

Design and Implementation of ROS2-based Autonomous Tiny Robot Car with Integration of Multiple ROS2 FPGA Nodes

Hayato Mori¹, Hayato Amano², Akinobu Mizutani², Eisuke Okazaki³, Yuki Konno³, Kohei Sada³,

Tomohiro Ono^{2,4}, Yuma Yoshimoto⁵, Hakaru Tamukoh^{2,6}, Takeshi Ohkawa³, Midori Sugaya¹

¹ Graduate School of Engineering and Science, Shibaura Institute of Technology, Koto, Tokyo, Japan

² Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Kitakyushu, Fukuoka, Japan

³ Department of Embedded Technology, Tokai University, Minato, Tokyo, Japan

⁴ Research Fellow of Japan Society for the Promotion of Science, Chiyoda, Tokyo, Japan

⁵ National Institute of Technology, Kitakyushu College, Kitakyushu, Fukuoka, Japan

⁶ Research Center for Neuromorphic AI Hardware, Kyushu Institute of Technology, Kitakyushu, Fukuoka, Japan

Abstract— This paper introduces an autonomous tiny robot car equipped with a camera-based lane detection function and a traffic signal/obstacle, pedestrian recognition function. Each function is integrated by Robot Operating System 2 (ROS2), a middleware for robot system development. Autonomous driving without the need for a driver requires not only lane-following driving but also traffic signal recognition and obstacle recognition. These functions are implemented on FPGA, and we evaluated them. According to these results, the execution time of traffic signal recognition by FPGA was 1.2 to 3.4 times faster than CPU execution. YOLOv4 is used for obstacle recognition, which improved mAP by 3.79 points compared to YOLO v3-Tiny.

Keywords— *FPGA, Robot Operating System 2, YOLOv4-tiny*

I. INTRODUCTION

The research and development of fully autonomous driving are actively conducted in recent years. Field Programmable Gate Array (FPGA) is a reconfigurable device characterized by low power consumption and high parallel processing performance and is expected to be applied to autonomous driving car. The International Conference on Field-Programmable Technology (ICFPT) 2022 FPGA Design Competition aims to promote research on FPGA technology required for Level 5 autonomous driving system.

Robot Operating System (ROS) [1], a middleware commonly used in the development of robotic systems, has attracted attention in the field of autonomous driving, and Autware [2], an open-source autonomous driving software, uses ROS and ROS2. In this study, an FPGA implementation of image recognition processing wrapped with ROS/ROS2 is developed in order to improve the productivity of development by constructing a system with each FPGA component as a ROS/ROS2 node.

We developed our robot car based on Ultra_bot[3], which won first place in the ICFPT 2021 FPGA Design Competition. In addition to the functions of Ultra_bot, we implemented a traffic signal recognition function, and updated the YOLOv3-tiny used for obstacle recognition to YOLOv4-tiny. Furthermore, while Ultra_bot operated only the lane detection function as a ROS2-FPGA node, our robot car integrates not only the lane detection but also the traffic signal recognition into our system as a ROS2-FPGA node.

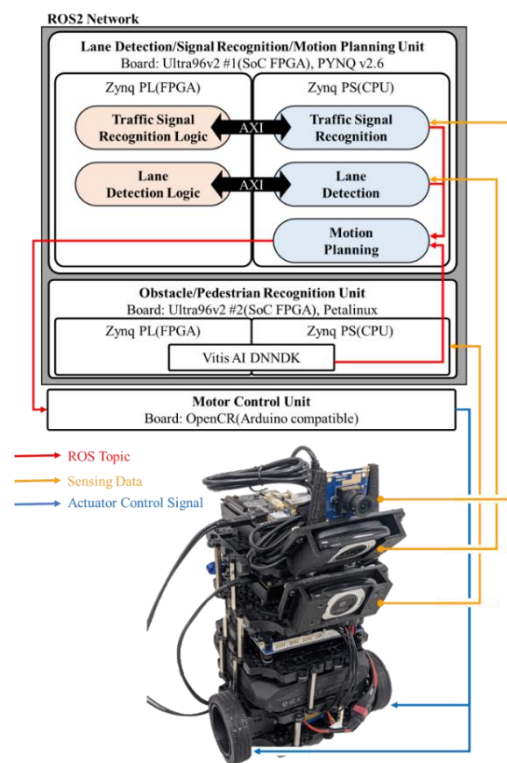


Fig. 1. Robot car configuration

In this paper, we describe the structure of the autonomous tiny robot car designed for design competition and the integration of multiple FPGA components using ROS2 for an autonomous driving system.

II. OVERVIEW OF OUR ROBOT CAR

This section describes the configuration of the robot car and the processing flow of the system. The main functions of the robot car are (1) lane detection, (2) obstacle/pedestrian recognition, (3) signal recognition, (4) motion planning, and (5) motor control.

A. Hardware Configuration

Fig. 1. shows the configuration of the robot car. Our robot car is designed based on TurtleBot3 Burger [4] and detached the LiDAR sensor and Raspberry Pi from that. Ultra96v2#1

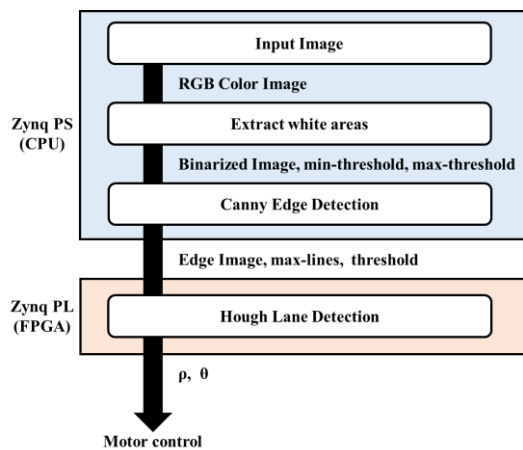


Fig. 2. Processing flow for lane detection

performs motor control, lane detection, and traffic signal recognition. Ultra96v2#2 performs obstacle and pedestrian recognition using YOLOv4 (You Only Look Once version 4). OpenCR, an Arduino-compatible board, controls motors. These components are integrated using ROS2, to improve debugging efficiency and development productivity by isolating faults points, for example, checking image data sent to a topic and visualizing ROS2 node graph.

B. Processing Flow

Ultra96v2#1 publishes lane detection results (theta, rho) and traffic light recognition results (red, yellow, green) as a ROS2 topic. Ultra96v2#2 sends recognized obstacle or pedestrian information (class label, bounding box) to Ultra96v2#1 via a ROS2 topic.

Ultra96v2#1 performs motion planning based on these data. The robot car follows the detected line. When the traffic signal turns red, the robot car stops until it turns green. Ultra96v2#1 controls the robot car when it receives recognition information from Ultra96v2#2. If it is recognized as a class of the pre-defined obstacles, the robot car avoids that. If it is a pedestrian, the robot car stops until the pedestrian passes through a crosswalk. These functions enable the robot car to drive autonomously.

III. LANE DETECTION

In the design competition, the robot car needs to follow the lane without a driver. The Hough transform is a straight-line detection algorithm commonly used in autonomous driving systems [5]. It is computationally intensive and spends runtime with CPU execution for embedded systems. Therefore, we implement the Hough transform process of lane detection on FPGA.

Fig. 2. shows the lane detection processing flow. First, this node converts the input image to an image in the HSV color space. Then, it extracts the white area where is the area corresponding to the lane from the image. After that, it performs edge extraction on the output image using the Canny method. Finally, it transfers the edge image to the FPGA part, detects a straight line using the Hough transform, and calculates the angle θ and length ρ from the origin of the detected straight line.

The lane detection node publishes these results to the motion planning node via the ROS2 topic.

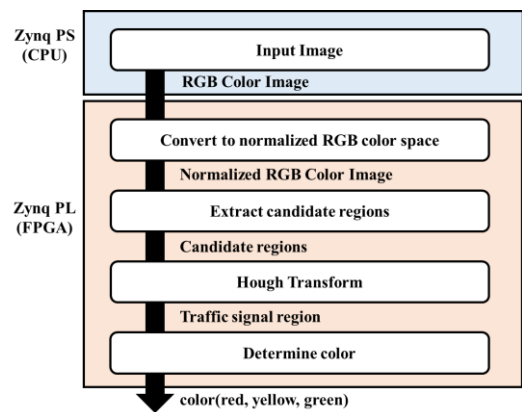


Fig. 3. Processing flow for traffic signal recognition

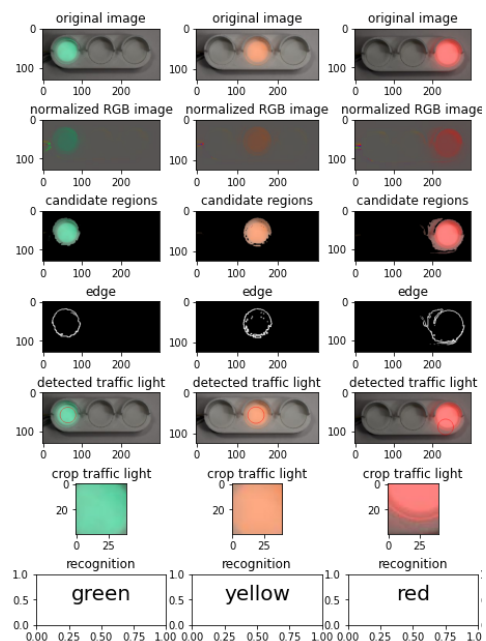


Fig. 4. Example images at each step of the traffic signal recognition

IV. TRAFFIC SIGNAL RECOGNITION

We implemented the traffic signal recognition in IP core on FPGA regarding a previous study [6] that combined color information and circle detection by the hough transform.

A. Processing Flow

Fig. 3. shows the processing flow of traffic signal recognition, and Fig. 4. shows examples of images at each step. First, the traffic signal node converts the input image to a normalized RGB color space to reduce the influence of lighting conditions. Next, it extracts candidate areas of traffic signals using RGB color information. For the extracted candidate areas, it detects circles using the Hough transform, and the color information (red, yellow, and green) is output as the traffic signal area. Also, we implemented traffic signal recognition in FPGA and CPU with the same algorithm.

B. Evaluation of Runtime

In this section, the runtime performance of the ROS2 node for traffic signal recognition is evaluated by comparing CPU execution with FPGA execution. In addition, we evaluated the

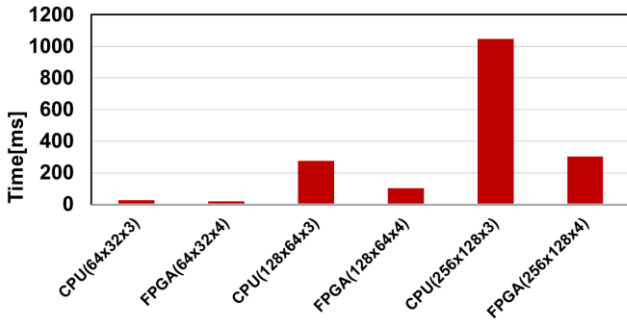


Fig. 5. Runtime CPU and FPGA execution

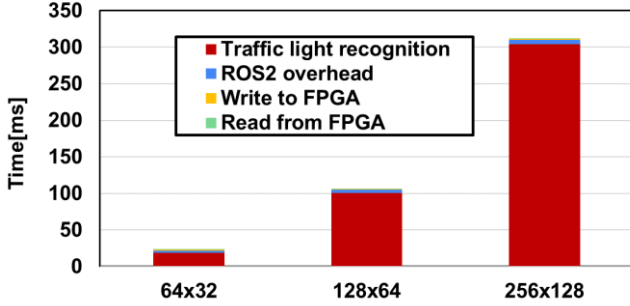


Fig. 6. Runtime of traffic signal recognition ROS2-FPGA node by each image

runtime of the ROS2-FPGA node for three different image sizes.

The Ultra96v2 boards mounted on the robot car are used for evaluation, with an Arm Cortex-A53 Quad Core (1.2GHz) CPU and Xilinx Zynq UltraScale+ MPSoC ZU3EG FPGA (set to 100MHz). The measurement targets are (1) time to recognize traffic signals, (2) ROS2 communication overhead, (3) time to write input data from the CPU to FPGA, and (4) time to read processing results from FPGA. We used three image sizes 256x128, 128x64 and 64x32 for the evaluation. The input image is 24-bit RGB (3 channels), but we padded the input image by 8 bits to achieve a 32-bit DMA transfer.

Fig. 5. shows a comparison of runtimes between CPU and FPGA execution. The runtime for traffic signal recognition with FPGA is 23.3 ms for a 64x32 image, 106.3 ms for a 128x64 image, and 303.9 ms for a 256x128 image, which is 1.2 to 3.4 times faster than CPU execution.

Next, Fig. 6. shows the results of the experiment using three different image sizes. The total latency of ROS2 overhead included pub/sub communication and FPGA I/O communication was 18.1% of the total runtime for 64x32, 5.5% for 128x64, and 2.5% for 256x128 images.

C. Hardware Resource Usage for Traffic Signal Recognition and Lane Detection

Table 1 shows the hardware resource usage for traffic signal recognition and lane detection. Traffic signal recognition and lane detection were implemented separately, and the total usage is the sum of the usage of each IP core and does not include other IP cores such as AXI DMA Controller.

The total hardware resource usage of the two IP cores was 100.5% for LUT, 19.4% for FF, 57.9% for BRAM18K, and 38.2% for DSP48E. The results show that the LUT exceeded 100%. Therefore, we will employ two FPGA boards to

Table 1. Hardware resource usage for traffic signal recognition and lane detection

	Lane Detection	Traffic Signal Recognition	Total
LUT	56,142	14,745	70,887/70,560(100.5%)
FF	16,578	10,831	27,409/141,120(19.4%)
BRAM18K	205	45	250/432(57.9%)
DSP48E	5	133	138/360(38.3%)

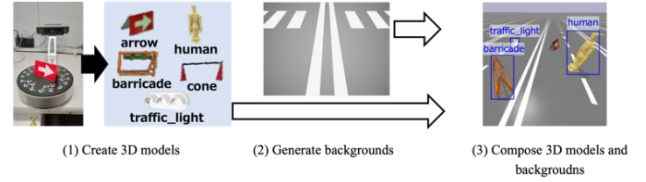


Fig. 7. Dataset generation method

overcome the resource shortage in the design competition of ICFPT 2022. As a future work, we plan to reduce the resources of the lane detection module, which accounts for 79% of the LUT usage, in order to implement it on a single Ultra96v2 board.

V. OBSTACLE/PEDESTRIAN RECOGNITION

A. YOLOv4 and Vitis AI

The obstacle or human recognition system with high accuracy and speed is necessary for the autonomous driving system. We use an object detection algorithm, You Only Look Once (YOLO) v4 [7].

YOLOv4 is an algorithm that detects object areas with high accuracy and low computational cost using neural networks. YOLOv4-tiny is an improved version for edge devices based on YOLOv4. The network structure is simplified, the parameters are reduced, and achieved up to seven times speedup compared to YOLOv4. We implemented the network using Vitis AI [8] provided by AMD Xilinx.

B. A Dataset Generation Method for the Training of the Obstacle Recognition System

In general, object recognition by deep learning requires a large amount of image and annotation data that indicates the position and label of objects in the image for training. Large amounts of high-quality datasets are necessary for a high-accurate object detection system by neural networks. Creating datasets by humans requires a large amount of time and human resources. In addition, the dataset created by humans can contain human errors and the quality of it may vary by annotators.

As a method of automating the creation of datasets, we modified the automatic dataset generation method on a 3D simulator proposed by Ono et al. [9].

The following is the flow of the dataset generation as shown in Fig. 7.

1) Scanning 3D Models

Create 3D models of obstacles using a 3D scanner EinScan-SP [10], because a provided 3D model has no texture.

2) Creating Environments

In this step, a simulator environment like the competition environment is prepared. The simulator environment has

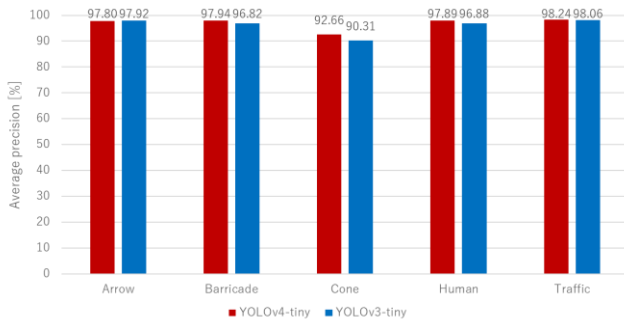


Fig. 8. Recognition results of 1) simulator images

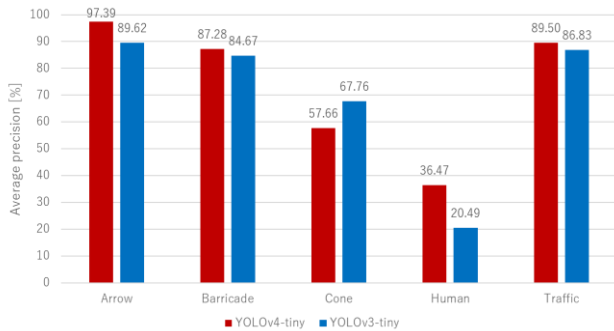


Fig. 9. Recognition results of 2) real images

background images (several types of the road) and obstacles that used in competition. Each 3D model (obstacles and doll) is randomly placed on a background image.

3) Dataset Generation

The simulation system captures the environment with randomly placed objects and outputs a dataset with annotation.

C. Evaluation of Classification Precision

We compared the mean average precision (mAP) of YOLOv4-tiny and YOLOv3-tiny, which we used in a previous competition. First, we trained both networks using 150,000 training data and 15,000 verification data. Then, in the experiment, we used 1) 15,000 test images generated on the simulator and 2) 100 test images taken in a real environment. To create dataset of real images, we put objects on the course used in the competition and we annotated the images manually.

1) Evaluation of Simulator Images

The result on 1) simulator images by category is shown in Fig. 8. The mAP on YOLOv4-tiny was 96.91 points, and YOLOv3-tiny was 96.00 points. The mAP of YOLOv4-tiny was improved by 0.91 points compared with YOLOv3-tiny.

2) Evaluation of Real Images

The result on 2) real images by category is shown in Fig. 9. The mAP on YOLOv4-tiny was 73.66 points, and YOLOv3-tiny was 69.87 points. The mAP of YOLOv4-tiny was improved by 3.79 points compared with YOLOv3-tiny.

The total accuracy was improved by using YOLOv4-tiny. We expect that the recognition mistakes will be reduced which lead to more stable robot car control and safety. The precision of the cone was low compared with other categories. Two cones and a bar compose this object and the detected bounding box contains more background images compared with other

objects. We consider that the ratio of the background image of an object affects the recognition precision.

VI. CONCLUSION

This paper describes the configuration and various functions of our ROS2-based autonomous robot car developed for the ICFPT 2022 FPGA Design Competition.

The ROS2 node for traffic light recognition recognizes traffic light colors based on color information and circle detection, referring to previous studies. The runtime for traffic signal recognition with FPGA is 23.3 ms for 64x32 images, 106.3 ms for 128x64 images, and 303.9 ms images for 256x128 images, which is 1.2 to 3.4 times faster than CPU execution.

The obstacle recognition with YOLOv4 recognizes obstacles and pedestrians on the road. The mAP for obstacle recognition is 73.66% points, which improved points by 3.79 points compared to YOLOv3-Tiny.

We will integrate IP cores for lane detection and traffic signal recognition in the future.

ACKNOWLEDGMENT

This work was partly supported by JST CREST Grant Number JPMJCR19K1, Japan.

REFERENCES

- [1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in Proc. ICRA Workshop Open Source Software, 2009, pp. 1–6.
- [2] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda and T. Hamada, "An Open Approach to Autonomous Vehicles," in IEEE Micro, vol. 35, no. 6, pp. 60-68, Nov.-Dec. 2015, doi: 10.1109/MM.2015.133.
- [3] Hayato Amano, Hayato Mori, Akinobu Mizutani, Tomohiro Ono, Yuma Yoshimoto, Takeshi Ohkawa, Hakaru Tamukoh, "A dataset generation for object recognition and a tool for generating ROS2 FPGA node," 2021 International Conference on Field-Programmable Technology (ICFPT), 2021, pp. 1-4, doi: 10.1109/ICFPT52863.2021.9609880.
- [4] ROBOTIS, "turtlebot3," <https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>, 14 Sep. 2022 Accessed.
- [5] Ismaïl El Hajjoui, Salah Mars, Zakariae Asrih, Aimad El Mourabit, "A novel FPGA implementation of Hough Transform for straight lane detection", Engineering Science and Technology, an International Journal, Volume 23, Issue 2, 2020, Pages 274-280, ISSN 2215-0986.
- [6] Masako Omachi and Shinichiro Omachi, "Traffic light detection with color and edge information", 2009 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 284-287, doi: 10.1109/ICCSIT.2009.5234518.
- [7] Alexey Bochkovskiy, Chien-Yao Wang, and Hong Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". arXiv preprint arXiv:2004.10934, 2020.
- [8] Xilinx, "Vitis AI," <https://www.xilinx.com/products/designtools/vitis/vitis-ai.html>, 14 Sep. 2022 Accessed.
- [9] Tomohiro Ono, Daiju Kanaoka, Tomoya Shiba, Shoshi Tokuno, Yuga Yano, Akinobu Mizutani, Ikuya Matsumoto, Hayato Amano & Hakaru Tamukoh (2022) "Solution of World Robot Challenge 2020 Partner Robot Challenge (Real Space)", Advanced Robotics, DOI: 10.1080/01691864.2022.2115315.
- [10] EinScan-SP, <https://www.einscan.com/desktop-3d-scanners/einscan-sp>, 14 Sep. 2022 Accessed