Traffic Signal Recognition and Application Algorithm for the Autonomous Vehicle in V2X Unable Areas

Yejin Gu
Department of ICT Convergence R&D
Korea Automotive Technology Institute
Cheonan, Republic of Korea
e-mail: yjgu@katech.re.kr

Daejun Kang
Department of ICT Convergence R&D
Korea Automotive Technology Institute
Cheonan, Republic of Korea
e-mail: djkang@katech.re.kr

Abstract— **Recognizing traffic signals is essential to operate autonomous driving on urban roads. The V2I is the priority in recognizing traffic signals in autonomous vehicles because it can send the exact Signal Phase and Timing (SPaT) of the traffic signal. However, there are many V2I unable areas currently, and even if it is available, it might have trouble due to communication delay or fail-operation. Accordingly, we present the traffic signals recognition framework using the convergence of Vehicle-to-Infrastructure(V2I) and camera detection to complement V2I. This study uses the signal recognized by the camera sensor in the area when the V2I signal is unable. By changing the existing one-way decision-making method, we implemented a customized communication system that dynamically changes in real time and fits the infrastructure situation. We identified that the proposed method works well by deploying an autonomous driving pilot system on the designated segment in Sejong-si, South Korea, where the V2I is partially available.**

Keywords-safety and traffic efficiency applications; object detection; C-V2X; image processing; CNN; deep learning

I. INTRODUCTION

Recognizing the driving environment is critical autonomous vehicles [1][2]. In particular, to operate autonomous vehicles on city roads, it is essential to recognize signals such as intersections. One method for recognizing the traffic signal is the vehicle to infrastructure (V2I) communication to receive traffic signals. Since accurate information is required for autonomous driving, Signal Phase and Timing (SpaT) information is preferentially used through V2I. However, there are V2I unable areas due to communication errors or insufficient system construction. Furthermore, though V2I is available, autonomous vehicles might have trouble recognizing traffic signals via V2I due to communication delay or breakdown. Accordingly, it is essential to complement recognizing traffic signals by using other sensors because of fail-safety.

In this paper, a camera is mainly used to complement V2I. In many studies, traffic signal recognition generally defines pre-segment a scene to find a region of interest. Ruta et al. [3] propose a general detector purification procedure based on mean moving clustering. Heuristics, in which traffic signal recognition utilizes prior knowledge of color and shape, have also been developed to recognize traffic signals [4][5].

However, it is challenging to process real-time camera-based signal recognition in real environments. Therefore, real-time image processing was performed using TenorRT [6], which can achieve fast computation speed with excellent performance. In addition, it is necessary to transmit accurate signal information to autonomous vehicles in a section where there is no V2I communication. Existing research showed the accuracy of signal recognition based on dataset, but it is not possible to show test results that are processed in real-time by being mounted on an actual autonomous vehicle [7][8]. In addition, research has not been conducted on the application of autonomous driving systems by fusion with V2I information in actual road areas.

In Section 1, a real-time signal recognition inference engine was implemented engine, and in Section 2, it was mounted on the system and fused with V2I in the real driving environment and applied to the autonomous driving system. The real-time video was taken with one camera, and the traffic light area was decided using computer vision-based technology.

II. METHODOLOGY

The dataset used in this study was collected from Sejong City, South Korea. Therefore, we propose an algorithm to detect a traffic signal in an image frame through a real-road environment as input, as shown in Fig.1 The camera was installed in the center of the vehicle's dashboard. The image is saved as a JPEG with RGB values of 2048x1536 pixels using a Blackfly S USB3. All images were captured under natural light conditions, such as variations in sunlight and complexity of the background. In particular, these conditions greatly increase the difficulty of detecting traffic lights in the field.

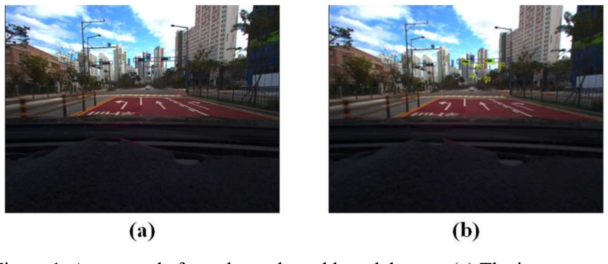


Figure 1. An example from the real-world road dataset. (a) The input camera image (b) The traffic light detection result.

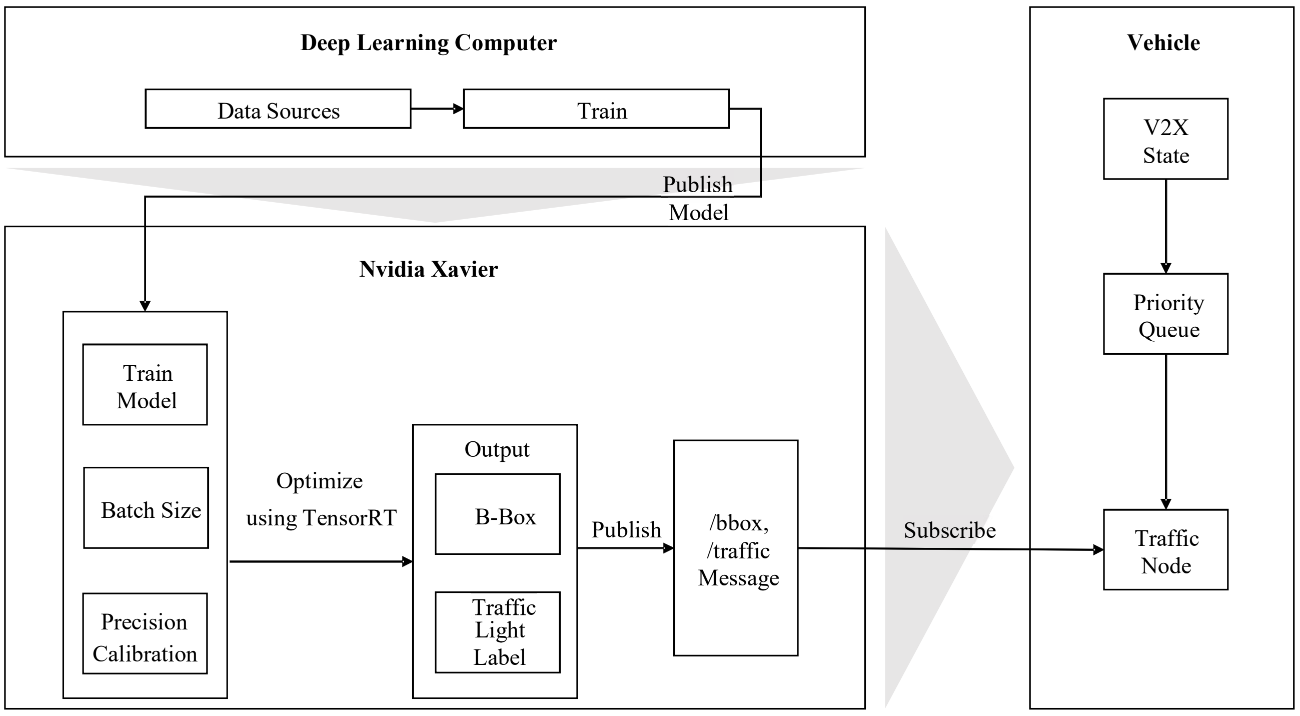


Figure 2. The overall pipeline of traffic light recognition system. Batch size : Total number of training examples present in a single batch. Precision : Precision calibration. B-Box : bounding box center offset and scales

The Figure 2 shows the overall flowchart of the training and detection process for constructing an applicable detection model in this study. Our training model is reliable, fast, capable of detecting signals of various scales, and generalizes to new data sets obtained from the training set in the real world.

A. Data Construction

Labeled data requires class labels and locations for all ground-truth bounding boxes in the training image. In order to apply deep learning, traffic lights in the real driving environment were manually extracted and identified. Therefore, the dataset we collected using the camera is 2M images annotated with labels and bounding boxes. The model used in this study can detect four object classes {stop, go, turn, turn&go}. The trained model was tested using an image resolution of 832x832 pixels with the batch size set to 32 for consistency with the training image resolution. Because of the small size of the traffic lights in the whole image, it can be difficult to recognize the correct category, so you need to resize it to the appropriate image size.

B. Traffic recognition system

We next describe the model used for traffic light recognition and how it fuses with V2I in real-time.

1) Detection model

To detect object in an image, we use an algorithm that detects in a specific area instead of processing the entire image in high resolution. Each grid cell is an image divided into squares. Slide a rectangular cell over the image to detect and classify objects. Then, each value calculated in the cell is analyzed for significant differences between the different cells, and the bounding area is calculated, outputting a bounding box. Each box is given as a tuple (x, y, w, h) and a confidence measure, where (x, y) is the center of the predicted box relative to the cell boundary, and (w, h) is the width and height of the bounding box relative to the image size. The neural network computes the features of the extracted object regions and classifies them into categories. This classification is scored on a predefined scale in the network based on the scores of the pre-classified objects. We use a model architecture that directly predicts the object bounding box of an image in a one- stage model.

Raw images are sent from the camera to the computing system NVIDIA Xavier for processing. During training, the input to the model is a 3 channel color image with an annotated bounding box around each traffic light. Image data is propagated through multiple convolutional layers. In order to define the boundary of the object in the learning model, a grid size 832x832 is first defined. Then an object prediction is performed for each cell. A vector of size $(C+5)\cdot B$ is created for each grid element, where B defines the number of masks in one layer and C defines the number of classes. Parameter 5 shows the center coordinates of the grid cell inner boundary, the height and width, and the probability that the boundary is defined correctly. This results in a prediction bounding box of size 106x106xB. The framework for the traffic light detection can be seen in Figure 3.

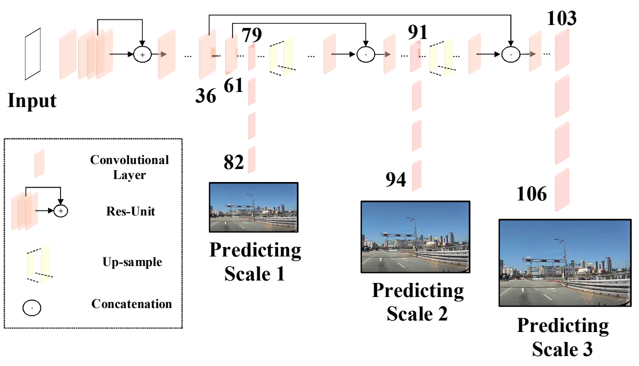


Figure 3. Framework for the traffic light detection

2) *Real-time Inference Engine (model acceleration)*

Optimization of the neural network architecture can be used to speed up processing and keep the accuracy at the same level. There are ways to speed up neural network processing without spending a lot of time changing programs. TensorRT is a platform that uses algorithms to optimize architectures and speed up neural network processing using algorithms that use the power of NVIDIA GPUs to increase computation. This is convenient in many cases because it can speed up programs without spending a lot of resources changing code. The process of using TensorRT to create a network for Xavier is shown in Figure 4.

TensorRT analyzes graphs that represent network models. If there are repeating elements in the graph, TensorRT merges them. As a result, the network size becomes smaller.

3) *Traffic Signal Recognition using the Convergence of V2I and Camera Detection*

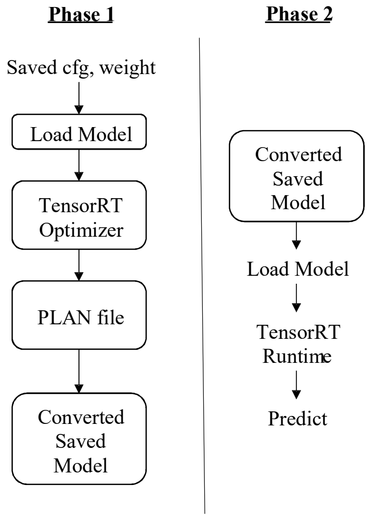


Figure 4. A flow of the network creating on TensorRT

In the proposed system, traffic lights on the actual road were captured by a camera attached to the vehicle. For object detection, a camera captures an object and then extracts features and learns it from a deep learning model. When a traffic light is identified, it is waiting in the form of a Robot Operating System (ROS) topic message so that data can be transmitted via ROS. In general, the braking distance information consists of distance-speed calculation, and the existing braking distance calculation map is presented in Table 1 [9]. Between 30 and 40 km/h, the braking distance is 10 to 18m, and according to equation (1), the response time is 1.38 to 2.5 seconds. In this study, 1.5 seconds is used as the required response time because the vehicle is traveling at 30 km/h approximately. If the V2I data is blank for 1.5 seconds, the priority is switched to the image detector, and the current signal state can be determined by receiving the image detector's ROS topic.

TABLE I. BRAKING DISTANCE CALCULATION MAP

Speed (km/h)	Braking distance (m)	Needed Response Time (s)
30	10	2.5
40	18	1.38

$$Needed\ Response\ Time = \frac{Braking\ distance}{Speed\ of\ Vehicle} \quad (1)$$

The raw image of camera array C, the distance D from the vehicle to the traffic light transmitted via V2I communication, is given as input; the data transmission period is represented by the tick variable. As long as image data exists, the algorithm iterates. Traffic lights are detected using a pre-trained model from the input image data. Traffic light information including the location and label of the bounding box is created and published in the form of a ROS topic. If data is not input for more than 1.5 seconds, the result derived from the learning model is subscribed to the vehicle.

The traffic light state communication algorithm is shown in Figure 5.

III. IMPLEMENTATION DETAILS

In this study, a TensorRT model was loaded on NVIDIA Jetson AGX Xavier, a powerful inference engine with JetPack 4.6 provided by NVIDIA. CUDA 10.2, cuDNN 8.2.1, TensorRT 8.0.1, and OpenCV 3.4.0 are required for TensorRT to work properly. The system uses the Python programming language. Also, a file containing the trained model weights (weights) and network configuration(cfg) is needed to create the plan file. We found that there is a significant difference between the number of frames in Xavier and the number of frames when TensorRT is applied.

Algorithm 1 Traffic Light State Communication Algorithm

```
1:   Data: raw camera image in C;
2:          stand-by time tick;
3:   Result: traffic light state topic
4:   initialization;
5:     while C is not empty do
6:       detect traffic light;
7:       publish traffic light state topic (Xavier);
8:       Calculate ticktime;
9:       if tick > 1.5 then
10:        subscribe traffic light state topic (Xavier → Vehicle) ;
11:      end
12:    end
```

Figure 5. Traffic Light State Communication Algorithm

Table 1 compares the hardware specifications of the NVIDIA developer kit. NVIDIA AGX Xavier is powered by the new NVIDIA Xavier processor and delivers over 20x performance and 10x energy efficiency, especially over the NVIDIA Jetson TX2.

As shown in Table 1, the number of real-time video frames was measured. The number of video frames is one of the important factors that can affect real-time image processing. According to the test results, when TensorRT is applied, 14.9 frames are compared to 6.6 frames when TensorRT is not applied. When TensorRT is a difference of about 2.25 times compared to when it is not applied. The accuracy of detection did not decrease after using TensorRT.

TABLE II. HARDWARE SPECIFICATION

specification	Jetson AGX Xavier	Jetson Nano	Jetson TX2
GPU	512 NVIDIA CUDA Cores and 64 Tensor Cores	128 NVIDIA CUDA Cores	256 cores NVIDIA Pascal GPU
CPU	8 cores NVIDIA Carmel Armv8.2 64bit CPU 8MB L2+4MB L3	Quad cores ARM Cortex-A57 MPCore Processor	Dual cores Denver 2 64 bit CPU & Quad cores Arm Cortex-A57 MPCore Processor
Memory	32GB 256bit LPDDR4x 136.5GB/s	4GB 64bit LPDDR4	8GB 128bit LPDDR4 59.7GB/s

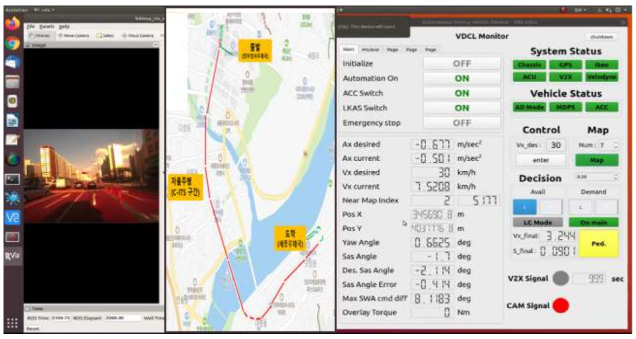
TABLE III. FRAME DIFFERENCE BETWEEN STANDARD AND TENSORRT

type	***FPS***
Without tensorRT	6.6
Adjust tensorRT	14.9

An accelerated engine model is generated and analyzed to determine one of four classes to quickly and accurately determine traffic light object recognition in a vehicle in real-time. A NVIDIA Xavier equipped with an algorithm is implemented in the vehicle and transmits the traffic light recognition result to the main server that controls the entire vehicle using ROS topic message. In this way, it can be easily applied to other vehicles, resulting in high scalability. As a result of the experiment, it was found that the proposed system provides accurate traffic signal information in the unabled section of V2I in the city image with a complex background in Figure 6.

IV. CONCLUSION

In this study, we proposed a traffic light recognition algorithm that applied TensorRT to minimize image resource usage in the section where V2I is unabled. Performance was analyzed based on real-world scenarios tested in neural networks. To compensate for sections without V2I using camera sensors, we used a deep learning architecture based on Convolution Neural Network (CNN) to label traffic lights. As a result of our experiments, our algorithm was able to accelerate in terms of speed with the same accuracy. This function was available in real-time driving, where processing speed is an important value. Our method is simple but effective, and yields significant improvements in demanding road traffic object detection and image classification datasets.


VDCL Monitor

Initialize OFF
Automation On ON
ACC Switch ON
LKAS Switch ON
Emergency stop OFF

Ax desired -0.677 m/sec²
Ax current -0.501 m/sec²
Vx desired 30 km/h
Vx current 7.5208 km/h
Near Map Index 2 5177
Pos X 345690.8 m
Pos Y 4037176.8 m
Yaw Angle 0.6625 deg
Sas Angle -1.7 deg
Des. Sas Angle -2.114 deg
Sas Angle Error -0.414 deg
Max SWA cmd diff 8.1183 deg
Overlay Torque 0 Nm

System Status: Chassis, GPS, Ibeo, ACU, V2X, Velodyne
Vehicle Status: AD Mode, MDPS, ACC
Control: Vx_des: 30
Map: Num: 7
Decision: Avail, Demand, LC Mode, On main
Vx_final: 3.244
S_final: 0.0901
Ped.
V2X Signal 999 sec
CAM Signal

Figure 6. Experiment driving and Trajectories

ACKNOWLEDGMENT

This work was supported in part by the Development of chip level solid-state sensor using nano-photonics structure for autonomous driving under grant 2020485, funded by the Ministry of Trade, Industry and Energy.

REFERENCES

[1] M. Priisalu, A. Pirinen, C. Paduraru, and C. Sminchisescu, “Generating scenarios with diverse pedestrian behaviors for autonomous vehicle testing,” *Conference on Robot Learning*, PMLR, pp. 1247-1258, 2022.

[2] R. Greer, J. Isa, N. Deo, A. Rangesh, and M. M. Trivedi, “On Salience-Sensitive Sign Classification in Autonomous Vehicle Path Planning: Experimental Explorations with a Novel

Dataset," *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 636-644, 2022.

[3] A. Ruta, F. Porikli, S. Watanabe, and Y. Li, "In-vehicle camera traffic sign detection and recognition," *Machine Vision and Applications*, vol.22.2, pp. 359-375, 2011.

[4] G. Piccioli, E. De Micheli, P. Parodi, and M. Campani, "Robust method for road sign detection and recognition," *Image and Vision Computing*, vol. 14.3, pp. 209-223, 1996.

[5] G. Loy, N. Barnes, D. Shaw, and A. Robles-Kelly, "Regular polygon detection," *Proc. of the 10th IEEE Int. Conf. on Computer Vision*, vol. 1, pp. 778-785, 2005.

[6] H. Vanholder, "Efficient inference with tensorrt," *GPU Technology Conference*, vol. 1, pp. 2, 2016.

[7] M. Omachi and S. Omachi, "Traffic light detection with color and edge information," *2009 2nd IEEE International Conference on Computer Science and Information Technology*, IEEE, pp. 284-287, 2009.

[8] M. P. Philipsen, M. B. Jensen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Traffic light detection: A learning algorithm and evaluations on challenging dataset," *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, pp. 2341-2345, 2015.

[9] P. Greibe, "Braking distance, friction and behaviour," *Trafitec, Scion-DTU*, 2007.