (2019b). An adaptive emergency first intelligent scheduling algorithm for efficient task management and scheduling in hybrid of hard real-time and soft real-time embedded iot systems. *Sustainability*, 11(8):2192.

Maruyama, Y., Kato, S., and Azumi, T. (2016). Exploring the performance of ros2. In *Proceedings of the 13th International Conference on Embedded Software*, pages 1–10.

Shulga, D., Kapustin, A., Kozlov, A., Kozyrev, A., and Rovnyagin, M. (2016). The scheduling based on machine learning for heterogeneous cpu/gpu systems. In *2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*, pages 345–348. IEEE.

Stroustrup, B. (2013). *The C++ programming language*. Pearson Education.

Thammachantuek, I., Kosolsombat, S., and Ketcham, M. (2018a). Support vector machines for road images recognition in autonomous car. In *2018 18th International Symposium on Communications and Information Technologies (ISCIT)*, pages 291–293. IEEE.

Thammachantuek, I., Kosolsomnbat, S., and Ketcham, M. (2018b). Comparison of machine learning algorithm's performance based on decision making in autonomous car. In *2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, pages 1–6. IEEE.

Thomas, D., Woodall, W., and Fernandez, E. (2014). Next-generation ROS: Building on DDS. In *ROSCon Chicago 2014*, Mountain View, CA. Open Robotics.

Werling, M., Ziegler, J., Kammel, S., and Thrun, S. (2010). Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. IEEE.

Yingzi, W., Xinli, J., Pingbo, H., and Kanfeng, G. (2009). Pattern driven dynamic scheduling approach using reinforcement learning. In *2009 IEEE International Conference on Automation and Logistics*, pages 514–519. IEEE.