Optimizing YOLO Performance for Traffic Light Detection and End-to-End Steering Control for Autonomous Vehicles in Gazebo-ROS2

Hoang Tran Ngoc, Khang Hoang Nguyen, Huy Khanh Hua, Huynh Vu Nhu Nguyen, Luyl-Da Quach
FPT University, Can Tho 94000, VietNam

***Abstract*—Autonomous driving has become a popular area of research in recent years, with accurate perception and recognition of the environment being critical for successful implementation. Traditional methods for recognizing and controlling steering rely on the color and shape of traffic lights and road lanes, which can limit their ability to handle complex scenarios and variations in data. This paper presents an optimization of the You Only Look Once (YOLO) object detection algorithm for traffic light detection and end-to-end steering control for lane-keeping in the simulation environment. The study compares the performance of YOLOv5, YOLOv6, YOLOv7, and YOLOv8 models for traffic light signal detection, with YOLOv8 achieving the best results with a mean Average Precision (mAP) of 98.5%. Additionally, the study proposes an end-to-end convolutional neural network (CNN) based steering angle controller that combines data from a classical proportional integral derivative (PID) controller and the steering angle controller from human perception. This controller predicts the steering angle accurately, outperforming conventional open-source computer vision (OpenCV) methods. The proposed algorithms are validated on an autonomous vehicle model in a simulated Gazebo environment of Robot Operating System 2 (ROS2).**

***Keywords*—Yolo models; PID; CNN; gazebo; ROS2; traffic-light; lane-keeping; autonomous**

I. Introduction

The increasing number of vehicles on the road has raised concerns about traffic accidents and fatalities caused by various factors [1]. To address this, research has been conducted to create technologies that enhance driving safety, such as Advanced Driver Assistance Systems (ADAS) and autonomous driving systems. Traffic light signal recognition is a crucial component of these systems as it helps to detect the current status of traffic lights and provides real-time information for the vehicle control system to make accurate decisions. The conventional approaches for early detection of traffic lights rely on identifying manual features and color characteristics of signal lights, as [2]-[7]. These methods typically employ feature matching, color matching, or similar techniques to detect traffic lights based on their shape and color. Other approaches involve utilizing offline location information, which includes traffic light data from maps and GPS-based data, to track both the vehicle's present location and the status of traffic lights, as discussed in references [8]-[10]. Conventional approaches to detecting traffic lights are constrained by technical aspects like the type of camera used and the surrounding installation conditions. Additionally, these methods rely on offline location data that needs to be constantly refreshed and is susceptible to security risks.

In recent years, deep learning approaches, such as convolution neural network (CNN) [11], single shot multibox detector (SSD) [12], or YOLO architecture [13]-[15], have been used to accurately identify traffic lights and provide a real-time solution for traffic light detection. Among these approaches, the YOLO method has emerged as the best performer in detecting and recognizing traffic signals. Compared to other deep learning techniques, it offers smoother, more accurate results, and achieves real-time performance. Joseph et al. presented YOLO and an enhanced version called YOLOv2, which were utilized for detecting and classifying traffic lights [16]-[17]. Possatti et al. employed YOLOv3 [18] along with prior maps to identify crucial states of traffic lights for vehicles in their investigation. Tai et al. made advancements to YOLOv4 to enhance its precision in classifying green, red, and yellow traffic lights [19]. However, the past year has seen the release of a series of updated versions of YOLO, including YOLOv5 [20] [21], YOLOv6 [22], YOLOv7 [23], and YOLOv8 [24], each of which shows improved accuracy and performance in implementation on the COCO dataset.

To apply these improved models in the field of autonomous vehicles, we present our traffic light detection algorithm that utilizes the latest version of YOLO, YOLOv8 [25]. Additionally, we demonstrate the superior performance of YOLOv8 compared to other models (YOLOv5-v6-v7), in terms of accuracy and real-time processing. We conduct experiments on the traffic light dataset obtained and augmented data from multiple sources, including the CinTA_v2 dataset, and GathoTF datasets shown in Fig. 1.

Furthermore, research focused on utilizing camera images for angle prediction in autonomous vehicles to ensure proper lane-keeping has garnered considerable interest and made significant advancements. Earlier research employed image processing techniques, including color and edge detection, as well as critical lane regions such as the Canny edge detector, Hough Transform technique, and Isolate Region of Interest (IROI), to predict the steering angle [26]-[29]. The steering angle was then used in combination with a classic PID controller to ensure lane-keeping [30]-[33]. However, when faced with more complex scenarios, such as intersections, where lane images may be interrupted, a hybrid angle controller must be employed to account for driver perception. To overcome these limitations, autonomous vehicle control techniques have emerged that utilize deep learning to emulate human driving behavior.

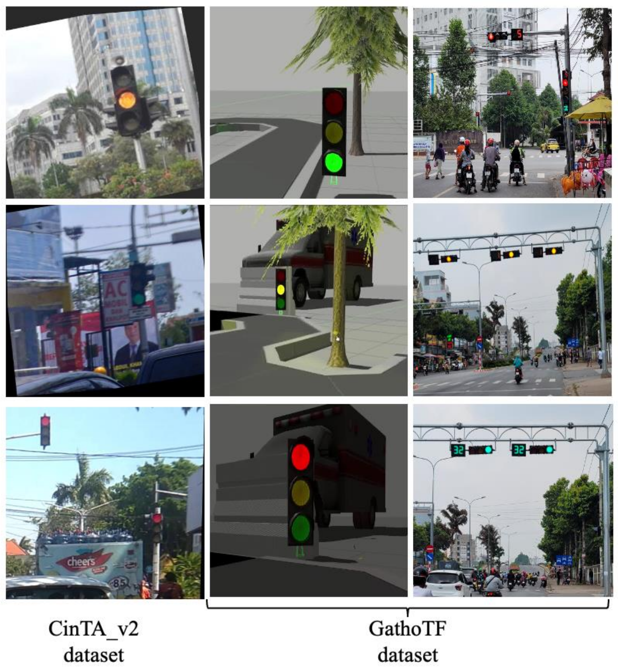

Fig. 1. Traffic light dataset.

In this research, we present a YOLOv8 architecture for traffic light recognition combined with an end-to-end CNN steering controller for lane-keeping in a simulated Gazebo environment of Robot Operating System 2 (ROS2). While the vehicle is following the lane, the traffic light signals are also detected and classified to send to the central controller. The primary contributions of this research are:

- The new YOLOv8 model is applied in our traffic light detection algorithm. Using the multiple source traffic light dataset, the YOLOv8 model is trained, evaluated, and compared accuracy and real-time performance with previous models.
- Based on the end-to-end convolution network, a steering angle predictor is designed. The model not only re-learns human driving behavior but is also trained with a data set of OpenCV+PID steering angle control methods to increase the accuracy of lane keeping and to give a steering angle when the lane is discontinuous.
- To assess the effectiveness of our proposed method for traffic light detection and lane-keeping, we collected a dataset by driving a donkey car model following the lane and incorporating traffic lights into the Gazebo-ROS2 simulation environment. We compared the performance of our method with that of previous methods using the same dataset.

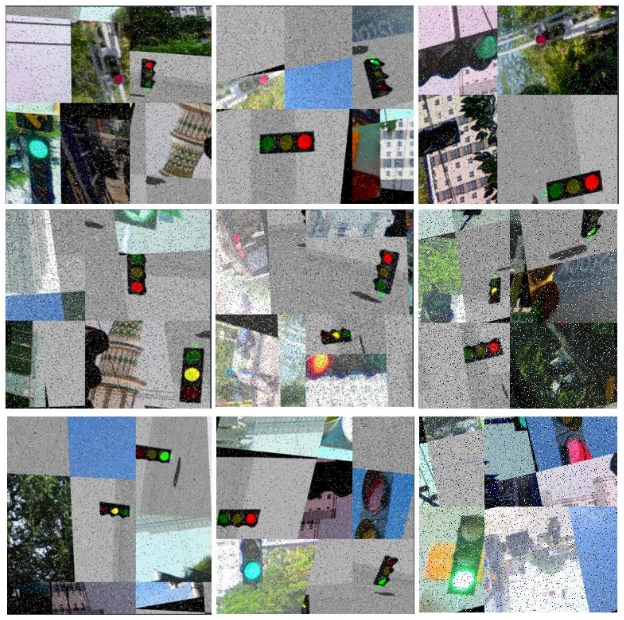

Fig. 2. Augmentation dataset.

This paper is structured as follows: Section II describes the architecture of YOLO models used for traffic light detection and classification. Section III provides a detailed description of the end-to-end CNN steering angle predictor, including the data collection process, evaluation metrics, and network architecture. Section IV presents the experimental results and provides a discussion of the findings. Finally, Section V concludes the paper and suggests areas for future research.

II. Traffic Light Detection and Classification Using YOLO Models

A. *Preprocessing Data*

1) Data collection: To evaluate the effectiveness of YOLO in traffic light detection, experiments are conducted using a synthetic dataset, which was collected from two different sources, the CinTA_v2 traffic light datasets, and GathoTF datasets.

(https://www.kaggle.com/datasets/ngochoangtran1992/traffic-light-dataset).

The CinTA_v2 dataset is a freely available traffic light dataset that was public in Roboflow. It contains 999 images captured under various weather and lighting conditions, and for this study, 999 images were randomly picked for training and evaluation. Each image has a resolution of 1280 x 960 pixels.

The GathoTF dataset consists of 2025 images and includes traffic light images taken in Can Tho city and traffic light images taken in a virtual environment created by the Gazebo/ROS2 program. In this environment, a self-driving car was simulated, and a camera mounted on the car took pictures at a frame rate of 30 fps and a resolution of 1024 x 600 pixels.

TABLE I. DETAILED DIFFERENCES BETWEEN YOLO MODELS

	Model	Backbone	Neck	Head	Loss Function
YOLOv5 [36]	The YOLOv5 algorithm is a fast and efficient object detection system that uses anchor-free detectors, a CSPDarknet53 backbone, a PAN neck, and AutoML for anchor box optimization and employs a Mosaic data augmentation technique to improve generalization.	CSPDarkent53 Focus structure	PANet	B x (5 + C) output layer B: No, of bounding boxes C: Class score	Binary Cross Entropy and Logit Loss Function
YOLOv6 [37]	YOLOv6 is an updated version of the YOLO algorithm that incorporates a RepVGG backbone, VariFocal Loss for dynamically adjusting object contribution during training, SIoU and GIoU for classification and regression loss, and a Focal Attention mechanism for improved region detection.	RepVGG And CSPRepStack	RepPAN	Decoupled Classification and Detection Head	Varifocal Loss for Classification and Distribution Focal Loss for Detection
YOLOv7 [38]	YOLOv7 is an object detection algorithm that uses an Extended Efficient Layer Aggregation backbone with group convolution, a Gradient Flow Propagation module for re-parameterization, and an auxiliary head for improved prediction accuracy.	EELAN	PANet	Lead Head for final output, Auxiliary Head for middle layer outputs	BCE with Focal Loss for Classification, IoU loss for Detection
YOLOv8 [39]	YOLOv8 is an object detection algorithm that uses a CSP-inspired C2f module instead of the C3 module, allowing for more abundant gradient flow information while maintaining a lightweight design.	CSP same as that of YOLOv5 but C3 module replaced by C2f module	PAN-FPN	Decoupled-Head	VFL Loss and DFL Loss+CIOU

Using our datasets to train and test evaluate its effectiveness in traffic light recognition and detection. The variety of datasets enabled us to enhance and improve the model's resilience and generalizability by exposing it to a variety of different scenarios and conditions.

Algorithm 1: Data augmentation

Input: Load the dataset of traffic light images and their corresponding labels.

1. For each image in the dataset, randomly select one or more of the following augmentation techniques:

1.1: Image flipping: randomly flip the image horizontally or vertically.

1.2: Image rotation: randomly rotate the image by 15 degrees to the right or left.

1.3: Blur: apply a Gaussian blur with a kernel size of 8% to the image.

1.4: Random cropping: randomly crop the image by 3%.

1.5: Noise generation: add random Gaussian noise with a standard deviation of 8% to the image.

1.6: Decolorization: randomly remove 7% of the color channels from the image.

2. Apply the selected augmentation techniques to the image and save the new image as a separate file in the dataset folder.

3. Update the label of the new image with the same label as the original image.

4. Repeat steps 1-3 for all images in the dataset.

Result: Save the augmented dataset and use it to train and test the traffic light recognition system.

2) Data augmentation: In order to enhance the precision of the traffic light recognition system and expand the dataset, Algorithm 1 was implemented.

Following the augmentation procedure, the labeled data was reintegrated into the dataset, thereby augmenting the quantity of inherent angle data. The labeled data was then used to train a YOLO model. As a result of the data augmentation techniques, the original dataset of 4017 images was expanded to include 6695 additional images, resulting in a more diverse and robust dataset. Fig. 2 shows the example of the dataset after the augmentation. Next section, the different YOLO models are compared and analyzed when applying these models to traffic light detection and classification.

B. Detection and Classification using YOLO Models

This study employs four versions of YOLO architecture for traffic light recognition. YOLO architecture, originally introduced by Redmon et al. (2016) [34], approaches the detection task as a regression problem based on the Darknet architecture. Unlike popular Region Proposal Networks (RPN), YOLO predicts both bounding boxes (Bbox.) and class probabilities (Cls.) in a single network. This approach is based on a user-defined size grid cell responsible for detecting the object if it falls into the cell. To accurately evaluate the performance of these YOLOv models, we analyze the differences as well as the advantages and disadvantages of each model before applying them to the identification and classification of traffic lights. The comparison between structures of YOLOv5, YOLOv6, YOLOv7, and YOLOv8 are shown in Table I. The features of the modules are as follows:

- YOLOv5 is an advanced object detection algorithm that utilizes anchor-free detectors to detect multiple objects in real-time [20], [35]. It uses a CSPDarknet53 backbone and PAN (Path Aggregation Network).

- Network for faster and more efficient detection. The Mosaic data augmentation technique combines multiple images into a single image to enhance generalization, while AutoML optimizes anchor boxes of varying sizes and aspect ratios for each grid cell. YOLOv5 offers superior speed and accuracy in object detection compared to other methods.

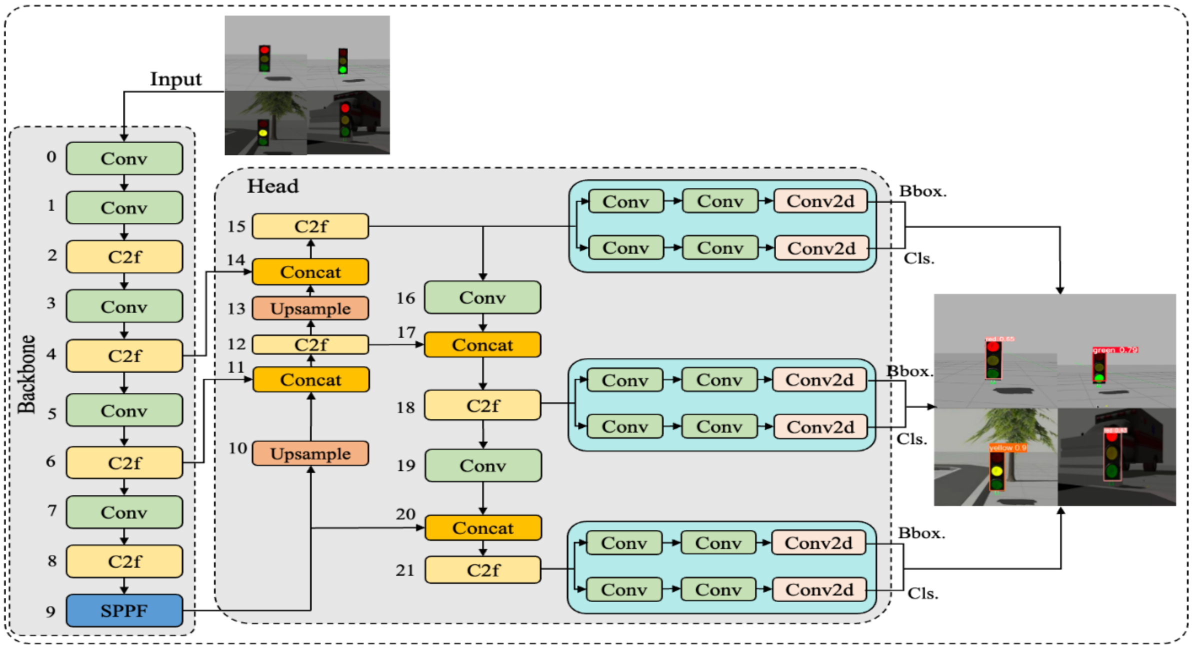

Fig. 3. YOLOv8 model.

- The YOLOv7 model incorporates an innovative architecture known as Extended Efficient Layer Aggregation (EELA) as its backbone. This novel design employs group convolution to broaden the computational block's channel, thereby enhancing the model's overall performance [38]. The algorithm also uses a new module called the Gradient Flow Propagation (GFP) module, which helps to determine which modules need re-parameterization in order to improve the model's accuracy. Finally, YOLOv7 includes an auxiliary head that is designed to provide a coarse-to-fine definition for better predictions lightweight.

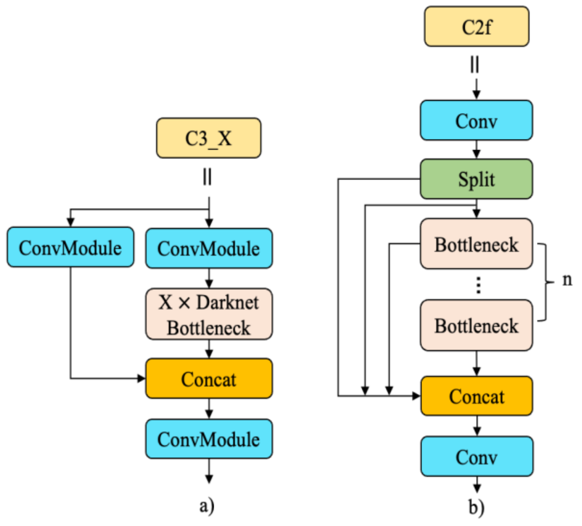

Fig. 4. Comparison of the structure of CS_X (YOLOv5) and C2f (YOLOv8) in backbone.

- The Backbone part of YOLOv8 is basically the same as that of YOLOv5, and the C3 module is replaced by the C2f module based on the CSP idea [39]. The C2f module learned from the ELAN idea in YOLOv7 and combined C3 and ELAN to form the C2f module so that YOLOv8 could obtain more abundant gradient flow information while ensuring lightweight. We use the model YOLOv8 for detecting, and classification traffic lights, as shown in Fig. 3. Fig. 4 is a comparison of the structure of CS_X (YOLOv5) and C2f (YOLOv8) in the backbone.

Four YOLO models are proposed to compare and test the applicability of these models to traffic light signal recognition for autonomous vehicle systems. There are a total of three active traffic light states: s = {red, yellow, green}. The models will be trained on our augmentation datasets

III. END-TO-END CNN STEERING ANGLE CONTROLLER

A. *Preprocessing Data*

1) Data collection: To ensure the vehicle is always in the center of the lane, a series of multi-step image processing using the OpenCV method and a PID steering controller is applied [33]. The steering angle will be predicted after image processing to detect the two lines of the lane. Combined with the PID controller, the vehicle will move more precisely. At this point, we will collect the input image data and the output steering angle to train the end-to-end CNN steering angle controller model. It contains 10,000 images captured from the camera mounted on the vehicle in the simulation environment Gazebo/ROS2. However, when the car went through the intersection and encountered complicated situations, the estimated steering angle from the lane detection was interrupted, so we used the joystick to control the car with human perception. In this case, 10,000 pictures continue to be collected along with the steering angle from human perception. Fig. 5 shows the dataset for the end-to-end CNN steering angle controller model.

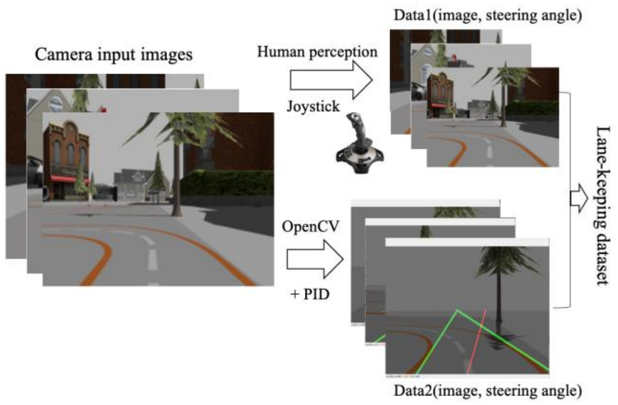

Fig. 5. Dataset for the end-to-end CNN steering angle controller model.

2) Data augmentation and normalization: To improve the accuracy and avoid issues with gradient explosion or vanishing of the lane-keeping model, data augmentation and normalization techniques are employed as Algorithm 2.

Algorithm 2: Data Augmentation and Normalization

```
Input: Training Dataset
1. AUGMENT (image, steering angle):
    1.1. if random () < 0.5;
             image = pan(image) // crop out a smaller image from the left or right side
    1.2. if random () < 0.5;
             image = zoom(image) // crop out a smaller image from the center
    1.3. if random () < 0.5;
             image = blur(image) // a Gaussian blur
    1.4. if random () < 0.5;
             image = adjust_brightness(image) // adjust brightness of the image
    1.5. image, steering angle = random flip (image, steering angle) // perform a horizontal flip on the image, which means flipping it from left to right, and adjust the corresponding steering angle accordingly.
    1.6. return image, steering angle
2. NORMAL (image):
    2.1. height = image.height.
    2.2. image = image[height/2+100] //Remove top half of the image, as it is not relevant for the lane following
    2.3. image = convert_Color(image, cv2.COLOR_RGB2YUV) //The optimal choice is to utilize the YUV color space.
    2.4. image = GaussianBlur(image, (3,3), 0)
    2.5. image = resize (image, (200,66)) // input image size (200,66) our model
    2.6. image = image / 255 # normalizing, the processed image becomes black for some reason
    2.7. return image
Result: Augmented and normalized image data with updated steering angles.
```

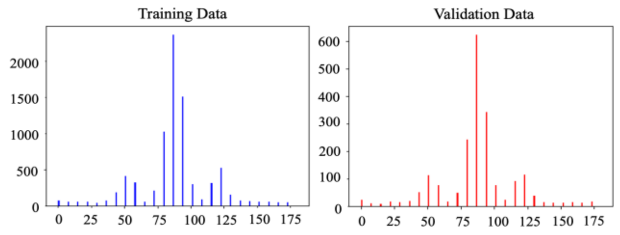

Fig. 6. Normalisation and augmentation of dataset.

As depicted in Fig. 6, our dataset has been subjected to both normalization and augmentation. Meanwhile, the distribution of the steering angle within our dataset has been illustrated in Fig. 7.

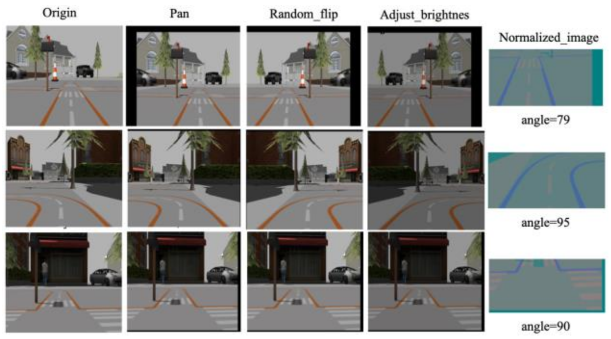

Fig. 7. Distribution of the steering angle within our dataset.

B. *Lane Keeping Using End-to-End CNN Model*

The proposed network architecture used an end-to-end neural network called Nvidia_model, Fig. 8 shows a convolutional neural network architecture with five Conv2D layers, followed by a flattened layer and three fully connected (FC) layers, and an output layer with a single neuron.

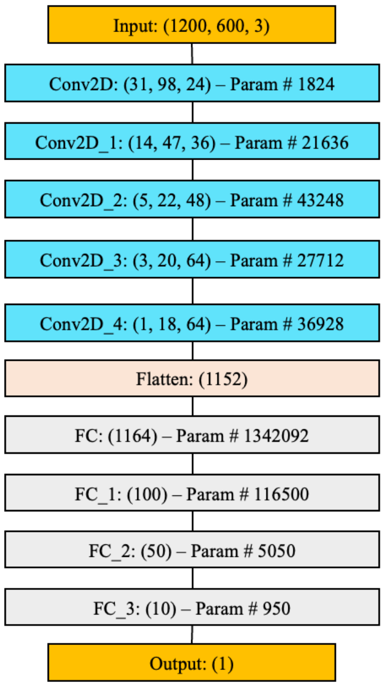
Input: (1200, 600, 3)
Conv2D: (31, 98, 24) – Param # 1824
Conv2D_1: (14, 47, 36) – Param # 21636
Conv2D_2: (5, 22, 48) – Param # 43248
Conv2D_3: (3, 20, 64) – Param # 27712
Conv2D_4: (1, 18, 64) – Param # 36928
Flatten: (1152)
FC: (1164) – Param # 1342092
FC_1: (100) – Param # 116500
FC_2: (50) – Param # 5050
FC_3: (10) – Param # 950
Output: (1)

Fig. 8. Convolutional neural network architecture.

The input to the network is a 3-dimensional tensor with shape (1200, 600, 3), which suggests that the input is an image with a width of 1200 pixels, a height of 600 pixels, and 3 color channels (e.g., RGB). The first layer in the network is a convolutional layer with 24 filters, each with a shape of (31, 98). This layer has 1,824 parameters, which are learned during training.

The next layer is another convolutional layer with 36 filters, each with a shape of (14, 47). This layer has 21,636 parameters. The third layer is a convolutional layer with 48 filters, each with a shape of (5, 22). This layer has 43,248 parameters. The fourth layer is a convolutional layer with 64 filters, each with a shape of (3, 20). This layer has 27,712 parameters. Finally, the fifth layer is a convolutional layer with 64 filters, each with a shape of (1, 18). This layer has 36,928 parameters.

After the last convolutional layer, the output is flattened into a 1-dimensional tensor with shape (1152). This flattened output is then fed into a fully connected (FC) layer with 1,164 units. This FC layer has 1,342,092 parameters. The output of this layer is then fed into another FC layer with 100 units, which has 116,500 parameters. The next FC layer has 50 units and 5,050 parameters. Finally, there is a FC layer with 10 units and 950 parameters. The total number of parameters in this model is 1,595,511. The output of the fourth fully connected layer is passed through and the output layer contains a single neuron, which predicts the steering angle.

IV. Experimental Result

A. Experimental System

To evaluate the performance of our proposed algorithms, we conducted experiments in a simulated environment using Gazebo-ROS2. The simulation was run on an Ubuntu 20.04 platform with an Intel Core i7 processor and 16 GB RAM. The simulated vehicle was equipped with a front-facing camera, simulated based on the actual parameters of the WGE100 camera. The camera captured images with a resolution of 1024×600 at a frame rate of 30 fps. The simulation environment was set up to include a variety of traffic light scenarios and lane configurations to test the robustness of our algorithms.

The experimental system consisted of two parts: traffic light detection and end-to-end steering control. For traffic light detection, we trained our YOLOv8 model using the CinTA_v2 and GathoTF datasets, which were augmented using Algorithm 1 to increase their size and diversity. We evaluated the performance of the model in terms of accuracy, precision, and recall on a test set of 20% of the total dataset. We also compared the performance of YOLOv5, YOLOv6, and YOLOv7 models for traffic light detection.

For end-to-end steering control, we designed a convolutional neural network-based steering angle controller that combines data from a classical PID controller and human perception. The model was trained on the same dataset used for traffic light detection and evaluated on a separate test set. We compared the performance of our model with that of a traditional PID controller and analyzed the results.

In order to assess the effectiveness of our proposed algorithms, we employed a Donkey Car model, in which the steering angle was controlled based on the predictions made by our model. The Donkey Car was driven on a predefined route in the simulated environment as shown in Fig. 9, which included a variety of traffic light scenarios and lane configurations. We collected data from the camera and the steering angle sensor to evaluate the performance of our algorithms in real time.

Overall, our experimental system allowed us to evaluate the effectiveness and robustness of our proposed algorithms for traffic light detection and end-to-end steering control in a simulated environment. The results of our experiments are presented and analyzed in the next section.

B. Evaluation Metrics

The use of evaluation metrics is critical in comparing and assessing the performance of machine learning algorithms. In this study, we compare four different object detection algorithms - YOLOv5, YOLOv6, YOLOv7, and YOLOv8, using various metrics such as F score, and mAP. To provide a comprehensive evaluation of the proposed algorithm's performance, we employ several evaluation metrics, including precision, recall, mAP, F-score, and FPS. However, to avoid potential biases in the evaluation process, we utilize different evaluation criteria that are based on different aspects of the algorithms' performance.

In order to evaluate the effectiveness of the proposed algorithm, this study utilizes several metrics including precision (P), recall (R), average precision (mAP), F1-Score, and Frames Per Second (fps). Precision is a critical metric used to assess the accuracy of the evaluation object by determining the ratio of correctly predicted positive samples to the total number of predicted positive samples.

$$P = \frac{TruePositive}{TruePositive + FalsePositive} \tag{1}$$

TruePositive indicates the count of positive samples accurately predicted as positive, whereas FalsePositives denotes the count of negative samples incorrectly predicted as positive.

Recall measures the ratio of correctly predicted positive samples to the total number of actual positive samples. It indicates whether the evaluation object is detected in its entirety or not.

$$R = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2}$$

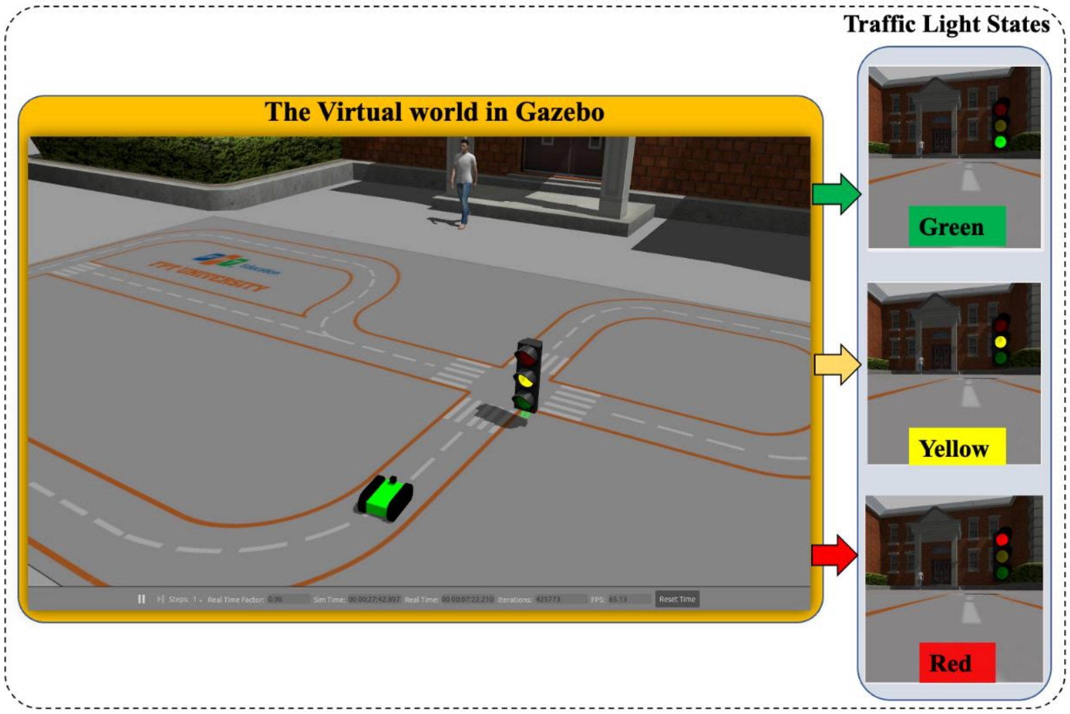

Fig. 9. The virtual world in Gazebo.

FalseNegative signifies the count of positive samples erroneously predicted as negative samples.

Mean average precision (mAP) is an essential metric utilized for evaluating the overall performance of the algorithm. It is computed by averaging the average precision (AveP) values across all classes.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AveP_k \tag{3}$$

AveP_k represents the average precision of class k, where n denotes the total number of classes.

The F1_Score is a metric that combines precision and recall into a single measurement by taking their weighted harmonic mean. It provides a balanced evaluation by considering both precision and recall.

$$F1\_Score = (1+\alpha^2)\,\frac{P.R}{\alpha^2.P + R} \tag{4}$$

The value of α is employed to achieve a balanced weighting of precision and recall in the calculation of the F-score. A higher F1_Score implies that the algorithm has a better balance between precision and recall. Finally, Frames Per Second is a critical metric in evaluating the speed and efficiency of the algorithm. A higher fps score indicates that the algorithm can process a large number of frames in a shorter time.

In conclusion, using these evaluation metrics in this study provides a comprehensive and unbiased assessment of the performance of the object detection algorithms.

C. Traffic Light Detection and Classification Results

In this section, we present the results of the traffic light detection and classification experiments using different YOLO models. We used a combined dataset of CinTA_v2 and GathoTF traffic light datasets for training and testing. The dataset was split into three parts: Training Set (76%), Validation Set (19%), and Testing Set (5%).

We trained YOLOv5, YOLOv6, YOLOv7, and YOLOv8 models on the combined dataset, and evaluated their performance on the Testing Set. The results are summarized in Table II. The YOLOv8 model achieved the best performance in terms of precision, recall, and F1 score, with an F1 score of 0.8947, and mAP_0.5 of 0.9192 outperforming other models by a significant margin. The results demonstrate the effectiveness of the proposed approach in detecting and classifying traffic lights accurately and efficiently.

Additionally, we assessed the influence of data augmentation on the performance of the YOLOv8 model. We trained the model with and without augmentation and subsequently compared their respective performances on the Testing Set. The findings of this evaluation are reported in Table II. The augmented dataset significantly improved the performance of the model, with an increase of 0.0313 in F1 score, and 0.0148 in mAP_0.5 indicating that data augmentation is an effective technique for enhancing the robustness and generalizability of the model.

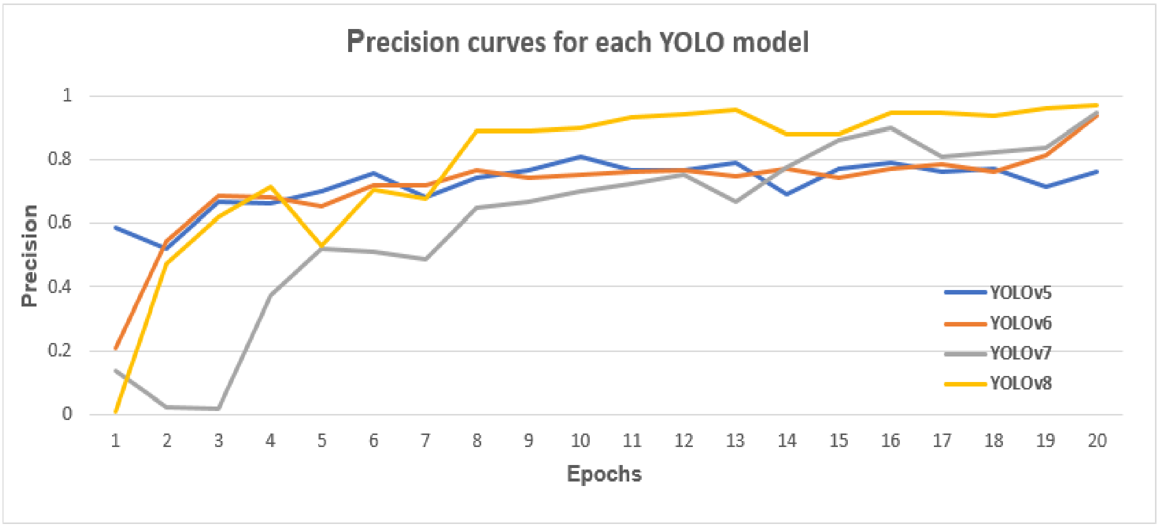

Fig. 10. Precision for traffic light detection and classification using YOLO models.

TABLE II. COMPARISON OF YOLOV5, YOLOV6, YOLOV7 AND YOLOV8 FOR TRAFFIC LIGHT DETECTION

YOLO Model	Precision	Recall	F1_Score	mAP_0.5
YOLOv5	0.7821	0.7787	0.7733	0.8216
YOLOv6	0.9139	0.8106	0.8631	0.8929
YOLOv7	0.9201	0.8289	0.8675	0.9028
YOLOv8	0.9317	0.8427	0.8947	0.9192

To further analyze the performance of the proposed approach, we generated precision curves for each YOLO model, as shown in Fig. 10. The curves indicate that the YOLOv8 model achieved the highest precision values for detecting and classifying traffic lights, followed by YOLOv7, YOLOv6, and YOLOv5 models.

Furthermore, we visualized the resulting images generated by the YOLOv8 model to illustrate the effectiveness of our approach in detecting and classifying traffic lights. Fig. 11 shows sample images from the Testing Set with bounding boxes and labels generated by the YOLOv8 model. The model successfully detected and classified the traffic lights, with high precision and recall values.

Overall, the results demonstrate that the proposed approach using YOLOv8 with data augmentation achieves superior performance in traffic light detection and classification, providing a real-time solution for autonomous driving systems in complex scenarios.

D. Lane-Keeping Results

In addition to traffic light detection, our research also focuses on improving lane-keeping performance for autonomous vehicles. We propose a convolutional neural network (CNN)-based steering angle controller that combines data from a classical PID controller and human perception to predict the steering angle. In order to evaluate the performance of our proposed steering controller, we conducted experiments in a simulated environment using the Gazebo-ROS2 platform.

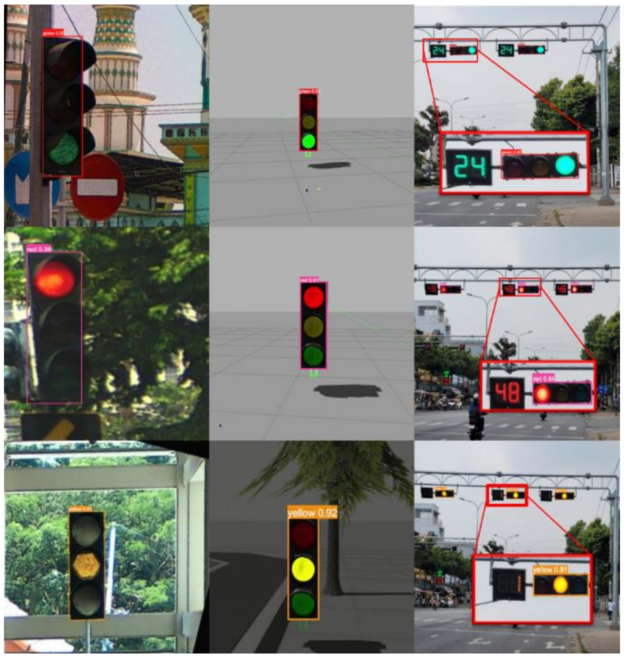

Fig. 11. Sample images with traffic light detection and classification results.

We collected a dataset of driving behaviors from a human driver using the OpenCV+PID steering angle control method, which we used to train and validate our CNN-based steering controller. The dataset consists of a donkey car model driving in a simulated environment with different road conditions and lighting conditions.

To evaluate the performance of our steering controller, we conducted experiments in the same simulated environment. We compared the performance of our proposed method with a baseline PID controller and a CNN-based steering controller trained with a traditional CNN architecture.

In order to evaluate the performance of our proposed steering controller, we conducted experiments in the same simulated environment and compared our method with a baseline PID controller and a CNN-based steering controller trained with a traditional CNN architecture. We use a mathematical equation to calculate the percentage of accuracy by summing the prediction error, dividing by the overall validation angle, and multiplying by 100 to measure the accuracy of our proposed method.

$$accuracy = 100 - \frac{\left|\sum_{i=1}^{N}\left(y_{actual_i} - y_{pred_i}\right)\right|}{\sum_{i=1}^{N}\left(y_{true_i}\right)} * 100 \quad (5)$$

where y_{actual} is the actual angle value, and y_{pred} is the predicted angle value.

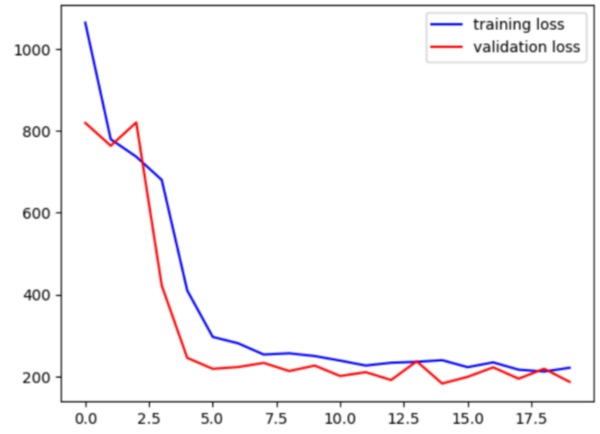

Fig. 12. The learning curve for the End-to-End CNN model using the Mean Squared Error (MSE) loss function.

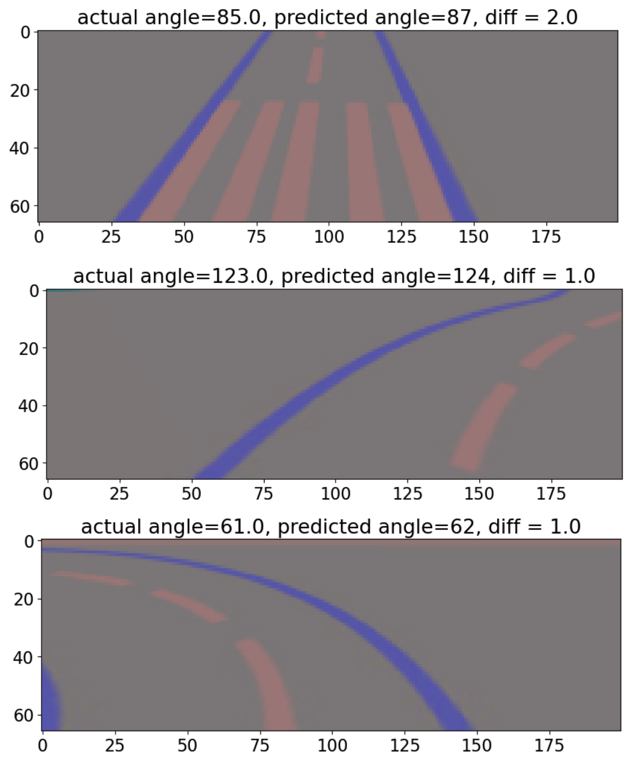
actual angle=85.0, predicted angle=87, diff = 2.0
actual angle=123.0, predicted angle=124, diff = 1.0
actual angle=61.0, predicted angle=62, diff = 1.0

Fig. 13. The true steering and predicted steering, the diff is the difference between true steering and predicted steering.

The evaluation of our lane-keeping performance involved a comparison between the actual steering angle of the vehicle and the predicted steering angle generated by the End-to-End CNN model. This analysis is depicted in Fig. 13.

We also evaluated the performance of our model using the MSE loss function, and the learning curve for the End-to-End CNN model using the MSE loss function is depicted in Fig. 12. After 20 epochs, the MSE value was 230.8, and the learning curve shows a decreasing trend. The curve indicates that the model's performance improves as the number of epochs increases, with diminishing returns after a certain point. Although the curve appears to be approaching a stable solution, further training may be required to confirm this.

Our results demonstrate that our proposed CNN-based steering controller outperforms the baseline PID controller and the traditional CNN-based steering controller in terms of accuracy and smoothness of the steering control. Our proposed method achieved an accuracy of 86.46%, as measured by the percentage of accurately predicted steering angles. The results indicate that our proposed method can effectively predict the steering angle and improve the lane-keeping performance of autonomous vehicles.

V. Conclusion

In conclusion, the study presents an optimized approach for traffic light detection and End-to-End steering control for autonomous vehicles using YOLOv8 and a CNN-based steering angle controller. The proposed methods are evaluated in a simulated environment and achieved high performance in both traffic light detection and lane-keeping tasks. The results show that YOLOv8 outperforms other YOLO models in traffic light detection, while the CNN-based steering angle controller achieves a high accuracy rate. The study contributes to the development of advanced autonomous driving systems that can improve driving safety and reduce traffic accidents.

References

[1] Road Traffic Injuries. Available online: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries (accessed on 30 June 2022).

[2] W.-C. S. Cheng-Chin, Ming-Che Ho, "Detecting and recognizing traffic lights by genetic approximate ellipse detection and spatial texture layouts," ICIC 2011, 2011.

[3] T. H.-P. Tran, C. C. Pham, T. P. Nguyen, T. T. Duong, and J. W. Jeon, "Real-time traffic light detection using color density," in 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia). IEEE, pp. 1–4, 2016.

[4] C. Jang, S. Cho, S. Jeong, J. K. Suhr, H. G. Jung, and M. Sunwoo, "Traffic light recognition exploiting map and localization at every stage," Expert Systems with Applications, vol. 88, pp. 290–304, 2017.

[5] J. Levinson, J. Askeland, J. Dolson, and S. Thrun, "Traffic light mapping, localization, and state detection for autonomous vehicles," Int. Conf. on Robotics and Automation (ICRA), pp. 5784–5791, 2011.

[6] H. T. Vo, H. N. Tran, and L. Quach, "An Approach to Hyperparameter Tuning in Transfer Learning for Driver Drowsiness Detection Based on Bayesian Optimization and Random Search" International Journal of Advanced Computer Science and Applications(IJACSA), 14(4), 2023.

[7] P. H. Phan, A. Q. Nguyen, L. Quach, and H. N. Tran. 2023. "Robust Autonomous Driving Control using Auto-Encoder and End-to-End Deep Learning under Rainy Conditions". In Proceedings of the 2023 8th International Conference on Intelligent Information Technology (ICIIT '23). Association for Computing Machinery, New York, NY, USA, 271–278.

[8] N. Fairfield and C. Urmson, “Traffic light mapping and detection,” IEEE Proc. Int. Conf. on Robotics and Automation, pp. 5421–5426, 2011.

[9] V. John, K. Yoneda, B. Qi, Z. Liu, and S. Mita, “Traffic light recognition in varying illumination using deep learning and saliency map,” Intelligent Transportation Systems (ITSC), 2014 IEEE 17th Int. Conf. on, pp. 2286–2291, 2014.

[10] V. John, K. Yoneda, Z. Liu, and S. Mita, “Saliency Map Generation by the Convolutional Neural Network for Real-Time Traffic Light Detection Using Template Matching,” IEEE Transactions on Computational Imaging, vol. 1, no. 3, pp. 159–173, 2015.

[11] R. Kulkarni, S. Dhavalikar and S. Bangar, "Traffic Light Detection and Recognition for Self-Driving Cars Using Deep Learning," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-4.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016.

[13] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.

[14] F.-A. Redmon, Joseph, “Yolov3: An incremental improvement,” Tech. Rep., 2018.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016.

[16] Joseph R, Santosh D, Ross G, Ali F. You Only Look Once: Unified, Real-Time Object Detection[C], IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, IEEE, 2016: 779-788.

[17] Joseph R, Ali F. YOLO9000: Better, Faster, Stronger[C], IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, IEEE, 2017: 6517-6525.

[18] F.-A. Redmon, Joseph, “Yolov3: An incremental improvement,” Tech. Rep., 2018.

[19] Tai. H. P. Tran and J. W. Jeon, "Accurate Real-Time Traffic Light Detection Using YOLOv4," 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), Seoul, Korea (South), 2020, pp. 1-4.

[20] V. D. Nguyen, T. D. Trinh and H. N. Tran, "A Robust Triangular Sigmoid Pattern-Based Obstacle Detection Algorithm in Resource-Limited Devices," in IEEE Transactions on Intelligent Transportation Systems.

[21] H. K. Hua, K. H. N., L. Quach, and H. N. Tran. 2023. “Traffic Lights Detection and Recognition Method using Deep Learning with Improved YOLOv5 for Autonomous Vehicle in ROS2”. In Proceedings of the 2023 8th International Conference on Intelligent Information Technology (ICIIT '23). Association for Computing Machinery, New York, NY, USA, 117–122.

[22] R. Kaur and J. Singh, "Local Regression Based Real-Time Traffic Sign Detection using YOLOv6," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 522-526.

[23] C.-Y. Wang, A. Bochkovskiy, and H.-Y. Mark Liao, ‘‘YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,’’ 2022, arXiv:2207.02696.

[24] Jocher, G.; Chaurasia, A.; Qiu, J. YOLO by Ultralytics. 2023. Available online: https://github.com/ultralytics/ultralytics/blob/main/CITATION.cff (accessed on 3 March 2023).

[25] Pallavi V. Ingale and Prof. K. S. Bhagat.: Comparative Study of Lane Detection Techniques, in International Journal on Recent and Innovation Trends in Computing and Communication, vol. 4, no. 5, 2016.

[26] Ammu M Kumar and Philomina Simon.: Review of Lane Detection and Tracking Algorithms in Advanced Driver Assistance System, in International Journal of Computer Science & Information Technology (IJCSIT), vol. 7, no. 4, 2015.

[27] Sekehravani, E.A., Babulak, E., Masoodi, M.: Implementing Canny Edge Detection Algorithm for Noisy Image. Bull. Electr. Eng. Inform, vol. 9, no. 4, pp. 1404–1410, 2020.

[28] P. Subhasri, S. Santhoshkumar and A. Sumath, Edge Filtering through Recursive Application using Canny Edge Detector algorithm on small sub-blocks in an Image, 2020 International Conference on Smart Electronics and Communication (ICOSEC), pp. 563-566, 2020.

[29] Messom C. H., Sen Gupta G. and Demidenko S.N.: Hough Transform Run Length Encoding for Real-Time Image Processing, IEEE Trans. Instrum. Meas., vol. 56, no. 3, pp. 962-967, 2007.

[30] W. Farag and Z. Saleh, Tuning of PID track followers for autonomous driving, 2018 Int. Conf. Innov. Intell. Informatics, Comput. Technol. 3ICT 2018, pp. 1–7, 2018.

[31] A. Simorgh, A. Marashian, and A. Razminia, Adaptive PID Control Design for Longitudinal Velocity Control of Autonomous Vehicles, Proc. - 2019 6th Int. Conf. Control. Instrum. Autom. ICCIA 2019, pp. 1–6, 2019.

[32] V. Robila, L. Paulino, M. Rao, I. Li, M. Zhu, and W. Wang, Design and Implementation of PID-Based Steering Control for 1/10-Scale Autonomous Vehicle, 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pp. 0758-0762, 2021.

[33] Hoang, T. N.; Quach, Luyl-Da. International Journal of Advanced Computer Science and Applications; West Yorkshire Vol. 13, Iss. 10, (2022). DOI:10.14569/IJACSA.2022.0131086

[34] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788.

[35] Ziwen Chen, Lijie Cao, Qihua Wang, "YOLOv5-Based Vehicle Detection Method for High-Resolution UAV Images", Mobile Information Systems, vol. 2022, Article ID 1828848, 11 pages, 2022. https://doi.org/10.1155/2022/1828848

[36] Norkobil Saydirasulovich, Saydirasulov, Akmalbek Abdusalomov, Muhammad Kafeel Jamil, Rashid Nasimov, Dinara Kozhamzharova, and Young-Im Cho. 2023. "A YOLOv6-Based Improved Fire Detection Approach for Smart City Environments" Sensors 23, no. 6: 3161. https://doi.org/10.3390/s23063161

[37] R. Kaur and J. Singh, "Local Regression Based Real-Time Traffic Sign Detection using YOLOv6," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 522-526, doi: 10.1109/ICAC3N56670.2022.10074236.

[38] Zhang, Yuan, Youpeng Sun, Zheng Wang, and Ying Jiang. 2023. "YOLOv7-RAR for Urban Vehicle Detection" Sensors 23, no. 4: 1801. https://doi.org/10.3390/s23041801.

[39] G. Jocher, A. Chaurasia, and J. Qiu, “YOLO by Ultralytics.” https://github.com/ultralytics/ultralytics, 2023. Accessed: February 30, 2023.