Journal of Emerging Computer Technologies
Research Article
Received: *2021-01-26* | ***Reviewing:*** *2021-01-26* & *2021-02-07* | ***Accepted:*** *2021-02-07* | ***Online:*** *2021-02-07* | ***Issue Date:*** *2021-06-30*

Vehicle Detection with HOG and Linear SVM

Nikola Tomikj
Ss. Cyril and Methodius University
Faculty of Computer Science and Engineering
1000 Skopje, North Macedonia
tomikj.nikola@students.finki.ukim.mk

Andrea Kulakov
Ss. Cyril and Methodius University
Faculty of Computer Science and Engineering
1000 Skopje, North Macedonia
andrea.kulakov@finki.ukim.mk

***Abstract*—In this paper, we present a vehicle detection system by employing Histogram of Oriented Gradients (HOG) for feature extraction and linear SVM for classification. We study the influence of the color space on the performance of the detector, concluding that decorrelated and perceptual color spaces give the best results. An in-depth analysis is carried out on the effects of the HOG and SVM parameters, the threshold for the distance between features and the SVM classifying plane, and the non-maximum suppression (NMS) threshold on the performance of the detector, and we propose values that illustrate good performance for vehicle detection on images. We also discuss the issues of the approach and the reasons for its mediocre performance on videos. Finally, we address these issues by presenting ideas that can be considered for improving the system.**

***Index Terms*—computer vision, machine learning, HOG, SVM, color space, vehicle detection, autonomous vehicles**

I. Introduction

In the past few years, autonomous driving has been gaining a lot of interest and it is expected to be the next big thing in the automotive industry. One of the main challenges in the development of the intelligence that powers the autonomous vehicles is its ability to detect obstacles like pedestrians, other vehicles and objects on the road. This ability provides the safety required to make autonomous vehicles mainstream. Different techniques for preceding vehicle detection have been developed throughout the literature, from traditional computer vision techniques to deep learning ones.

Dalal and Triggs [1] describe the HOG method in their breakthrough paper. They use the method for human detection. They performed thorough experiments with the parameters of the method. The values for the parameters that they found to be optimal for human detection are akin to the values that we found to be optimal for vehicle detection.

Creusen, Wijnhoven, Herbschleb, *et al.* [2] experimented with different color spaces and concluded that the choice of a color space significantly influences the performance of the HOG detector, and that the optimal color space choice depends on the type of object that the detector is trying to detect. They showed that for decorrelated color spaces like HSV, the performance when the H-channel is used as a single channel detector is almost identical to the performance in the HSV space. They concluded that this indicates that saturation and intensity information is largely irrelevant, and that color is the dominant feature. They also concluded that HSV and RGB are less suitable for traffic sign detection and detection of objects that have large color variation in general. They found out that the LAB and YCrCb color spaces provide the best performance, and that is probably due to the availability of two dedicated color channels.

Mao, Xie, Huang, *et al.* [3] took very similar approach to the one described in this paper, using HOG and linear SVM as well, and they developed effective and robust preceding vehicle detection system that can achieve high reliability target detection and low false positive rate.

An interesting and very useful approach was proposed by Arróspide, Salgado, and Camplani [4]. They overcame the computational limitations of the standard HOG, which yields excellent performance but can hardly be used in a real-time environment. They developed alternative HOG descriptors which are designed to be cost effective by making use of the previous knowledge on vehicle appearance.

Lately, deep learning approaches have been gaining popularity and several authors have experimented and produced state-of-the-art results. Fast R-CNNs are used extensively for object detection as proposed by Girshick [5]. Another important mention is YOLO introduced by Redmon, Divvala, Girshick, *et al.* [6].

In this paper we develop a pipeline for detecting preceding vehicles using HOG and linear SVM. Our aim is to study the HOG and linear SVM applicability and potential for vehicle detection. We experiment with the HOG and SVM parameters, the threshold for the distance between features and the SVM classifying plane, and the NMS threshold, to examine their influence on the performance of the system and to reveal values that provide good performance.

II. Methodology

We implemented a vehicle detection pipeline in Python that detects vehicles in images and videos recorded with a dashcam, using HOG and Linear SVM. The pipeline is developed in Python 2.7 [7] and OpenCV 3.4.3 [8].

The details of each component in the pipeline are presented as follow:

A. Data Preprocessing

The labeled data come from a combination of the GTI vehicle image database [9] and KITTI vision benchmark suite [10]. The data are png color images with dimensions 64×64, which is convenient for computing the HOG descriptors of the images. The labeled data contain 8792 vehicle images and 8968 non-vehicle images. The non-vehicle data also contain some images extracted from a real dashcam video with hard negative mining to reduce the number of false positives.

Because all training images have the same dimensions, their HOG feature vectors have the same length and can be used to train an SVM. So, preprocessing is not necessary, however converting the images to certain color spaces could potentially increase the performance of the detector.

B. Feature Extraction

After the preprocessing stage, feature extraction is performed by computing the HOG descriptors of every preprocessed image from the labeled dataset. These descriptors are used to train and test a linear SVM.

C. Training a Linear SVM

This is 2-class classification problem, so the viable SVM type options that OpenCV offers are C_SVC and NU_SVC types. As they are very similar, we decided on using NU_SVC.

We were bound to use linear SVM because only the primal form of a linear SVM can be passed as an argument to the HOGDescriptor struct setSVMDetector function. This function allows the usage of the performant detectMultiScale function which this pipeline is trying to make use of.

D. Vehicle Detection

The last step is to perform vehicle detection on real dashcam data. For this purpose, we used the detectMultiScale function to detect vehicles in images and videos. This function performs a sliding windows search using windows with different sizes, so that vehicles with arbitrary dimensions can be detected in the input image or video.

III. Results

A. Color Space

To determine the color space that provides the best performance, experiments using various color spaces were performed, using constant values for the HOG and SVM parameters.

Table I gives comparison of the performance for 8 different color spaces based on the error as a percent of misclassified images from the test set when the images are in certain color space. The values of the HOG and SVM parameters used for the experiments are given in table II. These are some common default values for these parameters.

Oddly enough, BGR gives the best performance, or the least percentage of misclassified samples. This was unexpected because the BGR color space has strongly correlated channels and is non-perceptual. But, although it achieves the best performance during the testing phase, it performs poorly when detection is performed on real images and videos. We think that the reason is that real images and videos have a lot of background color variation, so using BGR which has strongly correlated channels will cause many misdetections. That is the case because HOG calculates separate gradients for each color channel and the one with the largest norm is taken as the pixel's gradient vector. LAB, LUV, YCrCb and YUV all achieve similar performance, and follow after BGR. Any of these 4 color spaces is a viable option because the difference between their performance is very small. We decided to use the YUV color space because it has the smallest error percentage in table I and it works well with our test data.

Table I
COLOR SPACE PERFORMANCE

Color Space	**Error**
BGR	2.5732%
GRAY	3.5867%
LAB	3.3531%
LUV	3.2573%
HLS	5.7742%
HSV	6.0923%
YCrCb	3.2404%
YUV	3.1757%

Table II
PARAMETERS TESTING VALUES

Parameter	**Value**
Detection window size	64×64
Block size	16×16
Block stride	8×8
Cell size	8×8
Bins	9
Discrete derivative mask size	1
Gaussian smoothing window parameter	-1 (means no smoothing)
Block normalization type	L2-Hys
L2-Hys threshold	0.2
Gamma correction	False
Signed gradient	False
Nu	0.09

B. HOG Parameters

HOG is the key component in the pipeline, so the choice of the values for the HOG parameters is a very important one. We experimented with different values for these parameters and found out that the ones suggested by Dalal and Triggs [1], shown in table II are performing reasonably well. The specific effects that the values of these parameters have on the performance i.e. the linear SVM testing error, are the following:

- Because the dataset images are 64×64 pixels, it is reasonable to use a 64×64 detection window size. Decreasing or increasing this size will decrease the performance because the images will contain less information, or it will make the descriptor more sensitive to noise, respectively.
- Block size, block stride and cell size depend on each other's value and influence the performance together, so they must be tested together. Table III shows the results of the experiments and gives insight about the effects of these three parameters on the performance. The interpretation of these results is that overlap of $3/4$ slightly improves the performance, however it drastically

Table III
HOG PARAMETERS PERFORMANCE

Block Size	Block Stride	Cell Size	Error
8×8	4×4	4×4	3.2751%
8×8	8×8	8×8	6.5991%
16×16	4×4	4×4	2.3341%
16×16	8×8	8×8	2.6858%
16×16	16×16	16×16	7.7984%
32×32	8×8	8×8	2.3649%
32×32	16×16	16×16	2.7703%
32×32	32×32	32×32	16.4527%

increases the time required for extracting the HOG features and training the SVM which makes it not worth it for such a small performance increase. $1/2$ overlap is enough.

- Discrete derivative masks of size 1 without Gaussian smoothing work best. Using larger masks decreases performance and smoothing decreases performance significantly. However, larger masks and Gaussian smoothing can be useful on real videos.
- Using more than 9 bins with unsigned gradients will not improve the performance significantly. Decreasing the number of bins decreases the performance. So, it is appropriate to use 9 bins for vehicle detection.
- The only block normalization type currently available in OpenCV is L2-Hys. The optimal L2-Hys threshold is 0.2, increasing or decreasing this threshold decreases the performance.
- Using gamma correction increases the error for around 0.5%, however it can be useful on real videos.
- Dalal and Triggs [1] used unsigned gradients for human detection in their paper, but they suggested that for some other tasks like vehicle detection sign information helps substantially. However, we found out that this is not the case. Signed gradients seem to cause overfitting of the SVM and cause false negatives on real images and videos. Using signed gradients also increases the time required for extracting features and training the SVM, because the feature vectors become longer (assuming the number of bins is 18) and therefore, the pipeline is slower. So, we think that one should stick to unsigned gradients for dashcam vehicle detection.

C. Linear SVM Parameters

The optimal Nu value was determined to be 0.09 with the trainAuto function using 10-fold cross-validation. However, although this value of the parameter minimized the testing error, we observed bad detection on real images and videos. After experimenting, we found out that for $Nu \approx 0.25$ the pipeline performs very good and the detections are very accurate. In this case the testing error is larger, but it improves the performance by significantly reducing the false positives on dashcam data. We assume that the vehicles and non-vehicles images from the dataset have more distinguishable HOG feature vectors, so smaller Nu value produces smaller error. On the other hand, sliding windows of a real image that contain or do not contain vehicles seem to have similar HOG feature vectors which requires larger misclassification cost or larger Nu value, thus reducing the number of false positives. So, the Nu value depends on the training data and the input images or videos.

D. detectMultiScale parameters

The optimal values of the detectMultiScale parameters depend on the input on which the detection is performed. The hitThreshold parameter is a threshold for the distance between the features and the SVM classifying plane. Setting its value too low will cause false positives and setting its value too high will lead to false negatives. In our experience, for best results the hitThreshold value should be between 1 and 2 depending on the finalThreshold value and the input image or video, when $Nu = 0.25$. The finalThreshold parameter is an NMS threshold. The function will classify a region as a vehicle if there are more positives in the region than this given threshold, otherwise it will classify the region as non-vehicle. Setting its value too low will cause false positives and setting its value too high will cause to false negatives. In our experience, for best results the finalThreshold value should be between 0.5 and 1.5 depending on the hitThreshold value and the input image or video, for $Nu = 0.25$. Figure 1 shows the results with hitThreshold set to 1.25 and finalThreshold set to 0.75, values that were found to give the desired results.

E. Real Data Performance

Although the described pipeline performs reasonably well on real dashcam images, its performance on real dashcam videos is not satisfying. It is unusual that there are lots of false positives in a video frame, while there are none in an image that seems identical to that video frame. This is the case because video frames contain artifacts which are caused by the application of lossy compression. HOG descriptors describe the shapes or the edges of an object, so the HOG feature vectors of those artifacts can be very similar to the ones of real vehicles. That will cause false positive misclassifications in unusual regions, like in the sky or the asphalt. As mentioned before, gamma correction, larger discrete derivative mask and Gaussian smoothing can be applied on videos to improve the detector's performance. However, vehicle detection in lossily compressed images and videos still suffers from false positives as they are not fully eliminated by tweaking these HOG parameters.

IV. DISCUSSION

The described approach performs reasonably well on images and achieves mediocre performance on videos. There are numerous proposed solutions for this problem. Unfortunately, tuning the HOG parameters is not one of them. Even if there are some optimal values for these parameters, performing detection on every video frame is computationally expensive. Bear in mind that those values will not be optimal for other videos. The right approach would be to implement vehicle

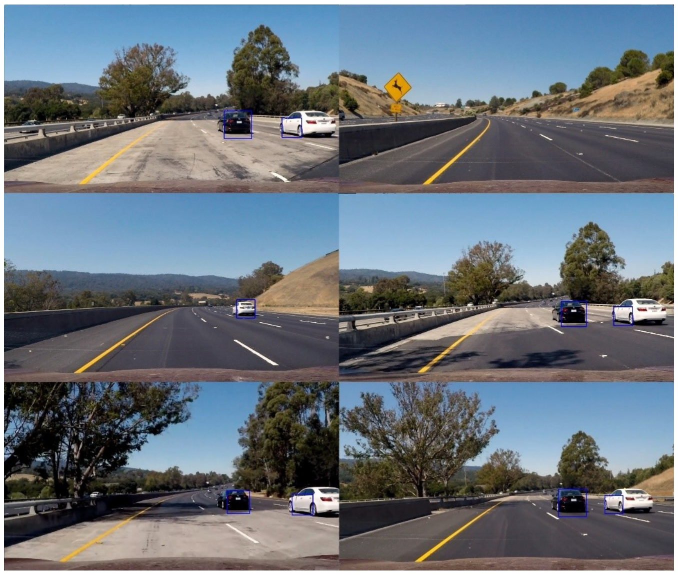

Figure 1. Vehicle detection in images with hitThreshold=1.25 and finalThreshold=0.75 with mean shift grouping

tracking, which is computationally inexpensive, rather than performing detection on every frame.

Some latest trends prefer using CNNs, RNNs or other deep learning approaches for these types of problems. We agree that these approaches are faster and more robust. However, our goal was to use classic image processing techniques instead of neural networks, because we wanted to get some perspective and experience in image processing and computer vision. So, although this approach cannot be used for real-time detection and is susceptible to false positives, it helps in understanding color spaces, image gradients, and SVM classifiers.

V. Conclusion and Future Work

We studied the influence of the color spaces on the performance and concluded that decorrelated and perceptual color spaces work best. We also studied the influence of the HOG parameters on the performance and concluded that in most cases, the optimal values of the HOG parameters for vehicle detection problems are the same as the proposed HOG parameters for human detection problems given in the original HOG paper [1]. HOG, SVM and detectMultiScale parameters are highly correlated and the choice of their values has profound effects on the performance of the detection. However, the performance of these parameters also depends on the input image or video and the goal should be finding the values for these parameters that generally work reasonably well with a lot of real images and videos and allow few misclassifications, instead of finding the perfect values for the parameters for only one image or video. The few misclassifications should be handled with other techniques.

We have shown that using HOG and linear SVM is a viable approach for vehicle detection in images, while it has some limitations for vehicle detection in videos. However, by using some simple techniques and extending the pipeline, this approach can easily overcome these limitations. Implementation of a vehicle tracking system is one of the future steps that will be considered for improving this pipeline. The false positives can be eliminated by checking whether the positive detections in a region are appearing in more consecutive frames. There are numerous HOG extensions and improvements that can be used, and SVMs with more complex kernels or modern and more sophisticated classification algorithms can be considered. These improvements can make the system resistant to artifacts and can provide overall better detection.

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *international Conference on computer vision & Pattern Recognition (CVPR'05)*, IEEE Computer Society, vol. 1, 2005, pp. 886–893.

[2] I. M. Creusen, R. G. Wijnhoven, E. Herbschleb, and P. de With, "Color exploitation in hog-based traffic sign detection", in *2010 IEEE International Conference on Image Processing*, IEEE, 2010, pp. 2669–2672.

[3] L. Mao, M. Xie, Y. Huang, and Y. Zhang, "Preceding vehicle detection using histograms of oriented gradients", in *2010 International Conference on Communications, Circuits and Systems (ICCCAS)*, IEEE, 2010, pp. 354–358.

[4] J. Arróspide, L. Salgado, and M. Camplani, "Image-based on-road vehicle detection using cost-effective histograms of oriented gradients", *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 1182–1190, 2013.

[5] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[7] G. Rossum, "Python reference manual", Amsterdam, The Netherlands, The Netherlands, Tech. Rep., 1995.

[8] G. Bradski, "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

[9] I. P. G. at UPM, *Gti vehicle image database*, https://www.gti.ssr.upm.es/data/, 2011.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset", *International Journal of Robotics Research (IJRR)*, 2013.