IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 9, 2022, accepted May 26, 2022, date of publication June 3, 2022, date of current version June 9, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3179999

Universal Detection-Based Driving Assistance Using a Mono Camera With Jetson Devices

DUONG NGUYEN-NGOC TRAN, (Graduate Student Member, IEEE), LONG HOANG PHAM, (Member, IEEE), HUY-HUNG NGUYEN, TAI HUU-PHUONG TRAN, HYUNG-JOON JEON, AND JAE WOOK JEON, (Senior Member, IEEE)

Department of Electrical and Computer Engineering, Sungkyunkwang University, Suwon 16419, South Korea

Corresponding author: Jae Wook Jeon (jwjeon@skku.edu)

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] under Grant 2020R1A2C3011286.

ABSTRACT Advanced Driver Assistance Systems (ADAS) are a collection of intelligent solutions integrated into next-generation vehicles to assist in safe driving. When building ADAS systems, the main goals are that they are stable, flexible, easy to maintain, and allow for error tracing. If a driving assistance algorithm is designed to be implemented on one machine or in one model, there is a potential disadvantage that if one component fails, then the entire system would stop. We work on modularizing the ADAS system to be flexible to accommodate any changes or improvements based on up-to-date requirements. Using advanced current edge (or network) devices, we propose a Detection-based Driving Assistance algorithm, which can collaborate or integrate with an existing system in a vehicle. The core of any process is to ensure that the system has a predictable level of functionality and that any misbehavior can be easily traced to the root cause. The proposed system shows fast, real-time performance on edge devices with limited computing power.

INDEX TERMS Advanced driver-assistance systems (ADAS), autonomous driving, scene understanding, situational awareness, edge device.

I. INTRODUCTION

Advanced Driver Assistance Systems are intelligent systems located inside a vehicle that assist the human driver in various ways. These systems present essential information about traffic, closures, congestion of the roads ahead, congestion levels, and suggested routes that avoid congestion. These systems can also assess driver fatigue and distraction, and then provide precautionary warnings or assess driving performance or make related recommendations. Advanced Driver Assistance Systems have become critical technologies studied in intelligent vehicles.

Most autonomous vehicle (AV) industry efforts focus on advanced driver assistance systems since they are the first step for fully self-driving cars. The reports [1], [2] show critical reasons for car accidents. They found that human factors are the primary reason or contributory element in 94% of car accidents. Vehicles, environmental factors, and other unknown reasons are responsible for 2% of crashes each. The three primary human factors most frequently cited in the study are speeding, inattentiveness, and improper lookout. Most of them can be avoided with ADAS. The role of ADAS is to prevent deaths and injuries by reducing the number of car accidents and reducing the severity of accidents that cannot be avoided. Essential safety-critical ADAS applications include pedestrian and vehicle detection/avoidance, lane departure warnings/corrections, and traffic light and traffic sign recognition. [3] shows that an ADAS system can have a crash avoidance effectiveness ranging from 9.3% to 33.3% for light vehicles [4], [5], while forward collision warnings (FCWs) may be able to prevent 23% to 50% of light vehicle rear-end crashes [6]. Moreover, [7] summarizes the effectiveness of twenty ADAS technologies of both light vehicles and heavy trucks. These lifesaving systems are vital to ensuring the success of ADAS applications, incorporating the latest interface standards, and running multiple vision-based algorithms to support real-time multimedia, vision co-processing, and sensor fusion subsystems.

These ADAS functions are usually based on one front camera or a front stereo-vision camera. Sometimes the

The associate editor coordinating the review of this manuscript and approving it for publication was Zhongyi Guo.

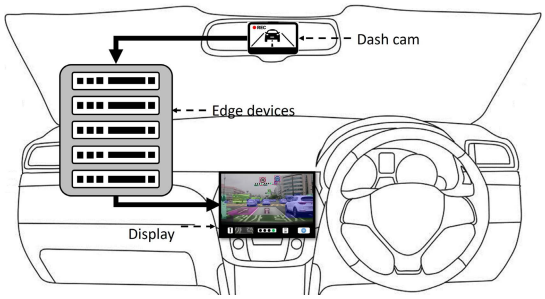


FIGURE 1. Structure of ADAS.

camera information is supplemented with information from other sensors, like light detection and ranging (LIDAR) or radio detection and ranging (RADAR). Against the front windshield, ADAS cameras are typically located inside the car behind the central rearview mirror. The ADAS camera field of view is located in the wiper area to keep the glass in front of the camera as clean as possible. Sometimes, RADAR sensing, vision sensing, and data fusion are combined in a single module.

This paper proposes ADAS using information obtained from a mono color camera and cluster edge devices, as shown in Fig. 1. The intelligent computation of ADAS is implemented using Machine Learning (ML) software that makes decisions based on critical observations of objects in surroundings. The cluster of edge devices operates the model, including traffic lights and signs, road markings and lanes, vehicles and pedestrians, synchronization, and scenarios. Then, the final caution appears on display.

We experimented with optimizing performance in the testing stage using the Korean dataset provided by KATECH (Korea Automotive Technology Institute) [8], [9] by showing the time required for processing critical scenarios in ADAS. There are several key contributions of this work:

- We implement a modularized system that can flexibly implement any change and any improvement based on up-to-date requirements.
- The proposed thread-based approach demonstrably maintains stable operation because the crash or failure of one thread-based module cannot affect the performance of other parallel modules. This approach is adopted because any process that affects human safety must guarantee that the system has a predictable operational level, and that any malfunctions can be effortlessly traced to the root cause.
- The work shows very acceptable real-time performance on edge devices with limited computational power.

In terms of the physical world's specific design of both machines and technology, Operation Technology (OT) is the physical machines themselves and the systems that control, monitor, and interface with them. In the OT, the Operational Level is the manufacturing operations management, which manages production workflow. An autonomous car must run continuously to capture any action outside the vehicle. If the vehicle process included sending and receiving cautions from the server, it would be dependent on the connection among many vehicles. Therefore, it would be problematic if there is any problem with the connection. We need the AI embedded and edge devices to isolable process the signal from the sensors on the vehicle. It should be emphasized that the satisfied requirements for real-world processing are in high demand, especially on embedded computational devices or edge device. Manufacturing companies provide various application-specific integrated circuits, such as field-programmable gate arrays (FPGAs), digital signal processors (DSPs), or graphics processing units (GPUs). In this study, the proposed method has been implemented and tested on an NVIDIA GPU-based computer and on NVIDIA embedded computing platforms of the Jetson TX2 [10], Jetson Xavier NX [11], and Jetson AGX Xavier [12]. The information for Jetson devices is shown in Table. 1.

TABLE 1. Hardware comparison of Jetson modules with Titan X.

Hardware feature	Jetson TX2	Jetson Xavier NX	Jetson AGX Xavier	Titan X
CPU (ARM)	4-core ARM Cortex-A57, 2-core Denver2	6-core ARM Carmel v8.2	8-core ARM Carmel v8.2	
GPU	256-core Pascal	384-core Volta	512-core Volta @ 1.37 GHz	3584 cores, 1417MHz
Memory	8 GB 128-bit LPDDR4, 58.3 GB/s	8 GB 128-bit LPDDR4, 51.2GB/s	16 GB 256-bit LPDDR4, 137 GB/s	12GB, 480.4GB/s
Storage	32 GB eMMC 5.1	16 GB eMMC 5.1	32 GB eMMC 5.1	
Tensor cores	-	48	64	
Power	7.5W / 15W	10W / 15W	10W / 15W / 30W	250W

The structure of this paper is as follows. Section II presents the related works. Section III presents the system architecture and describes each stage of the system. Next, the experiments and results are presented in Section IV. Finally, conclusions and directions for future work are discussed in Section V.

II. RELATED WORKS

In this section, we review common approach ADAS solutions and discuss their advantages and disadvantages. After describing the current boundaries, we demonstrate our strategy to improve these limitations.

A. DRIVING ASSISTANCE ON ONE MACHINE

The procedure mainly focuses on building a model that can work with many tasks while maintaining high accuracy. For further details, the model aims to learn better representations through information shared among multiple tasks. For illustration, A CNN-based multi-task

TABLE 2. Definition of classes in system.

Class	Details
Vehicle	Vehicle - Car
	Vehicle - Bus
	Vehicle - Motorcycle
	Vehicle - Truck
Pedestrian	Pedestrian
	Bicycle
Traffic light	Traffic light arrow - Red
	Traffic light arrow - Yellow
	Traffic light arrow - Green
	Traffic light arrow - Red arrow
	Traffic light arrow - Green arrow
Traffic sign	Traffic sign - Speed
	Traffic sign - Else
Road Mark	Road mark - Crosswalk
	Road mark - Number
	Road mark - Character
Road Mark Arrow	Road mark arrow - Straight
	Road mark arrow - Left
	Road mark arrow - Right
	Road mark arrow - Straight left
	Road mark arrow - Straight right
	Road mark arrow - U-turn
	Road mark arrow - Else

learning method mainly performs convolutional sharing of the network structure. MultiNet [13] completes the three scene perception tasks of scene classification, object detection, and segmentation of the driving area simultaneously by sharing an encoder and three independent decoders. DLT-Net [14] inherits the encoder-decoder structure and contributively constructs context tensors between sub-task decoders to share designated information among tasks. Moreover, [15] proposes one encoder for feature extraction and three decoders to handle the specific tasks. Meanwhile, it proposes a novel loss function to constrain the lane line to the outer contour of the lane area so that they will overlap geometrically. More importantly, the training paradigm of a multi-task model also requires very careful consideration. Reference [16] states that the joint training is appropriate and beneficial only when all those tasks are indeed related; otherwise, it is necessary to adopt alternating optimization strategies.

However, in [17] and [18], the multi-task models have many issues: they require significantly more computational power to obtain higher performance and accuracy. Compared with a single-task model, they achieve low accuracy because many tasks using only one feature extractor. Broadly, it is very difficult to simultaneously train many tasks and get a better result, This may be because the tasks must be learned at different rates or because one task may dominate the learning leading to poor performance on other tasks. For keeping good outcomes without losing the performance of any process, one solution is to apply an individual model for each task. The level of performance demanded by ADAS platforms will require increasingly larger and more powerful GPUs, thus impacting the manufacturing bills of materials for autonomous vehicles. To mitigate this expense, platform vendors have sought to increase the value and functionality of the GPU by using it to perform multiple workloads in the vehicle. Virtualized GPUs have obvious applicability for autonomous vehicles and ADAS scenarios, as a single GPU can power multiple applications, from the visualization of maps and operations of entertainment consoles to the processing of environmental sensor data to identify roadway obstacles. However, enabling multiple virtual operations from a single GPU in automotive applications is only safe and effective if the GPU has rock-solid support for hardware-accelerated virtualization. Virtualization software is most dependable when hardware enforces entirely separate managed address spaces for each virtual instance and enables the restart, or flushing, of a single instance that is not operating correctly. This workload isolation is key to allowing the shared use of the GPU while keeping critical software, such as driver-assistance systems, from being corrupted by any other process.

ADAS running on one shared machine has many issues. First, processing many tasks on one machine may lead to high computation requirements. Also, when we upgrade each part of ADAS, the whole system has to be updated. In a dangerous scenario, if a single task fails, it may lead to the crash of the whole system. Broadly speaking, it is hard to deploy a model that does not affect other models running on the same machine.

B. DEPLOYMENT ON EDGE COMPUTING

To keep satisfactory results without losing the performance of any operation and processing with low computation requirements, we can use Jetson clusters. In [19], they mention the embedded edge computing by deploying a deep model on Jetson embedded boards. They compare the outcome with the computer simulation and get a comparable result with high-speed performance. For additional information, [20] notes that the core benefits of deploying the trained machine learning (ML) model on edge devices include: (1) The edge hardware is more energy-efficient since it requires fewer energy resources than computer and server machines. (2) Edge-based inference hardware costs considerably less than other computational hardware such as field-programmable gate arrays (FPGA) and GPUs. In our paper, we deploy and run all modules of the ADAS on the Jetson cluster to keep the benefit of edge computing while obtaining good outcomes.

III. PROPOSED METHOD

This section describes each module we use in the system, including traffic lights and signs, road markings and lanes, vehicles and pedestrians, synchronization, and scenarios. Each module has a specific mission and provides important information for driving assistance. For a more detailed description of objects we consider in this system, Table. 2 shows a list of the considered objects from the parent to child leaves.

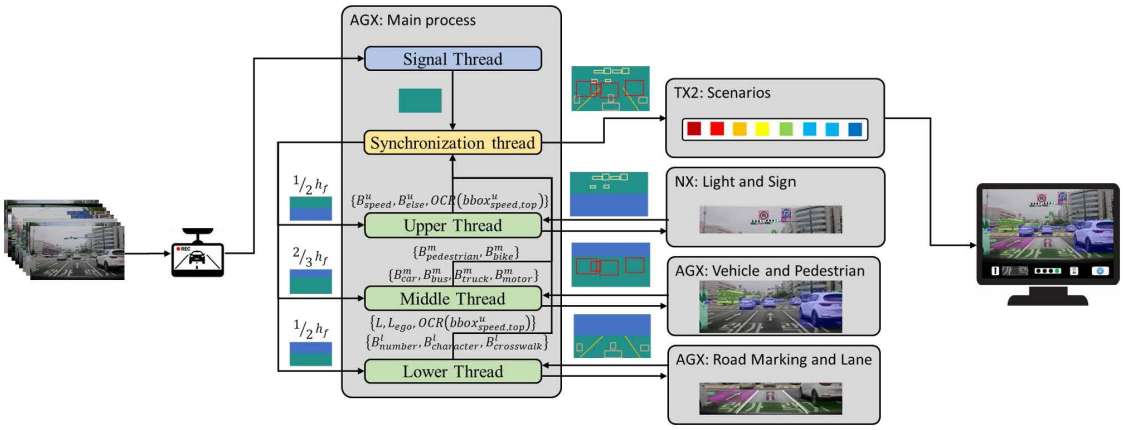


FIGURE 2. The framework of the system.

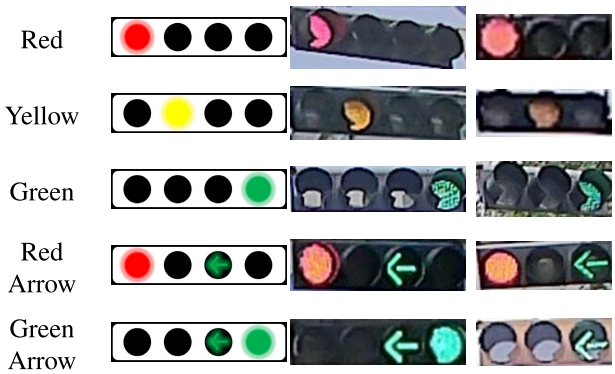


FIGURE 3. Different stages of traffic lights, with or without arrow.

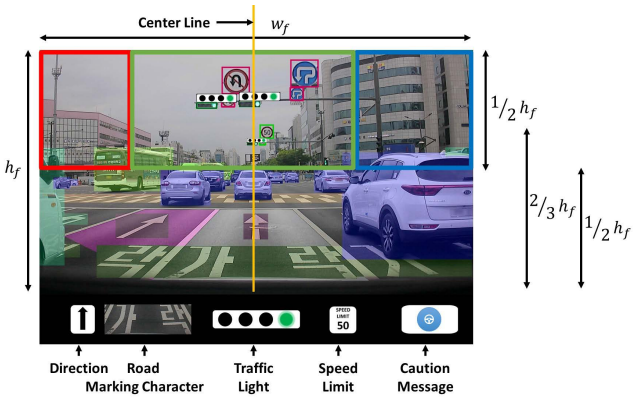


FIGURE 4. Display of ADAS.

A. PIPELINE OF SYSTEM

The overall details of the system pipeline are shown in Fig. 2. In the Main process, five threads have the job of keeping the system stable and running.

1) From the beginning, the frame from dashcam transfers to the Jetson AGX and is responsible for the primary process. The Signal thread's task is to connect and receive the buffer frame from a dashcam. After getting the input frame, the synchronizing thread does the job of cropping the image to a specific ratio, sending it to the three threads for subsequent processing before waiting for the threads to finish. If the connection is down, it will reconnect again to keep the system working.
2) The three threads, including the Upper Thread, Middle Thread, and Lower Thread, correspond with the Light and Sign module, Vehicle and Pedestrian module, and Road marking and Lane modules, respectively. Each of the three threads sends the suitably cropped frame to the proper Jetson edge devices for processing and waits for the information to return. If the information or result is not sent back in time, the thread will be terminated, and a new thread will be created to and the input signal is resent.
3) After all the threads successfully return, the Synchronization Thread combines and sends all the information with the same time frame to the scenarios module to investigate the situation and show the assistance information. Finally, all the assistance information appears in the display of the dashboard in a vehicle.

B. TRAFFIC LIGHT AND SIGN MODULE

The signal module covers the detection work for traffic lights and traffic signs. We only use a one-stage detector to make sure it runs in real-time. We do not need a complex or large model because the traffic light and traffic sign have a similar sample in most cases, and do not widely vary, like a pedestrian or car. The model we use is Scaled-YOLOv4 [21]. Likewise, we have used the TensorRT [22] inference optimizer and runner for better optimization and further reduce the inference time. We converted the scaled-YOLOv4 for model simplification and FP16 acceleration using TensorRT network definition APIs, which are based on an up-to-date version of the operating system of Jetson devices. For more detail, the model is first implemented in PyTorch. We train and export a weights file from the model. Afterward, we define

(a) Traffic Sign – Speed

(b) Traffic Sign – Else

FIGURE 5. Different types of traffic signs. The first row shows Traffic Sign – Speed, the second to the seventh row show the Traffic sign – Else.

the network using TensorRT, load the extracted weights file, and do inference tasks. The list of traffic lights and traffics sign classes are shown in Fig. 3 and Fig. 5. The input frame has the shape of $\{w_f, h_f, c_f\}$ which represent the width, height, and number of channels, respectively. For this module, we predefine three upper regions:

$$ROI_u = \{roi_l, roi_m, roi_r\} \tag{1}$$

with the corresponding factor u_l, u_m, u_r where $u_l + u_m + u_r = 1$. Each region has a height that is equal to that of the sent cropped image, and the width is calculated by $w_i = u_i * w_f$. The priority of the region is defined by $roi_m > roi_r > roi_l$, which means that the middle region is the most important part, the second is on the right, and the final is on the left. Each bounding box has a value that that matches that of Pascal VOC [23] format $bbox = \{x_{min}, y_{min}, x_{max}, y_{max}\}$. We use the three regions above to calculate the priority of traffic lights and traffic signs. Using the priority, we ascertain the leading traffic light or traffic sign for the assisted vehicle. In Fig. 4, the red, green, and blue bounding boxes represent the regions roi_l, roi_m, roi_r, respectively. Moreover, we notate that the point p is inside the bounding box, represented by the formula: $p \in bbox$

1) TRAFFIC LIGHT RECOGNITION

We only focus on non-occluded instances of traffic light detection to reduce ambiguities. All occluded traffic lights are removed from the training set and validation set to achieve this goal. In some cases, the deep learning networks still detect traffic lights on the boundaries of images. Our system uses two policies to decide whether or not a considered traffic light is irrelevant: In the case of the top boundary, more than half of the traffic light bulbs must be visible. When candidate signals are on the left or right edges of the images, all of the bulbs must be visible.

While in an intersection or roundabout, many traffic lights are detected and recognized with different stages. The issue is which one provides a signal meant for our car. We address this issue and increase recognition performance by adjusting the positions of ROIs based on individual image analysis. By simplifying [24], we identify the region-of-interest (ROI) containing the traffic light in each frame using the vehicular pose and a prior traffic light pose. As shown in Fig. 4, the order of precedence is green, blue, and red. In each ROI, the priority is above to below.

$$B^u_{light} = \left\{bbox^u_{light,1}, bbox^u_{light,2}, \ldots\right\} \tag{2}$$

which have the center point $\left\{c^u_{light,1}, c^u_{light,2}, \ldots\right\}$ with $c^u_{light,i} = \left\{x^u_{light,c,i}, y^u_{light,c,i}\right\}$. We find the top priority of traffic light bounding box by these conditions: the bounding box belongs to the upper region when $c^u_{light,i} \in bbox^u_{roi_j}$ and $roi_j \in \{roi_l, roi_m, roi_r\}$.

$$bbox^u_{light,i} > bbox^u_{light,j} \begin{cases} roi_i > roi_j \\ y^u_{light,i} < y^u_{light,j} \; with \; roi_i = roi_j \end{cases} \tag{3}$$

Therefore, the top priority traffic light, which is considered the current one for the vehicle, is:

$$bbox^u_{light,top} = argmax\, B^u_{light} \tag{4}$$

2) TRAFFIC SIGN RECOGNITION

Traffic signs have different structures and forms in different countries, the essential types of traffic signs are prohibitory, danger, mandatory, and text-based signs. The prohibitory, dangerous, or mandatory signs often have standard shapes, such as circles, triangles, and rectangles, and often have standard colors, such as red, blue, and yellow. The text-based signs usually do not have fixed shapes and contain informative text. Based on the KATECH dataset, we only consider the Traffic Sign Else class (including danger class, mandatory class, and prohibitory class) and the Traffic Sign Speed class:

- Traffic sign - Speed class: include the speed limited in range. We take the localized sign from the detection result and recognize and classify it (as shown in Fig. 5 - (a))
- Traffic sign - Else class: is designed to provide warnings vividly and instantly, including prohibitory or mandatory restrictions (as shown in Fig. 5 - (b))

In the traffic sign part, we only consider the main traffic sign speed for the vehicle because it is used for the post process in The Scenario Module. After detection, we have the list of bounding boxes

$$B^u_{speed} = \left\{bbox^u_{speed,1}, bbox^u_{speed,2}, \ldots\right\} \tag{5}$$

$$B^u_{else} = \left\{bbox^u_{else,1}, bbox^u_{else,2}, \ldots\right\} \tag{6}$$

FIGURE 6. Visualization of lane detection.

which have the center point $\{c^u_{speed,1}, c^u_{speed,2}, \ldots\}$ with $c^u_{speed,i} = \{x^u_{speed,c,i}, y^u_{speed,c,i}\}$. We find the top priority of the traffic light bounding box by these conditions: the bounding box belongs to the region if $c^u_{speed,i}$ in $bbox^u_{roi_j}$ and $roi_j \in \{roi_m, roi_r\}$

$$bbox^u_{speed,i} > bbox^u_{speed,j} \begin{cases} roi_i > roi_j \\ y^u_{speed,i} < y^u_{speed,j} \text{ with } roi_i = roi_j \end{cases} \tag{7}$$

Therefore, the top priority traffic sign - speed, which is considered as the current bounding speed of the vehicle, is:

$$bbox^u_{speed,\ top} = argmax\ B^u_{speed} \tag{8}$$

Because traffic sign – speed contains only digits, we use LPRNet [25] for fast Optical Character Recognition (OCR) and get the main speed limit value from the traffic sign – speed, which is

$$OCR\left(bbox^u_{speed,\ top}\right) \tag{9}$$

C. VEHICLE AND PEDESTRIAN MODULE

Automotive standards need to be followed for any system to obtain enhanced stability, predictability, and reliability. The most important priority is that the ADAS be safe and secure. Since the misbehavior of systems in a vehicle may result in hazardous situations to the passengers and other vehicles or pedestrians on the road, care should be taken to ensure system reliability. We use the TensorRT version of scaled-YOLOv4 to detect pedestrians and vehicles in the detection, which the same version as one in traffic light and sign module. We simplify the model in the same process as the method using in the Traffic Light and Sign module. The processing takes on $\frac{2}{3}$ of an image from the bottom up. Based on [26], reducing the computational complexity reduces the search space instead of limiting the window scale and position. Let us suppose that unnecessary portions of the image, including the image background and the areas of the scene where objects of interest are not expected, can be excluded from the search space. In that case, there will be considerable savings in computational cost. After detection, we get the set of bounding boxes: B^m_{car}, B^m_{bus}, B^m_{truck}, B^m_{motor}, $B^m_{pedestrian}$, and B^m_{bike}.

D. ROAD SURFACE MARKING AND LANE MODULE

1) LANE DETECTION

Reference [27] indicates that the lane detection algorithm must ensure good reliability, real-time operation, and robustness to meet practical requirements. With the development of autonomous driving technology, we need a lot of actual tests on the road. At the same time, a considerable overhead of resources will be consumed, and there are particular dangers to slow or erroneous computation.

We only keep $\frac{1}{2}$ image from below to extract the result, and we use Ultra-Fast Structure-aware Deep Lane Detection [28] to get the lane. We get the list of lanes

$$L = \{l_1, l_2, \ldots\} \tag{10}$$

in which $l_i = \{p^i_1, p^i_2, \ldots\}$. We find the ego lane to determine the main road marking in the post process. Ego-lane detection detects the current lane and its boundary and is mainly applied online so that autonomously driving cars can stay in the current lane with the aid of lane departure detection. The white segment is drawn in Fig. 6, and the purple cover is the regular lane. To find the ego-lane, we find two lines nearest the screen centerline (the orange line in Fig. 4) on the left and right. The variable for the screen centerline:

$$l_{sc}(x) = m_{sc} \times x + c_{sc} \tag{11}$$

The position of a point is

$$\text{pos}_{p_i} = l_{sc}(x_i) = \begin{cases} < 0 \text{ on the left} \\ 0 \text{ on the cente} \\ > 0 \text{ on the right} \end{cases} \tag{12}$$

The number of points on the left is:

$$num_{p_l} = \sum_i^n \text{pos}_{p_i} < 0 \tag{13}$$

and on the right is:

$$num_{p_r} = \sum_i^n \text{pos}_{p_i} > 0 \tag{14}$$

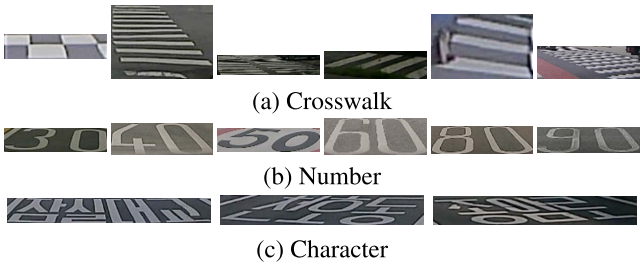
To find whether the line is on the left or right of the screen centerline, we use:

$$\text{pos}_{l_i} \begin{cases} left & \text{if } num_{p_l} > num_{p_r} \\ right & \text{if } num_{p_l} > num_{p_r} \end{cases} \tag{15}$$

To find the nearest distance, we use the Hausdorff distance:

$$d_H(l_i, l_{sc}) = \max\left\{\max_{p^i \in l_i}\left\{\min_{p^{sc} \in l_{sc}} d\left(p^i, p^{sc}\right)\right\}, \max_{p^{sc} \in l_{sc}}\left\{\min_{p^i \in l_i} d\left(p^i, p^{sc}\right)\right\}\right\} \tag{16}$$

The nearest distance on the left is $argmin\ d_H\left(l_{i,left}, l_{sc}\right)$, and the nearest distance on the right is $argmin\ d_H\left(l_{i,right}, l_{sc}\right)$.



(a) Crosswalk

(b) Number

(c) Character

FIGURE 7. Different types of road marks.

The ego lane is

$$L_{ego} = \left\{ l_{argmin\ d_H(l_{i,left}, l_{sc})}, l_{argmin\ d_H(l_{i,right}, l_{sc})} \right\} = \left\{ l_{left,sc}, l_{right,sc} \right\} \quad (17)$$

2) ROAD MARKING RECOGNITION

Based on [29], to avoid the conflict of arrows in traffic lights with information on the road, we only consider five main directions of a traffic arrow, including Straight, Left, Right, Straight Left, Straight Right (as shown in Fig. 8). In addition, we add two more. These are U-turn and Else. Traffic arrow – Else means that other directions are different from the above arrows. Moreover, we consider the other road mark, such as crosswalk, number, and character (as shown in Fig. 7). The "Road mark - Number" shows the speed limit of the current lane. Using the lane information and the number given by OCR, the system can show the speed needed for this lane. After getting the detection, we get the detections: B^u_{number}, $B^u_{character}$, and $B^u_{crosswalk}$ which have the center point $\left\{c^l_{number,1}, c^l_{number,2}, \cdots\right\}$ with $c^l_{number,i} = \left\{x^l_{number,c,i}, y^l_{number,c,i}\right\}$, and the point $\left\{c^l_{character,1}, c^l_{character,2}, \cdots\right\}$ with $c^l_{character,i} = \left\{x^l_{character,c,i}, y^l_{character,c,i}\right\}$. We only consider the post-process for Number and Character Road Markings. We filter these bounding boxes and keep processes belonging to the ego lane, L_{ego}, by finding the center of the bounding box of the polygon created by two lines of the ego lane. The list of points of the polygon is:

$$P_{ego} = \left\{ p^l_{left,sc,1}, p^l_{left,sc,2}, \ldots, p^l_{right,sc,1}, p^l_{eight,sc,1}, \ldots \right\} \quad (18)$$

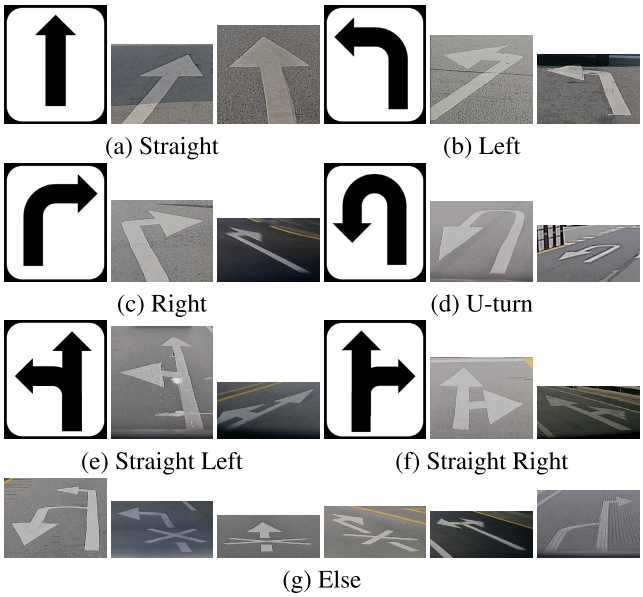
The list of bounding boxes that remain after filtering are:

$$B^l_{number,ego} = \left\{ bbox^u_{number} \middle| c^l_{number} \in P_{ego} \right\} \quad (19)$$

$$B^l_{character,ego} = \left\{ bbox^u_{character} \middle| c^l_{character} \in P_{ego} \right\} \quad (20)$$

We find the top priority of the Number and Character by these conditions:

$$bbox^l_{number,i} > bbox^l_{number,j} \quad where\ y^l_{number,c,i} > y^l_{number,c,j} \quad (21)$$



(a) Straight (b) Left

(c) Right (d) U-turn

(e) Straight Left (f) Straight Right

(g) Else

FIGURE 8. Different types of road mark arrows.

$$bbox^l_{character,i} > bbox^l_{character,j} \quad where\ y^l_{character,c,i} > y^l_{character,c,j} \quad (22)$$

Therefore, the top priority is given by the pair:

$$bbox^l_{number,\ top} = max\ B^l_{number,ego} \quad (23)$$

$$bbox^l_{character,\ top} = max\ B^l_{character,ego} \quad (24)$$

We next consider the instruction from "Road mark - Character." The letters and words from each country are different, so using the individual modules can help change, we choose to show the closet Character on the panel.

E. SYNCHRONIZATION MODULE

Elaborating on the content of Fig. 2, we implement our proposed solution from the video input in a multi-threaded CPU-, GPU-utilizing manner. As the four modules mentioned above are mostly independent of one another, we take advantage of the current hardware development for Internet of Things (IoT) devices with both multi-core CPU and CUDA-compatible GPU support and propose a thread-level parallelism framework [30]. Our approach best performs with at least five threads and a GPU, where each thread processes inputs continuously throughout the given frame sequence. From the primary device (Jetson AGX), we deploy Synchronization Threads for receiving and sending signals for all necessary components of the solution (other Jetson devices) and deploy the other four threads for continuous processing.

- **Thread #1 (Signal Thread)**: This thread is responsible for receiving the input frame. This thread has to make sure that the connection with the camera is stable and online. If there is an error in the connection or buffer frame from the camera, this thread reconnects and waits for the signal.

- **Thread #2 (Synchronization Thread)**: This thread is responsible for sending and receiving the crop frame and results from other threads. It is the most crucial thread because it can terminate and create a new thread. The thread must keep the time frame in order and the results corresponding with this time frame. After receiving the input frame, the thread crops it and sends it to the other threads, and then it waits for the results. After getting a result back, the thread packs it with the corresponding time frame and sends it to the Scenarios module in the Jetson TX2. If one of the modules does not send the result on time, the thread calls the primary process to terminate the non-responsive thread and creates a new one.
- **Thread #3 (Upper Thread)**: This thread is responsible for sending and receiving the Light and Sign module result from the Jetson NX. The edge device performs the object detection for the traffic light and traffic sign by running the traffic light and sign module (mentioned above). Finally, the result is sent back to the edge device primary process, and the upper thread handles it and sends it to the Synchronization Thread.
- **Thread #4 (Middle Thread)**: This thread is responsible for sending and receiving the result from the Vehicle and Pedestrian module from the Jetson AGX. The edge device performs object detection for cars, buses, trucks, and pedestrians by running the vehicle and pedestrian module (mentioned above). Finally, the result is sent back to the edge device primary process, and the middle thread handles it and sends it to the Synchronization Thread.
- **Thread #5 (Lower Thread)**: This thread is responsible for sending and receiving the result of the Road Marking and Lane module from the Jetson AGX. The edge device performs both lane detection and road marking detection. Finally, the result is sent back to the edge device primary process, and the lower thread handles it and sends it to the Synchronization Thread.

In case of adding a new module in the future, a new thread can be added to the current threads, and the Synchronize Thread would handle it.

F. SCENARIOS MODULE

This section shows the work of the scenarios module on the Jetson TX2. The module processes all the information and displays it along with the information and the assistance signal [31], [32]. The scenarios can be updated and customized according to the regulations of each country. From Fig. 4, the display of ADAS has five components, and each of them has the result from this module.

- **Direction panel**: The panel shows the current direction of the lane based on the default or the road marking arrow on the ego lane (shown in Fig. 8).
- **Road marking character panel**: The panel shows any words or characters in the ego lane. If we have many of them, it shows the lowest and closest word in the ego lane.
- **Traffic light panel**: This panel shows the current traffic light. The current light governing the vehicle is the result of the traffic light module process.
- **Speed limit panel**: The panel shows the speed limit. The default value is 60 kph, which is the speed limit for vehicles on most city streets and rural two-lane roads in Korea. The value is changed based on the Traffic Light and Sign module and Road Marking and Lane module. We show the speed limit with kilometer per hour (km/h) units.
- **Caution panel**: The panel shows the assistance message from the ADAS. There are three main caution messages: NORMAL, WARNING, DANGER. The NORMAL indicator means that the driving condition is safe. The WARNING indicator means that the driver must consider slowing down and pay more attention to obstacles or objects on the road. The DANGER indicator means that the driver should be ready to brake, and potential collisions and danger lie in front of the car. The danger signal indicates danger for the driver, pedestrians, or other drivers.

We prioritize caution: the first priority is to treat the pedestrian with caution and the second is to treat the vehicle with caution. The traffic light and traffic sign serve as supporting pieces of information that encourage driver caution.

1) PEDESTRIAN CAUTION

The scenarios module always initiates before the detection of any case of a pedestrian. For many cases, the caution signal regarding a pedestrian is stop. For example, such cases could be the pedestrian in the crosswalk in front of the car, or the pedestrian in any lane (especially, ego lane.). The way to determine the pedestrian state is by using Intersection over Union (IoU) for object detection. The equations to check the pedestrian's state in front of a car are

$$IoU\left(bbox^{m}_{pedestrian,i}, bbox^{l}_{crosswalk,j}\right) = bbox^{m}_{pedestrian,i} \cap bbox^{l}_{crosswalk,j} \tag{25}$$

$$IoU\left(bbox^{m}_{bike,i}, bbox^{l}_{crosswalk,j}\right) = bbox^{m}_{bike,i} \cap bbox^{l}_{crosswalk,j} \tag{26}$$

$$IoU\left(bbox^{m}_{pedestrian,i}, L_{ego}\right) = bbox^{m}_{pedestrian,i} \cap L_{ego} \tag{27}$$

$$IoU\left(bbox^{m}_{bike,i}, L_{ego}\right) = bbox^{m}_{bike,i} \cap L_{ego} \tag{28}$$

For example, in the case of Fig. 9 - (a)(b), the pedestrians go over the crosswalk, (25) has a value greater than zero. In the case of Fig. 9 - (c), the bike goes over the ego lane, and the

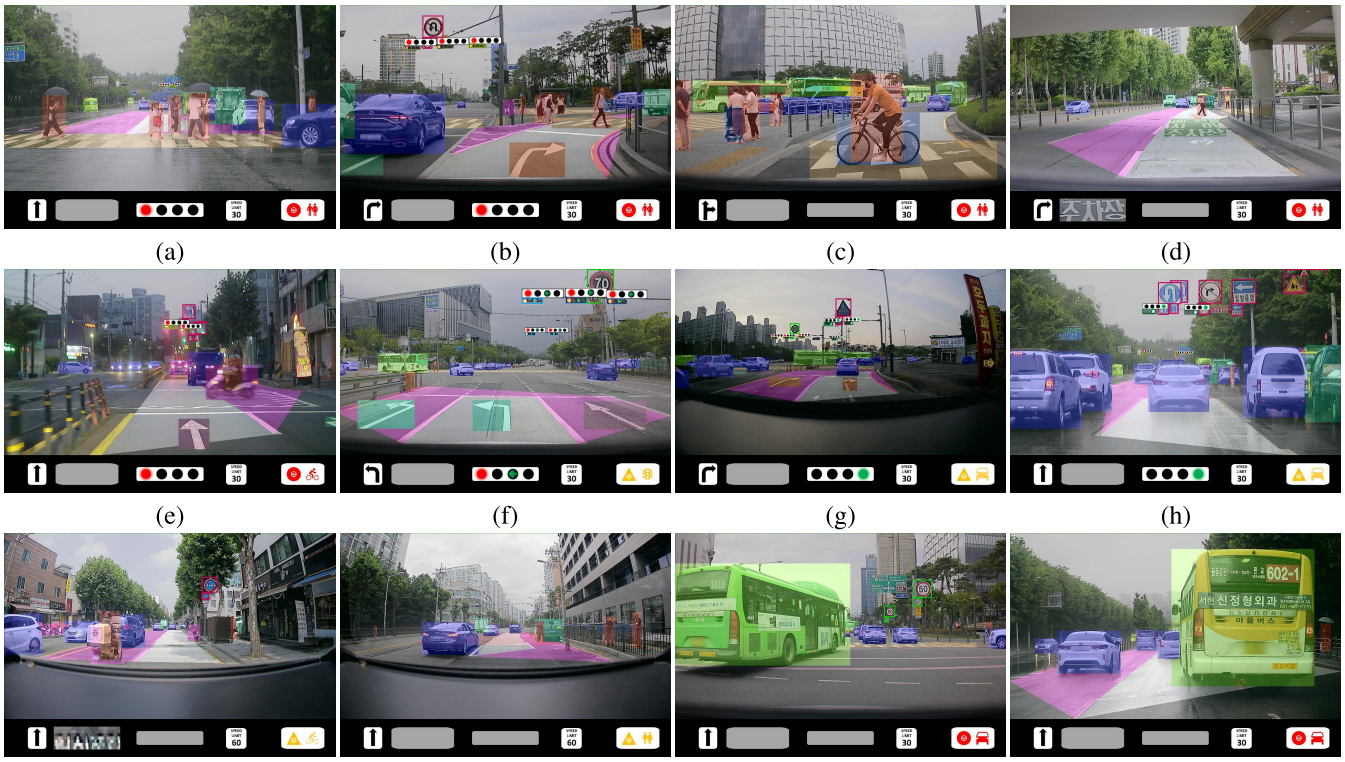


FIGURE 9. Specific cases. (a)-(c) The pedestrians are in crosswalks. (d) The pedestrian is in the ego-lane. (e) A motorbike is in front of the vehicle (f)-(h) Multiple and different types of traffic lights. (i) The motorbike is parked on the pavement. (k)-(l) The bus suddenly changes lanes.

value of (26) is greater than zero. In the case of Fig. 9 - (d), the bike goes over the ego lane, and the value (27) is greater than zero. If one of these IoUs has a value greater than zero, the caution is DANGER. If the IoU with the lane (not ego-lane, (29) and (30) are still zero):

$$IoU\left(bbox^{m}_{pedestrian,i}, L\right) = bbox^{m}_{pedestrian,i} \cap L \quad (29)$$

$$IoU\left(bbox^{m}_{bike,i}, L\right) = bbox^{m}_{bike,i} \cap L \quad (30)$$

If the case (29) has a value more than zero (as shown in Fig. 9 - (j)), the caution is WARNING.

2) VEHICLE CAUTION

For vehicle caution, the top priority is motorbike, car, bus, and the last is truck. We have a union of detection in vehicle detection.

$$B^{m}_{vehicle} = B^{m}_{motor} \cup B^{m}_{car} \cup B^{m}_{bus} \cup B^{m}_{truck} \quad (31)$$

The equation to check the position of the vehicle is:

$$\text{pos}_{vehicle,i} = \frac{y^{m}_{vehicle,i}}{h_f} \quad (32)$$

The caution is WARNING whenever any vehicle on the ego lane in more than $\frac{1}{2}h_f$ (Fig. 9 - (k)). And the caution is DANGER whenever any vehicle on the ego lane in more than $\frac{2}{3}h_f$ (as shown in Fig. 9 - (l)).

IV. EXPERIMENT RESULT

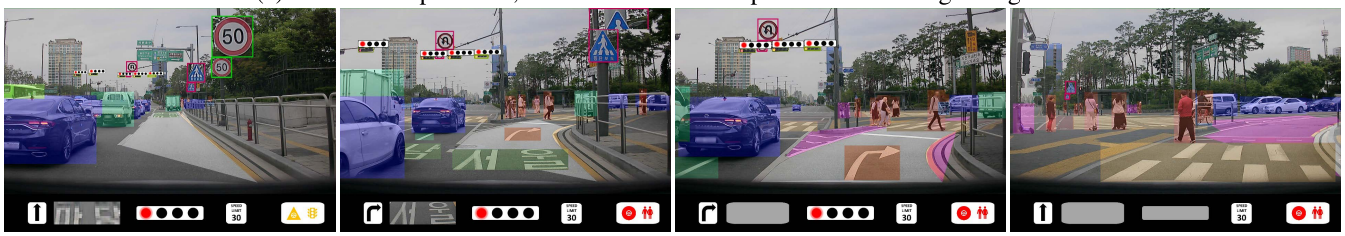
In this section, we evaluate the performance in analyzing some scenarios while driving to show the effectiveness of the proposed ADAS. In addition to showing caution messages in the display, we show the speed performance of the selected model running on Jetson devices.

A. EXPERIMENTAL SETTING

The experiment using the KATECH dataset contains more than 160,000 images for the training, validation, and testing sets. The image resolutions in the dataset are 1280 × 720 pixels, 1280 × 672 pixels (1280), or 1920 × 1080 pixels. The times of the captured images are daytime, dawn, and nighttime. Moreover, the dataset includes sunny, overcast, and rainy weather. We set up three Jetson AGX Xavier devices, one Jetson Xavier NX device, and one Jetson TX2 device. The input resolution we used was 1280, and the output is displayed at the end. We labeled 10000 images for the scenario test case. The label is based on four panels in the final display. Six direction types are labeled for the Direction panel, including Straight, Left, Right, Straight-Left, Straight-Right, and U-turn. For the Traffic Light panel, the varieties are shown in Fig. 3, including Red, Yellow, Green, Red Arrow, and Green Arrow. For the Speed Limit panel, we labeled it in the range of 30 to 150 kph. For the Caution Message panel, we labeled for Driving messages (including Safety, Warning,

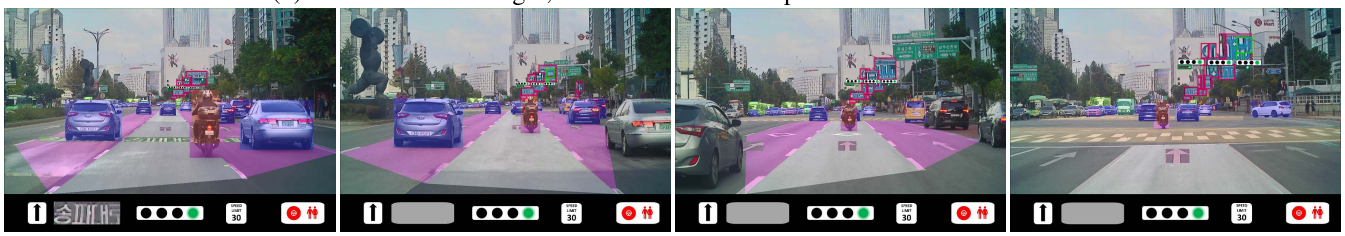
(1) (2) (3) (4)

(a) The vehicle proceeds, while cautious of the pedestrian crossing the ego lane.



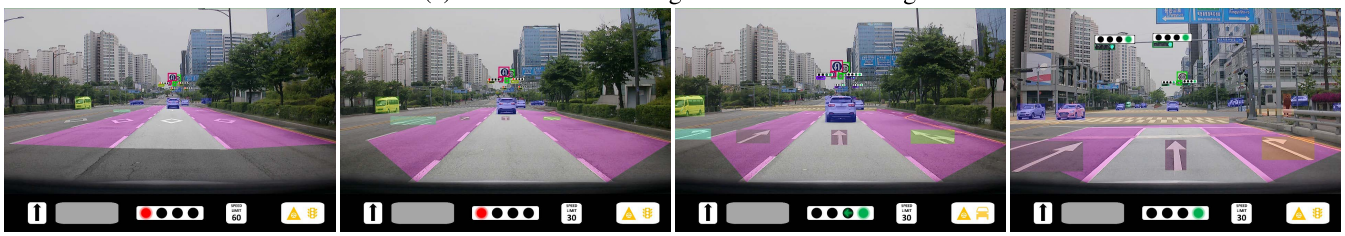
(1) (2) (3) (4)

(b) The vehicle turns right, while cautious of the pedestrian in the crosswalk.



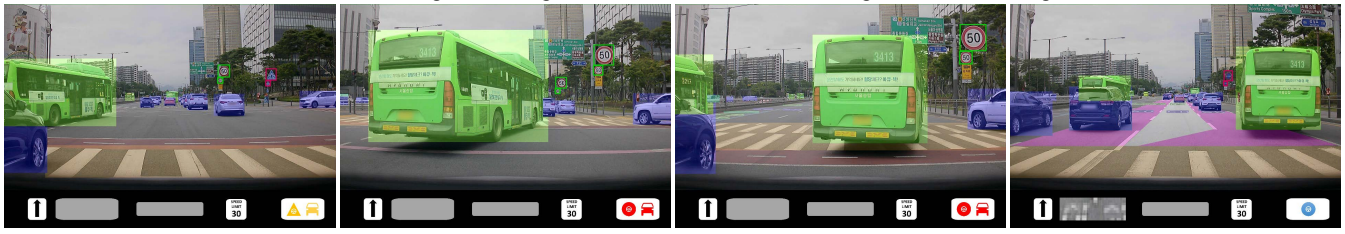
(1) (2) (3) (4)

(c) The motorbike changes lanes while running.



(1) (2) (3) (4)

(d) The vehicle goes through the intersection with a change in the traffic light.



(1) (2) (3) (4)

(e) The bus suddenly changes lanes.

FIGURE 10. **Visualization of scenarios in the series of frames.**

Danger) and Icon message (including Vehicle, Cycle, and Pedestrian).

B. CASE SCENARIOS

Fig. 10 shows some scenarios on the KATECH dataset that we have highlighted:

In the case of Fig. 10 - (a), the pedestrian is crossing the street without observation. The ADAS detects and alerts the driver about the pedestrian in the crosswalk with the WARNING signal. In the third image, the vehicle is near the walker, so the ADAS alerts DANGER and asks the driver to be ready to stop. In the last image, while the pedestrian is on

TABLE 3. Accuracy of panels.

Panel	Accuracy
Direction	98.1
Traffic Light	97.5
Speed Limit	95.6
Caution Message - Driving - Safety	94.3
Caution Message - Driving - Warning	98.5
Caution Message - Driving - Danger	95.1
Caution Message - Icon - Warning	94.8
Caution Message - Icon - Danger	92.7

TABLE 4. Accuracy (mAP) comparison of traffic object detection.

Classes	Our	DLT-Net	YOLOP
Car	98.0	78.4	84.7
Bus	95.9	76.7	82.9
Motorcycle	82.4	65.9	71.2
Truck	96.0	76.8	82.9
Pedestrian	70.9	56.7	59.6
Bicycle	61.4	36.8	38.7
Traffic light arrow - Red	89.2	66.9	68.9
Traffic light arrow - Yellow	76.7	57.5	59.3
Traffic light arrow - Green	88.4	66.3	68.3
Traffic light arrow - Red arrow	77.4	58.1	59.8
Traffic light arrow - Green arrow	86.3	64.7	66.7
Traffic sign - Speed	96.3	81.9	84.3
Traffic sign - Else	96.0	81.6	84.0
Road mark - Crosswalk	80.8	48.5	49.4
Road mark - Number	76.4	45.8	46.8
Road mark - Character	79.1	47.5	48.4
Straight	93.7	60.9	62.1
Left	79.8	51.9	52.9
Right	80.1	52.1	53.1
Straight left	78.1	50.8	51.8
Straight right	83.1	54.0	55.1
U-turn	80.8	52.5	53.6
Else	65.2	42.4	43.2

the sidewalk, the caution panel returns to WARNING, and the vehicle continues to drive without the alert DANGER.

In the case of Fig. 10 - (b), the vehicle can turn left with a speed limit of 50 km/h. However, when the left turn is ongoing, the ADAS detects pedestrians on the crosswalk in front, the WARNING caution is displayed, and the vehicle has to slow down. Then, with the crosswalk near the car, the caution DANGER is displayed, and it waits for all pedestrians to cross the street, then it turns to the NORMAL indicator.

In Fig. 10 - (c), the ADAS detects one motorbike near the vehicle and in the ego-lane and the WARNING caution is shown. Then, the NORMAL indicator is displayed when the motorbike is at a safe distance from the vehicle.

In Fig. 10 - (d), the WARNING caution alerts the driver that the red light is on, and that the vehicle must slow down and stop. Nevertheless, the NORMAL indicator allows vehicle to safely run while reaching the intersection when the green light is on.

In Fig. 10 - (e), a dangerous situation is presented because the bus changes lanes too fast to reach the bus stop. The ADAS alerts DANGER for the driver to be ready to stop. The caution indicator returns to NORMAL again after the bus is at the bus stop.

TABLE 5. Speed performance of the system.

Light and Sign	Lane and Road Marking	Vehicle and Pedestrian	Non-TensorRT (FPS)	TensorRT-FP16 (FPS)
v			20.87	33.39
	v		19.10	30.56
		v	17.93	28.69
	v	v	15.38	23.07
v	v		16.33	24.50
v		v	15.80	23.70
v	v	v	14.01	19.61

TABLE 6. Comparison of devices power consumption.

Devices	Power (W)
2 AGX, 1 TX2	75
3 AGX, 1 TX2	105
2 AGX, 1 NX, 1 TX2	90
3 AGX, 1 NX, 1 TX2	120
1 Titan X	250
3 Titan X	750

Finally, the system's accuracy on the scenario case is shown in Table. 3. Most of the number is higher than 90, proving that the system runs well in most cases.

C. PERFORMANCE

Besides the accuracy in the case scenario, we measure the accuracy in traffic object detection among our system with the one multi-task model, as shown in Table. 4. As can be seen, the DLT-Net and YOLOP have lower mAP than Ours because both models have to share the feature extractor with other tasks, and the training step for the multi-task model is more complicated than the specialized model.

Because these algorithms run in independent modules on hardware with limited power, the real-time performance of the system must be maintained. We find that the proposed system satisfies real-time reactions to the outside environment. Table. 5 shows the processing time of each case while using or neglecting to use each module in the system. In the meantime, the Table. 6 shows the recommended system power for each case. Therefore, in the case of running all modules in one machine with one or three Titan X GPUs, the energy required for it is higher than for the group of Jetson devices. Additionally, the stack of models may demand more memory than one Titan X's available memory, which means we need more than one GPU and more power consumption in one system.

V. CONCLUSION AND FUTUREWORK

This paper proposes a modular system that can flexibly implement any changes or improvements based on updated requirements. Herein, experiments show that the proposed ADAS maintains stability and that a crash in one module cannot affect the performance of others. The core tenet of any process that affects human safety is to ensure that the system has a predictable level of performance and that any misbehavior can be easily traced to the root cause. The work shows good execution speed with proper timing on

edge devices. In the future, we will upgrade the module to analyze more traffic rules and attempt to allow the ADAS to transact with the physical driving system, by implementing emergency braking, for instance, to improve the level of the autonomous vehicle system.

REFERENCES

[1] L. Dieter, "Saving lives: Boosting car safety in the EU," KOCH, Eur. Parliament, Strasbourg, France, Tech. Rep. 2017/2085(INI), Nov. 2017.

[2] T. Stewart, "Overview of motor vehicle crashes in 2020," U.S. Dept. Transp. Nat. Highway Traffic Saf. Admin., NHTSA Tech. Rep. DOT HS 813 266, Mar. 2022.

[3] L. Yue, M. A. Abdel-Aty, Y. Wu, and A. Farid, "The practical effectiveness of advanced driver assistance systems at different roadway facilities: System limitation, adoption, and usage," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3859–3870, Sep. 2020.

[4] E. Nodine, A. Lam, S. Stevens, M. Razo, and W. Najm, "Integrated vehicle-based safety systems (IVBSS) light vehicle field operational test independent evaluation," United States Nat. Highway Traffic Saf. Admin., Tech. Rep. DOT-VNTSC-NHTSA-11-02; DOT HS 811 516, Oct. 2011.

[5] T. Gordon, H. Sardar, D. Blower, M. L. Aust, Z. Bareket, M. Barnes, A. Blankespoor, I. Isaksson-Hellman, J. Ivarsson, B. Juhas, K. Nobukawa, and H. Theander, "Advanced crash avoidance technologies (ACAT) program—Final report of the Volvo-Ford-UMTRI project: Safety impact methodology for lane departure warning—Method development and estimation of benefits," United States Nat. Highway Traffic Saf. Admin., Washington, DC, USA, Tech. Rep. DOT HS 811 405, 2010.

[6] J. B. Cicchino, "Effectiveness of forward collision warning and autonomous emergency braking systems in reducing front-to-rear crash rates," *Accident Anal. Prevention*, vol. 99, pp. 142–152, Feb. 2017.

[7] L. Yue, M. Abdel-Aty, Y. Wu, and L. Wang, "Assessment of the safety benefits of vehicles' advanced driver assistance, connectivity and low level automation systems," *Accident Anal. Prevention*, vol. 117, pp. 55–64, Aug. 2018.

[8] Korea Automotive Technology Institute. *Leading a Future With Creativity & Innovation*. Accessed: Jan. 4, 2022. [Online]. Available: http://www.katech.re.kr/eng

[9] D. N.-N. Tran, H.-H. Nguyen, L. H. Pham, and J. W. Jeon, "Object detection with deep learning on drive PX2," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCE-Asia)*, Seoul, South Korea Nov. 2020, pp. 1–4.

[10] *Jetson TX2 Module*. Accessed: Jan. 17, 2022. [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx2

[11] *Jetson Xavier NX Developer Kit*. Accessed: Jan. 17, 2022. [Online]. Available: https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit

[12] *Jetson AGX Xavier Developer Kit*. Accessed: Jan. 17, 2022. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agxxavier-developer-kit

[13] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "MultiNet: Real-time joint semantic reasoning for autonomous driving," May 2016, *arXiv:1612.07695*.

[14] Z. Kang, K. Grauman, and F. Sha, "Learning with whom to share in multi-task feature learning," in *Proc. 28th Int. Conf. Int. Conf. Mach. Learn.*, Madison, WI, USA, 2011, pp. 521–528.

[15] D. Wu, M. Liao, W. Zhang, X. Wang, X. Bai, W. Cheng, and W. Liu, "YOLOP: You only look once for panoptic driving perception," 2021, *arXiv:2108.11250*.

[16] N. K. Ragesh and R. Rajesh, "Pedestrian detection in automotive safety: Understanding state-of-the-art," *IEEE Access*, vol. 7, pp. 47864–47890, 2019.

[17] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn, "Efficiently identifying task groupings for multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 34, 2021, pp. 27503–27516.

[18] T. Standley, A. R. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, "Which tasks should be learned together in multi-task learning?" in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, vol. 119, Jul. 2020, pp. 9120–9132.

[19] S. Ullah and D.-H. Kim, "Federated learning using sparse-adaptive model selection for embedded edge computing," *IEEE Access*, vol. 9, pp. 167868–167879, 2021.

[20] M. A. Farooq, P. Corcoran, C. Rotariu, and W. Shariff, "Object detection in thermal spectrum for advanced driver-assistance systems (ADAS)," *IEEE Access*, vol. 9, pp. 156465–156481, 2021.

[21] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13029–13038.

[22] *NVIDIA Tensorrt for Developers*. Accessed: Jan. 18, 2022. [Online]. Available: https://developer.nvidia.com/tensorrt

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[24] A. Avramovic, D. Sluga, D. Tabernik, D. Skocaj, V. Stojnic, and N. Ilc, "Neural-network-based traffic sign detection and recognition in high-definition images using region focusing and parallelization," *IEEE Access*, vol. 8, pp. 189855–189868, 2020.

[25] S. Zherzdev and A. Gruzdev, "LPRNet: License plate recognition via deep neural networks," Jun. 2018, *arXiv:1806.10447*.

[26] S. Lee, M. Younis, A. Murali, and M. Lee, "Dynamic local vehicular flow optimization using real-time traffic conditions at multiple road intersections," *IEEE Access*, vol. 7, pp. 28137–28157, 2019.

[27] N. Ma, G. Pang, X. Shi, and Y. Zhai, "An all-weather lane detection system based on simulation interaction platform," *IEEE Access*, vol. 8, pp. 46121–46130, 2020.

[28] Z. Qin, H. Wang, and X. Li, "Ultra fast structure-aware deep lane detection," in *Computer Vision—ECCV*, vol. 12369. Cham, Switzerland: Springer, 2020, pp. 276–291.

[29] Y. Qian, J. M. Dolan, and M. Yang, "DLT-Net: Joint detection of drivable areas, lane lines, and traffic objects," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4670–4679, Nov. 2020.

[30] D. N.-N. Tran, L. H. Pham, H.-H. Nguyen, T. H.-P. Tran, H.-J. Jeon, and J. W. Jeon, "A region-and-trajectory movement matching for multiple turn-counts at road intersection on edge device," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Nashville, TN, USA, Jun. 2021, pp. 4082–4089.

[31] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[32] S. Jeon, J. Son, M. Park, B. S. Ko, and S. H. Son, "Driving-PASS: A driving performance assessment system for stroke drivers using deep features," *IEEE Access*, vol. 9, pp. 21627–21641, 2021.

DUONG NGUYEN-NGOC TRAN (Graduate Student Member, IEEE) received the B.S. degree in computer science and the M.S. degree in information technology management from International University, Ho Chi Minh City, Vietnam, in 2014 and 2018, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Sungkyunkwan University, Suwon, South Korea. His current research interests include computer vision, image processing, and deep learning.

LONG HOANG PHAM (Member, IEEE) received the B.S. degree in computer science and the M.S. degree in information technology management from International University, Ho Chi Minh City, Vietnam, in 2013 and 2017, respectively, and the Ph.D. degree in electrical and computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2021. His current research interests include computer vision, image processing, and deep learning.

HUY-HUNG NGUYEN received the B.S. degree in computer science and the M.E. degree in information technology management from International University—Vietnam National University, Vietnam, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Sungkyunkwan University, Suwon, South Korea. His research interests include computer vision, image processing, and deep learning.

TAI HUU-PHUONG TRAN received the B.S. degree in computer science and engineering from International University—Vietnam National University, Vietnam, in 2015. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Sungkyunkwan University, Suwon, South Korea. His current research interests include image processing, computer vision, and deep learning.

HYUNG-JOON JEON received the B.S. degree in computer science and engineering from Yonsei University, Seoul, South Korea, in 2013, and the M.S. degree in electrical and computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2019, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. From 2013 to 2016, he was an Engineer with Samsung Electronics, Suwon. His research interests include computer vision, deep learning, and system software.

JAE WOOK JEON (Senior Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1984 and 1986, respectively, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1990. From 1990 to 1994, he was a Senior Researcher with Samsung Electronics, Suwon, South Korea. Since 1994, he has been with Sungkyunkwan University, Suwon, where he was first an Assistant Professor with the School of Electrical and Computer Engineering and is currently a Professor with the School of Information and Communication Engineering. His current research interests include robotics, embedded systems, and factory automation.