electronics

MDPI

Article

Accurate and Lightweight RailNet for Real-Time Rail Line Detection

Zhen Tao, Shiwei Ren, Yueting Shi, Xiaohua Wang and Weijiang Wang *

School of Information and Electronics, Beijing Institute of Technology, 5 South Zhongguancun Street, Haidian District, Beijing 100081, China; taozhen3120200703@163.com (Z.T.); renshiwei@bit.edu.cn (S.R.); shiyueting@bit.edu.cn (Y.S.); xh_wong@bit.edu.cn (X.W.)
* Correspondence: wangweijiangbit@163.com

Abstract: Railway transportation has always occupied an important position in daily life and social progress. In recent years, computer vision has made promising breakthroughs in intelligent transportation, providing new ideas for detecting rail lines. Yet the majority of rail line detection algorithms use traditional image processing to extract features, and their detection accuracy and instantaneity remain to be improved. This paper goes beyond the aforementioned limitations and proposes a rail line detection algorithm based on deep learning. First, an accurate and lightweight RailNet is designed, which takes full advantage of the powerful advanced semantic information extraction capabilities of deep convolutional neural networks to obtain high-level features of rail lines. The Segmentation Soul (SS) module is creatively added to the RailNet structure, which improves segmentation performance without any additional inference time. The Depth Wise Convolution (DWconv) is introduced in the RailNet to reduce the number of network parameters and eventually ensure real-time detection. Afterward, according to the binary segmentation maps of RailNet output, we propose the rail line fitting algorithm based on sliding window detection and apply the inverse perspective transformation. Thus the polynomial functions and curvature of the rail lines are calculated, and rail lines are identified in the original images. Furthermore, we collect a real-world rail lines dataset, named RAWRail. The proposed algorithm has been fully validated on the RAWRail dataset, running at 74 FPS, and the accuracy reaches 98.6%, which is superior to the current rail line detection algorithms and shows powerful potential in real applications.

Keywords: RailNet; rail line detection; sliding window detection; convolutional neural network

Citation: Tao, Z.; Ren, S.; Shi, Y.; Wang, X.; Wang, W. Accurate and Lightweight RailNet for Real-Time Rail Line Detection. *Electronics* **2021**, *10*, 2038. https://doi.org/10.3390/electronics10162038

Academic Editor: Stefanos Kollias

Received: 28 July 2021
Accepted: 18 August 2021
Published: 23 August 2021

1. Introduction

As an essential national infrastructure, railway transportation has received significant attention from society for its safety [1]. With the rapid development and popularization of high-speed rail technology, higher requirements are put forward for the speed and security of trains running on rail lines. In addition to the respective scheduling issues during train operation, it is also necessary to consider how to enhance the detection of road conditions during train operation [2]. With the application of railway video intelligent monitoring systems and the development of a new generation of the fully automatic driving signal system, the realization of intelligent monitoring of rail lines has become a hot topic of research [3,4], such as track obstacle recognition [5,6], rail cracks detection [7,8], road condition foreign body intrusion [9], and other issues. However, the factors that cause rail accidents are complex and changeable, such as bad weather, obsolete train tracks, malfunctions of electronic equipment, and the status of drivers.

In realizing the intelligentization and automation of railway transportation, the primary task is to predict the railway tracks in front during operation to provide trains with basic information about the environment ahead in time [10]. In this way, the train can sense the track's condition in advance, and adjust the speed in time, so as to avoid rail traffic accidents such as speeding and derailment in the curve. Simultaneously, the rail

lines area is detected in advance to prevent foreign matter intrusion, which can help frame the detection range and reduce the amount of processing. In this way, the operation safety of the train can be ensured in real-time. At present, the detection of rail lines based on computer vision is the mainstream method of railway detection. The rail line detection algorithm based on computer vision can be divided into two directions: one is based on the image processing algorithms, using image edge detection and other algorithms to search for rail lines features and curve fitting. The other is based on deep convolutional neural networks, which have powerful semantic information extraction capabilities to obtain advanced feature information such as the edge, color, and texture of the rail lines and segment the railway tracks and background face of more complex images information.

Before the rise of deep learning, rail line detection mainly used traditional image processing technology, that is, based on the difference of a specific attribute of the entity pixel in the image in the field. This type of algorithm uses the change law of the entity pixel and the surrounding environment to determine the railway lines target in the image, so as to carry out line detection. As one of the early works, Zhong Ren et al. [11] proposed a rail recognition algorithm based on prior knowledge. The critical technology of the algorithm is rail modeling and template matching. By matching, the position of the railroad tracks in the current picture is determined. Although this method has some drawbacks, such as susceptibility to environmental interference and low accuracy, it has set a precedent for rail line detection. Afterward, according to the characteristics of the rails in the monitoring images, Q Wang et al. [12] proposed a rail line identification and detection method based on the Radon transform idea and the Bresenham straight lines detection algorithm. However, the applicability of this method is not strong, and it is only suitable for straight-line sections. Based on traditional image processing, Zhao Wu et al. [13] added postprocessing methods such as segment merging, slope culling, single-frame comprehensive decision-making, segment rebuilding, and multi-frame recognition result fusion to improve the accuracy of rail recognition, but only for straight-line detection. Lei Zhang et al. [1] studied the method of extracting rail tracks from infrared images, by obtaining the target area and edge of rail tracks through image segmentation, refining the extracted target area, and finally obtaining the curve based on the shape and location of railroad tracks. However, this method still needs a lot of improvement in both detection speed and detection accuracy. The proposed curve model directly influences the accuracy and computational complexity of the rail line detection algorithm. Kaleli [14] and Badino et al. [15] suggested extracting line features based on median filter and using dynamic programming to detect lines. Still, the model is susceptible to environmental interference, and the robustness needs to be improved. Although the complex curve model can fit more different boundary curves, it has a weak anti-jamming ability and is susceptible to noise interference. Recently, Yunze Wang et al. [16] used a curvature map-based orbital recognition algorithm to identify near-distance orbits and then obtain seed points from near-distance orbits recognition results, based on local gradient information, to recognize long-distance trajectories improved seed area growth algorithm to introduce directions. The algorithm overcomes the shortcomings of the previous methods, but it needs to be improved in identifying multi-rail lines. At the same time, the accuracy and real-time of rail line detection algorithms based on traditional image processing still need further breakthroughs.

With the success of deep learning, researchers have also gradually investigated its application in dealing with rail line detection. Ziguan Wang et al. [17] were among the first to use deep understanding in railway track detection. Their model is based on Mask R-CNN, which scans the picture and produces a candidate box containing the rails, calculates the position of the box containing the tracks, creates a mask covering the rails, and finally gets the position of the rails in the picture. They obtained photos from a surveillance video of a subway company and fabricated them into a dataset for training and evaluating their system. However, the presence of speculation in the final result of their output compromises the recognition effect. Moreover, they fail to release the accuracy and detection speed of their study, which hinder further comparisons. Recently, Xiaoyong Guan et al. [5] used

ResNet101 and Feature Pyramid Networks (FPN) as the backbone network. Input pictures can generate feature maps of various sizes, forming pyramids of feature maps at different levels, making the network further enhanced in extracting features. By making railway datasets, building network models, and training network parameters, the recognition and segmentation of rail area, metro train, and signal lamps can be realized. The network can adapt to the changes in metro train operation environment. Nevertheless, although the complex network structure ensures the accuracy of detection, it hinders the real-time performance of rail line detection.

In this paper, an algorithm based on state-of-the-art deep learning convolutional neural networks is proposed to overcome the deficiencies of the aforementioned detection methods. This algorithm is mainly used in local trains and city railways. First, the RailNet is designed to preprocess images, extracting the key information and output the binary segmentation maps, which is robust to unnecessary noise. The rail lines are segmented from the background, and the feature of tracks are preserved without interference from other objects [18]. Afterward, the binary segmentation maps pass through the post-processing part of the RailNet, namely the sliding window detection algorithm. The algorithm is mainly composed of three steps: Inverse Perspective Transformation (IPT), Feature Point Extraction (FPE), and Rail Lines Curve Fitting. Moreover, the fitting results are mapped to the original images, and the rail lines are finally marked on the authentic images. An overview of the entire process of the algorithm can be seen in Figure 1.

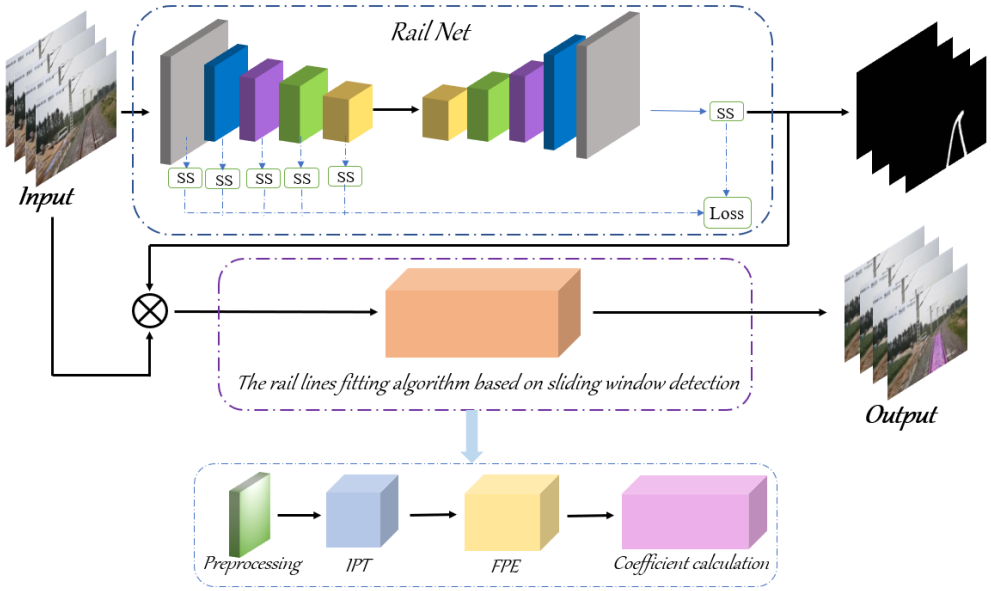


Figure 1. Overview of the proposal method.The RailNet part is responsible for extracting the rail lines features, which is trained to generate the binary segmentation maps of the rail lines. Afterward, the binary segmentation maps and the original images are processed by the rail lines fitting algorithm based on sliding window detection part. IPE and FPE respectively stand for Inverse Perspective Transformation and Feature Point Extraction of the rail lines. A second-order polynomial is fitted for each rail line, and the rail lines are reprojected onto the original images.

The main contributions of our algorithm are four-fold:

1. A novel lightweight deep learning network, RailNet, is proposed. The encoder-decoder structure of the RailNet ensures the accuracy of detection. The Depth Wise Convolution (DWconv) is introduced in the RailNet, which reduces the number of network parameters and eventually ensures real-time detection. Compared with the existing state-of-the-art methods of extracting features, the RailNet has solid detection speed and higher accuracy.
2. The Segmentation Soul (SS) module is creatively added to the RailNet structure, which can enhance the feature representation in the training phase and can be discarded in the testing phase. The SS module improves segmentation performance without any additional inference time.
3. A rail lines fitting algorithm based on sliding window detection is proposed as the post-processing part of the RailNet. The algorithm further improves the accuracy of detection. Simultaneously, the rail lines in the original image are accurately marked, and the mathematical expression and curvature of the tracks are calculated.
4. A dataset of rail lines, RAWRail, has been created for deep learning network training and testing. The dataset can be used for algorithm performance evaluation, which would help enrich the research and development of rail line detection.

2. Material and Methods

The main aim of the algorithm is to improve the accuracy and speed of rail line detection through the RailNet with its post-processing algorithm. We train the lightweight RailNet, which is realized by treating the rail line detection as a binary segmentation problem. The imbalance between the rail lines and background features can show whether the pixels belong to the tracks. Since RailNet outputs a set of pixels of railway lines, we still need to fit a curve through these pixels to improve detection accuracy. Therefore, this paper designs the rail lines fitting algorithm based on sliding window detection. It carries out postprocessing on the binary images output by the neural network and finally marks the rail lines on the original images.

2.1. RailNet

The detection of train tracks is essentially an image segmentation problem, which segments the tracks from the background and retains the characteristic information of the tracks. In this way, the network has a more vital anti-interference ability when extracting rail characteristics and can cope with changes in the number of rails. The RailNet model structure is mainly divided into two parts. Figure 2 shows the specific structure of the RailNet.

Encoder-decoder architectures are widely used in dense prediction tasks like semantic segmentation, which typically utilize convolutional layers and transpose convolution layers for feature encoding and decoding [19]. For a higher efficiency, the RailNet network adopts a light-weight encoder-decoder architecture. Table 1 shows details of the constituent layers. The encoder takes images of the front view of a rail as the input, and hierarchically extracts the features [19,20]. The decoder progressively recovers the resolution of the feature map and produce pixel-wise binary images.

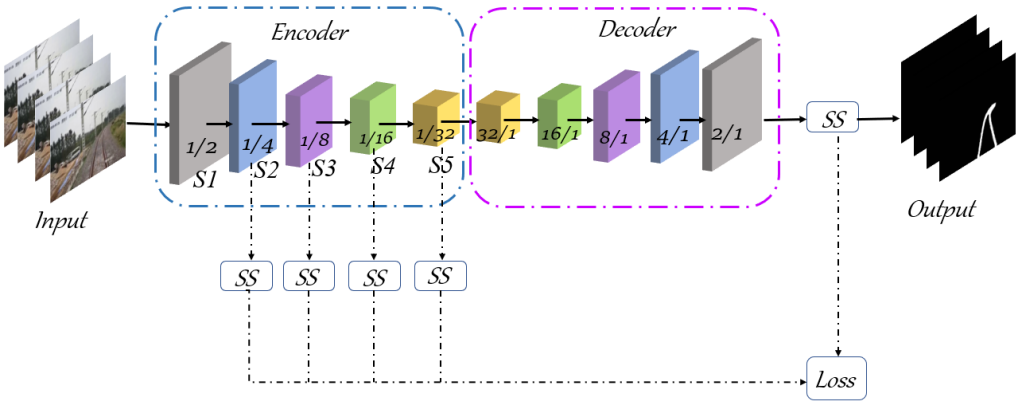


Figure 2. Overview of the RailNet architecture. From left to right, the model receives input images from a forward-looking camera and outputs binary segmentation images. The RailNet is mainly composed of two parts: the encoder and decoder. The detailed structural information of the network is shown in Table 1. SS refers to the Segmentation Soul section.

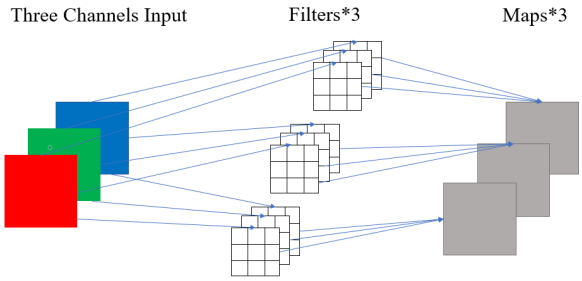
Table 1. Details of the RailNet. The structure of the decoder is relatively simple, and there is a more detailed description in the text, so only the details of the encoder are shown in the table. The input size of the picture is 640 × 320.

	Type	# Filters	Kernel Size/Stride	Output Size
S1	Conv+BN+ReLU	16	3 × 3/2	160 × 320
S2	Conv+BN+ReLU	16	1 × 1	160 × 320
	Conv+BN+ReLU	16	3 × 3/2	80 × 160
	Conv+BN+ReLU	16	3 × 3	80 × 160
S3	DWConv+BN	32	3 × 3/2	40 × 80
	Conv+BN+ReLU	32	1 × 1	40 × 80
	DWConv+BN	32	3 × 3	40 × 80
	Conv+BN+ReLU	32	1 × 1	40 × 80
S4	DWConv+BN	64	3 × 3/2	20 × 40
	Conv+BN+ReLU	64	1 × 1	20 × 40
	DWConv+BN	64	3 × 3	20 × 40
	Conv+BN+ReLU	64	1 × 1	20 × 40
S5	DWConv+BN	128	3 × 3/2	10 × 20
	Conv+BN+ReLU	128	1 × 1	10 × 20
	DWConv+BN	128	3 × 3	10 × 20
	Conv+BN+ReLU	128	1 × 1	10 × 20
	GAPooling+BN		3 × 3	10 × 20
	Conv+BN+ReLU	128	1 × 1	10 × 20
	Conv	128	3 × 3	10 × 20

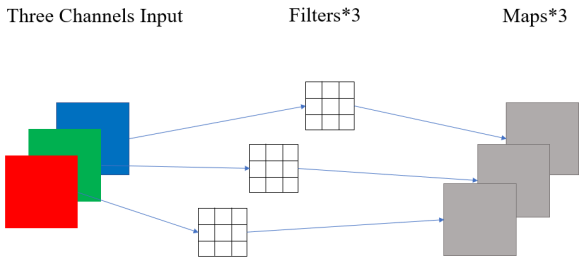
2.1.1. Encoder

The backbone network extracts image features, which is also the encoding part of the network. Inspired by Bisenetv2, RailNet is designed to extract semantic information features of images [20]. The encoder of the RailNet replaces the standard convolution

operations by the Depth Wise Convolutions (DWconv) to significantly lower the computational cost [21]. The details of the DWconv reducing the calculation cost are shown in Figure 3. To be more specific, the DWconv layers with a kernel size of 3 are stacked for progressive feature extraction. The 1×1 convolution layer is designed to follow each DWConv layer, which benefits for channel-wise information aggregation. As noted above, there exists a great deal of objects that share similar local appearance with rail lines in the input images. In order to improve detection accuracy, the context information should be properly extracted and preserved in the encoding stage. The network is designed as two DWConv layers followed by a 1×1 convolution layer, which is used for feature extraction on one particular feature resolution [22]. The first DWConv layer has a dilation rate of 1, while the following layer uses dilation rate of 2. This enlarges the reception field. The structure design of the encoder gives proper consideration to the efficiency and accuracy of the detection.



(**a**) The working principle of Conventional Convolution.



(**b**) The working principle of Depth Wise Convolution.

Figure 3. Comparison between Conventional Convolution and Depth Wise Convolution. Different from the Conventional Convolution operation, one convolution kernel of Depth Wise Convolution is responsible for one channel, and one channel is calculated by only one convolution kernel. However, the Conventional Convolution kernels operate each channel of the input picture at the same time. It can be seen from the figure that the calculation amount of Conventional Convolution is three times that of the Depth Wise Convolution.

2.1.2. Decoder

After the backbone network is the binary segmentation part, which is the decoding part of the network. In order to recover the feature resolution and produce the rail line binary segmentation images, we design a decoder architecture that follows the encoder [19]. Although the transposed convolutional layer is mainly used to amplify the intermediate features in the neural network, it has the disadvantage of excessive calculation. Since the sub-pixel convolution layer has the advantages of no parameters and no computational cost, we use the sub-pixel convolution layer to gradually restore the feature resolution. The last layer of the decoder is the softmax layer, which is used to classify pixels. The decoder of RailNet has trained to output binary segmentation maps, indicating which pixels belong to a rail line or not [23].

2.1.3. Split Soul (SS) Module

To further improve the segmentation accuracy, we propose a booster training strategy [24], called the Split Soul (SS) module. This module consists of a 3×3 global average pooling layer, a 1×1 convolution layer, and a 3×3 convolution layer. The specific structure details of the SS are shown in Figure 4. More specifically, it is similar to a catalyst in chemical reactions: it can enhance the feature representation in the training phase and can be discarded in the testing phase. Accordingly, it increases little computation complexity in the testing phase. We can insert the Split Soul (SS) module to different positions of the RailNet. In general, it improves the segmentation performance without any extra testing time.

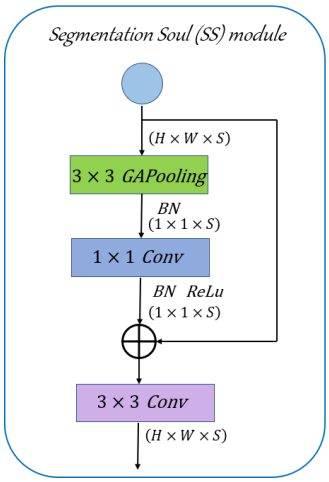


Figure 4. Detailed design of the SS. *GAPooling* is the global average pooling. *Conv* is the convolutional operation. *BN* denotes batch normalization. *ReLu* is the ReLu activation function. Simultaneously, 1×1, 3×3, indicates the kernel size and $H \times W \times S$ represents the tensor shape (height, width, depth).

2.1.4. Loss Function

The RailNet model applies the classical cross-entropy as the loss function, and the L_1 loss, L_2 loss, and cross-entropy loss are widely used in rail line detection. Among them, x_i is the input, y_i is the actual true value, that is, the known label, and y_i^* is the predicted value of the output. The cross-entropy loss uses an inter-class competition mechanism, and p_i is the probability that the sample belongs to class C. When $C = 2$, the cross-entropy loss can be defined as a binary classification problem, where y is the label of the sample, the positive class is one, and the negative class is zero. In railway line detection, the imbalance rate between the railway line and the background is considerable. In order to solve this problem, each category is given a different weight w_i. However, due to the existence of an inter-class competition mechanism, cross-entropy loss mainly represents the accuracy of prediction probability of correct tags. It ignores the difference of other wrong titles. To increase the intersection of predicted rail line pixels and actual rail lines pixels, we propose a loss function $L_{IoU-Rail}$ based on IoU:

$$L_{IoU-Rail} = 1 - \frac{M_p}{M_p + M_T + M_C} \tag{1}$$

where M_p is the predicted rails pixel, M_T is the real rails pixel, M_C and is the rail lines in the overlap area between the predicted rail lines area and the actual rail lines area.

2.2. The Rail Line Fitting Algorithm Based on Sliding Window Detection

As mentioned in the previous section, the RailNet outputs a set of pixels for the rail lines. It is not ideal to fit polynomials by these pixels in the original image space, so people have to resort to higher-order polynomials to deal with curved rail lines [23]. A generally accepted solution to this problem is to project the image into a "bird's eye" representation, where the rail lines are parallel to each other, so curved rail lines can be fitted with second to third-order polynomials.

The algorithm mainly consists of three steps: Inverse Perspective Transformation (IPT), Feature Point Extraction (FPE), Rail Lines Curve Fitting. Figure 5 shows the specific algorithm flow in the form of a flowchart.

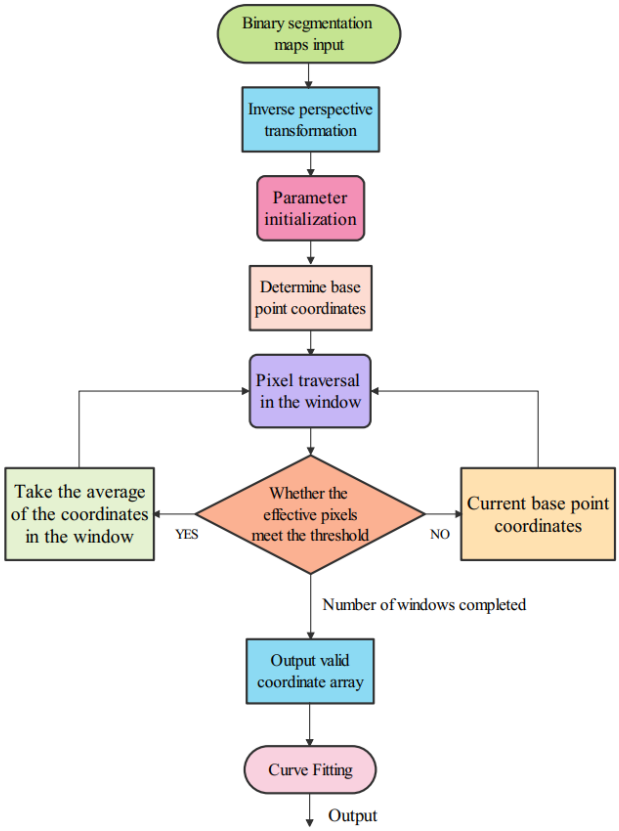


Figure 5. Flow diagram of the rail lines fitting algorithm based on sliding window detection.

2.2.1. Inverse Perspective Transformation (IPT)

Inverse perspective transformation is to remove the perspective effect of the camera and restore the parallel rail lines from the perspective of the top view. The inverse perspective transformation is as follows [25]:

$$\begin{cases} x' = C\frac{x_d cos\theta_1 - y_d sin\theta_1 cos\theta_2 + cos\theta_1 sin\theta_2}{sin\theta_1 + y_d cos\theta_1} + Acos\theta_2 + Bsin\theta_2 \\ y' = C\frac{-x_d sin\theta_1 - y_d sin\theta_1 cos\theta_2 + cos\theta_1 sin\theta_2}{sin\theta_1 + y_d cos\theta_1} - Asin\theta_2 + Bcos\theta_2 \end{cases} \quad (2)$$

where x_d and y_d satisfy:

$$\begin{cases} x_d = \frac{x - c_x}{f_x} \\ y_d = \frac{y - c_y}{f_y} \end{cases} \quad (3)$$

In Equation (2), $O_C(A, B, C)$ coordinates the optical center of the camera in the world coordinate system. Respectively, θ_1 and θ_2 are the pitch angle and yaw angle of the camera.

The point ($x^{'}$,$y^{'}$) is the corresponding point of the pixel (x,y) in the original image in the inverse perspective image.

2.2.2. Feature Point Extraction (FPE)

After the inverse perspective transformation, the feature points of the target are detected and collected using histogram, sliding window, and other algorithms.

The priority in this task is the determination of the coordinates of the initial sliding base points of the sliding window. With the bottom edge of the feature map set to the x-axis after inverse fluoroscopic transformation, the distribution of pixels in the vertical direction of the image for each abscissa on the x-axis is statistically derived using a histogram [26]. At this time, because most feature pixels belong to the tracks, there will be two apparent peaks near the abscissa of the rail lines on the left and right sides [27]. The coordinates of these two peaks are the starting points of sliding window detection.

After that is the sliding window detection of the feature map: First and foremost, the parameters are designed and initialized, including the number of sliding window detection, the height and width of the sliding window are obtained by image size and the number of detection [28]. Next, In the process of feature point detection in a sliding window, the window pixels are traversed, and the coordinates of non-zero pixel values are recorded. When the number of effective pixels in the window is less than the threshold, the window width is increased by the window height and width until the minimum number of pixels is met [29]. Furthermore, taking the average value of the abscissa of the effective pixels in the sliding window as the base point coordinate of the next sliding window, iterative detection is carried out until the total number of sliding windows is satisfied [30]. Last but not least, after the feature point detection deadline, the target array is the feature point of the detected target.

2.2.3. Rail Lines Curve Fitting

The detected feature points are fitted by the curve fitting algorithm. The curve fitting of the collected rail feature point array can well estimate the parameters of the rail lines, such as offset, inclination angle, curvature radius, and other information, so as to predict the direction of the tracks and provide help for the automatic train control system. The existing mainstream algorithm is to directly use the least square method to do quadratic or cubic curve fitting. For uncomplicated rail line detection, the fitting results are mostly quadratic curves, meeting the requirements.

$$f(x) = ax^2 + bx + c \tag{4}$$

where a, b and c are the quadratic term, the coefficient of the first term and the constant term respectively.

$$\ell = \frac{|2a|}{[1 + (2ax + b)^2]^{\frac{3}{2}}} \tag{5}$$

The curvature ℓ of the rail lines is easily derived from the above polynomial procedure.

3. Experimental Methodology

3.1. RAWRail

In order to realize the function of the network designed in this paper, we need to train and test the network. So as to verify the feasibility of the algorithm and reduce the difficulty of feature extraction, the experiment first collects the video stream of the rail lines in front of the train when the weather is good in the daytime and then converts it into pictures. When collecting images, we use cameras installed in front of local trains and city railways, which can capture objects about 350 m ahead during the daytime. However, due to the influence of the external environment and the inverse perspective transformation in the algorithm, the algorithm detects the rail lines distance up to 280 m. The dataset is

named RAWRail. A total of 3000 railroad track pictures with 640 × 360 are prepared, and the images with only 2 rail lines are first detected.

Secondly, we label the rail lines of all the rail images by using LABELME to get the JSON file used as the real rail lines during training and finally compared with the predicted rail lines [29]. In the actual training, the 3000 pictures are divided into the training set, verification set, and test set according to the ratio of 0.9:0.05:0.05. The specific information of RAWRail is shown in Table 2.

Table 2. Specific distribution of the RAWRail.

Number of Rails	Left Curved Tracks	Right Curved Tracks	Straight Tracks	In All
2	1000	1000	1000	3000

3.2. Evaluation Metrics

At present, it is rare to use deep learning neural networks for feature extraction of train tracks, so the existing evaluation index for rail line detection is not perfect. In terms of evaluation indexes, TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative) are commonly used in the field of image processing [31].

The accuracy is calculated as the average number of correct points per image:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} = \sum_i \frac{X_i}{Y_i} \tag{6}$$

where X_i the number of correct points and Y_i the number of ground-truth points. When the difference between the basic fact and the prediction point is less than a certain threshold, the point is correct. In addition to accuracy, FNR (False Negative Rate) and FPR (False Positive Rate) are also proposed.

$$FNR = \frac{FN}{TP + FN} = \frac{F_{pred}}{N_{gt}} \tag{7}$$

$$FPR = \frac{FP}{TN + FP} = \frac{N_{pred}}{F_{all}} \tag{8}$$

where F_{pred} is the number of rail lines that are initially correct but are predicted to be negative, N_{gt} is the number of all right rail lines, N_{pred} is the number of rail lines that are originally negative but predicted to be positive, F_{all} is the number of all wrong tracks.

3.3. Implementation Details

The hyperparameters of each experiment in this work are generally consistent. Although the dataset has 3000 images, this is far from enough. During training, these experiments use data enhancement appropriately, and apply data enhancement with a probability of 10/11. The transformations used are rotation with an angle in degrees $\theta \sim U(-10, 10)$, horizontal flip with a possibility of 0.5 [22]. The Adam optimizer is used, along with the Cosine Annealing learning rate scheduler with a batch size of 16 and an initial learning rate of 5×10^{-4} until convergence [26]. The training session runs for 1961 epochs, taking approximately 23 h on four GeForce RTX 2080Ti. In the post-processing curve fitting, a second-order polynomial degree is chosen to be the default. The Tensorboard is used for data visualization analysis, and the RailNet training process is shown in Figure 6.

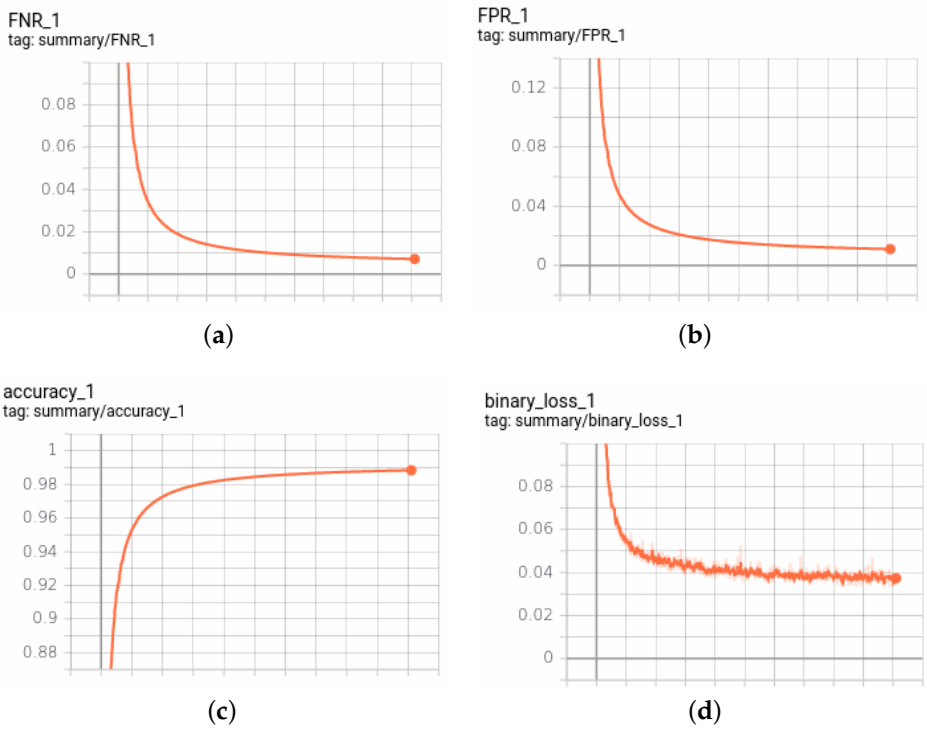


Figure 6. Training process of the RailNet. (**a**) FNR result. (**b**) FPR result. (**c**) Accuracy result. (**d**) Loss function result.

3.4. Results

In order to verify the rationality and superiority of the algorithm, the following three sets of experiments are designed. The detailed experimental diagram of each step of the algorithm is shown in Figure 7.

3.4.1. State-of-the-Art Comparison

The results of comparing our algorithm with other latest algorithms are shown in Table 3. Because these documents do not provide source code the evaluation index data can only be directly extracted from the original papers. Regarding detection accuracy and detection speed, it is not difficult to see that our algorithm is exceptionally competitive. Figure 8 shows the results of the algorithm detecting the rail lines area, which reflects the excellent performance in both straight and curved rail lines. The studies in [1,11,25] all lack the two evaluation metrics of FPR and FNR, and [11] also lacks the evaluation metric of FPS. Therefore, we introduce the two new evaluation indicators of FPR and FNR into these three algorithms and introduce FPS into [11]. We conduct supplementary experiments on FPR, FNR, and FPS evaluation metrics, and reproduce the algorithms. Morever, we let the four algorithms run under the same instrument and GPU conditions. In this way, the ACC, FPR, FNR, and FPS of the four algorithms can be compared comprehensively and clearly. In practical application, our algorithm not only detects the rail lines in real-time during train operation but also has strong robustness in bad weather.

3.4.2. Multi-Rail Line Detection

There will be some unexpected situations in practical application, such as several trains running in parallel and changing the rail lines in time when meeting the rail fork. At this time, it is of vital importance to identify multiple rail lines. It can be seen from Table 4 that the algorithm still has high accuracy in identifying multi-rail lines. This shows that the algorithm has strong robustness.

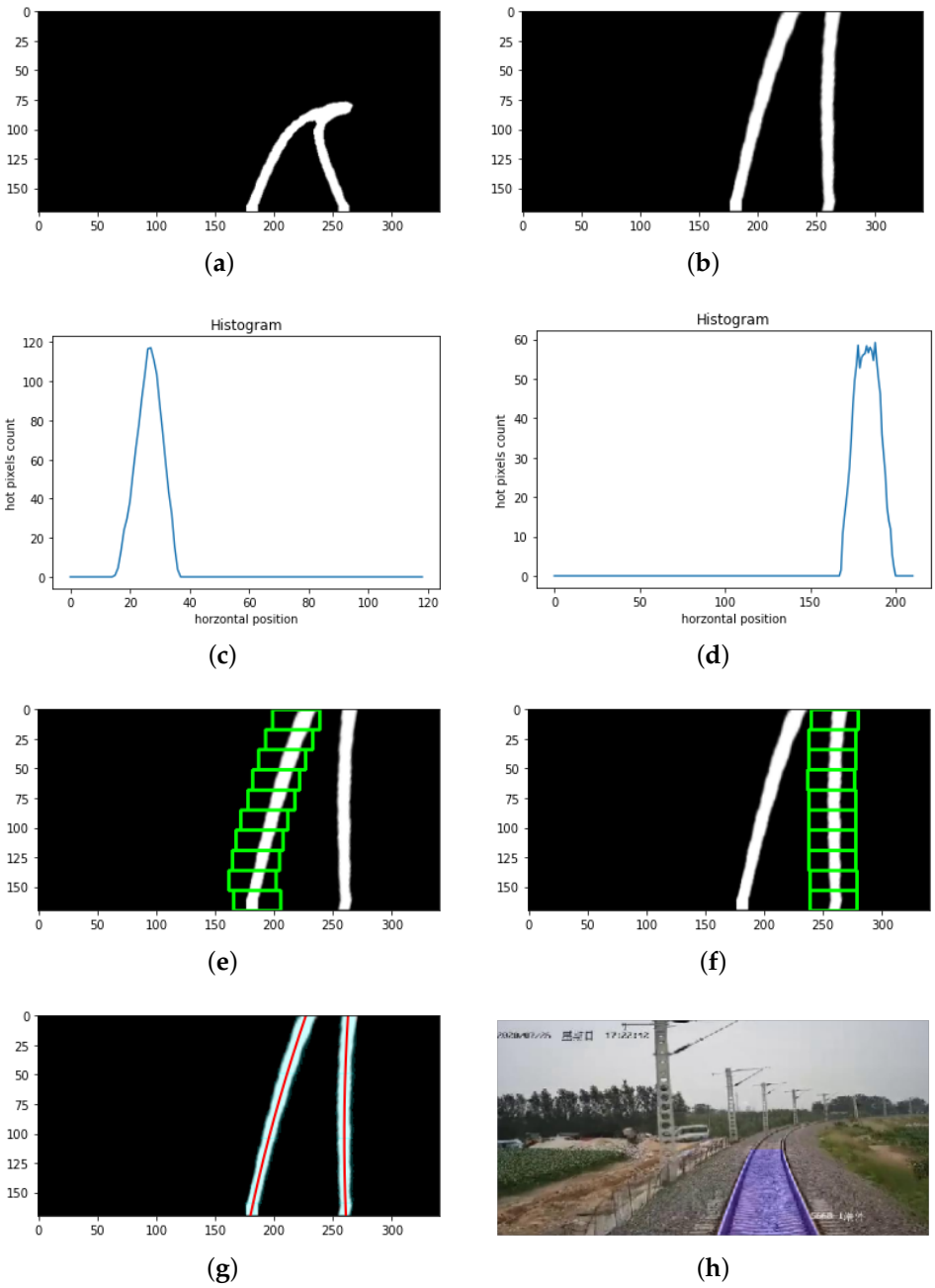


Figure 7. Experiment process of the algorithm. (**a**) Binary segmentation diagram. (**b**) Inverse perspective transformation diagram. (**c**) Histogram detection of left rail diagram. (**d**) Histogram detection of right rail diagram. (**e**) Result of sliding window detecting left rail. (**f**) Result of sliding window detecting right rail. (**g**) Schematic of characteristic point curve fitting. (**h**) The final rail line prediction.

3.4.3. Ablation Study

To investigate the impact of some of the decisions made for the proposed method [22], two ablation studies were carried out, using only RAWRail's training set for training and the validation set for testing. Different order polynomials are used to fit the rail lines in the curve fitting part of the rail lines fitting algorithm based on sliding window detection module. The experimental results are shown in Table 5. Due to the camera's angle of view, the detected track lines are mainly near the camera, and the curvature of this part is not particularly obvious. This is why there is little difference in the experimental results when other order polynomials are used to fit rail lines. Therefore, low order polynomials of

different orders have little influence on the fitting results. Even so, it can be seen from the data in the table that when the polynomial is of second-order, the detection accuracy is the highest, so the second-order polynomial is also used in the design of this algorithm.

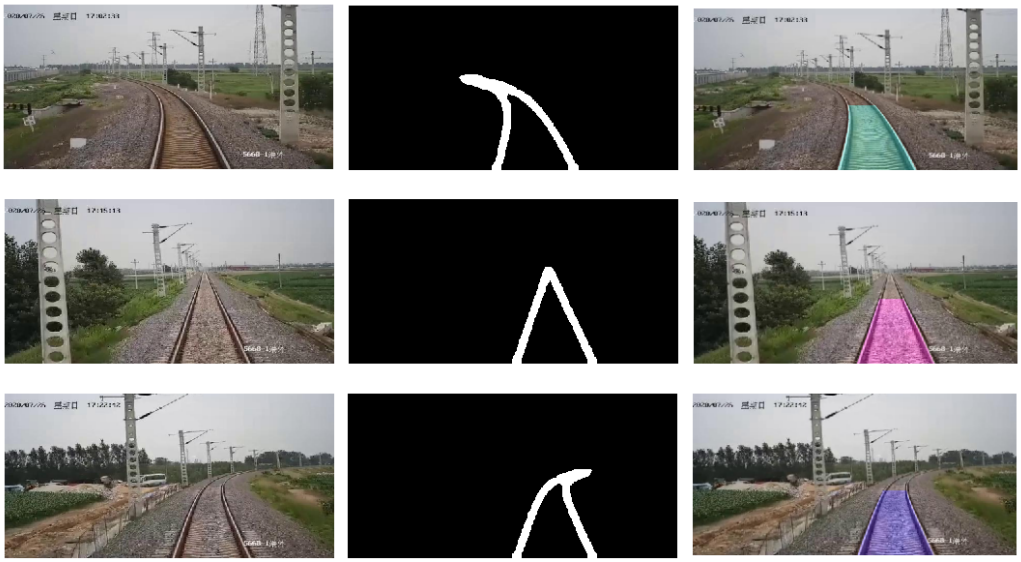


Figure 8. Test results of rail lines. The first column: the original images. The second column: the RailNet output. The third column: the final rail lines prediction.

Table 3. Comparison of the results of the state-of-the-art rail lines detection algorithms.

Method	ACC	FPR	FNR	FPS
Zhong Ren [11]	76.7%	0.21872	0.10423	37.4
Tong Zhang [25]	79.0%	0.18790	0.08953	25.6
Lei Zhang [1]	90.1%	0.09094	0.04325	22.7
Proposed Method	**98.6%**	**0.01104**	**0.00714**	**74.2**

Table 4. The results of different type of rail lines on RAWRail.

Rail Line Type	ACC	FPR	FNR
Multi-rail lines	94.16%	0.05958	0.02832

Table 5. The results of different polynomial degrees on RAWRail.

Polynomial Degrees	ACC	FPR	FNR
1st	97.71%	0.02309	0.01570
2nd	98.65%	0.01321	0.00864
3rd	98.42%	0.01570	0.01038

As to another ablation study we carried out, we can find that the resolution of the train camera is also the key factor affecting the results. Different sizes of images are input into the algorithm, and the specific experimental results are shown in Table 6. The experimental results show that reducing the image size will reduce the accuracy of rail lines prediction. At the same time, the detection speed of rail lines has increased significantly. In practical applications, combined with the characteristics of this algorithm, the best accuracy and detection speed can be found [32]. The image size in the RAWRail is 640×320.

Table 6. The results of different picture sizes on RAWRail.

Input Sizes	ACC	FPR	FNR	FPS
320 × 180	95.14%	0.04895	0.03520	79.2
480 × 270	97.01%	0.02880	0.01991	76.4
640 × 360	98.64%	0.01332	0.00872	72.6

4. Conclusions

In this paper, a novel method for rail line detection algorithm based on deep learning is proposed. Firstly, we propose the lightweight RailNet. The RailNet extracts the feature of tracks by converting the rail line detection into an image segmentation problem. The ingenious design of RailNet remarkably improves the accuracy and real-time of algorithm detection. Afterward, we design the rail lines fitting algorithm based on sliding window detection, which makes full use of the segmentation feature maps output by RailNet and finally marks the rail lines on the original images. For the training and testing of the RailNet, we collect a real-world rail lines dataset called RAWRail. Compared with the state-of-the-art methods, the proposed method is effective and efficient, while maintaining an accuracy of 98.6% and detection speed of 74 FPS. Furthermore, the proposed algorithm also works well with multi-rail lines, which provides wide application prospects.

Author Contributions: Conceptualization, Z.T. and Y.S.; methodology, Z.T.; investigation and validation, Z.T., S.R., and Y.S.; data curation and formal analysis, Z.T., X.W., and W.W.; writing—original draft preparation, Z.T. and Y.S.; writing—review and editing, S.R., X.W., and W.W.; funding acquisition, S.R. and W.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 61801024).

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

References

1. Zhang, L. Research on Railway Track Recognition Technology in Infrared Image. Master's Thesis, University of Electronic Science and Technology, Chengdu, China, 2012.
2. Pan, Z.; Qiang, H. The train sliding on-line monitoring system based on three times spline curve. *Microcomput. Inf.* **2012**, *28*, 39–41.
3. Xiao, Y.; Su, L. Research on Integration Technology of Rail Transit Fully Automatic Driving System. *China Railw.* **2015**, *5*, 109–113.
4. Narote, S.P.; Bhujbal, P.N.; Narote, A.S.; Dhane, D.M. A review of recent advances in lane detection and departure warning system. *Pattern Recognit. J. Pattern Recognit. Soc.* **2018**, *73*, 216–234. [CrossRef]
5. Guan, X. Research on Train Obstacle Detection and Recognition Technology Based on Deep Learning. Master's Thesis, Beijing Jiaotong University, Beijing, China, 2020.
6. Chen, R. Research on Image Detection Algorithm of Obstacles in Front of Train. Master's Thesis, Southwest Jiaotong University, Chengdu, China, 2012.
7. Zhu, S.; Du, J.; Li, Y. Method for bridge crack detection based on the U-Net convolutional networks. *J. Xidian Univ.* **2019**, *46*, 41–48.
8. Li, W.; Zhang, M.; Shen, Z.; Hu, W.; Li, P. Track Crack Detection Method in Complex Environment. In Proceedings of the 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 8–9 December 2018; Volume 1, pp. 356–359.
9. Zhang, C. Research on Detection System of Railway Foreign Body Intrusion Limit Based on Deep Learning. Master's Thesis, North China University of Technology, Beijing, China, 2017.
10. Maire, F.; Bigdeli, A. Obstacle-free range determination for rail track maintenance vehicles. In Proceedings of the International Conference on Control Automation Robotics and Vision(ICARCV), Singapore, 7–10 December 2010; pp. 2172–2178.
11. Ren, Z. Railway Recognition Based on Prior Knowledge. Master's Thesis, Wuhan University of Technology, Wuhan, China, 2007.
12. Wang, Q.; Liang, X.; Lu, Z. Railway rail identification detection method using machine vision. *J. Cent. South Univ. (Sci. Technol.)* **2014**, *45*, 2496–2502.
13. Wu, Z.; Tian, X.; Wang, B. An Improved Track Recognition Algorithm Based on Computer Vision Technology. *Sci. Technol. Vis.* **2019**, *5*, 32–33+24.
14. Nassu, B.T.; Ukai, M. A Vision-Based Approach for Rail Extraction and its Application in a Camera Pan–Tilt Control System. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 1763–1771. [CrossRef]

15. Badino, H.; Franke, U.; Mester, R. Free Space Computation Using Stochastic Occupancy Grids and Dynamic Programming. In Proceedings of the IEEE International Conference on Computer Vision(ICCV), Rio de Janeiro, Brazil, 14–20 October 2007; Volume 20.
16. Wang, Y. Design and Implementation of an Algorithm to Recognize the Rail Ahead of The Train. Master's Thesis, Zhejiang University, Hangzhou, China, 2017.
17. Wang, Z.; Shu, G. Research on Track Section Identification Based on Traditional Image Processing Algorithm and Deep Learning. *J. Electr. Autom.* **2019**, *41*, 111–114.
18. Zheng, Y.S.; Jin, Y.W.; Dong, Y. Rail Detection Based on LSD and the Least Square Curve Fitting. *Int. J. Autom. Comput.* **2021**, *18*, 85–95. [CrossRef]
19. Wang, Z.; Ren, W.; Qiu, Q. Lanenet: Real-time lane detection networks for autonomous driving. *arXiv* **2018**, arXiv:1807.01726.
20. Zhao, H.; Li, H.; Li, C. Improving Retinal Vessel Segmentation with Joint Local Loss by Matting. *Pattern Recognit.* **2019**, *98*, 107068. [CrossRef]
21. Hou, Y.; Ma, Z.; Liu, C.; Loy, C.C. Learning lightweight lane detection cnns by self attention distillation. *arXiv* **2019**, arXiv:1908.00821.
22. Tabelini, L.; Berriel, R.F.; Paixao, T.; Badue, C.; Souza, A.F.D.; Oliveira-Santos, T. Polylanenet: Lane Estimation via Deep Polynomial Regression. *arXiv* **2020**, arXiv:2004.10924.
23. Paszke, A.; Chaurasia, A.; Kim, S.; Culurciello, E. ENet: A deep neural network architecture for real-time semantic segmentation. *arXiv* **2016**, arXiv:1606.02147.
24. Yu, C.; Gao, C.; Wang, J.; Yu, G.; Shen, C.; Sang, N. BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation. *arXiv* **2020**, arXiv:2004.02147.
25. Zhang, T. Research on Railway Pedestrian Invasion Limit Detection System Based on Deep Learning. Master's Thesis, Beijing Jiaotong University, Beijing, China, 2020.
26. Xu, C. Track Detection and Recognition Based on UAV. *J. Hubei Polytech. Univ.* **2020**, *36*, 12–15.
27. Ko, Y.; Jun, J.; Ko, D.; Jeon, M. Key points estimation and point instance segmentation approach for lane detection. *arXiv* **2020**, arXiv:2002.06604.
28. Neven, D.; Brabandere, B.D.; Georgoulis, S.; Proesmans, M.; Gool, L.V. Towards End-to-End Lane Detection: An Instance Segmentation Approach. *arXiv* **2018**, arXiv: 1802.05591.
29. Abualsaud, H.; Liu, S.; Lu, D.; Situ, K.; Rangesh, A.; Trivedi, M. LaneAF: Robust Multi-Lane Detection with Affinity Fields. *arXiv* **2021**, arXiv:2103.12040.
30. Niu, J.; Lu, J.; Xu, M.; Lv, P.; Zhao, X. Robust Lane Detection using Two-stage Feature Extraction with Curve Fitting. *Pattern Recognit. J. Pattern Recognit. Soc.* **2016**, *59*, 225–233. [CrossRef]
31. Zou, Q.; Jiang, H.; Dai, Q.; Yue, Y.; Chen, L.; Wang, Q. Robust Lane Detection From Continuous Driving Scenes Using Deep Neural Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 41–54. [CrossRef]
32. Suder, J.; Podbucki, K.; Marciniak, T.; Dąbrowski, A. Low Complexity Lane Detection Methods for Light Photometry System. *Electronics* **2021**, *10*, *1665*. [CrossRef]