www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

An Algorithm For Detecting Vehicles In Live Video Streams Using Yolov4

[1] P. Venu Madhav, [2] C. Vaishnavi,

[1]Assistant Professor, Megha Institute of Engineering & Technology for Women, Ghatkesar.
[2] MCA Student, Megha Institute of Engineering & Technology for Women, Ghatkesar.

Abstract

Computer vision is a crucial technology for achieving safe driving of automobiles, which requires perceiving information about the vehicle's environment. When it comes to fast and accurate one-stage target identification algorithms, YOLO and SSD, RetinaNet are good examples. While the most recent algorithm in the YOLO series, YOLOv4, does a better job at detecting vehicle targets quickly and accurately than its predecessors, it is still far from real-time. With the current detection speed being insufficient, this research suggests an enhanced YOLOv4-based video stream vehicle target recognition system. The study begins with a theoretical introduction to the YOLOv4 method, then on to suggest an algorithmic procedure to increase detection speed, and concludes with real road trials. This paper's method, according to the experimental results, may increase the system's detection speed without sacrificing accuracy, which can form the foundation for decisions regarding safe vehicle driving.

Keywords

YOLOv4; Video Streaming, Vehicle detection; algorithm fusion

I. Introduction

With the country's economy and productivity continuing to rise, more and more Chinese are purchasing autos. While cars make many things easier, they also pose certain dangers. In order to mitigate these risks, vehicles need have the capability to detect their environment and react accordingly. Autos can detect and avoid hazards like other cars, pedestrians, and other road users thanks to computer vision technology, which is essential to autonomous driving. By combining data from other sensors, such as millimeter wave radar and lidar, vehicle safety and environmental consciousness might be improved. Currently, there are two dominant ideologies in the realm of vehicle target detection: one that prioritizes deep learning and another that sticks to more conventional image processing methods. Manually retrieving design information is the backbone of traditional image processing detection approaches. Though it's easy to understand and uses little computing power, this method isn't as resilient or generalizable as the algorithm. In contrast, deep learning-based methods may accomplish end-to-end detection and training thanks to the widespread usage of convolutional neural networks for learning and feature extraction. In terms of durability, generalizability, and detecting effects, convolutional neural networks are second to none. Depending on whether the candidate frame of the prospective target has to be obtained in advance, target identification based on convolutional neural networks may be one-stage or two-stage, as shown by two-stage algorithms like R-CNN, fast R-CNN, faster RCNN, R-FCN, etc. Some algorithms that just need one step include the YOLO series, SSD, and RetinaNet. Based on criteria for real-time accuracy, the YOLOv4 algorithm is selected for vehicle target recognition in this experiment.

II. Related Theories

Backbone, the feature extraction network, Neck, and YOLO Head are the three primary components of the YOLOv4 algorithm. Figure 1 shows the network topology when the input picture resolution is 608*608. In order to get the final prediction results, which include the target's class, the Bbox (bounding box), and the confidence, the YOLO Head uses the features extracted from the previous part, which are based on the model's feature pyramid structure, which is CSPDarkNet53. The model's neck part also includes the SPP and PANet structures. An upgrade over Darknet53 is the CSPDarknet53 network. In the Darknet53 network, there are 53 convolutional layers; in the core network, there are 52 convolutions (not including the final fully linked layer). A convolution kernel with thirty-two filters is the first stop for an incoming picture before it makes its way through five sets of residual units in a neural network. In each of the five remaining units, there is a single convolutional layer joined by a series of convolutional layers that are executed repeatedly with repetitions of 1, 2, 8, 4, and so on. Each of these layers first performs a 1 × 1 convolution operation and then a 3 × 3 convolution operation. The input of the convolutional layer before it is fused with the data that has been convolutionally processed before being used as the input of the subsequent layer in a residual network is a network that has undergone convolutional operation. This procedure can fix deep learning's gradient disappearance issue, allowing the network to keep learning and improving the model's feature extraction capabilities. All of the convolution operations in the first individual convolution layer of the residual unit have a stride of 2. Hence, after 5 residual units, the structure with an input of (608,608,32) becomes (19,19,1024).

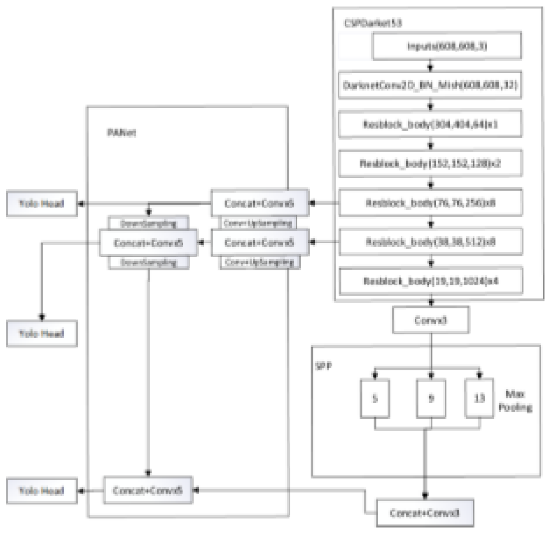

CSPDarknet53
Inputs(608,608,3)
DarknetConv2D_BN_Mish(608,608,32)
Resblock_body(304,304,64)x1
Resblock_body(152,152,128)x2
Resblock_body(76,76,256)x8
Resblock_body(38,38,512)x8
Resblock_body(19,19,1024)x4
Conv x3
SPP
5
9
13
Max Pooling
Concat+Conv x3
PANet
Concat+Conv x5
Conv+UpSampling
DownSampling
Yolo Head

Fig. 1. Network structure diagram of YOLOv4

CSPDarknet53 builds on Darknet53 and adds two major enhancements to it. An example of this is the following change to the DarknetConv2D module: it replaces the LeakyReLU function with the Mish activation function.

$$Mish = x \times \tanh(\ln(1 + e^{x})) \tag{1}$$

Its difference with the LeakyReLU activation function is shown in Figure 2

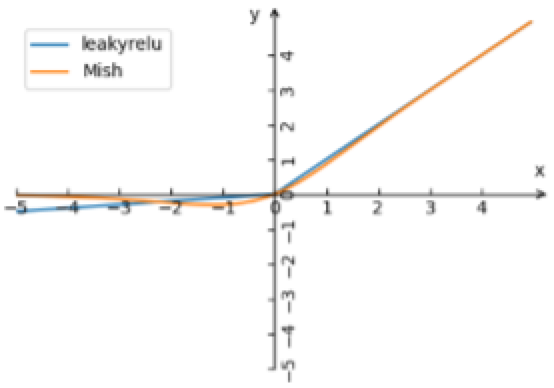


Fig. 2. Comparison of Mish with LeakyReLU

The unbounded nature, lack of saturation from capping, and minimal negative gradient flow of the Mish function guarantee information flow; the superiority of the information that can penetrate deeper into the neural network as a result of the Mish's smoother gradient is another benefit. Secondly, Resblock_body makes use of CSPNet, which involves dividing the original residual block into two halves. The first half continues the stack of the original residual block, while the second half acts as a residual edge, connecting directly to the end after a small amount of processing. This arrangement gives the impression that CSPNet has a large residual edge, as shown in Figure 3.

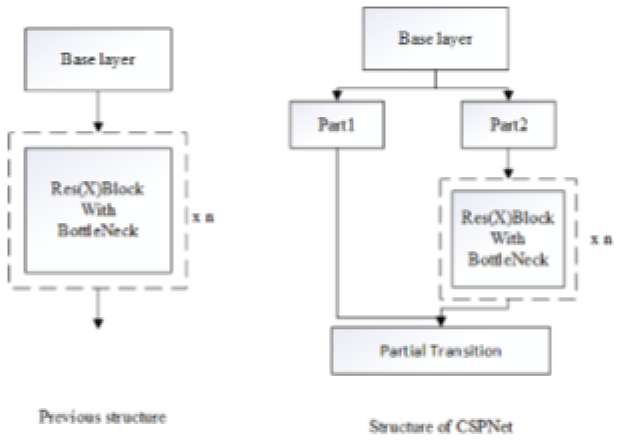


Fig. 3. Comparison of CSPNet structure with the previous structure

In the final feature layer of CSPDarknet53, the SPP structure is encased using a convolutional sandwich. In that order, the last feature layer of CSPDarknet53 undergoes three convolutions before using maximum pooling kernel sizes of 13×13, 9×9, 5×5, and 1×1 for four distinct scales. By including SPP, we are able to enhance the model's understanding and retrieve important contextual information. Figure 1 demonstrates that in order to perform feature extraction from both the bottom to the top and the top to the bottom of the feature pyramid, the PANet structure continually employs several up sampling and down sampling procedures. Three feature layers, with coordinates (19,19,1024), (38,38,512), and (76,76,256), are extracted using YOLOv4, as shown in Figure 1. These layers are effectively used by YOLOv4. Following PANet's splicing, convolution, upsampling, and downsampling, the feature layers provide three output layers: (19,19,75), (38,38,75), and (76,76.). The network's ultimate dimension is 75 due to its training on the 20-class VOC dataset. The output includes the confidence, center coordinates (b, y), height (h), and width () of the forecast box. Each feature layer uses three preselected boxes, and the approach uses a total of twenty-five dimensions. To acquire the detection results, NMS (NonMaximum Suppression) is performed once the final prediction structure is obtained. Learning rate cosine annealing decay, label smoothing, CIOU, and mosaic data augmentation are a few of the enhanced training approaches in YOLOv4. Both the model and the algorithm's feature extraction capabilities are enhanced by these strategies.

III. Algorithm Fusion Process

This paper presents an improved algorithm that takes video input frames at a time, generates a hash value from the difference between them, extracts a fingerprint from each frame, calculates their hamming distance, and then compares this distance to a threshold value to find how similar the two frames are. The average calculation time of the whole algorithm is improved by selecting the algorithm based on the similarity size. A. Hash Algorithm for Differences In comparison to other hash algorithms, such as the average hash algorithm and the perceptual hash algorithm, the difference hash algorithm (DAH) is superior for quick picture difference detection due to its combination of speed and quality. Here is how it works. First, hide the picture's features by reducing the input image to a 9×8 size; the final processed image contains 72 pixel points. 2. Make the downsized picture into a grayscale map with 64 levels of intensity, and change the original RGB three-channel three-dimensional comparison to a one-channel one-dimensional measurement. Third, figure out how much the difference is. To get the difference value, we compare the intensities of the pixels in each row; if one pixel's intensity is higher than the other, we get a value of 1, otherwise we get a value of 0. An 8x8 difference value matrix is generated after the procedure is complete. 4 were transformed into dHash values. After obtaining an 8x8 matrix in step 3, each row's values are treated as 1 bit. Then, a hexadecimal value is formed by 8 bits, and lastly, an 8-length hash string is produced. We need this text as the hash value.How many distinct bits are present on the corresponding bit when two strings are considered far apart? That's the Hamming distance.

$$d(x, y) = \sum x[i] \oplus y[i] \qquad (2)$$

You can find out how similar two frames are by calculating the Hamming distance between them using the dHash values of the current and previous images; a good threshold number to use is 5. So, we may say that the two frames are comparable if the Hamming distance is less than 5. B. The algorithm for monitoring Camshifts You might think of Camshift as an adaptive modification of the MeanShift method. Characterized by a fast processing speed and the ability to adjust the size of its own search box, its algorithmic idea is to process each frame of the video using the Mean-Shift algorithm. It then iterates using the results of the previous frame as the initial value to achieve tracking. The following is the particular flow of its algorithm, as depicted in Figure 4 of the document. 1. Convert the RGB space picture to HSV space, then use the resulting color histogram to build the color probability density distribution. Then, use reverse projection to get the final result. 2. Assuming that the pixel （x,） is situated in the tracking frame, the probability estimate for that pixel in the color probability estimation map is denoted by (x,). What follows is the calculation for the tracking window's zero-order moment $M00$ and its first-order moments $M10$, $M01$.

$$M_{00} = \sum\sum I(x,y) \qquad (3)$$

$$M_{10} = \sum\sum xI(x,y) \qquad (4)$$

$$M_{01} = \sum\sum yI(x,y) \qquad (5)$$

3. Determine the tracking frame's center location, height h, and length o.

$$x_c = \frac{M_{10}}{M_{00}},\quad y_c = \frac{M_{01}}{M_{00}} \qquad (6)$$

$$h = \sqrt{\frac{M_{00}}{256}}, l = 1.2 \cdot h \qquad (7)$$

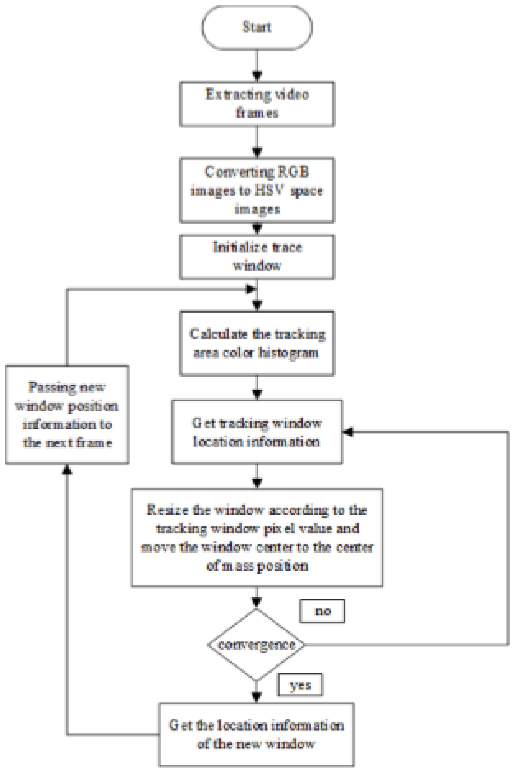

Start
Extracting video frames
Converting RGB images to HSV space images
Initialize trace window
Calculate the tracking area color histogram
Passing new window position information to the next frame
Get tracking window location information
Resize the window according to the tracking window pixel value and move the window center to the center of mass position
no
convergence
yes
Get the location information of the new window

Fig. 4. Camshift algorithm flow chart

C. A migration-learning-based YOLOv4 algorithm
The process of transferring the parameters of one model to another in order to train new models is known as transfer learning. We employ YOLOv4's weight file as a pre-training model for vehicle identification since it was formally trained using the VOC dataset, which includes 20 categories, and because the resulting model has high feature extraction and generalization capabilities. We train our system using 7132 photos labeled "car" and "person" taken from the open-source dataset KITTI, with an emphasis on front vehicle and pedestrian detection. The backbone feature extraction network's YOLOv4 pre-trained model may be fully used by freezing the network's model for the first 50 epochs. This speeds up training and prevents weight corruption at the beginning of the training process. The experimental platform used in this paper has the following specifications: a CPU i78750H with 2.21GHz, 16G RAM, an NVIDIA GeForce GTX1060 with 6G video memory, Python 3.6, CUDA 10.0, CUDNN 7.6.5, Keras 2.1.5, and Tensorflow-gpu 1.13.2. The training uses the Adam optimizer optimization algorithm with an initial learning rate of 0.0001 and a minimum learning rate

of 0.000001. The batch size is 16 when frozen and 8 when thawed. The mosaic data enhancement method and learning rate cosine annealing decays are employed. The training loss value changes as shown in Figure 5. The last step in vehicle recognition in video streams is to use the model weight file that was created by migration learning.

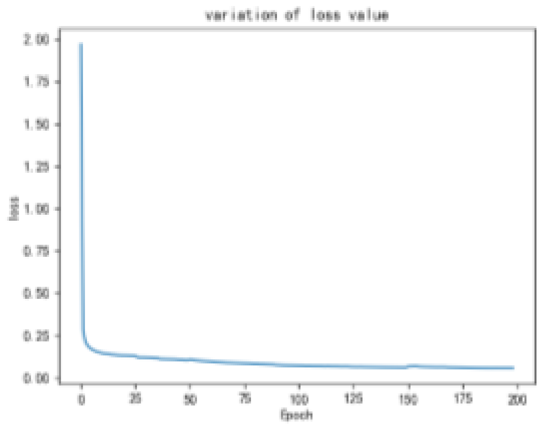


Fig. 5. Variation of loss during training

Fusion of Algorithms (D) Following the acquisition of the optimal weight file by migration learning, algorithmic fusion is carried out using this weight data. Prior to passing the first frame into the algorithm, the YOLOv4 vehicle target detection is executed and calculated. Then, the Camshift algorithm receives the tracking target and its position information. As seen in Figure 6's flow diagram, the YOLOv4 method is used for detection if there is a big change in the hash value between the current frame and the preceding one. In all other cases, the Camshift technique is utilized for tracking.

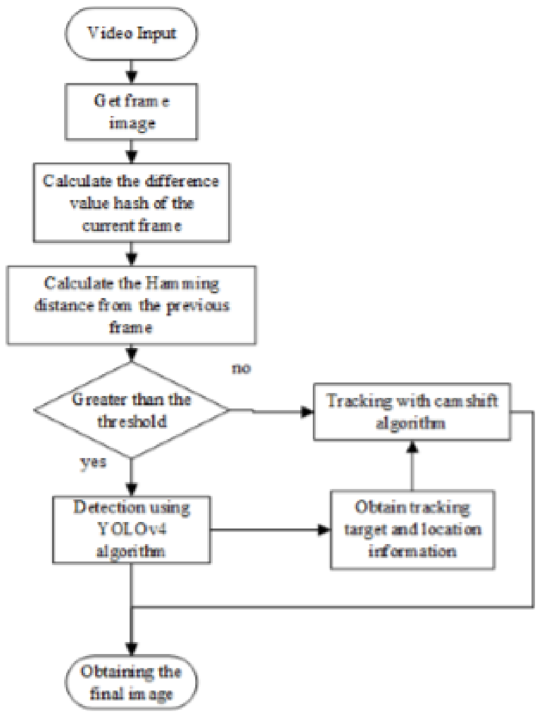
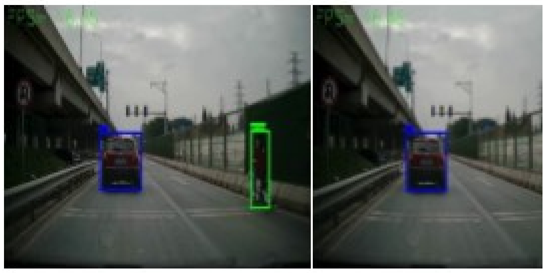

Video Input
Get frame image
Calculate the difference value hash of the current frame
Calculate the Hamming distance from the previous frame
Greater than the threshold
no
yes
Tracking with camshift algorithm
Detection using YOLOv4 algorithm
Obtain tracking target and location information
Obtaining the final image

Fig. 6. Algorithm flow chart

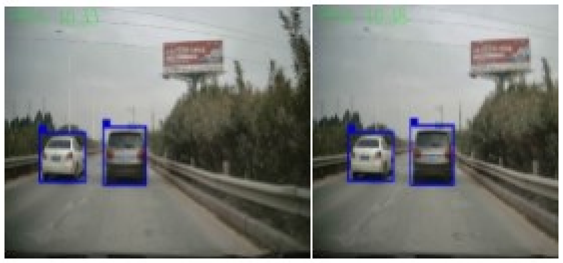
IV. Real-world road experiments

An industrial-grade U2291 USB camera was mounted to the vehicle's central rearview mirror in order to record footage from a real vehicle on a Guangzhou city road. Frames per second (FPS) were the primary metrics used to assess the algorithm's real-time performance and effectiveness. With a score of 0.5 as the default, the detection target will be shown whenever the score is higher. As shown in Figure 7, the acquired continuous frames' detecting effect is displayed.

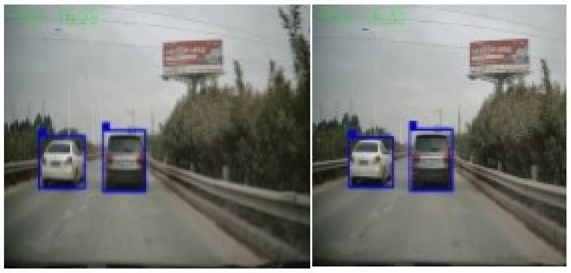
before algorithm fusion



after algorithm fusion



before algorithm fusion



after algorithm fusion

Fig. 7. Detection effect diagram

V. Conclusion

Experimental findings on real-world roads show that the upgraded YOLOv4 detection system is good at spotting approaching cars and people. Tracking using the Camshift method essentially does not diminish the detection effect, and the fused vehicle detection algorithm enhances the total detection speed from around 10 FPS to approximately 16 FPS. Hence, the technique developed for this article has enhanced the video stream detection speed.

REFERENCES

[1] GIRSHICK R. Fast R-CNN[C]//IEEE International Conference on Computer Vision. Santiago,2015.

[2] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 502–511, 2019.

[3] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN:Object detection via region-based fully convolutional networks. In Advances in Neural Information Processing Systems (NIPS), pages 379–387, 2016.

[4] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C Berg. RetinaMask: Learning to predict masks improves stateof-the-art single-shot detection for free. arXiv preprint arXiv:1901.03353, 2019

[5] Ross Girshick. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 1440–1448, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 37(9):1904–1916, 2015.

[7] J. G. Allen and J. S. Jin. Mean shift object tracking for a SIMD computer. In ICITA '05: Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) Volume 2, pages 692–697, Washington, DC, USA, 2005. IEEE Computer Society.

[8] Xiu C. and Ba F. 2016 2016 Chinese Control and Decision Conference (CCDC) (Yinchuan) Target tracking based on the improved Camshift method 3600-3604

[9] Wei W., Zhijun L., Zhiping C., Xiaomeng M., Rui Z. and Ruo Y. 2017 2017 International

Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII) (Wuhan) The Application of Improved Camshift Algorithm in Hand Tracking 87-90

[10] YOLOv4: Optimal speed and accuracy of object detection. arXiv 2020 A Bochkovskiy, CY Wang, HYM Liao - arXiv preprint arXiv:2004.10934, 2020