Demo Abstract: MATLAB/Simulink Benchmark Suite for ROS-based Self-driving System

Shota Tokunaga
Graduate School of Engineering Science, Osaka University

Noriyuki Ota
Yoshiharu Tange
NEXTY Electronics Corporation

Keita Miura
Takuya Azumi
Graduate School of Science and Engineering, Saitama University

ABSTRACT

This paper proposes a MATLAB/Simulink benchmark suite for an open-source self-driving system based on Robot Operating System (ROS). In recent years, self-driving systems have been developed around the world. One approach to the development of self-driving systems is the utilization of ROS which is an open-source middleware framework used in the development of robot applications. On the other hand, the popular approach in the automotive industry is the utilization of MATLAB/Simulink which is software for modeling, simulating, and analyzing. MATLAB/Simulink provides an interface between ROS and MATLAB/Simulink that enables to create functionalities of ROS-based robots in MATLAB/Simulink. However, it is not been fully utilized in the development of self-driving systems yet because there are not enough samples for self-driving, and it is difficult for developers to adopt co-development. Therefore, we provide a MATLAB/Simulink benchmark suite for a ROS-based self-driving system called Autoware. Autoware is popular open-source software that provides a complete set of self-driving modules. The provided benchmark contains MATLAB/Simulink samples available in Autoware. They help to design ROS-based self-driving systems using MATLAB/Simulink.

ACM Reference Format:
Shota Tokunaga, Noriyuki Ota, Yoshiharu Tange, Keita Miura, and Takuya Azumi. 2019. Demo Abstract: MATLAB/Simulink Benchmark Suite for ROS-based Self-driving System. In *10th ACM/IEEE International Conference on Cyber-Physical Systems (with CPS-IoT Week 2019) (ICCPS '19), April 16–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3302509.3313315

1 INTRODUCTION

Self-driving systems become more complex every day as the increased number of functionalities and hardware such as cameras, radar sensors, LiDARs, and GNSS. One approach to the development of the complicated systems is the utilization of Robot Operating System (ROS) [7] [8]. ROS can perform distributed computing by inter-process communication using *nodes* and *topics*. The *nodes* represent individual processes, and the *topics* hold inputs from and output data to the *nodes*. In addition, characteristics of ROS, such as abstracting hardware and improving code reusability, make the development more efficient. A ROS-based self-driving system is Autoware [3] [5]. Autoware is popular open-source software for self-driving that provides a complete set of self-driving modules such as sensing, localization, detection, planning, and actuation.

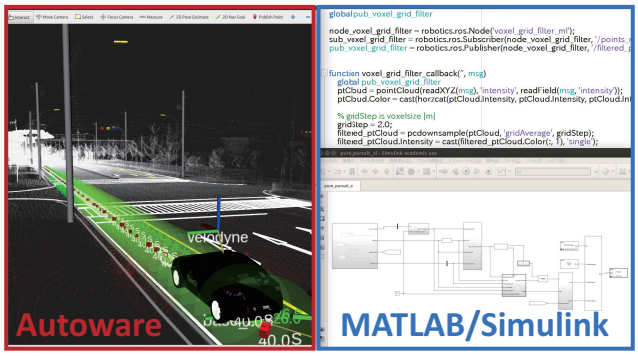

Figure 1: The co-simulation using Autoware and the provided benchmark.

In the automotive industry, the design of self-driving applications has often used MATLAB®/Simulink® which is software for modeling, simulating, and analyzing manufactured by MathWorks. MATLAB/Simulink can test and verify designed applications on ROS-based systems by using Robotics System Toolbox™ (RST) [6] which provides an interface between ROS and MATLAB/Simulink. The design using RST improves the development efficiency because it is unnecessary to generate ROS code and to incorporate it into ROS-based systems. However, RST has not been fully used to develop self-driving systems yet because there is not enough MATLAB/Simulink samples using RST for self-driving, so that it is difficult to co-develop using ROS-based systems and MATLAB/Simulink. Therefore, we propose a MATLAB/Simulink benchmark suite [4] for ROS-based self-driving systems, especially focused on Autoware. The proposed benchmark contains MATLAB code and Simulink models available in Autoware (Figure 1). It helps to design ROS-based self-driving systems using MATLAB/Simulink.

In this paper, after describing the details of the proposed benchmark, we describe future work.

2 MATLAB/SIMULINK BENCHMARK SUITE

The MATLAB/Simulink benchmark suite contains MATLAB code and Simulink models of which processes are executed as ROS *nodes* available in Autoware (Figure 2). In this section, we describe nodes

ICCPS '19, April 16–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6285-6/19/04…$15.00
https://doi.org/10.1145/3302509.3313315

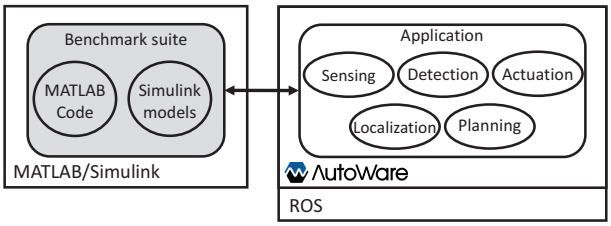

Figure 2: The system model of the proposed benchmark.

provided by the proposed benchmark for each module such as sensing, localization, detection, planning, and actuation. The provided benchmark is published on [4].

2.1 Sensing

Autoware mainly obtains information of road environments from sensors such as LiDARs and cameras. The obtained data are preprocessed by the sensing module. The proposed benchmark provides point cloud filter *nodes* and a image filter *node* as the preprocessing *nodes*.

Point cloud filter: The point cloud filter *nodes* downsample raw point cloud data as obtained from LiDARs. The proposed benchmark provides three downsampling nodes; using a box grid filter, with random sampling and without replacement, and using nonuniform box grid filter.

Image filter: The image filter *node* pre-processes image data as obtained from cameras. This *node* takes foggy images as input and outputs defogged images. The defogged images can be used for object detection and traffic light recognition, and the improvement in detection accuracy is expected.

2.2 Localization

The localization module is one of the crucial parts of self-driving systems. However the proposed benchmark currently contains only a *node*, we plan to provide multiple *nodes* in the future.

Connector: The connector *node* determines the velocity and pose of the vehicle used for subsequent processing. Autoware can calculate the vehicle velocity and pose in multiple methods such as using Controller Area Network (CAN) information, obtaining from the generated path, and estimating from localization. This *node* outputs the specified velocity and pose as current them.

2.3 Detection

Self-driving vehicles need to detect surrounding objects such as vehicles, people, and traffic signals. The proposed benchmark provides a detector *node* and tracker *nodes*.

Detector: The proposed benchmark provides a people detector *node* using aggregate channel features (ACF) [2]. This node detects people from image data and annotates the detected people with the bounding boxes and their detection scores. We plan to create the other object detector *nodes* such as vehicle detection and traffic light recognition.

Tracker: The tracker nodes track moving objects. The proposed benchmark provides the two tracker *nodes*: using point cloud data and using image data.

2.4 Planning

This module plans the path for self-driving. The path planning is broken into mission and motion planning.

Mission planner: The mission planner plans a global path based on the current location and the specified destination. The proposed benchmark provides a node which generates a global path considering traffic light.

Motion planner: The motion planner conducts local motion planning to determine the final path. The proposed benchmark provides a simulator *node* which generates a local path and determines the pose and the velocity of the self-driving vehicle to use Autoware in simulation mode.

2.5 Actuation

The self-driving vehicle follows the path generated by the motion planner. The proposed benchmark provides *nodes* generating actuation commands and filtering the commands.

Path following: The path following *node* generates actuation commands for the self-driving vehicle. The generation of the actuation commands is based on the pure pursuit algorithm [1].

Path filter: The path filter *node* conducts the filtering of the generated actuation commands to avoid dangerous acceleration, deceleration, and handling. The self-driving vehicle is controlled according to the filtered actuation commands.

3 CONCLUSION

In this research, we proposed a MATLAB/Simulink benchmark suite which contains MATLAB code and Simulink models available in Autoware. They help to design ROS-based self-driving systems using MATLAB/Simulink. In the future, we plan to add multiple samples to the provided benchmark. We will also publish sufficient document to help users introduce the provided benchmark.

ACKNOWLEDGMENTS

This work was partially supported by JST PRESTO Grant Number JPMJPR1751, Japan.

REFERENCES

[1] R Craig Coulter. 1992. *Implementation of the pure pursuit path tracking algorithm.* Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST.
[2] P. DollÃąr, R. Appel, S. Belongie, and P. Perona. 2014. Fast Feature Pyramids for Object Detection. *TPAMI* (2014).
[3] github.com. 2019. Autoware. https://github.com/CPFL/Autoware
[4] github.com. 2019. The proposed benchmark suite. https://github.com/CPFL/Autoware_Toolbox
[5] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. 2018. Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems. In *Proc. of ICCPS*.
[6] mathworks.com. 2019. robotics. https://www.mathworks.com/products/robotics.html
[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R Wheeler, and A. Ng. 2009. ROS: an open-souurce Robot Operatng System. In *Proc. of ICRA, Open-Source Software Workshop.*
[8] Y. Saito, T. Azumi, S. Kato, and N. Nishio. 2016. Priority and Synchronization Support for ROS. In *Proc. of ICCPS, Networks, and Applications.*