sensors

MDPI

Article

Steering-Angle Prediction and Controller Design Based on Improved YOLOv5 for Steering-by-Wire System

Cunliang Ye [1,2], Yunlong Wang [1], Yongfu Wang [1,*] and Yan Liu [3]

[1] School of Mechanical Engineering and Automation, Northeastern University, Shenyang 110819, China; yecunliang123@126.com (C.Y.); wangyl@mail.neu.edu.cn (Y.W.)
[2] School of Mechanical Engineering, Ningxia Institute of Science and Technology, Shizuishan 753000, China
[3] School of Automobile and Traffic Engineering, Liaoning University of Technology, Jinzhou 121001, China; dannieliuy@outlook.com
* Correspondence: yfwang@mail.neu.edu.cn

Abstract: A crucial role is played by steering-angle prediction in the control of autonomous vehicles (AVs). It mainly includes the prediction and control of the steering angle. However, the prediction accuracy and calculation efficiency of traditional YOLOv5 are limited. For the control of the steering angle, angular velocity is difficult to measure, and the angle control effect is affected by external disturbances and unknown friction. This paper proposes a lightweight steering angle prediction network model called YOLOv5Ms, based on YOLOv5, aiming to achieve accurate prediction while enhancing computational efficiency. Additionally, an adaptive output feedback control scheme with output constraints based on neural networks is proposed to regulate the predicted steering angle using the YOLOv5Ms algorithm effectively. Firstly, given that most lane-line data sets consist of simulated images and lack diversity, a novel lane data set derived from real roads is manually created to train the proposed network model. To improve real-time accuracy in steering-angle prediction and enhance effectiveness in steering control, we update the bounding box regression loss function with the generalized intersection over union (GIoU) to Shape-IoU_Loss as a better-converging regression loss function for bounding-box improvement. The YOLOv5Ms model achieves a 30.34% reduction in weight storage space while simultaneously improving accuracy by 7.38% compared to the YOLOv5s model. Furthermore, an adaptive output feedback control scheme with output constraints based on neural networks is introduced to regulate the predicted steering angle via YOLOv5Ms effectively. Moreover, utilizing the backstepping control method and introducing the Lyapunov barrier function enables us to design an adaptive neural network output feedback controller with output constraints. Finally, a strict stability analysis based on Lyapunov stability theory ensures the boundedness of all signals within the closed-loop system. Numerical simulations and experiments have shown that the proposed method provides a 39.16% better root mean squared error (RMSE) score than traditional backstepping control, and it achieves good estimation performance for angles, angular velocity, and unknown disturbances.

Keywords: steer-by-wire (SbW) systems; steering-angle prediction; autonomous vehicles (AVs); convolutional neural network (CNN); barrier Lyapunov function

check for updates

Citation: Ye, C.; Wang, Y.; Wang, Y.; Liu, Y. Steering-Angle Prediction and Controller Design Based on Improved YOLOv5 for Steering-by-Wire System. *Sensors* **2024**, *24*, 7035. https://doi.org/10.3390/s24217035

Academic Editors: Gaochang Wu, Zizhu Fan and Dong Pan

Received: 19 September 2024
Revised: 27 October 2024
Accepted: 30 October 2024
Published: 31 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Background

According to the latest World Health Organization (WHO) reports, approximately 1.3 million individuals lose their lives annually due to road accidents, with human errors accounting for an estimated 90% of all car crashes [1]. The statistics above have prompted the proposition of autonomous vehicles (AVs) as a means to mitigate human errors. AVs exemplify a burgeoning application of automotive technology, attracting significant attention from both academia and industry due to their advanced functionalities, such as

scene recognition, path planning, and motion control, which offer substantial driving convenience. AVs are now emerging as an innocuous alternative to human drivers, thereby significantly reducing the annual loss of thousands of lives [2]. Noteworthy ongoing research efforts in managing the diverse challenges faced by AVs include advancements in human recognition, traffic analysis, road and lane detection, steering controls, and path planning. Simultaneously, the prediction and control of the steering angle play a pivotal role in the realm of autonomous vehicles, garnering significant attention from researchers, manufacturers, and insurance companies alike [3].

1.2. Related Research

In the last few years, significant advancements have been made in the field of AVs. The current research methodologies on AVs can be classified into two primary approaches: mediated perception and end-to-end driving. The mediated perception approach focuses on breaking the driving task down (e.g., perception, localization, planning, and control to ensure safe driving) into standardized modules. Subsequently, it employs rule-based methods to establish connections between these distinct modules. The mediated perception method, however, exhibits inherent limitations. The primary disadvantage lies in the intricate task of developing and maintaining interconnections between all modules within the system. The modularity paradigm may be compromised in different scenarios that require varying connections between modules [4]. Additionally, constructing and sustaining such a pipeline incurs significant costs despite extensive efforts over many years; thus, it is far from achieving complete autonomy [5]. Consequently, these approaches necessitate a substantial endeavor in designing architectures that integrate all system components and are frequently susceptible to error propagation throughout the entire pipeline [6,7].

Deep learning (DL) techniques are gaining increasing popularity and have emerged as a valuable tool across various industries, including the automotive sector, owing to their exceptional capability to extract image features. DL has significantly impacted AVs' control, particularly in terms of steering-angle prediction, due to its efficient processing of unlabeled raw data and robust extraction of image features. These techniques have facilitated the emergence of the end-to-end driving approach, also called the behavior reflex approach, simplifying traditional subsystems considerably and alleviating the burden associated with vehicle modeling and control [8]. DL-based steering-angle prediction in the end-to-end driving approach offers several advantages, including error tolerance, rapid error identification, and an enhanced capability to manage unpredictable situations [9]. The work of [10] presents an intelligent driving assist system for the real-time prediction of the steering angle using DL and a raw data set collected from a real environment.

To date, many studies have employed deep learning techniques to predict steering angles and control lateral motion in autonomous vehicles. The earliest attempt at end-to-end driving can be traced back to ALVINN, where a three-layer, fully connected network was trained to predict the steering wheel angle based on camera and radar images [11]. Consequently, the researchers subsequently developed an end-to-end driving system specifically designed for off-road conditions. They extensively trained a six-layer convolutional network using a substantial amount of data to accurately simulate the obstacle avoidance behavior exhibited by human drivers [12], the small vehicle, known as DAVE. Ref. [13], expanded upon the DAVE system, training a three-camera model to perform steering control for a vehicle in a range of real-world driving scenarios. This work arguably brought end-to-end systems to the forefront of AV research. A convolutional neural network (CNN)-based end-to-end controller was proposed for steering autonomous vehicles [14]. The network was trained using road screenshots generated via the car simulator CARSIM and human driver steering angles. However, it is worth noting that the network architecture consists of only seven layers, which may limit its capacity, and the training data lack real-world derivation, leading to potential limitations in their generalization ability. The proposed framework in [15] introduces a multi-task demonstration learning (MT-LfD) approach, incorporating an end-to-end trainable network for emulating the driving commands of

an expert demonstrator. Supervised auxiliary task prediction is employed to guide the primary task of predicting driving commands. A neural motion planner is proposed in [16] for autonomous driving in complex urban scenarios, encompassing traffic light processing, concessions, and interaction with multiple road users. To accomplish this, the authors developed a comprehensive model that utilizes raw liDAR data and high-definition maps as inputs to generate interpretable intermediate representations in the form of 3D detections and their future trajectories.

The ease of amassing extensive human driving data renders the end-to-end approach highly effective for straightforward tasks. However, intricate and infrequent traffic scenarios continue to pose challenges for this methodology [17]. From the relevant research above, the primary challenge in DL-based steering-angle prediction for AVs lies in the scarcity of real-world data sets, necessitating a heavy reliance on data generated from simulated environments. The limited depth of existing CNN-based networks has also constrained their training performance. Increasing the network depth can enhance performance. However, it also introduces challenges such as vanishing or exploding gradients and accuracy saturation, followed by rapid degradation [18].

Despite the numerous shortcomings and challenges that persist, end-to-end driving remains the most promising approach for AVs based on previous research findings. This approach offers significant advantages in terms of reducing hardware costs and research complexity compared to alternative methods. Furthermore, its incorporation of diverse data sets enables versatility across various scenarios [6]. The work of [19] presents an innovative approach to end-to-end steering-angle prediction and its control in electric power-steering (EPS) systems. The methodology integrates transfer learning-based computer vision techniques for prediction and control with fuzzy signature-enhanced fuzzy systems. Successfully applied transfer learning-based computer vision technology to extract corresponding visual data without the need for large data sets reduces data collection and the computer load. The experiment shows that the proposed model achieves good performance. Recently, with the rapid advancement of deep learning techniques, computer vision-based object detection, combined with deep learning, has gradually emerged as the predominant method in this field of AVs. This methodology eliminates the need for manual feature extraction and can be categorized into two main groups: two-stage object detection methods represented by Region with CNN (RCNN), Fast RCNN, Faster RCNN, and Mask RCNN [20], and one-stage object detection methods, represented by Single Shot MultiBox Detector (SSD) detection methods series, You Only Look Once (YOLO) detection methods series, and RetinaNet [21–23]. Among these approaches, YOLOv5 stands out as a cutting-edge representation due to its superior speed, enhanced detection accuracy, and reduced file size. Consequently, it finds extensive applications requiring precise object detection tasks in various domains. In this study, drawing inspiration from object detection methodologies, we transform the task of predicting steering angles into an object-detection problem and propose a novel steering controller.

A lot of research has been conducted on steering control in AVs. In order to obtain accurate tracking performance for SbW systems, for example, model-based linear quadratic feedback control [24], trajectory tracking control [25], and model predictive control [26] are widely used. Ref. [27] proposed an adaptive sliding-mode control method to improve control accuracy using self-aligning torque and friction as an external disturbance. Ref. [27] proposed a robust sliding-mode learning control scheme and designed a sliding-mode learning controller to drive the sliding-mode variable in order to converge the tracking error to zero. They ignore system-parameter uncertainty and external disturbances. Ref. [28] proposed a robust, adaptive, integral terminal sliding-mode control strategy based on extreme learning machines, which ensures the finite time convergence of errors and effectively estimates the total uncertainty in a system using a single hidden layer feed-forward network. The work of [29] presents a new, vertical noncontact angle sensor based on the electromagnetic induction principle and conducted nonlinear optimization. A sensor prototype was made and tested in a laboratory. The experimental results show that the

nonlinearity of the sensor was significantly improved, making angle measurement more accurate. This provides accurate angle values in angle-control experiments, which helps improve control accuracy. However, although the above methods have achieved many results, there are still limitations in many areas. For example, it is necessary to accurately know the angular velocity signal, and in practice, additional sensors need to be installed, while using differential methods can amplify measurement noise. It cannot be guaranteed that the tracking error will always be within a specific range, and there may be sudden changes in the tracking error.

1.3. Motivation and Contributions

The remarkable advancements in automotive technology in recent years have been driven by a convergence of several interconnected trends, including the resurgence of deep learning, the rapid evolution of sensing devices and in-vehicle computing systems, the accumulation of annotated data, and significant breakthroughs in related research fields (particularly computer vision) [30]. The rapid progress of deep learning can be attributed to the emergence and extensive application of convolutional neural networks (CNNs) in computer vision and object detection, which has paved the way for autonomous vehicle development. However, numerous challenges still hinder the widespread adoption of AVs in practical applications. For instance, one such challenge pertains to vision-based steering-angle prediction for AV control and designing an effective lateral controller for dynamic path tracking in order to solve the existing problems in the corner prediction and tracking control of autonomous vehicles. For example, the traditional YOLOv5 applied to angle prediction entails high requirements for storage capacity and hardware. Unknown parameters are difficult to measure in steering-angle tracking control, the influence of external disturbance, and model uncertainty on the control effect.

In this paper, we address the challenges above by transforming the task of predicting steering angles into an object-detection problem, and we propose a steering controller. It imitates the driver's perception of road conditions, decision-making, and steering implementation during vehicle operation. It utilizes a camera positioned behind the windshield to translate the observed lane image into a steering control signal. The selection of a DL-based object detection algorithm plays a crucial role in addressing the challenge of steering-angle prediction. Undoubtedly, YOLOv5 has garnered significant attention and demonstrated remarkable accomplishments. However, the constraints on storage capacity and hardware limitations present substantial challenges for deploying the full-scale YOLOv5 network model in vehicular equipment [31]. Therefore, due to its performance merits, YOLOv5s is selected as the network model for predicting steering angles. To achieve accurate steering-angle prediction while enhancing computational efficiency, a lightweight steering-angle prediction network model called YOLOv5Ms is proposed, based on YOLOv5s.

To mimic the driver's behavior during autonomous driving, we categorized lane-line curves into 15 classes, based on their curvature, and we created a corresponding data set for training You Only Look Once version 5 with MobileNet version 3 (YOLOv5Ms) as a network model. During AVs' operation, a camera mounted on the vehicle captures information about lane lines on the road. The trained YOLOv5Ms model is utilized to predict steering angles using these camera images. The processed steering angle is then transmitted to the steering controller via a serial port. Subsequently, our proposed controller receives this data through the serial port to enable appropriate steering control.

In order to ensure that the steering system of AVs can accurately and in a timely manner track the reference angle signal, the steering system's modeling and the controller's design are crucial. In practical engineering, there are many unknown parameters in the SbW system, such as friction torque, self-aligning torque, external disturbance, and angular velocity, that are difficult to measure. With the development of adaptive control technology, neural networks can approximate nonlinear functions with high accuracy. Meanwhile, the state observer and disturbance observer exert a good estimation effect for the variables that are difficult to measure and the external disturbance. Therefore, neural networks,

state observers, and perturbation observers can be used to model SbW systems in order to improve model accuracy for better control accuracy. However, during the angle tracking of AVs, the tracking error should be guaranteed to be within a specific range, which may lead to accidents if sudden changes occur instantaneously. Therefore, we are motivated to explore a control method in an SbW system with model uncertainty, an external disturbance, and difficult-to-measure variables to improve the tracking accuracy while ensuring that the tracking error is always within a specific range.

The contributions of this paper can be summarized as follows:

1. The present paper proposes a lightweight steering-angle prediction network model, namely YOLOv5Ms, based on YOLOv5s to achieve model compression while maintaining detection accuracy and speed. To address the issues of low localization accuracy and slow regression speed in object detection boxes during training, we employ Shape-IoU_Loss as the regression loss function for bounding box improvement.
2. To ensure that the steering system of the autonomous vehicle responds quickly and accurately to the predicted steering-angle signal, an adaptive output feedback control scheme with output constraints based on a neural network is proposed in this paper. The advantage of this control method is that it can constrain the tracking error within a given range, improving the tracking accuracy. Meanwhile, the effect of model uncertainty, external disturbance, and unmeasurable variables on the system are compensated for.
3. To enhance the generalization capability of the proposed detection model in this study, we conducted an extended data collection experiment at Western Xia Park in Yinchuan City, building upon our previously created lane-line data set. To ensure consistency with the previously collected steering-angle information, we once again experimented using a Borgward SUV (BX5) vehicle. This time, we recorded 40 h of real-world driving videos and increased the number of images from 8000 to 20,000 in order to augment the data set's diversity while maintaining an image size of $640 \times 640 \times 3$ pixels. To our knowledge, this manually labeled data set utilizing ImageLabel is currently the most prominent one encompassing lane lines.

The rest of the paper is organized as follows. Section 2 presents the lane-line detection algorithm based on YOLOv5, while Section 3 details the data set's organization and network training. The steering controller design and stability proof are discussed in Section 4, followed by an experimental evaluation in Section 5. Finally, conclusions are drawn in Section 6.

2. Methods

2.1. Lane-Line Detection Algorithm Description

The YOLOv5 algorithm, as the current representative deep learning algorithm in the YOLO series (Appendix A), exhibits exceptional performance in object detection with an accelerated training speed, enhanced accuracy, and broader applicability. Consequently, it holds significant potential for practical applications. Four models are derived based on network depth and feature map width, YOLOv5s, m, l, and x. Specifically, the four models share a consistent network structure comprising input, backbone, neck, and prediction-head components. The YOLOv5 network is categorized as a single-stage, end-to-end detection framework that treats object detection as a regression problem by predicting bounding boxes and class probabilities across the entire image. This model encompasses three crucial processes: boundary-box prediction, class prediction, and feature extraction. The well-designed network architecture enables flexibility and selectivity in diverse scenarios. Therefore, we adopt YOLOv5's one-staged detection framework as the foundational architecture for steering-angle prediction.

The YOLOv5 network model has achieved quite good results, but one of the problems with the steering-angle prediction model-based YOLOv5 is that the model is huge, with many parameters and large calculation amounts, and it is difficult to apply in embedded devices; moreover, steering-angle prediction scenarios require low latency or response

speed. Imagine what terrible things could happen if the detection model for steering-angle prediction were slow. So, working on a small but efficient detection model is crucial in these scenarios, at least for now, although the hardware will also become faster in the future. Therefore, in the steering-angle prediction model based on YOLOv5, there is still room for improvement and enhancement. Optimizing the model architecture by employing smaller convolutional cores and reducing the number of pooling layers can effectively mitigate the parameters and computational load. Different improvement methods can be adopted, based on specific application scenarios with varying detection difficulties. Hence, this paper proposes a lightweight steering-angle prediction network model based on YOLOv5s to achieve a streamlined design while minimizing FLOPs, parameter counts, and the overall model size without compromising detection accuracy.

The MobileNet network exhibits superior advantages in lightweight neural networks due to its reduced size, decreased computational requirements, and enhanced accuracy. Furthermore, mobile models have been constructed using progressively more efficient building blocks. MobileNetV3 is an amalgamation of three models: MobileNetV1's depth-wise separable convolutions, MobileNetV2's inverted residual with a linear bottleneck, and MnasNet's lightweight attention model based on squeeze and excitation (SE) structures [32–34]. To optimize its performance on mobile phone CPUs, MobileNetV3 underwent a hardware-aware network architecture search (NAS) complemented by the NetAdapt algorithm, followed by further enhancements through novel architectural advancements. Two versions of MobileNetV3 are available: Large, for high-performance platforms, and Small, for low-performance platforms. Instead of using partial 3×3 deep convolutions, MobileNetV3 introduces a deep convolution of size 5×5. The SE module and h-swish (HS) activation function are incorporated to enhance model accuracy. These techniques can be effectively combined to discover optimized models tailored to specific hardware platforms.

YOLOv5 is a widely adopted object-detection algorithm, while MobileNetV3 represents a lightweight architecture for convolutional neural networks. Leveraging the strengths of these two models, we propose YOLOv5Ms, a lightweight steering prediction network model based on YOLOv5s. This model efficiently and accurately performs target-detection tasks, achieving a balance between accuracy and latency, making it suitable for deployment on mobile devices.

The schematic of steering-angle prediction and control based on the YOLOv5Ms network is illustrated in Figure 1. During autonomous driving, the vehicle-mounted camera captures road information in the form of images, which are then processed via a pre-trained YOLOv5 network. Subsequently, the predicted steering is transmitted to the steering controller. By mapping the steering-angle signal according to a predefined mapping relation, the input signal for the steering controller is obtained, resulting in corresponding steering outcomes through the actuation of the steering motor.

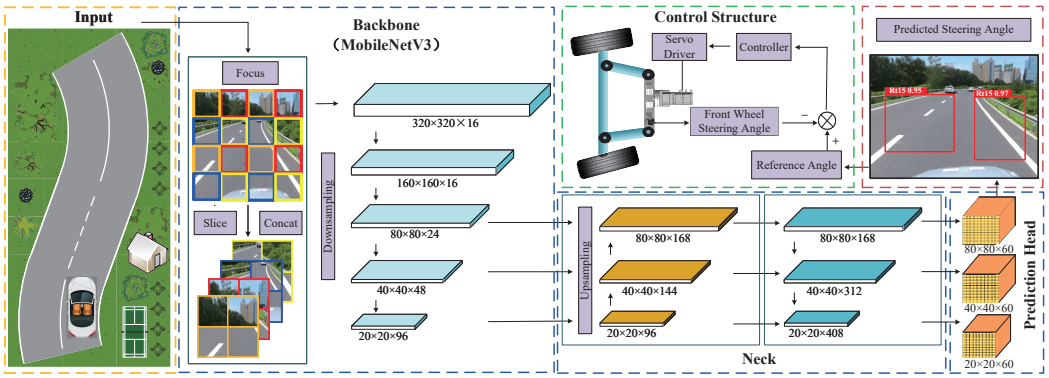


Figure 1. The lightweight steering-angle prediction network model, namely YOLOv5Ms, based on YOLOv5s and a control schematic network.

2.2. Improvement of YOLOv5Ms Network Architecture

In this section, we present a lightweight steering-angle prediction network model based on YOLOv5s. To enhance the performance of YOLOv5s, we have made two key improvements: replacing the original network backbone with MobileNetV3 and substituting the CIoU border regression function with Shape-IoU for better results. To reduce dimensionality without sacrificing features and computational efficiency, we have retained the focus structure at the input end of the network, as designed in YOLOv5. The resulting lightweight steering-angle prediction network model, named YOLOv5Ms, is built upon YOLOv5s, and its control schematic network is illustrated in Figure 1.

Focus: YOLOv5 incorporates a focused structure at the network input to reduce dimensionality without compromising features and computational efficiency. The focus structure divides the preprocessed image into four parts using slicing operations, and it concatenates them. A 20% overlap area is introduced between the two image parts to ensure that lane lines are not disrupted. The specific principles of slicing and concatenation are illustrated in Figure 2. This focus module achieves downsampling while increasing channel dimensions, reducing FLOPs, and improving speed.

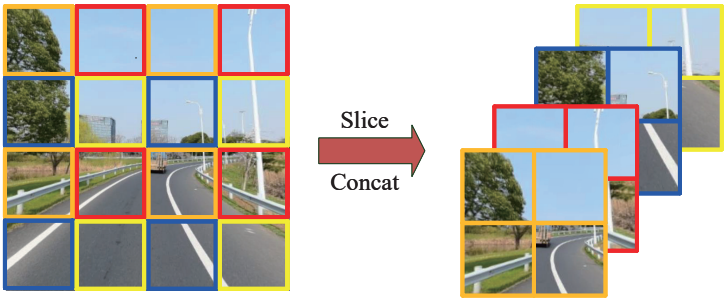


Figure 2. The specific slicing and concat principles.

Backbone: To achieve a lightweight network architecture, we modified the backbone of yolov5s by incorporating the large and small modules of MobileNetV3 as alternative backbones. We named them YOLOv5Ms and YOLOv5Ml, respectively. The specifications for YOLOv5Ms and YOLOv5Ml of the backbone can be seen in Tables 1 and 2. In the table, the input represents the shape transformation of each feature layer, while the operator signifies the block structure to be executed at each feature layer. The variables 'size' and '♯out' respectively denote the number of channels after the inverse residual structure is applied to the bottleneck and the number of channels in the feature layer when it is input into the bottleneck ('bk'). The term 'SE' indicates whether or not an attention mechanism, SE, is introduced in this module. In column six, 'NL' represents the type of activation function, where 'HS' denotes the h-swish, 'RE' represents ReLU, and 's' indicates the stride.

Table 1. Specifications for YOLOv5Ms of the backbone.

Input	Operator	Size	♯ Out	SE	NL	s
$640^2 \times 3$	2d, 3×3	–	16	–	*HS*	2
$320^2 \times 16$	bk, 3×3	16	16	✓	*RE*	2
$160^2 \times 16$	bk, 3×3	72	24	–	*RE*	2
$80^2 \times 24$	bk, 3×3	88	24	–	*RE*	1
$80^2 \times 24$	bk, 5×5	96	40	✓	*HS*	2
$40^2 \times 40$	bk, 5×5	240	40	✓	*HS*	1
$40^2 \times 40$	bk, 5×5	240	40	✓	*HS*	1
$40^2 \times 40$	bk, 5×5	120	48	✓	*HS*	1
$40^2 \times 48$	bk, 5×5	144	48	✓	*HS*	1
$40^2 \times 48$	bk, 5×5	288	96	✓	*HS*	2
$20^2 \times 96$	bk, 5×5	576	96	✓	*HS*	1
$20^2 \times 96$	bk, 5×5	576	96	✓	*HS*	1

Table 2. Specifications for YOLOv5Ml of the backbone.

Input	Operator	Size	♯ Out	SE	NL	s
$640^2 \times 3$	2d, 3×3	–	16	–	*HS*	2
$320^2 \times 16$	bk, 3×3	16	16	–	*RE*	1
$320^2 \times 16$	bk, 3×3	64	24	–	*RE*	2
$160^2 \times 24$	bk, 3×3	72	24	–	*RE*	1
$160^2 \times 24$	bk, 5×5	72	40	✓	*RE*	2
$80^2 \times 40$	bk, 5×5	120	40	✓	*RE*	1
$80^2 \times 40$	bk, 5×5	120	40	✓	*RE*	1
$80^2 \times 40$	bk, 3×3	240	80	–	*HS*	2
$40^2 \times 80$	bk, 3×3	200	80	–	*HS*	1
$40^2 \times 80$	bk, 3×3	184	80	–	*HS*	1
$40^2 \times 80$	bk, 3×3	184	80	–	*HS*	1
$40^2 \times 80$	bk, 3×3	480	112	✓	*HS*	1
$40^2 \times 112$	bk, 3×3	672	112	✓	*HS*	1
$40^2 \times 112$	bk, 5×5	672	160	✓	*HS*	2
$20^2 \times 112$	bk, 5×5	960	160	✓	*HS*	1
$20^2 \times 160$	bk, 5×5	960	160	✓	*HS*	1

The bottleneck structure is depicted in Figure 3, encompassing an inverse residual architecture that incorporates a linear bottleneck, depthwise separable convolution, and an SE attention mechanism. In the case of a stride value of 1 and input channels equal to output channels for the module, a shortcut connection is established between the input and output. Otherwise, subsequent operations follow the depthwise separable convolution operation, the SE attention mechanism, and the 1×1 point convolution operation. The depth separable convolution and squeeze-and-excitation block attention mechanism, SE, of the linear bottleneck are further described below.

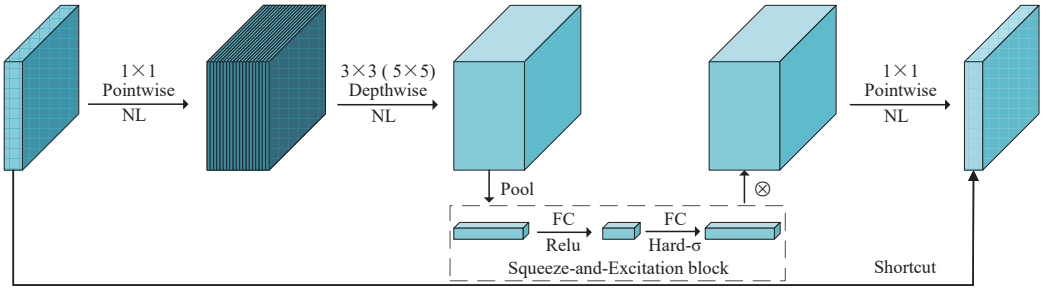


Figure 3. Bottleneck block structure.

(1) Depthwise separable convolution

The standard convolution operation takes an input tensor, F, of size $D_F \times D_F \times M$ and applies a convolutional kernel, $K \in R^{k \times k \times M \times N}$, to generate an output tensor, G, of size $D_G \times D_G \times N$. Here, D_F represents the spatial width and height of a square input feature map, M denotes the number of input channels (input depth), and D_G corresponds to the spatial width and height of a square output feature map. Finally, N signifies the number of output channels (output depth) [35].

The standard convolutional layer is parameterized by a square convolution kernel, K, with spatial dimensions of $D_K \times D_K \times M \times N$, where M represents the number of input channels, and N represents the number of output channels, as defined previously. Assuming stride one and padding, the computation for the output feature map in standard convolution can be expressed as follows:

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m} \quad (1)$$

Standard convolutions have the computational cost of

$$C_{s-cost} = D_F \times D_F \times D_K \times D_K \times M \times N \tag{2}$$

It is evident that the computational cost is multiplicatively dependent on various factors, including the number of input channels (M) and output channels (N), the kernel size ($D_K \times D_K$), and the feature map size ($D_F \times D_F$).

Depthwise separable convolutions serve as a fundamental component in numerous efficient neural network architectures, and we employed them in our current study as well. This form of factorized convolutions breaks a standard convolution down into two parts: depthwise convolution, which applies a single filter per input channel for lightweight filtering, and pointwise convolution (a 1×1 convolution), responsible for generating new features by computing linear combinations of the input channels [32]. This factorization significantly reduces the computation and model size. The standard and depthwise separable convolution principles are illustrated in Figures 4 and 5.

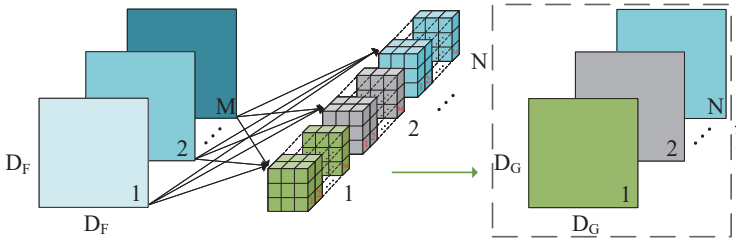


Figure 4. Standard convolution architecture.

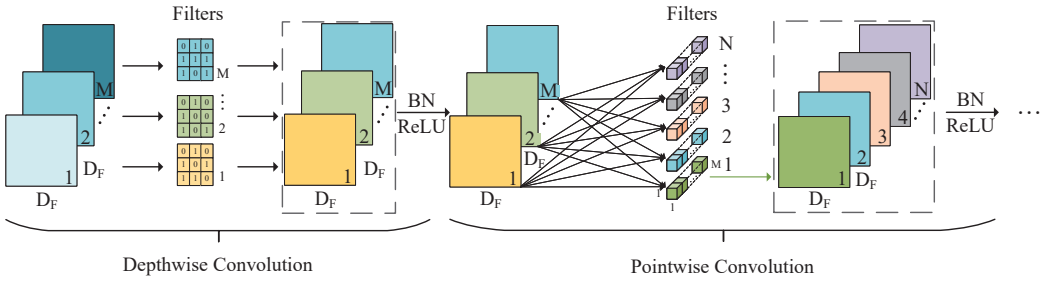


Figure 5. Depthwise separable convolution architecture.

Depthwise convolutions are employed to apply a singular filter for each input channel, while pointwise convolution, represented as a simple 1×1 convolution, is subsequently utilized to generate a linear combination of the output from the depthwise layer. The bottleneck block incorporates both batch normalization and rectified linear unit (ReLU) nonlinearities in both layers.

The depthwise convolution operation establishes a one-to-one correspondence between the convolution kernel and channel. Each channel is convolved with only one specific convolution kernel, generating feature map channels that exactly match the number of input channels. Mathematically, depthwise convolution can be represented as employing a single filter per input channel (input depth):

$$\widetilde{G}_{k,l,m} = \sum_{i,j} \widetilde{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m} \tag{3}$$

where $\widetilde{K}$ is the depthwise convolutional kernel of size $D_K \times D_K \times M$, and the m_{th} filter in $\widetilde{K}$ is applied to the m_{th} channel in F in order to produce the m_{th} channel of the filtered output feature map $\widetilde{G}$. Hence, depthwise convolution has a computational cost of

$$C_{Dw-cost} = D_F \times D_F \times D_K \times D_K \times M \tag{4}$$

The pointwise convolution operations resemble standard convolution operations, as they employ a $1 \times 1 \times M$ convolution kernel, where M represents the number of channels in the preceding layer. Consequently, this convolution operation amalgamates the previous step's map in a depth-wise manner with appropriate weights to generate a novel feature map. Multiple output feature maps are produced for each convolution kernel. Pointwise convolution and depthwise separable convolutions have the computational cost of

$$\begin{aligned} C_{Pw-cost} &= D_F \times D_F \times 1 \times 1 \times M \times N \\ &= D_F \times D_F \times M \times N \end{aligned} \tag{5}$$

$$C_{dp-cost} = {D_F}^2 \times {D_K}^2 \times M + {D_F}^2 \times M \times N \tag{6}$$

By expressing convolution as a two-step process of filtering and combining, we achieve a reduction in the computation of

$$\frac{{D_F}^2 \times {D_K}^2 \times M + {D_F}^2 \times M \times N}{{D_F}^2 \times {D_K}^2 \times M \times N} = \frac{1}{N} + \frac{1}{{D_K}^2} \tag{7}$$

The depthwise separable convolution effectively reduces the computation compared to traditional layers by a factor of approximately k^2. When a 3×3 convolution kernel is assumed to be utilized in the depthwise separable convolution, it requires only 8 to 9 times less computation than the standard convolution without considering bias while maintaining a negligible decrease in accuracy. Figure 6 illustrates the comparison between standard convolution and depthwise, separable convolution with the batch norm and ReLU.

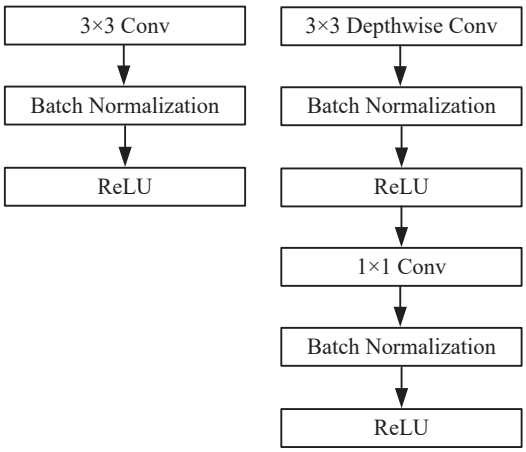


Figure 6. (**Left**): standard convolutional layer with batchnorm and ReLU. (**Right**): depthwise, separable convolution with depthwise and pointwise layers, followed by batchnorm and ReLU.

(2) Squeeze-and-excitation blocks

CNNs have emerged as valuable models for processing diverse visual tasks, making them the fundamental network structure employed in our study [36]. Recent research has demonstrated that incorporating attention mechanisms into CNN can enhance feature representations by effectively capturing spatial correlations. The attention mechanism is inspired by humans' ability to process external information, where individuals selectively attend to relevant information while filtering out irrelevant stimuli due to the limited processing capacity of the human brain [37,38].

In order to obtain a more robust representation, only the most significant attributes for predicting the steering angle in images captured via the front-mounted camera during driving are retained, thereby enhancing performance. We introduce a novel architectural unit called the squeeze-and-excitation (SE) block that explicitly models interdependencies between convolutional feature channels to improve network representations [39].

The structure of the SE building block is depicted in Figure 7. Assume that input feature maps of the SE block $\mathbf{X}$ have shape $W' \times H' \times C'$, where W' is the width, H' is height, and C' is the channels of feature maps. $\mathbf{F}_{tr}$ can be thought of as a standard convolution operator. $\mathbf{F}_{tr}$ maps an input $\mathbf{X} \in \mathbb{R}^{W' \times H' \times C'}$ to feature maps $\mathbf{U} \in \mathbb{R}^{W \times H \times C}$. In the notation that follows, we use $\mathbf{K} = [\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_C]$ to denote the learned set of filter kernels, where $\mathbf{k}_c$ refers to the parameters of the cth filter. We can then write the outputs as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_C]$, where $\mathbf{u}_c$ can be expressed mathematically as follows:

$$\mathbf{u}_c = \mathbf{k}_c * \mathbf{X} = \sum_{m=1}^{C'} \mathbf{k}_c^m * \mathbf{x}^m \tag{8}$$

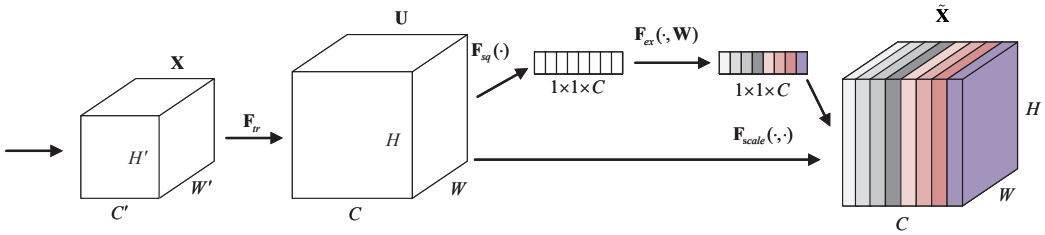


Figure 7. The structure of the SE building block.

Here, $\mathbf{k}_c = [\mathbf{k}_c^1, \mathbf{k}_c^2, \cdots, \mathbf{k}_c^{C'}]$, $*$ denotes a convolution operation, $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^{C'}]$, and $\mathbf{u}_c \in \mathbb{R}^{H \times W}$. $\mathbf{k}_c^m$ is a 2D spatial kernel representing a single channel of $\mathbf{k}_c$ that acts on the corresponding channel of $\mathbf{X}$. To enhance the clarity of notation, the inclusion of bias terms has been omitted.

In addition to high-level channel relationships, convolutional models inherently involve implicit and localized channel dependencies. We anticipate that explicitly modeling interdependencies between channels will enhance the learning of convolutional features, thereby improving the network's sensitivity to informative features that subsequent transformations can effectively utilize. Consequently, we aim to provide it with access to global information and recalibrate filter responses in two steps, namely squeezing and excitation, before they are fed into the next transformation. The SE attention mechanism enables the feature map to access global information and recalibrate the filter response through squeezing and excitation steps before proceeding with subsequent transformations [39].

After $\mathbf{F}_{tr}$ mapping an input, $\mathbf{X} \in \mathbb{R}^{W' \times H' \times C'}$, to feature maps, $\mathbf{U} \in \mathbb{R}^{W \times H \times C}$, to solve the problem of utilizing channel dependencies, we squeeze global spatial information into a channel description by using global average pooling operations to generate channel-wise statistics. The function of this descriptor is to generate an embedment of the corresponding global distribution of channel features, allowing all layers to use information from the global receptive domain (receptive field) of the network. Formally, the statistic $\mathbf{z} \in \mathbb{R}^C$ is generated via a shrinking of $\mathbf{U}$ through its spatial dimension $H \times W$, and the c-th element of $\mathbf{z}$ is calculated as follows:

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{M} u_c(i, j) \tag{9}$$

Here, $\mathbf{F}_{sq}$ represents the global average pooling of channels.

To take advantage of the information aggregated in the squeeze operation, the second operation (excitation operation) is performed to capture the channel-wise dependency fully. To limit the complexity of the model and facilitate generalization, parameterized gating mechanisms are parameterized by forming a bottleneck containing two fully connected layers (FCs) around the nonlinearity. The first fully connected layer compresses C channels into $\frac{C}{r}$ channels to reduce the computational load, and a ReLU nonlinear activation, δ, is then used. The second fully connected layer returns to the number of the channel dimension

to the original C channels and then obtains the weight $\mathbf{s}$ through sigmoid activation, σ. Therefore, the weight, $\mathbf{W}$, of C feature maps in $\mathbf{U}$ can be expressed as follows:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1\mathbf{z})) \tag{10}$$

where r refers to the proportion of compression, and we take $r = 4$. $\mathbf{s} \in \mathbb{R}^{1\times1\times C}$, $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r}\times C}$, $\mathbf{W}_2 \in \mathbb{R}^{C\times\frac{C}{r}}$. The final output of the squeeze-and-excitation block is obtained by rescaling $\mathbf{U}$ using the activation $\mathbf{s}$:

$$\widetilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c\mathbf{u}_c \tag{11}$$

where $\widetilde{\mathbf{X}} = [\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \ldots, \widetilde{\mathbf{x}}_C]$, and $\mathbf{F}_{scale}(\mathbf{u}_c, s_c)$ refer to channel-wise multiplication between the scalar s_c and the feature map $\mathbf{u}_c \in \mathbb{R}^{H\times W}$. SE blocks intrinsically introduce dynamics conditioned on the input, which can be regarded as a self-attention function on channels whose relationships are not confined to the local receptive field that the convolutional filters are responsive to. The SE blocks are also a flexible plug-and-play module. When a bottleneck module in the network backbone does not use SE, the bottleneck module can carry out the normal convolution process.

2.3. Loss Function

Object detection is a fundamental challenge in computer vision tasks, and bounding-box regression plays a pivotal role in accurately predicting the location of target objects. In our proposed YOLOv5Ms network, we make predictions for boxes at three different scales [40]. The final scale generates a 3D tensor that encodes information about bounding boxes, objectness scores, and class predictions. At each scale, we predict three boxes, resulting in a tensor of size $D \times D \times [3 \times (4 + 1 + 15)]$ to account for the four bounding box offsets, one objectness prediction, and fifteen class predictions. Herein, D represents the grid size of the detection head.

Regarding the evaluation metric for bounding-box regression, conventional object detectors typically employ the mean square error (MSE) to directly regress the center-point coordinates, height, and width of the bounding box (BBox). However, treating each point of the BBox as independent variables when estimating their coordinate values fails to consider the holistic nature of the object itself. To address this issue and overcome the neglect of object integrity, a novel intersection over union (IoU) loss function was proposed for bounding boxes [41]. This loss function takes into consideration the coverage of both predicted and ground-truth bounding-box areas. It jointly regresses all bound variables as a unified unit by directly enforcing the maximal overlap between the predicted bounding box and the ground truth, as illustrated in Figure 8. The schematic diagram in Figure 8 demonstrates the process of bounding-box regression for this loss function. The X-axis represents the horizontal direction, while the Y-axis represents the vertical direction. The computation of the IoU loss involves calculating four coordinate points of the BBox by comparing them with the ground truth and then connecting these generated results to form a complete code. The most popular metric of IoU is as follows:

$$IoU = \frac{|G_T \bigcap P_b|}{|G_T \bigcup P_b|} \tag{12}$$

$$G_T = (x_2 - x_1) \times (y_2 - y_1) \tag{13}$$

$$P_b = (x_4 - x_3) \times (y_4 - y_3) \tag{14}$$

where G_T is the ground truth area with lane lines, as labeled by our own bounding box, and P_b is the predicted area of the object bounding box. $M(x_1, y_1)$, $N(x_2, y_2)$, $P(x_3, y_3)$,

and $Q(x_4, y_4)$ are the left top and bottom right corners of the two bounding boxes, respectively. The intersection of G_T and P_b is calculated via Equation (15):

$$|G_T \bigcap P_b| = |(min(x_2, x_4) - max(x_1, x_3) \times (min(y_2, y_4) - max(y_1, y_3)| \quad (15)$$

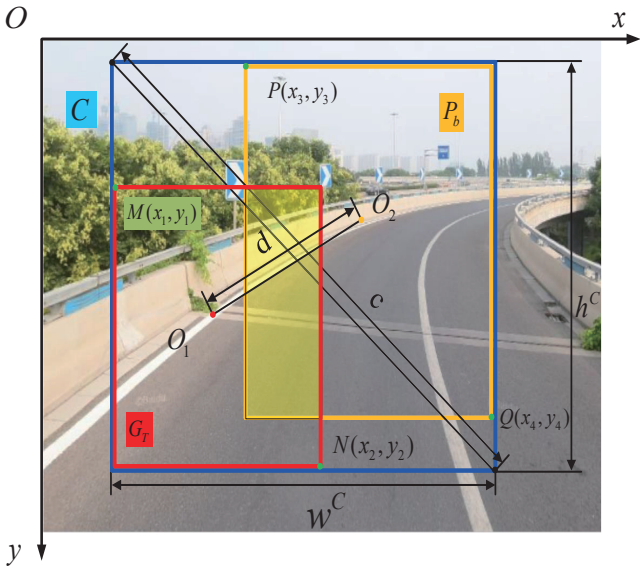


Figure 8. Schematic diagram of the bounding-box regression loss function.

Therefore, the IoU loss function can be expressed as follows:

$$L_{IoU} = 1 - \frac{|G_T \bigcap P_b|}{|G_T \bigcup P_b|} \quad (16)$$

However, the IoU loss only works when the bounding boxes have overlap and would not provide any moving gradient for non-overlapping cases, and then the generalized IoU loss (GIoU) was proposed in [42], adding a penalty term C expressed as follows:

$$|C| = |[\max(x_2, x_4) - \min(x_1, x_3)] \times [\max(y_2, y_4) - \min(y_1, y_3)]| \quad (17)$$

The GIoU loss function can be expressed as follows:

$$L_{GIoU} = 1 - IoU + \frac{|C - G_T \bigcup P_b|}{|C|} \quad (18)$$

where C is the smallest box covering G_T and P_b, as is shown in Figure 8. The GIoU loss includes the shape and orientation of the object, in addition to the coverage area. They proposed finding the smallest-area BBox that can simultaneously cover the predicted BBox and ground-truth BBox and using this BBox as the denominator to replace the denominator originally used in the IoU loss.

Introducing the penalty term facilitates the movement of the predicted box towards the target box in non-overlapping scenarios. Despite its ability to alleviate the gradient vanishing issue for such cases, GIoU still exhibits certain limitations. When bounding boxes enclose objects, the GIoU loss degrades to the IoU loss. A new approach called the distance-IoU (DIoU) loss is proposed for bounding-box regression to address this limitation. The DIoU loss incorporates a penalty term into the IoU loss to directly minimize the normalized distance between the central points of two bounding boxes, resulting in significantly faster convergence compared to the GIoU loss [43]. The objective of the DIoU

loss is to simultaneously consider both the overlap area and the central-point distance when evaluating bounding boxes. Generally, the DIoU loss can be defined as follows:

$$L_{DIoU} = 1 - IoU + \frac{\rho^2(O_1, O_2)}{c^2} \tag{19}$$

As shown in Figure 8, O_1 and O_2 denote the central points of G_T and P_b, respectively. $\rho(\cdot) = \|O_2 - O_1\|_2$ is the Euclidean central distance of the predicted BBox and ground-truth BBox, and c is the diagonal length of C, which is the smallest enclosing box covering the two boxes.

The loss function for bounding-box regression should incorporate three crucial geometric measures, namely the overlap area, central-point distance, and aspect ratio. Consequently, building upon the DIoU loss, ref. [43] proposed a comprehensive IoU (CIoU) loss that integrates these aforementioned geometric measures. This novel approach facilitates faster convergence and yields superior performance compared to both IoU and GIoU losses. Then, the loss function can be defined as follows:

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(O_1, O_2)}{c^2} + \alpha v \tag{20}$$

$$v = \frac{4}{\pi^2}(\arctan\frac{w^{G_T}}{h^{G_T}} - \arctan\frac{w^{P_b}}{h^{P_b}})^2 \tag{21}$$

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{22}$$

where α is a positive trade-off parameter, and v measures the consistency of the aspect ratio. w^{G_T} and h^{G_T} are the width and height of the ground-truth bounding box, while w^{P_b} and h^{P_b} represent the width and height of the predicted bounding box. The gradient of v with reference to w^{P_b} and h^{P_b} should be specified as follows:

$$\begin{aligned}\frac{\partial v}{\partial w^{P_b}} = \frac{8}{\pi^2}(\arctan\frac{w^{G_T}}{h^{G_T}} - \arctan\frac{w^{P_b}}{h^{P_b}}) \\ \times \frac{h^{P_b}}{{w^{P_b}}^2 + {h^{P_b}}^2}\end{aligned} \tag{23}$$

$$\begin{aligned}\frac{\partial v}{\partial h^{P_b}} = -\frac{8}{\pi^2}(\arctan\frac{w^{G_T}}{h^{G_T}} - \arctan\frac{w^{P_b}}{h^{P_b}}) \\ \times \frac{w^{P_b}}{{w^{P_b}}^2 + {h^{P_b}}^2}\end{aligned} \tag{24}$$

The dominator ${w^{P_b}}^2 + {h^{P_b}}^2$ is usually a small value for the cases w^{P_b} and h^{P_b}, with a range of [0, 1], which is likely to yield gradient explosion, and thus, in our implementation, the dominator ${w^{P_b}}^2 + {h^{P_b}}^2$ is simply removed for stable convergence, through which the step size $\frac{1}{{w^{P_b}}^2 + {h^{P_b}}^2}$ is replaced with 1, and the gradient direction is still consistent with Equations (23) and (24).

Since the v only reflects the discrepancy of the aspect ratio, the CIoU loss may optimize the similarity unreasonably. To address the above problems, the work of [44] revised the CIoU loss and proposed a more efficient version of IoU (EIoU), the EIoU loss, which is defined as follows:

$$\begin{aligned}L_{EIoU} = 1 - IoU + \frac{\rho^2(O_1, O_2)}{c^2} + \frac{\rho^2(w^{G_T}, w^{P_b})}{(w^C)^2} \\ + \frac{\rho^2(h^{G_T}, h^{P_b})}{(h^C)^2}\end{aligned} \tag{25}$$

where h^C and w^C are the width and height of the smallest enclosing box, C, covering the two boxes.

Based on previous research, SIoU further considers the influence of the angle between the bounding boxes on the bounding box regression, which aims to accelerate the convergence process by decreasing the angle between the anchor box and the G_T box, which is the horizontal or vertical direction [45]. Its definition is as follows:

$$L_{SIoU} = 1 - IoU + \frac{\Delta + \Omega}{2} \tag{26}$$

$$\Delta = \sum_{t=w^{P_b},h^{P_b}} (1 - e^{-\gamma\rho_t}), \gamma = 2 - \Lambda \tag{27}$$

$$\Lambda = \sin(2\sin^{-1} \frac{min(|x_c^{G_T} - x_c^{P_b}|, |y_c^{G_T} - y_c^{P_b}|)}{\sqrt{(x_c^{G_T} - x_c^{P_b})^2 + (y_c^{G_T} - y_c^{P_b})^2 + \epsilon}}) \tag{28}$$

$$\begin{cases} \rho_{w^{P_b}} = (\frac{x_c^{G_T} - x_c^{P_b}}{w^C})^2 \\ \rho_{h^{P_b}} = (\frac{y_c^{G_T} - y_c^{P_b}}{h^C})^2 \end{cases} \tag{29}$$

$$\Omega = \sum_{t=w^{P_b},h^{P_b}} (1 - e^{-\omega_t})^\theta, \theta = 4 \tag{30}$$

where the value of θ defines how much the shape costs and its unique value for each data set. The value of θ is a significant term in this equation; it controls how much attention should be paid to the cost of the shape. If a value of θ is set to be 1, it will immediately optimize the shape, thus harming the free movement of a shape. To calculate the value of θ, the genetic algorithm is used for each data set; experimentally, the value of θ near to 4, and the range that the author defined for this parameter is from 2 to 6.

$$\begin{aligned} \omega_{w^{P_b}} &= \frac{|w^{G_T} - w^{P_b}|}{max(w^{G_T}, w^{P_b})} \\ \omega_{h^{P_b}} &= \frac{|h^{G_T} - h^{P_b}|}{max(h^{G_T}, h^{P_b})} \end{aligned} \tag{31}$$

In conclusion, the previous bounding box regression methods mainly achieve more accurate regression by adding new geometric constraints on top of IoU. The above methods considered the influence of the distance, shape, and angle of the G_T box and the anchor box on the bounding-box regression but neglected the fact that the shape and scale of the bounding box itself will also exert influences on the bounding-box regression. In order to further improve the accuracy of the regression, the authors proposed a new generation of the bounding regression loss: Shape-IoU in [46].

$$w_{wt} = \frac{2 \times (w^{G_T})^{scale}}{(w^{G_T})^{scale} + (h^{G_T})^{scale}} \tag{32}$$

$$h_{wt} = \frac{2 \times (h^{G_T})^{scale}}{(w^{G_T})^{scale} + (h^{G_T})^{scale}} \tag{33}$$

$$D_{shape} = h_{wt} \times \frac{(x_c^{P_b} - x_c^{G_T})^2}{c^2} + w_{wt} \times \frac{(y_c^{P_b} - y_c^{G_T})^2}{c^2} \tag{34}$$

$$\Omega_{shape} = \sum_{t=w^{P_b},h^{P_b}} (1 - e^{-\omega_t})^\theta, \theta = 4 \tag{35}$$

$$\begin{cases} \omega_{w^{P_b}} = h_{wt} \times \frac{|w^{G_T} - w^{P_b}|}{max(w^{G_T}, w^{P_b})} \\ \omega_{h^{P_b}} = w_{wt} \times \frac{|h^{G_T} - h^{P_b}|}{max(h^{G_T}, h^{P_b})} \end{cases} \tag{36}$$

where scale is the scale factor, which is related to the scale of the target in the data set, and w_{wt} and h_{wt} are the weight coefficients in the horizontal and vertical directions, respectively, whose values are related to the shape of the GT box. D_{shape} is the distance shape. Its corresponding bounding box regression loss is as follows:

$$L_{Shape-IoU} = 1 - IoU + D_{shape} + 0.5 \times \Omega_{shape} \tag{37}$$

YOLOv5 previously used GIoU; for the above reasons, in this paper, we use $L_{Shape-IoU}$ as the bounding-box regression-loss function. Moreover, the excellent performance of Shape-IoU as a bounding-box regression-loss function has been verified in [46].

The loss function used in the training of the YOLOv5Ms in the prediction of the steering angle mainly included the bounding-box location loss($L_{Shape-IoU}$), confidence loss ($L_{confidence}$), and classification loss (L_{class}), as follows:

$$Loss = L_{Shape-IoU} + L_{confidence} + L_{class} \tag{38}$$

$$\begin{aligned} Loss = {} & 1 - IoU + D_{shape} + 0.5 \times \Omega_{shape} - \\ & \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} [\, \tilde{C_i^j} \log(\acute{C_i^j}) + (1 - \tilde{C_i^j}) \log(1 - \acute{C_i^j})] - \\ & \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{noobj} [\, \tilde{C_i^j} \log(\acute{C_i^j}) + (1 - \tilde{C_i^j}) \log(1 - \acute{C_i^j})] \\ & - \sum_{i=0}^{S^2} I_i^{obj} \sum_{\tilde{c} \in class} [\, \tilde{p}_i(\tilde{c}) \log(\acute{p}_i(\tilde{c})) + \\ & (1 - \tilde{p}_i(\tilde{c})) \log(1 - \acute{p}_i(\tilde{c}))] \end{aligned} \tag{39}$$

where S is the number of grids (the image is divided into S×S grids to find the position containing lane lines one by one), and B is the number of anchors for each grid to predict the lane line. I_{ij}^{obj} denotes that the *jth* bounding box predictor in cell *i* is responsible for that prediction; its value is 1 if there is an object in the jth anchor of the ith grid, and otherwise, it is 0. I_i^{obj} denotes whether the object (lane line) appears in cell *i*. $\tilde{C_i^j}$ and $\acute{C_i^j}$ stand for confidences. I_{ij}^{noobj} denotes that the *jth* bounding-box predictor in cell *i* is responsible for that prediction. λ_{noobj} is a parameter to decrease the loss from confidence predictions for boxes that do not contain lane lines. $\tilde{p}_i(\tilde{c})$ and $\acute{p}_i(\tilde{c})$ represent the probability that the lane line in cell *i* belongs to class $\tilde{c}$, while $\tilde{c}$ is the number of classes (lane-camber category).

3. Data Sets' Descriptions and Detector Model Training

In this section, we present our own data set, elaborate on the training process, and report the evaluation results of the proposed network model.

3.1. Data Sets' Descriptions

The task of creating custom data sets has been previously addressed in our prior research. To collect lane lines and steering angles, a Borgward SUV (BX5) was utilized at 20 km per hour within the Shenyang Qipanshan Scenic Area. Building upon our previous work, we conducted an expansion acquisition experiment at Western Xia Park in Yinchuan. To ensure consistency with the collected steering-angle information from before, we employed the BX5 vehicle once again for experimentation purposes. This time,

we recorded 40 h of real-time driving video and augmented the data set by increasing the number of images from its original count of 8000 to 20,000, maintaining the image size of 640 × 640 × 3 pixels. Following statistical analysis on the distribution of collected steering angles, which ranged between −50 and 60 degrees, we divided these angles into 15 categories, ranging from −60 to +60 degrees, as illustrated in Table 3. The label denotes the sequential number of the data sets' labels. The classification of images in a data set is exemplified in Figure 9. Subsequently, we partitioned our data set into three subsets: a training set (70%), a validation set (20%), and a test set (10%), based on a predetermined ratio.

Table 3. Lane-turning categories and steering angles.

Classes	L60	L45	L35	L25	L15	L10	L05	G	R05	R10	R15	R25	R35	R45	R60
Angle	−60	−45	−35	−25	−15	−10	−5	0	5	10	15	25	35	45	60
Label	14	13	12	11	10	9	8	0	1	2	3	4	5	6	7



Figure 9. Illustrative images extracted from the acquired unprocessed training data.

3.2. Measurement Metrics

The performance evaluation of the proposed YOLOv5Ms, based on YOLOv5s, was conducted using precision, recall, mean average precision (mAP@0.5; mAP@0.5:0.95), average detection processing time, parameter count, FLOPs, and model size as measurement metrics [47].

Precision is calculated as the proportion of the number of positive samples correctly predicted to those predicted as positive. It is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{40}$$

Recall is calculated as the proportion of all targets that are correctly predicted. It is defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{41}$$

The average precision (AP) represents the mean precision rate, which should be maximized while ensuring accuracy as the recall rate gradually increases from 0 to 1. The calculation formulas of mAP@0.5 and mAP@0.5:0.95 are as follows:

$$mAP = \frac{\sum_{\hat{c}=1}^{\hat{C}} AP(\hat{c})}{\hat{C}} \tag{42}$$

where TP, FP, and FN represent the counts of true positive (TP) cases (indicating correctly predicted lane lines in the model), false positive (FP) cases (denoting situations where no lane line exists but a lane is falsely predicted in the model), and false negative (FN) cases (referring to scenarios where lane lines exist but are incorrectly predicted as absent in the model), respectively. Additionally, $\hat{C}$ denotes the number of curvature categories for lane lines. In this study, we categorized lane-line curves into 15 classes, resulting in $\hat{C}$ being equal to 15. The term mAP@0.5 refers to the mean average precision across all categories

when the intersection over union (IoU) is set to 0.5, while mAP@0.5:0.95 represents the average mAP at various IoU thresholds, ranging from 0.5 to 0.95, with a step size of 0.05.

3.3. Training and Test Results

Training object detection is a computationally intensive task, particularly following the completion of data-set collection and tagging. In supervised training, the network's weights or independent parameters are iteratively adjusted to optimize performance for a specific training data set. The proposed model's training process is executed using the PyTorch framework on the Windows operating system. The software environment includes CUDA 11.0 and Python 3.8. For data-set training, an Advanced Micro Devices (AMD) Ryzen 2700X processor was employed, alongside an NVIDIA GeForce RTX 2080Ti with 11 GB of memory.

Details of the training process and results are shown in Table 4 and Figures 10 and 11. The number of training epochs was set to 500 during network training; hence, 'Epoch' in Figures 10 and 11 corresponds to a total of 500 iterations. After the training, the weights of the 6 models were 10.1 M, 36.2 M, 14.5 M, 40.3 M 88.7 M, and 173.3 M, respectively. The total number of parameters of the six models is 5.07×10^6, 17.85×10^6, 7.05×10^6, 20.93×10^6, 46.19×10^6, and 86.32×10^6, respectively. Comparing the training results of our proposed YOLOv5Ms models and the four YOLOv5 models can reveal that the total number of parameters in YOLOv5Ms is reduced by 1.98×10^6 compared to YOLOv5s, resulting in a 28.09% decrease and a weight reduction of 4.4 M, which represents a decline of 30.34%. In terms of the training time, under identical hardware conditions, YOLOv5Ms exhibits a training speed that is 5.19 h faster than that of YOLOv5s. Furthermore, regarding precision metrics such as mAP@0.5 and mAP@0.5:0.95, the performance improvement achieved using YOLOv5Ms surpasses that of YOLOv5s, with enhancements reaching 6.59%, 0.41%, and 5.01%, respectively. The experimental results demonstrate that incorporating an attention mechanism into YOLOv5s effectively enhances model precision in reducing both total parameters and weights.

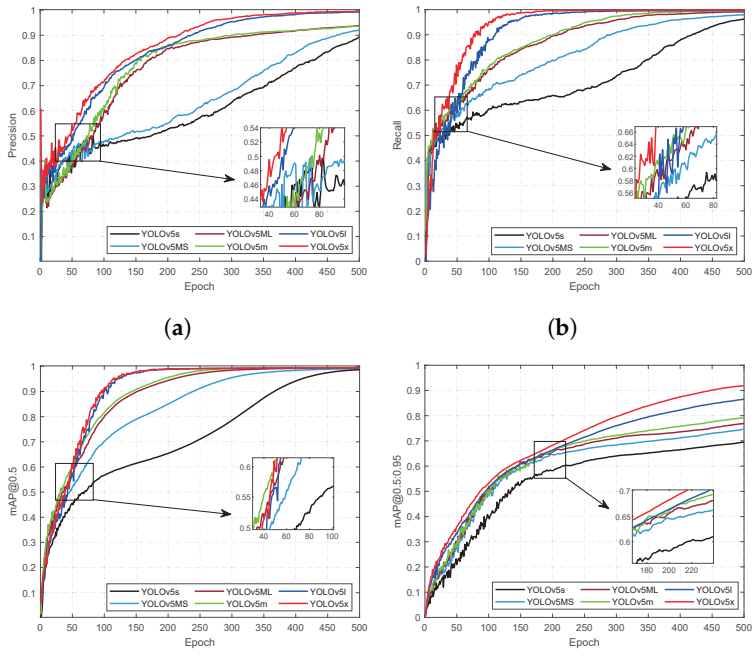


Figure 10. Comparison of the precision, recall rate, mAP@0.5, and mAP@0.5:0.95 of the six models of YOLOv5 in the training and verification stages: (**a**) precision, (**b**) recall, (**c**) mAP@0.5, and (**d**) mAP@0.5:0.95.

Table 4. Details of training process and results.

Model	Params ($\times 10^6$)	Weight (M)	Loss ($\times 10^{-3}$)			Pre (%)[2]	Rec (%)[2]	mAP (%)		$T(h)$[2]
			T_C^1	T_O^1	T_{cl}^1			$m_{0.5}^2$	$m_{.5:.95}^2$	
YOLOv5Ms	5.07	10.1	19.80	11.57	5.71	95.83	96.28	99.10	74.60	16.67
YOLOv5Ml	17.85	36.2	18.82	10.51	3.54	97.32	97.93	99.42	76.89	25.95
YOLOv5s	7.05	14.5	19.59	12.61	7.55	89.24	90.09	98.69	69.57	21.86
YOLOv5m	20.93	40.3	15.52	9.18	3.32	99.05	99.09	99.49	79.16	41.67
YOLOv5l	46.19	88.7	14.49	7.85	2.71	99.22	99.40	99.41	86.59	58.09
YOLOv5x	86.32	173.3	13.38	6.59	2.14	99.50	99.64	99.42	91.85	76.27

[1] T_C, T_O, and T_{cl}: represent each error of training, respectively. [2] Pre: precision; Rec: recall; $\mathbf{m}_{0.5}$: mAP@0.5; $\mathbf{m}_{.5:.95}$: mAP@0.5:0.95; T(h): training time (hours).

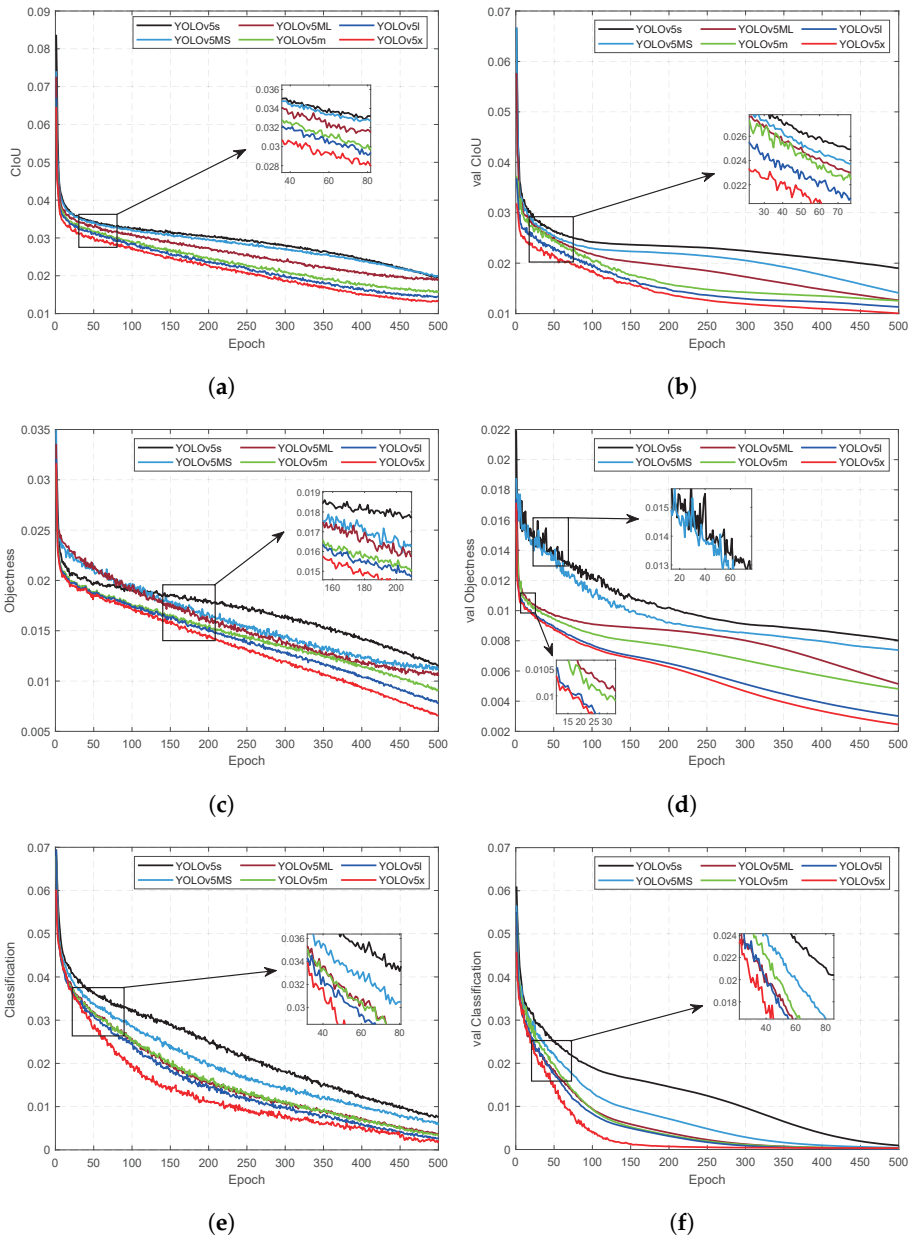


Figure 11. Comparison of the bounding-box regression, confidence, and classification of the six models of YOLOv5 in the training and validation stages: (**a**) CIoU, (**b**) val CIoU, (**c**) objectness, (**d**) val objectness, (**e**) classification, and (**f**) val classification.

Through a comparison between our proposed YOLOv5Ml model and YOLOv5s and YOLOv5m models, we observed that the various indicators of YOLOv5Ml outperform those of YOLOv5s but are similar to those of YOLOv5m. However, its overall performance is slightly inferior to that of YOLOv5m due to dominant factors such as model size, weight size, and training time. Compared with YOLOv5m, the total number of parameters in our model was reduced by 3.08×10^6, the weight size decreased by 4.1 M, and the training time was shortened by 15.72 h.

YOLOv5x and YOLOv5l exhibit superior performance, irrespective of the associated time and hardware costs. However, in practical applications, a common approach is to balance the model cost and overall performance. Therefore, it is generally preferred to consider lower-cost factors such as hardware utilization and execution time while ensuring that the model's performance meets real-world requirements.

The test results of the trained model are shown in Figure 12. In Figure 12, the "tags" represent the YOLOv5Ms network subsequent to 500 epochs of training. Following the evaluation of images containing lane lines, the letters denote their respective categories, while the numbers indicate their corresponding probabilities. It can be seen from the test results that the training of the network achieved the expected performance. We discuss the real-time predictive performance of the trained YOLOv5Ms detector by verifying experiments in Section 5.

This paper proposes a composite controller that integrates a novel adaptive updating law and disturbance observer to ensure that AVs' steering system promptly and accurately responds to the predicted steering-angle signal. Subsequently, we elaborate on the proposed controller in the following section of this paper and conduct a comparative experiment to validate its control performance.

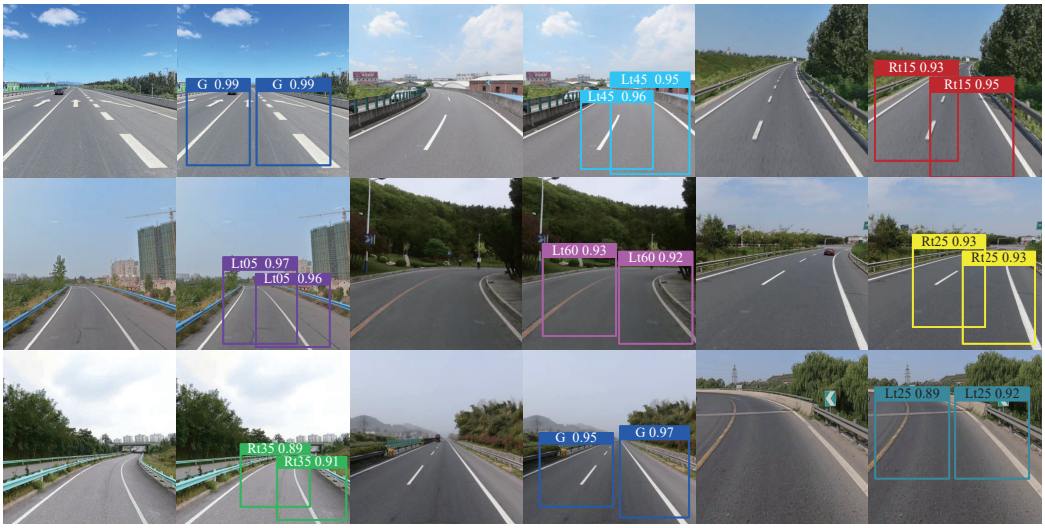


Figure 12. Images labeled with tags and the outcomes of tests.

4. Mathematical Model of SbW System and Controller Design

4.1. Dynamics Model

Figure 13 shows the schematic diagram of the SbW system. The SbW system consists mainly of mechanical and electrical structures. It mainly includes a handwheel, a steering column, a steering motor, a motor controller, a gear reducer, an angle sensor, and a feedback motor. On the handwheel side, the angle sensor is mounted on the steering column, and it measures the angle of the handwheel. The motor controller receives the angle signal through CAN to control the motor rotation, and the motor rotation drives the front-wheel rotation through the gear reducer. The sensors mounted on the steering-tie rods measure the steering angle of the front wheels and feedback to the motor controller. The motor controller receives the feedback signal for closed-loop control to track the reference angle.

For simplicity, the modeling of the steering system consists of the steering motor and the front wheel, and the dynamic equations of the steering motor are as follows:

$$J_{sm}\ddot{\delta}_s + B_{sm}\dot{\delta}_s + \tau_1 + \tau_d = \tau_m \tag{43}$$

where δ_s is the angle of the steering motor shaft. J_{sm} is the inertia of the steering motor. B_{sm} is the viscous friction of the steering motor. τ_1 represents the torque applied via the front wheels to the steering motor shaft. τ_d is the disturbance torque. τ_m is the torque output via the steering motor.

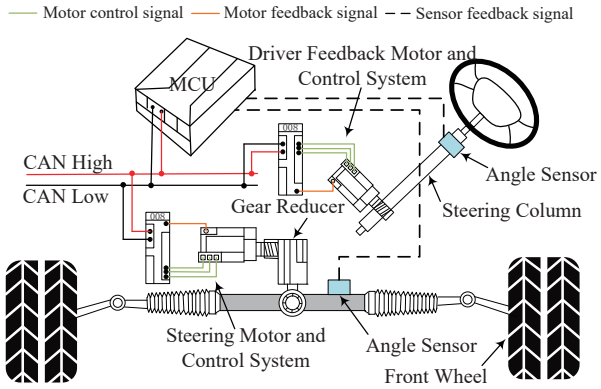


Figure 13. Overall structure diagram of SbW system.

The dynamic equation for the front wheels is as follows:

$$J_{fw}\ddot{\delta}_f + B_{fw}\dot{\delta}_f + \tau_f + \tau_e = \tau_2 \tag{44}$$

where δ_f is the angle of the steering motor shaft. J_{fw} is the inertia of the front wheels. B_{fw} is the viscous friction of the steering motor. τ_f is the friction torque on the front wheels. τ_e is the self-aligning torque on the front wheels. τ_2 represents the torque transmitted via the steering motor to the front-wheel steering arms. The self-aligning moment is expressed as a hyperbolic tangent function:

$$\tau_e = \varrho \tanh(\delta_f) \tag{45}$$

where ϱ is the positive constant related to the road adhesion.

The friction model friction torque is defined as follows [48]:

$$\tau_f = \alpha_1[\tanh(\beta_1\dot{\delta}_f) - \tanh(\beta_2\dot{\delta}_f)] + \alpha_2 \tanh(\beta_3\dot{\delta}_f) + \alpha_3\dot{\delta}_f \tag{46}$$

where α_i and β_i are positive constants. According to (46), the friction model contains many features of other friction models. Therefore, the friction model (46) provides a more comprehensive description of friction and provides a more accurate model for controller design.

Regardless of the backlash exiting the rack and pinion gear teeth, we have the following relationship:

$$\frac{\delta_f}{\delta_s} = \frac{\dot{\delta}_f}{\dot{\delta}_s} = \frac{\ddot{\delta}_f}{\ddot{\delta}_s} = \frac{1}{i_{mc}} = \frac{\tau_1}{\tau_2} \tag{47}$$

where i_{mc} is the steering ratio.

Using (47) and eliminating δ_s, we obtain the following:

$$J_e\ddot{\delta}_f + B_e\dot{\delta}_f + i_{mc}\tau_d + \tau_f + \tau_e = i_{mc}\tau_m \tag{48}$$

where $J_e = J_{fw} + i_{mc}^2 J_{sm}$ is the moment of inertia. $B_e = B_{fw} + i_{mc}^2 B_{sm}$ is the the system damping.

We rewrite (48) as follows:

$$\ddot{\delta}_f = \frac{i_{mc}}{J_e}\tau_m - \frac{B_e\dot{\delta}_f + \tau_e + \tau_f}{J_e} - \frac{i_{mc}\tau_d}{J_e} \tag{49}$$

We define $x_1 = \delta_f$, $x_2 = \dot{\delta}_f$ and rewrite (49) as the state space equation as follows:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = i_e\tau_m - \tau_{ef} - \tau_D \end{cases} \tag{50}$$

where $i_e = \frac{i_{mc}}{J_e}$. $\tau_{ef} = \frac{B_e\dot{\delta}_f+\tau_e+\tau_f}{J_e}$ is the lumped uncertainty. $\tau_D = \frac{i_{mc}\tau_d}{J_e}$ is the external disturbance. The design of the controller requires accurate model parameters such as J_{fw}, J_{sm} and i_{mc}. In engineering practice, these parameters can be obtained through offline parameter identification methods such as the half-period integration method [49] and the zero-mean sinusoidal perturbation method [50]. The transmission ratio i_{mc} can be calculated from the number of gear teeth. However, the accurate friction model τ_f and self-aligning torque τ_e present difficulty in obtaining accurate values. The coefficients α_i and β_i of the friction model are related to road adhesion, and it is difficult to obtain accurate values. The self-aligning torque, τ_e, is an estimated model, and the coefficient ρ is related to road adhesion. Therefore, the friction model and the self-aligning torque need to be approximated using other methods in the controller design.

Lemma 1 ([51]). *For any $(x, y) \in \mathcal{R}^2$, the following Young's inequality is satisfied:*

$$xy \le \frac{\varpi^p}{p}|x|^p + \frac{1}{q\varpi^q}|y|^q \tag{51}$$

where $\varpi > 0$, $p > 1$, and $q > 1$ are constants satifying that $(q-1)(p-1)$=1.

Lemma 2 ([52]). *A function, $F(\boldsymbol{X})$: $\Omega_{\boldsymbol{X}} \to \mathcal{R}$, with a compact set, $\Omega_{\boldsymbol{X}} \in \mathcal{R}^m$, can be approximated using a radial basis function neural network (RBFNN):*

$$F(\boldsymbol{X}) = \boldsymbol{W}^{*T}\boldsymbol{G}(\boldsymbol{X}) + \varepsilon^* \tag{52}$$

where $\boldsymbol{W}^* = [W_1^*, W_2^*, ..., W_l^*]^T \in \mathcal{R}^l$ is the ideal weight vector. $\boldsymbol{X} = [X_1, X_2, ..., X_m]^T \in \mathcal{R}^m$ is the input vector. ε^* is an optimal approximation. There exists a positive constant, $\bar{\varepsilon}_1$, that satisfies $\varepsilon^* < \bar{\varepsilon}_1$. $\boldsymbol{G}(\boldsymbol{X}) = [G_1(\boldsymbol{X}), G_2(\boldsymbol{X}), ..., G_l(\boldsymbol{X})] \in \mathcal{R}^l$ is the basis function vector. The Gaussian function is chosen as the basis function as follows.

$$G_j(\boldsymbol{X}) = \exp\left(-\frac{\|\boldsymbol{X} - \boldsymbol{c}_j\|^2}{2b_j^2}\right) \tag{53}$$

where $\boldsymbol{c}_j = [c_{1j}, c_{1j}, ..., c_{mj}]^T$ is the center, and b_j is the width of the Gaussian function.

Remark 1. *The disturbance τ_D satisfies the following conditions: $\tau_D \le \tau_{D\max}$ and $\dot{\tau}_D \le \varepsilon_D$ ($\tau_{D\max}$; $\varepsilon_D \in R^+$). In much of the previous literature [53], it has been assumed that the perturbations and the derivatives of the perturbations are bounded. In this paper, the external disturbances $\tau_D = \frac{i_{mc}\tau_d}{J_e}$, i_{mc} and J_e are intrinsic parameters of the system, which are related only to the system itself and not to the external world. In engineering practice, τ_d is mainly an external disturbance including the following: internal disturbances in the steering motor, sensor measurement noise, external forces acting on the SbW system, and so on. These disturbances and their derivatives exist in practice with unknown maximum values. Therefore, the unknown $\tau_{D\max}$ and ε_D exist such that $\tau_D \le \tau_{D\max}$ and $\dot{\tau}_D \le \varepsilon_D$.*

4.2. Observer Design

Given that the angular position can be measured directly by the sensor in practice, the angular velocity needs to be measured with an additional sensor, and it is difficult to find a suitable location for the sensor. In observer design, the parameters in the state space equations need to be known. However, model uncertainty and external disturbance make it difficult to obtain accurate values. According to Lemma 2, τ_{ef} can be approximated via RBFNN as follows:

$$\tau_{ef}(x_1, \hat{x}_2) = \mathbf{W}_0^{*T}\mathbf{G}_0(x_1, \hat{x}_2) + \varepsilon_0^* \tag{54}$$

where $\mathbf{W}_0^*$ is the ideal weight. $\hat{x}_2$ is the angular velocity obtained by the state observer designed later.

Define the estimate of τ_{ef} as follows:

$$\hat{\tau}_{ef} = \hat{\mathbf{W}}_0\mathbf{G}_0(x_1, \hat{x}_2) \tag{55}$$

Define $\Delta\tau_{ef} = \tau_{ef}(x_1, x_2) - \tau_{ef}(x_1, \hat{x}_2)$, and satisfy $|\Delta\tau_{ef}| \leq h_f|x_2 - \hat{x}_2|$; h_f is a designed positive constant. Therefore, the dynamic of x_2 can be represented as follows:

$$\dot{x}_2 = i_e\tau_m - \Delta\tau_{ef} - \mathbf{W}_0^{*T}\mathbf{G}_0(x_1, \hat{x}_2) + \varepsilon_0^* - \tau_D \tag{56}$$

Define $\hat{x}_1$ as the estimate of the angular position, $\hat{x}_2$ as the estimate of angular velocity, $\tilde{x}_1 = x_1 - \hat{x}_1$ as the angular position estimation error, and $\tilde{x}_2 = x_2 - \hat{x}_2$ as the angular velocity estimation error. Inspired by [54], the state observer design is as follows:

$$\begin{cases} \dot{\hat{x}}_1 = \hat{x}_2 + k_1\tilde{x}_1 \\ \hat{x}_2 = S_1 + k_3\tilde{x}_1 \\ \dot{S}_1 = i_e\tau_m - \hat{\mathbf{W}}_0\mathbf{G}_0(x_1, \hat{x}_2) - \hat{\tau}_D + k_2\tilde{x}_1 \end{cases} \tag{57}$$

where $k_i (i = 1, 2, 3)$ is the designed positive constants. $\hat{\tau}_D$ is the estimate of τ_D. According to (50), (56), and (57), the following error dynamics equation is obtained:

$$\begin{cases} \dot{\tilde{x}}_1 = -k_1\tilde{x}_1 + \tilde{x}_2 \\ \dot{\tilde{x}}_2 = -\Delta\tau_{ef} - \tilde{\mathbf{W}}_0\mathbf{G}_0(x_1, \hat{x}_2) - \tilde{\tau}_D + \tilde{x}_1(k_1k_3 - k_2) \\ \qquad -k_3\tilde{x}_2 \end{cases} \tag{58}$$

where $\tilde{\tau}_D = \tau_D - \hat{\tau}_D$ is the disturbance estimation error.

To estimate the disturbance, the auxiliary variable λ is introduced as follows:

$$\lambda = \tau_D + \alpha x_2 \tag{59}$$

where α is a designed constant. The disturbance is estimated as follows:

$$\hat{\tau}_D = \hat{\lambda} - \alpha\hat{x}_2 \tag{60}$$

Deriving λ with respect to time gives the following:

$$\dot{\lambda} = \dot{\tau}_D + \alpha(i_e\tau_m - \tau_{ef} - \lambda + \alpha x_2) \tag{61}$$

The design of $\dot{\hat{\lambda}}$ is as follows:

$$\dot{\hat{\lambda}} = \alpha(i_e\tau_m - \hat{\lambda} + \alpha\hat{x}_2) \tag{62}$$

Define $\tilde{\lambda} = \lambda - \hat{\lambda}$ and combine it with (61) and (62) with the derivation for time to obtain the following:

$$\begin{aligned} \dot{\tilde{\lambda}} =& \dot{\lambda} - \dot{\hat{\lambda}} \\ =& \dot{\tau}_D + \alpha(-\mathbf{W}_0^*\mathbf{G}_0(x_1, \hat{x}_2) + \varepsilon_0^* + \Delta\tau_{ef} - \tilde{\lambda} + \alpha\tilde{x}_2) \end{aligned} \tag{63}$$

The Lyapunov function is selected as follows:

$$V_0 = \frac{1}{2}\tilde{x}_1^2 + \frac{1}{2}\tilde{x}_2^2 + \frac{1}{2}\tilde{\lambda} \tag{64}$$

Taking the derivative of (64) yields the following:

$$\begin{aligned}\dot{V}_0 =& \tilde{x}_1\dot{\tilde{x}}_1 + \tilde{x}_2\dot{\tilde{x}}_2 + \tilde{\lambda}\dot{\tilde{\lambda}} \\ =& \tilde{x}_1\tilde{x}_2 - k_1x_1^2 + \tilde{x}_2\Delta\tau_{ef} - \tilde{x}_2\tilde{\mathbf{W}}_0^T\mathbf{G}_0(x_1,\hat{x}_2) + \tilde{x}_2\varepsilon_0^* \\ & - \tilde{x}_2\tilde{\tau}_D + \tilde{x}_1\tilde{x}_2(k_1k_3 - k_2) - k_3\tilde{x}_2^2 + \tilde{\lambda}\dot{\tau}_D + \alpha\tilde{\lambda}\varepsilon_0^* \\ & + \alpha\tilde{\lambda}\Delta\tau_{ef} - \alpha\tilde{\lambda}^2 + \alpha^2\tilde{\lambda}\tilde{x}_2 - \alpha\tilde{\lambda}\mathbf{W}_0^{*T}\mathbf{G}_0(x_1,\hat{x}_2)\end{aligned} \tag{65}$$

Using Lemma 1 yields the following:

$$\begin{cases}\tilde{x}_1\tilde{x}_2 \le \frac{\mu_0}{2}\tilde{x}_1^2 + \frac{1}{2\mu_0}\tilde{x}_2^2 \\ \tilde{x}_2\Delta\tau_{ef} \le |h_f|\tilde{x}_2^2 \\ \tilde{x}_2\tilde{\mathbf{W}}_0^T\mathbf{G}_0(x_1,\hat{x}_2) \le \frac{\mu_2}{2}\tilde{x}_2^2 + \frac{1}{2\mu_1}\tilde{\mathbf{W}}_0^T\tilde{\mathbf{W}}_0 \\ \tilde{x}_2\tilde{\tau}_D \le (\frac{\mu_2}{2} + \frac{\alpha^2}{\mu_2})x_2^2 + \frac{1}{\mu_2}\tilde{\lambda}^2 \\ \tilde{\lambda}\dot{\tau}_D \le \frac{\mu_3}{2}\tilde{\lambda}^2 + \frac{1}{2\mu_3}\varepsilon_D^2 \\ \alpha\tilde{\lambda}\mathbf{W}_0^{*T}\mathbf{G}_0(x_1,\hat{x}_2) \le \frac{\mu_4}{2}\tilde{\lambda}^2 + \frac{1}{2\mu_4}\alpha^2\mathbf{W}_0^{*T}\mathbf{W}_0^* \\ \alpha^2\tilde{\lambda}\tilde{x}_2 \le \frac{\mu_5}{2}\tilde{\lambda}^2 + \frac{1}{2\mu_5}\alpha^4\tilde{x}_2^2 \\ \alpha\tilde{\lambda}\Delta\tau_{ef} \le \frac{\mu_6}{2}\tilde{\lambda}^2 + \frac{1}{2\mu_6}\alpha^2h_f^2\tilde{x}_2^2 \\ \alpha\tilde{\lambda}\varepsilon_0^* \le \frac{\mu_7}{2}\tilde{\lambda}^2 + \frac{\alpha^2}{2\mu_7}\varepsilon_0^{*2} \\ \tilde{x}_2\varepsilon_0^* \le \frac{\mu_8}{2}\tilde{x}_2^2 + \frac{1}{2\mu_8}\varepsilon_0^{*2}\end{cases} \tag{66}$$

According to the above inequalities, we can get the following:

$$\begin{aligned}\dot{V}_0 \le & -(k_1 - \frac{\mu_0(k_1k_3 - k_2 + 1)}{2})\tilde{x}_1^2 - [k_3 - \frac{k_1k_3 - k_2 + 1}{2\mu_0} \\ & + \frac{\mu_8}{2} - \frac{\mu_1}{2} - (\frac{\mu_2}{2} + \frac{\alpha^2}{\mu_2}) - \frac{1}{2\mu_5}\alpha^4 - \frac{1}{2\mu_6}\alpha^2h_f^2]\tilde{x}_2^2 \\ & - (\alpha - \frac{1}{\mu_2} - \frac{\mu_3}{2} - \frac{\mu_4}{2} - \frac{\mu_5}{2} - \frac{\mu_6}{2} - \frac{\mu_7}{2})\tilde{\lambda}^2 + \frac{1}{2\mu_3}\varepsilon_D^2 \\ & + \frac{1}{2\mu_1}\tilde{\mathbf{W}}_0^T\tilde{\mathbf{W}}_0 + \frac{1}{2\mu_4}\alpha^2\mathbf{W}_0^{*T}\mathbf{W}_0^* + (\frac{\alpha^2}{2\mu_7} + \frac{1}{2\mu_8})\varepsilon_0^{*2}\end{aligned} \tag{67}$$

where $\kappa_1 = k_1 - \frac{\mu_0(k_1k_3-k_2+1)}{2}$, $\kappa_2 = k_3 - \frac{k_1k_3-k_2+1}{2\mu_0} + \frac{\mu_8}{2} - \frac{\mu_1}{2} - (\frac{\mu_2}{2} + \frac{\alpha^2}{\mu_2}) - \frac{1}{2\mu_5}\alpha^4 - \frac{1}{2\mu_6}\alpha^2h_f^2$, $\kappa_3 = \alpha - \frac{1}{\mu_2} - \frac{\mu_3}{2} - \frac{\mu_4}{2} - \frac{\mu_5}{2} - \frac{\mu_6}{2} - \frac{\mu_7}{2}$, $\kappa_4 = \frac{1}{2\mu_3}\varepsilon_D^2 + \frac{1}{2\mu_1}\tilde{\mathbf{W}}_0^T\tilde{\mathbf{W}}_0 + \frac{1}{2\mu_4}\alpha^2\mathbf{W}_0^{*T}\mathbf{W}_0^* + (\frac{\alpha^2}{2\mu_7} + \frac{1}{2\mu_8})\varepsilon_0^{*2}$.

According to (67), if $\kappa_1 > 0$, $\kappa_2 > 0$, $\kappa_3 > 0$, and κ_4 is bounded, it follows from Lyapunov's stability theory that the estimation errors $\tilde{x}_1$, $\tilde{x}_2$, and $\tilde{\lambda}$ are ultimately and consistently bounded. Therefore, this paper aims to design a suitable adaptive control

method to guarantee that all states of the closed-loop system, including the observer, are bounded.

4.3. Controller Design

In order to improve the tracking accuracy, a barrier Lyapunov function is used on the basis of the traditional backstepping control method. The proposed controller design is divided into two steps.

Step 1: Define the variables $z_1 = x_1 - x_d$ and $z_2 = \hat{x}_2 - \alpha_1$. Select the barrier Lyapunov function as follows:

$$V_1 = \frac{1}{2} \ln \frac{z_1^2}{b_1^2 - z_1^2} \tag{68}$$

where b_1 is a designed positive constant for the boundary.

The derivation of (68) yields the following:

$$\dot{V}_1 = \frac{z_1 \dot{z}_1}{b_1^2 - z_1^2} = \frac{z_1}{b_1^2 - z_1^2}(z_2 + \alpha_1 + \tilde{x}_2 - \dot{x}_d) \tag{69}$$

The virtual control law α_1 is designed as follows:

$$\alpha_1 = -k_{d1} z_1 + \dot{x}_d - \frac{z_1}{b_1^2 - z_1^2} - \frac{5}{2}(b_1^2 - z_1^2) z_1 \tag{70}$$

Substituting (70) into (69) yields the following:

$$\begin{aligned} \dot{V}_1 =& \frac{z_1}{b_1^2 - z_1^2}(z_2 - k_{d1} z_1 + \dot{x}_d - \frac{z_1}{b_1^2 - z_1^2} - \frac{5}{2}(b_1^2 - z_1^2) \\ & - \dot{x}_d + \tilde{x}_2) \\ =& \frac{z_1 z_2}{b_1^2 - z_1^2} - \frac{k_{d1} z_1^2}{b_1^2 - z_1^2} - \frac{z_1^2}{(b_1^2 - z_1^2)^2} - \frac{5}{2} z_1^2 + \frac{z_1 \tilde{x}_2}{b_1^2 - z_1^2} \end{aligned} \tag{71}$$

Using Lemma 1 yields the following:

$$\begin{cases} \dfrac{z_1 z_2}{b_1^2 - z_1^2} \leq \dfrac{1}{2} \dfrac{z_1^2}{(b_1^2 - z_1^2)^2} + \dfrac{1}{2} z_2^2 \\ \dfrac{z_1 \tilde{x}_2}{b_1^2 - z_1^2} \leq \dfrac{1}{2} \dfrac{z_1^2}{(b_1^2 - z_1^2)^2} + \dfrac{1}{2} \tilde{x}_2^2 \end{cases} \tag{72}$$

Substituting (72) into (71) yields the following:

$$\dot{V}_1 \leq -\frac{k_{d1} z_1^2}{b_1^2 - z_1^2} + \frac{1}{2} z_2^2 - \frac{5}{2} z_1^2 + \frac{1}{2} \tilde{x}_2^2 \tag{73}$$

Step 2: Select the barrier Lyapunov function as follows:

$$V_2 = \frac{1}{2} \ln \frac{z_2^2}{b_2^2 - z_2^2} + \frac{1}{2 r_0} \tilde{\mathbf{W}}_0^T \tilde{\mathbf{W}}_0 + \frac{1}{2 r_1} \tilde{\mathbf{W}}_1^T \tilde{\mathbf{W}}_1 \tag{74}$$

where b_2 is a designed positive constant for the boundary. r_0 and r_1 are positive constants.

The derivation of (74) yields the following:

$$\begin{aligned} \dot{V}_2 =& \frac{z_2 \dot{z}_2}{b_2^2 - z_2^2} - \frac{1}{r_0} \tilde{\mathbf{W}}_0^T \dot{\mathbf{W}}_0 - \frac{1}{r_0} \tilde{\mathbf{W}}_1^T \dot{\mathbf{W}}_1 \\ =& \frac{z_2}{b_2^2 - z_2^2}(i_e \tau_m - \hat{\tau}_{ef} - \hat{\tau}_D + k_3 \tilde{x}_2 + (k_2 - k_1 k_3) \tilde{x}_1 \\ & - \dot{\alpha}_1) - \frac{1}{r_0} \tilde{\mathbf{W}}_0^T \dot{\mathbf{W}}_0 - \frac{1}{r_1} \tilde{\mathbf{W}}_1^T \dot{\mathbf{W}}_1 \end{aligned} \tag{75}$$

Define $F = \dot{\alpha}_1$; according to Lemma 2, F can be approximated via RBFNN as follows:

$$F = \mathbf{W}_1^{*T}\mathbf{G}_1(x_1, \hat{x}_2) + \varepsilon_1^* \tag{76}$$

Define the estimate of F as follows:

$$\hat{F} = \hat{\mathbf{W}}_1\mathbf{G}_1(x_1, \hat{x}_2) \tag{77}$$

The control input u is designed as follows:

$$\begin{aligned}\tau_m =& \frac{1}{i_e}[\hat{\tau}_{ef} + \hat{\tau}_D - (k_2 - k_1k_3)\tilde{x}_1 - k_{d2}z_2 - \frac{k_3^2 z_2}{b_2^2 - z_2^2} \\ & - \frac{3z_2}{2(b_2^2 - z_2^2)} - z_1 - \frac{5}{2}(b_2^2 - z_2^2)z_2 + \hat{F}]\end{aligned} \tag{78}$$

The neural network adaptive law is designed as follows:

$$\begin{cases}\dot{\hat{\mathbf{W}}}_0 = r_0(z_1 + z_2)\mathbf{G}_0 - m_0\hat{\mathbf{W}}_0 \\ \dot{\hat{\mathbf{W}}}_1 = r_1(z_1 + z_2)\mathbf{G}_1 - m_1\hat{\mathbf{W}}_1\end{cases} \tag{79}$$

where $r_i, m_i (i = 1, 2)$ is the designed positive constants.

Substituting (77)–(79) into (75) yields the following:

$$\begin{aligned}\dot{V}_2 =& \frac{k_3 z_2 \tilde{x}_2}{b_2^2 - z_2^2} - \frac{z_2}{b_2^2 - z_2^2}\tilde{\mathbf{W}}_1^T\mathbf{G}_1 + \frac{z_2}{b_2^2 - z_2^2}\varepsilon_1^* - \frac{k_{d2}z_2^2}{b_2^2 - z_2^2} \\ & - \frac{3z_2^2}{2(b_2^2 - z_2^2)^2} - \frac{k_3^2 z_2^2}{2(b_2^2 - z_2^2)^2} - \frac{z_1 z_2}{b_2^2 - z_2^2} - \frac{5}{2}z_2^2 \\ & - \tilde{\mathbf{W}}_0^T(z_1 + z_2)\mathbf{G}_0 + \frac{m_0}{r_0}\hat{\mathbf{W}}_0 - \tilde{\mathbf{W}}_1^T(z_1 + z_2)\mathbf{G}_1 + \frac{m_1}{r_1}\hat{\mathbf{W}}_1\end{aligned} \tag{80}$$

Using Lemma 1 yields the following:

$$\begin{cases}\dfrac{k_3 z_2 \tilde{x}_2}{b_2^2 - z_2^2} \le \dfrac{k_3^2 z_2^2}{2(b_2^2 - z_2^2)^2} + \dfrac{1}{2}\tilde{x}_2^2 \\ \dfrac{z_2}{b_2^2 - z_2^2}\tilde{\mathbf{W}}_1\mathbf{G}_1 \le \dfrac{z_2^2}{2(b_2^2 - z_2^2)^2} + \dfrac{1}{2}\tilde{\mathbf{W}}_1^T\tilde{\mathbf{W}}_1 \\ \dfrac{z_2\varepsilon_1^*}{b_2^2 - z_2^2} \le \dfrac{z_2^2}{2(b_2^2 - z_2^2)^2} + \dfrac{1}{2}\varepsilon_1^{*2} \\ \dfrac{z_1 z_2}{b_2^2 - z_2^2} \le \dfrac{z_2^2}{2(b_2^2 - z_2^2)^2} + \dfrac{1}{2}z_1^2 \\ -\tilde{\mathbf{W}}_0^T(z_1 + z_2)\mathbf{G}_0 + \dfrac{m_0}{r_0}\hat{\mathbf{W}}_0 \le z_1^2 + z_2^2 + \dfrac{1}{2}\tilde{\mathbf{W}}_0^T\tilde{\mathbf{W}}_0 \\ -\dfrac{m_0}{2r_0}\tilde{\mathbf{W}}_0^T\tilde{\mathbf{W}}_0 + \dfrac{m_0}{2r_0}\mathbf{W}_0^{*T}\mathbf{W}_0^* \\ -\tilde{\mathbf{W}}_1^T(z_1 + z_2)\mathbf{G}_1 + \dfrac{m_1}{r_1}\hat{\mathbf{W}}_1 \le z_1^2 + z_2^2 + \dfrac{1}{2}\tilde{\mathbf{W}}_1^T\tilde{\mathbf{W}}_1 \\ -\dfrac{m_1}{2r_1}\tilde{\mathbf{W}}_1^T\tilde{\mathbf{W}}_1 + \dfrac{m_1}{2r_1}\mathbf{W}_1^{*T}\mathbf{W}_1^*\end{cases} \tag{81}$$

The control principles and procedures of the constraint-based adaptive neural network-output feedback controller proposed in this paper is shown in Figure 14.

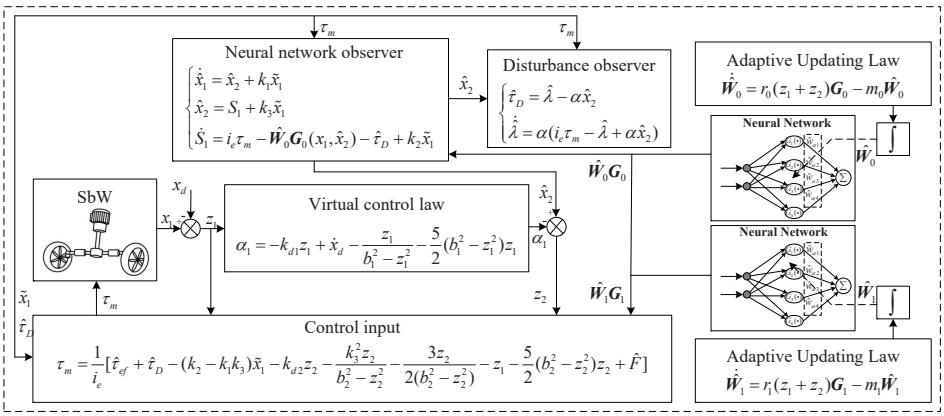


Figure 14. The control principles and procedures of the method proposed in this paper.

Substituting (81) into (80) yields the following:

$$\begin{aligned}\dot{V}_2 \leq & \frac{1}{2}\tilde{x}_2^2 - \frac{k_{d2}z_2^2}{b_2^2 - z_2^2} - (\frac{m_0}{2r_0} - \frac{1}{2})\tilde{\mathbf{W}}_0^T\tilde{\mathbf{W}}_0 - (\frac{m_1}{2r_1} - \frac{1}{2})\tilde{\mathbf{W}}_1^T\tilde{\mathbf{W}}_1 \\ & + \frac{1}{2}\varepsilon_1^* - \frac{1}{2}z_2^2 + \frac{5}{2}z_1^2 + \frac{m_0}{2r_0}\mathbf{W}_0^{*T}\mathbf{W}_0^* + \frac{m_1}{2r_1}\mathbf{W}_1^{*T}\mathbf{W}_1^*\end{aligned} \tag{82}$$

4.4. Stability Analysis

Theorem 1. *For the SbW control system (50), the neural network state observer (57) and disturbance observer (60), the virtual controller (70), and the controller (78) are used. The adaptive law of the neural network is defined in (79); then, all signals of the closed-loop system converge to a compact set.*

Proof. Select the Lyapunov function as follows:

$$V = V_0 + V_1 + V_2 \tag{83}$$

Substituting (67), (73), and (82) into (83) yields the following:

$$\begin{aligned}\dot{V} \leq & -(k_1 - \frac{\mu_0(k_1k_3 - k_2 + 1)}{2})\tilde{x}_1^2 - (k_3 - \frac{k_1k_3 - k_2 + 1}{2\mu_0} \\ & + \frac{\mu_8}{2} - \frac{\mu_1}{2} - \frac{\mu_2}{2} - \frac{\alpha^2}{\mu_2} - \frac{1}{2\mu_5}\alpha^4 - \frac{1}{2\mu_6}\alpha^2h_f^2 - 1)\tilde{x}_2^2 \\ & - (\alpha - \frac{1}{\mu_2} - \frac{\mu_3}{2} - \frac{\mu_4}{2} - \frac{\mu_5}{2} - \frac{\mu_6}{2} - \frac{\mu_7}{2})\tilde{\lambda}^2 \\ & - (\frac{m_0}{2r_0} - \frac{1}{2} - \frac{1}{2\mu_1})\tilde{\mathbf{W}}_0^T\tilde{\mathbf{W}}_0 - (\frac{m_1}{2r_1} - \frac{1}{2})\tilde{\mathbf{W}}_1^T\tilde{\mathbf{W}}_1 \\ & - k_{d1}\frac{z_1^2}{b_1^2 - z_1^2} - k_{d2}\frac{z_2^2}{b_2^2 - z_2^2} + \frac{1}{2\mu_3}\varepsilon_D^2 + \frac{1}{2\mu_4}\mathbf{W}_0^{*T}\mathbf{W}_0^* \\ & + (\frac{\alpha^2}{2\mu_7} + \frac{1}{2\mu_4})\varepsilon_0^{*2} + \frac{1}{2}\varepsilon_1^* + \frac{m_0}{2r_0}\mathbf{W}_0^{*T}\mathbf{W}_0^* + \frac{m_1}{2r_1}\mathbf{W}_1^{*T}\mathbf{W}_1^*\end{aligned} \tag{84}$$

where $C_1 = 2(k_1 - \frac{\mu_0(k_1k_3-k_2+1)}{2})$, $C_2 = 2(k_3 - \frac{k_1k_3-k_2+1}{2\mu_0} + \frac{\mu_8}{2} - \frac{\mu_1}{2} - \frac{\mu_2}{2} - \frac{\alpha^2}{\mu_2} - \frac{1}{2\mu_5}\alpha^4 - \frac{1}{2\mu_6}\alpha^2h_f^2 - 1)$, $C_3 = 2(\alpha - \frac{1}{\mu_2} - \frac{\mu_3}{2} - \frac{\mu_4}{2} - \frac{\mu_5}{2} - \frac{\mu_6}{2} - \frac{\mu_7}{2})$, $C_4 = 2k_{d1}$, $C_5 = 2k_{d2}$, $C_6 = 2(\frac{m_0}{2r_0} - \frac{1}{2} - \frac{1}{2\mu_1})$, $C_7 = 2(\frac{m_1}{2r_1} - \frac{1}{2})$, $C_0 = +\frac{1}{2\mu_3}\varepsilon_D^2 + \frac{1}{2\mu_4}\mathbf{W}_0^{*T}\mathbf{W}_0^* + (\frac{\alpha^2}{2\mu_7} + \frac{1}{2\mu_4})\varepsilon_0^{*2} + \frac{1}{2}\varepsilon_1^* + \frac{m_0}{2r_0}\mathbf{W}_0^{*T}\mathbf{W}_0^* + \frac{m_1}{2r_1}\mathbf{W}_1^{*T}\mathbf{W}_1^*$.

Define the compact set $\Omega_z = \{z_i||z_i| \leq b_i, i = 1,2\}$; then, on Ω_z, the following is obtained: $\ln \frac{z_i^2}{b_i^2 - z_i^2} \leq \frac{z_i^2}{b_i^2 - z_i^2}$. Therefore, (84) can be expressed as follows:

$$\dot{V} \leq -aV + C_0 \tag{85}$$

where $a = \min\{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$.

According to A, $\tilde{x}_1$, $\tilde{x}_2$, λ, z_1, z_2, $\tilde{\mathbf{W}}_0$, and $\tilde{\mathbf{W}}_1$are bounded. Since α is bounded, it follows that $\hat{x}_2 = z_2 + \alpha$ is bounded, and $x_2 = \hat{x}_2 + \tilde{x}_2$ is bounded. It can be inferred from (70) and (78) that the virtual control law and control input are also bounded. Theorem 1 has been proven. □

5. Simulation and Experimental Validation

In this section, to verify the effectiveness of the proposed controller in the online control SbW system, we conducted simulations and hardware loop experiments.

5.1. Simulation

The parameters of the SbW system are chosen to be the same with [22], which are listed in Table 5. In this numerical simulation, the parameters are set as follows: $\alpha_1 = 0.25$, $\alpha_2 = 20$, $\alpha_3 = 0.01$, $\beta_1 = 100$, $\beta_2 = 1$, and $\beta_3 = 100$. To verify the robustness under different loads, different external disturbance are added as follows:

$$\tau_D = \begin{cases} 2, 0 < t \leq 40 \\ 1 + 0.5tanh(0.1(t-40)), 40 < t \leq 80 \\ 2 + 0.3tanh(0.1(t-80)), 80 < t \leq 120 \\ 3 - 120tanh(0.1(t-120)), 120 < t \leq 160 \end{cases} \tag{86}$$

To verify the superiority of the proposed method, three methods are selected for comparison. In order to ensure a fair comparison of the three methods, the parameters of each method are adjusted to achieve the best control effect under the same conditions.

Table 5. Parameters for SbW system.

Parameters	Value
J_{fw} (kg· m^2)	3.8
J_{sm} (kg· m^2)	0.0045
B_{fw} (Nms/rad)	10
B_{sm} (Nms/rad)	0.05
i_{mc}	18

(1) The controller proposed in Theorem 1 is expressed as follows:

$$\begin{aligned} \tau_m = & \frac{1}{i_e}[\hat{\tau}_{ef} + \hat{\tau}_D - (k_2 - k_1 k_3)\tilde{x}_1 - k_{d2} z_2 - \frac{k_3^2 z_2}{b_2^2 - z_2^2} \\ & - \frac{3z_2}{2(b_2^2 - z_2^2)} - z_1 - \frac{5}{2}(b_2^2 - z_2^2)z_2 + \hat{F}] \end{aligned} \tag{87}$$

The parameters of the proposed method are set as follows: $k_1 = 0.001$, $k_2 = 150$, $k_3 = 50$, $r_0 = 100$, $m_0 = 20$, $r_1 = 100$, $m_1 = 15$, $k_{d1} = k_{d2} = 10$, $b_1 = b_2 = 5$, $\alpha = 1$. The neural network parameters are set as follows: $b_j = 5$, $c_1 = [0,0]^T$, $c_2 = [0.05, 0.05]^T$, $c_3 = [0.1, 0.1]^T$, $c_4 = [0.15, 0.15]^T$, $c_5 = [0.2, 0.2]^T$, $c_6 = [0.25, 0.25]^T$, $c_7 = [0.3, 0.3]^T$, $c_8 = [0.35, 0.35]^T$, $c_9 = [0.4, 0.4]^T$, $c_{10} = [0.45, 0.45]^T$, $c_{11} = [0.5, 0.5]^T$, $c_{12} = [0.55, 0.55]^T$, $c_{13} = [0.6, 0.6]^T$, $c_{14} = [0.65, 0.65]^T$, $c_{15} = [0.7, 0.7]^T$, $c_{16} = [0.75, 0.75]^T$, $c_{17} = [0.8, 0.8]^T$, $c_{18} = [0.85, 0.85]^T$, $c_{19} = [0.9, 0.9]^T$, $c_{20} = [0.95, 0.95]^T$, $c_{21} = [1,1]^T$.

(2) To verify the effectiveness of the disturbance observer, this method removes the disturbance observer from the proposed method, and the controller parameters remain consistent with the proposed method.

(3) The traditional backstepping control method (BSC) differs from the proposed method by removing the state constraints and keeping the other parts the same. The controller is expressed as follows:

$$\begin{cases} \alpha_1 = -k_{d1} z_1 + \dot{x}_d \\ \tau_m = \dfrac{1}{i_e}(\hat{\tau}_{ef} + \hat{\tau}_D + \dot{\alpha}_1 - (k_2 - k_1 k_3)\tilde{x}_1 - k_{d2} z_2 - z_1) \end{cases} \tag{88}$$

The parameters of BSC are set as follows: $k_{d1} = k_{d2} = 70$. The parameters of the state observer and the disturbance observer are consistent with the proposed method. The simulation reference signal is set to $x_d = \sin(t)$. In order to evaluate the performance of the three controllers, a performance index is introduced: the root mean square error (RMSE), RMSE=$\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$. The simulation results are shown in Figure 15.

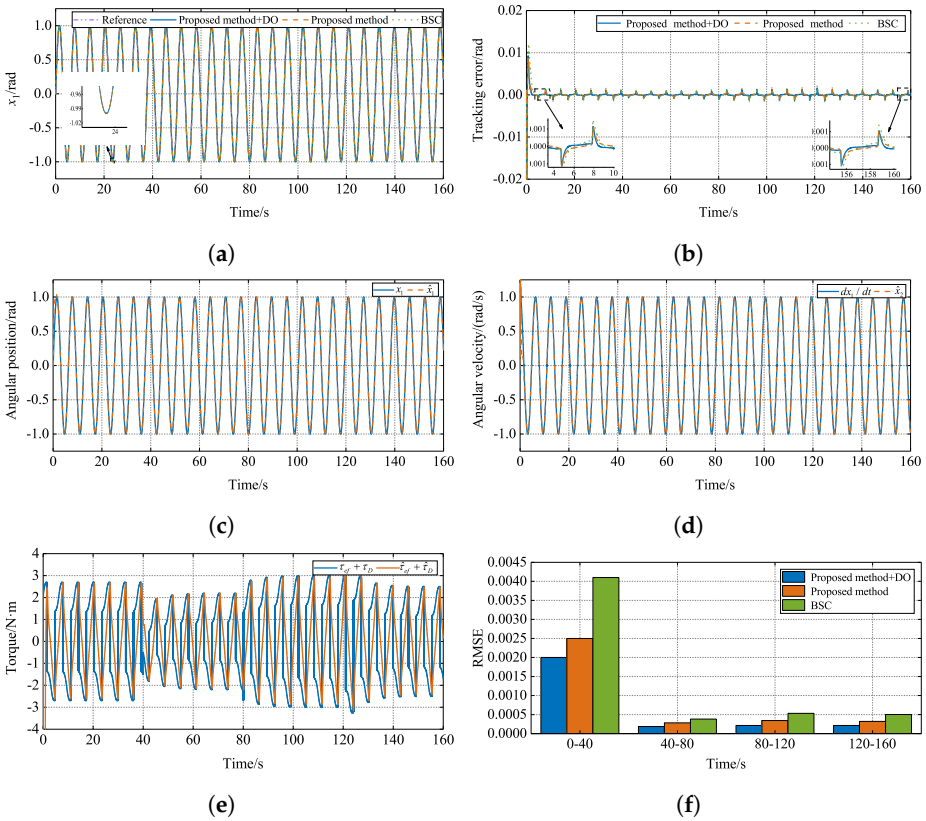


Figure 15. The simulation results. (**a**) Position-tracking performance; (**b**) tracking error; (**c**) angular position; (**d**) angular velocity; (**e**) friction and torque; (**f**) RMSE.

Figure 15 shows the curves of the numerical simulation. Figure 15a,b show the position-tracking performance and tracking error of the three methods, respectively. From Figure 15a, it can be seen that all three methods can track the reference signal well, the errors are very close to each other, and it is difficult to see the difference between the three methods intuitively. However, the tracking error curve shows that the tracking error of BSC is more than 0.01 rad in the starting tracking stage, while the other two methods are within 0.01 rad, which indicates that the proposed method achieves a faster response speed. During the tracking process, the tracking error of BSC is larger than that of the other two methods. The tracking error of the proposed method with a disturbance observer is smaller, and the tracking error converges near zero faster. Figure 15c,d show the curves of the estimation

results of the state observer for x_1 and x_2, respectively. As can be seen from Figure 15c, at the beginning stage, the curves of $\hat{x}_1$ and x_1 have a slightly larger error and do not fit perfectly. But soon after that, the two curves almost coincide, and the state observer can estimate the angular position well. As can be seen from Figure 15d, since the simulation is an ideal environment to derive the angular velocity directly for the angular position without noise, a smooth curve is obtained. The curves of $\hat{x}_2$ and dx_1/dt almost coincide, and the state observer can estimate the angular velocity well. Figure 15e shows the curve of the neural network approximation of friction and the disturbance observer estimation of external disturbance; from the simulation results, it can be seen that the neural network and the disturbance observer are quickly adjusted by updating the adaptive law to accurately estimate the friction and the external disturbance when the load changes. Figure 15f shows RSME histograms of the three methods, from which it can be seen that, in each period, the proposed method with the disturbance observer achieved lower values in RMSE, indicating better tracking results.

5.2. Experiment

In order to better validate the effectiveness of the proposed method, and for safety reasons, hardware-in-the-loop (HIL) experiments were conducted to verify the effectiveness of the proposed control method in practice. The schematic diagram of the steering control YOLOv5-based end-to-end for an autonomous vehicle is shown in Figure 16. The SbW system mainly consists of a steering test bench, dSPACE1401, a monocular camera, and a computer. The steering test bench mainly consists of a steering motor, a reducer, a servo controller, a tension controller, a magnetic powder brake, a steering gear, a cable sensor, and a table frame. In order to verify the proposed method's effectiveness and superiority in comparison with two other methods, firstly, the steering-angle signal received via the controller is artificially given, which causes the steering wheel to traverse the categorized steering angles sequentially, starting from the original position, which ranges between $-60°$ and $60°$.

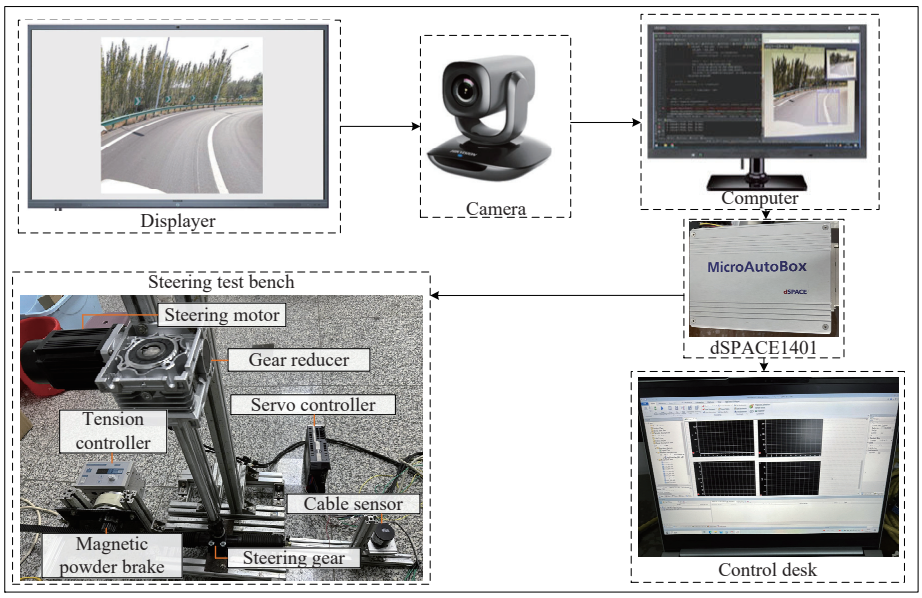


Figure 16. The schematic diagram of the YOLOv5-based end-to-end steering control for an autonomous vehicle.

The parameters of the methods in the hardware-in-the-loop experiments are different from those in the simulation, and the parameters of the three methods are set as follows:

(1) The parameters of the proposed method are set to $k_1 = 0.001$, $k_2 = 180$, $k_3 = 100$, $k_{d1} = 70$, $k_{d2} = 1$, $b_1 = 0.2$, $b_2 = 0.3$, and $\alpha = 0.7$. The parameters of the neural network are consistent with those in the simulation.

(2) The parameters of BSC are set to $k_{d1} = 85$, $k_{d2} = 2$. The other parameters are consistent with the proposed method.

The experimental results are shown in Figure 17. Figure 17a,b show the position-tracking performance and tracking error, respectively. From Figure 17a, it can be seen that the tracking performance curve is not as smooth as in the simulation due to the noise in the angular position of the sensor measurement in the experiment. Compared to the other two methods, the proposed method with the disturbance observer exhibits a faster response and a smaller steady-state error. This is more intuitive in the tracking error curve. As Figure 17b shows, the maximum tracking error of the BSC is 0.055 rad, and the maximum tracking errors of the other two methods are within 0.055 rad. Within 50 s to 120 s, it is clearly seen that the proposed method with perturbation observer has a smaller tracking error. In the final stage of tracking, the proposed method with the perturbed observer has a smaller steady-state error, and the tracking error is closer to zero. Figure 17c,d show the estimated curves for the angular position and angular velocity, respectively. From Figure 17c, it can be seen that the $\hat{x}_1$ curve derived from the state observer is very close to that of x_1. Between 0 and 50 s, due to the initial operation of the SbW system, there is significant resistance between various components, such as the gap between the gear and the rack, and the gap between the steering motor and the gear reducer. After running for a period of time, the system operates in normal working condition. This can also be seen in Figure 17e, where the estimation of friction and external disturbances from 0 to 50 s is larger than after 50 s. Therefore, the estimation of the angle position has a certain impact, and the error from 0 to 50 s is larger than that after 50 s. From Figure 17d, it can be seen that the derivation of x_1 leads to an amplification in the noise due to the presence of noise in the sensor measurements of the experiment. The $\hat{x}_2$ derived from the state observer, on the other hand, is much closer to the real value of x_2, has much less noise, and meets the requirements of the controller design. Figure 17e shows the neural network approximation friction and the external disturbance curve estimated via the disturbance observer. Figure 17f shows the RMSE curves of the three methods; at each stage, the proposed method has smaller values, indicating that the proposed method with the disturbance observer achieves better tracking performance.

From the above simulation and experiment, it can be seen that, although the experimental conditions added in the simulation are different from the environment in the actual experiment, it is difficult to achieve consistency with the simulation in the actual experimental environment; for instance, noise is present in the sensors, the actuator has a response delay, added friction occurs, and the external perturbation is inconsistent, etc. However, the trend in the results obtained from the numerical simulation and HIL experiment is consistent, indicating the superiority of the method proposed in this paper.

For safety reasons, we verified the real-time steering prediction and steering control of the trained YOLOv5 on the SbW experimental bench. The experimental results are shown in Figure 18. Figure 18a shows the randomly selected real-time validation images; the trained YOLOv5 outputs the predicted steering angle based on the image and sends it to follow the controller via a serial protocol. Figure 18b shows the output predicted steering angle. When the predicted angle changes, the reference signal is a step-signal change. While, in practice, there are delays in the communication between the steering motor and the controller, it is difficult to track the step signal, and the actual tracking performance is fully and smoothly shifted. Therefore, a filter is added after the predicted steering angle of the output to obtain a smoothed reference signal, as shown in Figure 18c. The adaptive neural network output feedback controller proposed in this paper is used to track the reference signal derived earlier, as shown in Figure 18d, from which it can be seen that, in real-time turn angle variation, the method proposed in this paper can still track the reference angle well and achieves good tracking performance. Figure 18e,f show the estimation curves of the state observer, from which it can be seen that $\hat{x}_1$ and $\hat{x}_2$ derived from the state observer achieve good accuracy when the reference turning angle varies randomly, which is consistent with the previous simulation and experiment. From the

experimental results in Figure 18, it can be seen that the trained detector and the proposed controller achieve better prediction and control capabilities.

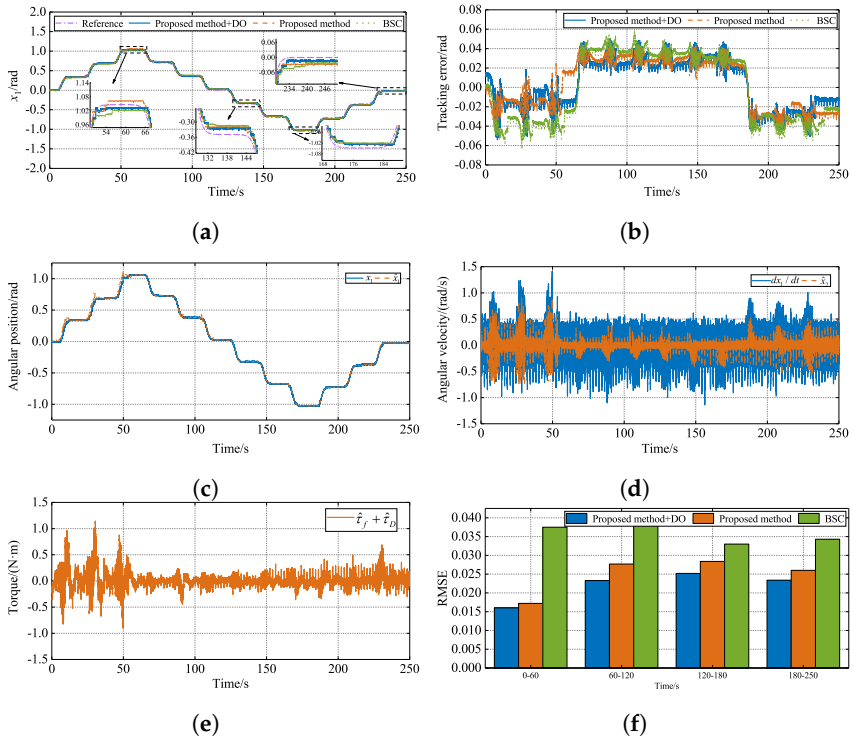


Figure 17. The experimental results. (**a**) Position-tracking performance; (**b**) tracking error; (**c**) angular position; (**d**) angular velocity; (**e**) friction and torque; (**f**) RMSE.

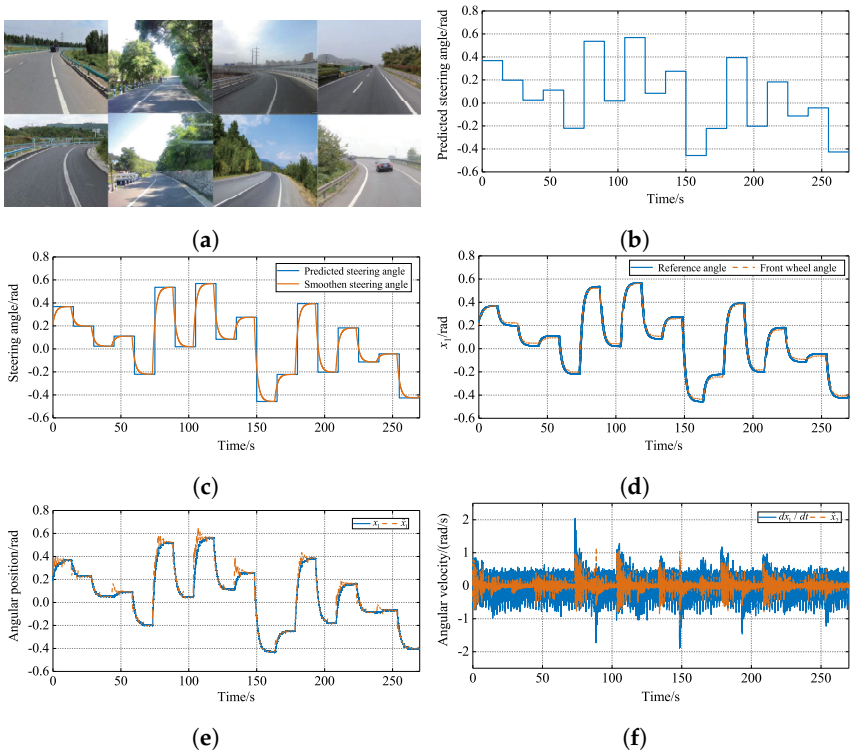


Figure 18. The real-time prediction and execution of the steering angle. (**a**) Randomly selected images; (**b**) predicted angle; (**c**) reference angle; (**d**) position tracking performance; (**e**) angular position; (**f**) angular velocity.

6. Conclusions

In this paper, a lightweight steering-angle prediction network model based on YOLOv5 and an adaptive output feedback control scheme with output constraints based on neural networks has been introduced to regulate the predicted steering angle of YOLOv5Ms effectively. We used YOLOv5Ms as a detector to solve the challenging task of steering-angle prediction in this paper. Meanwhile, an adaptive output feedback control scheme with output constraints based on neural networks has been proposed to ensure that the steering system responds quickly and accurately to the desired steering-angle signal. To train the YOLOv5Ms detector and enhance the generalization capability of the proposed detection model in this study, we conducted an extended data-collection experiment at Western Xia Park in Yinchuan City, building upon our previously created lane-line data set. The images of the data sets were labeled one by one by comparing the steering angle collected in the videos. The accuracy and response speed of the detector can meet the actual requirements after real-time testing. Furthermore, the proposed controller can improve the convergence of the tracking error and eliminate the influence of disturbance. Meanwhile, the proposed controller enhances the steering control accuracy of the steering angle predicted via YOLOv5Ms. The experimental results show that the trained network and the proposed controller perform well. The research in this paper can be further extended to other aspects. Firstly, due to the limited experimental conditions, the prediction and tracking of the angle are only experimented in the SbW experimental bench, and they will be experimentally verified in the actual car in the future. In the controller design, only angle tracking was considered, and the angle tracking of the SbW system will be combined with the trajectory tracking control of the AV in the future.

Author Contributions: Conceptualization, C.Y. and Y.L.; methodology, C.Y. and Y.W. (Yunlong Wang); software, C.Y. and Y.L.; validation, C.Y., Y.L., and Y.W. (Yunlong Wang); formal analysis, C.Y.; investigation, Y.W. (Yongfu Wang); resources, Y.L. and Y.W. (Yongfu Wang); data curation, C.Y. and Y.W. (Yongfu Wang); writing—original draft preparation, C.Y.; writing—review and editing, C.Y. and Y.L.; visualization, C.Y. and Y.L.; supervision, Y.L.; project administration, Y.L.; funding acquisition, C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of the Ningxia Hui Autonomous Region (2022AAC03338), supported by the National Natural Science Foundation of China under Grant 52405535, supported by the Postdoctoral Science Foundation of Northeastern University under Grant 20240103, supported by the Fundamental Research Funds for the Central Universities of China under Grant N2303026, supported by the Natural Science Foundation of Liaoning Province under Grant 2023-BSBA-101.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

AVs	Autonomous vehicles	GIoU	Generalized intersection
SbW	Steer by wire	CNN	Convolutional neural network
WHO	World Health Organization	DL	Deeping learning
MT-LfD	Multi-task demonstration learning	SE	Squeeze and excitation
HS	h-swish	ReLU	Rectified linear unit
FCs	Fully connected layers	MSE	Mean square error
BBOX	Bounding box	IoU	Intersection over union
DIoU	Distance-IoU	CIoU	Comprehensive IoU

EIoU	Efficient version of IoU	AP	Average precision
TP	True positive	FP	False positive
FN	False negative	AMD	Advanced Micro Devices
CAN	Controller area network	RBFNN	Radial basis function neural network
BSC	Backstepping control method	HIL	Hardware-in-the-loop
NAS	Network architecture	RCNN	Region with CNN
SSD	Single Shot MultiBOX Detector	YOLO	You Only Look Once
YOLOv5Ms	You Only Look Once version 5 with MobilNet version 3		

Appendix A

The algorithm followed in this study is presented in Algorithm A1.

Algorithm A1 Steering Angle prediction and control

Require: Input image sequences
Ensure: Controlled motor position
1. nput_size = (640, 640, 3), nc= 15, anchors = [(10, 13), (16, 30), (33, 23), (30, 61), (62, 45), (59, 119), (116, 90),
(156, 198), (373, 326)], numanchors = len(anchors)
2. Input picture: Image = read_image(path)
3. Image preprocess: Processed_image = preprocess_image(Image)
4. Loading model: Model = load_model('YOLOv5Ms')
5. Image reasoning: Outputs = Model(Processed_image)
6. Analytical prediction result: Predictions = Parse_predictions(Outputs)
7. Filter prediction result: Filtered_predictions = Filter_predictions(Predictions)
8. Output the final prediction result to controller: Final prediction result=Ym
9. Estimate angular velocity, model uncertainty, and external disturbance: $\hat{x}_2$, $\hat{\tau}_{ef}$, $\hat{\tau}_D$.
10. Adaptive neural network output feedback controller provides output τ_m, and controls motor position
based on this controller output.
11. Adjust motor position.
12. **return** Controlled motor position

References

1. Elallid, B.B.; Benamar, N.; Hafid, A.S.; Rachidi, T.; Mrani, N. A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 7366–7390. [CrossRef]
2. Fagnant, D.J.; Kockelman, K. Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Transp. Res. Part A Policy Pract.* **2015**, *77*, 167–181. [CrossRef]
3. Gidado, U.M.; Chiroma, H.; Aljojo, N.; Abubakar, S.; Popoola, S.; Mohammed, A. A survey on deep learning for steering angle prediction in autonomous vehicles. *IEEE Access* **2020**, *8*, 163797–163817. [CrossRef]
4. Peng, B.; Sun, Q.; Li, S.E.; Kum, D.; Yin, Y.; Wei, J.; Gu, T. End-to-end autonomous driving through dueling double deep Q-network. *Automot. Innov.* **2021**, *4*, 328–337. [CrossRef]
5. Ma, Y.; Wang, Z.; Yan, G.; Yang, L. Artificial intelligence applications in the development of autonomous vehicles: A survey. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 315–329. [CrossRef]
6. Tampuu, A.; Matiisen, T.; Semikin, M.; Fishman, D.; Muhammd, N. A Survey of End-to-End Driving: Architectures and Training Methods. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 1364–1384. [CrossRef]
7. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* **2020**, *8*, 58443–58469. [CrossRef]
8. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision* **2015**, *115*, 211–252. [CrossRef]
9. Polack, P.; Altch, F.; Andr, B.; Fortelle, A. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 812–818.
10. Mohammed, M.S.; Abduljabar, A.M.; Faisal, M.M.; Mahmmod, B.M.; Abdulhussain, S.H.; Khan, W.; Liatsis, P.; Hussain, A. Low-cost autonomous car level 2: Design and implementation for conventional vehicles. *Results Eng.* **2023**, *17*, 100969. [CrossRef]
11. Pomerleau, D.A. Alvinn: An autonomous land vehicle in a neural network. *Adv. Neural Inf. Process. Syst.* **1988**, *1*, 305–313

12. Muller, U.; Ben, J.; Cosatto, E.; Flepp, B.; Cun, Y. Off-road obstacle avoidance through end-to-end learning. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2005; Volume 18.
13. Bojarski, M.; Del, T.D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D. End to end learning for self-driving cars. *arXiv* **2016**, arXiv:1604.07316.
14. Mohseni, F.; Voronov, S.; Frisk, E. Deep learning model predictive control for autonomous driving in unknown environments. *IFAC-PapersOnLine* **2018**, *51*, 447–452. [CrossRef]
15. Mehta, A.; Subramanian, A. Learning end-to-end autonomous driving using guided auxiliary supervision. In Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing, Hyderabad, India, 18–22 December 2018; pp. 1–8.
16. Zeng, W.; Luo, W.; Suo, S.; Sadat, A.; Yang, B.; Casas, S.; Urtasun, R. End-to-end interpretable neural motion planner. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8660–8669.
17. Le M.; Yi, D.; Dianati, M.; Mouzakitis, A. A survey on imitation learning techniques for end-to-end autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 14128–14147.
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
19. Karadeniz, A.M.; Ballagi, Á.; Kóczy, L.T. Transfer Learning-Based Steering Angle Prediction and Control with Fuzzy Signatures-Enhanced Fuzzy Systems for Autonomous Vehicles. *Symmetry* **2024**, *16*, 1180. [CrossRef]
20. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef]
21. Bochkovskiy, A.; Wang, C.; Liao, H.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
22. Ye, C.; Wang, Y.; Wang, Y.; Tie, M. Steering angle prediction yolov5-based end-to-end adaptive neural network control for autonomous vehicles. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2022**, *236*, 1991–2011. [CrossRef]
23. Dong, X.; Yan, S.; Duan, C. A lightweight vehicles detection network model based on yolov5. *Eng. Appl. Artif. Intell.* **2022**, *113*, 104914. [CrossRef]
24. Marumo, Y.; Nagai, M. Steering control of motorcycles using steerby-wire system. *Veh. Syst. Dyn.* **2007**, *45*, 445–458. [CrossRef]
25. Setlur, P.; Wagner, J.R.; Dawson, D.M.; Braganza, D. A trajectory tracking steer-by-wire control system for ground vehicles. *IEEE Trans. Veh. Technol.* **2006**, *55*, 76–85. [CrossRef]
26. Huang, C.; Naghdy, F.; Du, H. Delta operatorbased fault estimation and fault-tolerant model predictive control for steer-by-wire systems. *IEEE Trans. Control Syst. Technol.* **2018**, *26*, 1810–1817. [CrossRef]
27. Sun, Z.; Zheng, J.; Man, Z.; Wang, H. Robust control of a vehicle steer-by-wire system using adaptive sliding mode. *IEEE Trans. Control Syst. Technol.* **2016**, *63*, 2251–2262. [CrossRef]
28. Ye, M.; Wang, H. Robust adaptive integral terminal sliding mode control for steer-by-wire systems based on extreme learning machine. *Comput. Electr. Eng.* **2020**, *86*, 106756. [CrossRef]
29. Chen, J.; Guo, Y. Design and Non-Linearity Optimization of a Vertical Brushless Electric Power Steering Angle Sensor. *Sensors* **2024**, *24*, 2469. [CrossRef]
30. Garcia, G.; Molina, C.; Luque, B.; Rafael, M.; Juan, M. Road pollution estimation from vehicle tracking in surveillance videos by deep convolutional neural networks. *Appl. Soft Comput.* **2021**, *113*, 107950. [CrossRef]
31. Yao, J.; Qi, J.; Zhang, J.; Shao, H.; Yang, J.; Li, X. A real-time detection algorithm for kiwifruit defects based on yolov5. *Electronics* **2021**, *10*, 1711. [CrossRef]
32. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Wey, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
33. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
34. Zhao, L.; Wang, L. A new lightweight network based on mobilenetv3. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 1–15.
35. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
36. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*. [CrossRef]
37. Niu, Z.; Zhong, G.; Yu, H. A review on the attention mechanism of deep learning. *Neurocomputing* **2021**, *452*, 48–62. [CrossRef]
38. Guo, M.H.; Xu, T.X.; Liu, J.J.; Liu, Z.N.; Jiang, P.T.; Mu, T.J.; Zhang, S.H.; Martin, R.R.; Cheng, M.M.; Hu, S.M. Attention mechanisms in computer vision: A survey. *Comput. Vis. Media* **2022**, *8*, 331–368. [CrossRef]
39. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
40. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
41. Yu, J.; Jiang, Y.; Wang, Z.; Cao, Z.; Huang, T. Unitbox: An advanced object detection network. In Proceedings of the 24th ACM international conference on Multimedia, Amsterdam, The Netherlands, 15–19 October 2016; pp. 516–520.

42. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 658–666.
43. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-iou loss: Faster and better learning for bounding box regression. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 12993–13000. [CrossRef]
44. Zhang, Y.F.; Ren, W.; Zhang, Z.; Jia, Z.; Wang, L.; Tan, T. Focal and efficient iou loss for accurate bounding box regression. *Neurocomputing* **2022**, *506*, 146–157. [CrossRef]
45. Gevorgyan, Z. Siou loss: More powerful learning for bounding box regression. *arXiv* **2022**, arXiv:2205.12740.
46. Zhang, H.; Zhang, S. Shape-iou: More accurate metric considering bounding box shape and scale. *arXiv* **2023**, arXiv:2312.17663.
47. Visa, S.; Ramsay, B.; Ralescu, A.L.; Van Der Knaap, E. Confusion matrix-based feature selection. *Maics* **2011**, *710*, 120–127.
48. Na, J.; Chen, Q.; Ren, X.; Guo, Y. Adaptive prescribed performance motion control of servo mechanisms with friction compensation. *IEEE Trans. Ind. Electron.* **2014**, *61*, 486–494. [CrossRef]
49. Kim, S. Moment of inertia and friction torque coefficient identification in a servo drive system. *IEEE Trans. Ind. Electron.* **2019**, *66*, 60–70. [CrossRef]
50. Liu, K.; Zhu, Z. Determination of moment of inertia of permanent magnet synchronous machine drives for design of speed loop regulator. *IEEE Trans. Ind. Electron.* **2017**, *25*, 1816–1824. [CrossRef]
51. Deng, H.; Krstic, M. Stochastic nonlinear stabilization-i: A backstepping design. *Syst. Control Lett.* **1997**, *32*, 143–150. [CrossRef]
52. Li, Y.; Qiang, S.; Zhuang, X.; Kaynak, O. Robust and adaptive backstepping control for nonlinear systems using rbf neural networks. *IEEE Trans. Neural Netw.* **2004**, *15*, 693–701. [CrossRef]
53. Chen, M.; Ge, S. Direct adaptive neural control for a class of uncertain non-affine nonlinear systems based on disturbance observer. *IEEE Trans. Cybern.* **2013**, *43*, 1213–1225. [CrossRef]
54. Peng, Z.; Wang, J.; Wang, D. Distributed containment maneuvering of multiple marine vessels via neurodynamics-based output feedback. *IEEE Trans. Ind. Electron.* **2017**, *64*, 3831–3839. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.