An Upgraded Object Detection Model for Enhanced Perception and Decision Making in Autonomous Vehicles.

Oshin Rawlley Shashank Gupta
Department of Computer Science and Information Systems
Birla Institute of Technology and Science, Pilani Rajasthan, India

p20200063@pilani.bits-pilani.ac.in, shashank.gupta@pilani.bits-pilani.ac.in

2022 IEEE International Conference on Communications Workshops (ICC Workshops) | 978-1-6654-2671-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCWORKSHOPS53468.2022.9814532

Abstract—**In Internet of Vehicles (IoV), accurate object detection is one of the basic requirements for Autonomous Vehicles (AVs) and Vision-based Self Driver Assistance System (VSDAS). With restricted processing power of sensor nodes and network bandwidth, enhanced object detection with low energy consumption and acceptable rate of accuracy is still a major issue that makes VSDAS untrustworthy and unsustainable.Considering this situation, in this paper, the authors propose an upgraded Object Detection model for simulating the enhanced perception and decision making in autonomous vehicles. In this paper, we evaluate the performance of three methods namely Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP), HAAR utilizing KITTI dataset. The results reveal that Haar exhibits higher detection rate than the other two methods. For the enhanced object detection, we utilize a frame similarity difference technique for filtering out the duplicate frames and generating key frames. Finally, an upgraded Haar-cascade classification algorithm is proposed for accurate and fast object detection. Our comprehensive experimental outcomes on the eminent publicly available dataset (KITTI) showed that our model not only significantly improves the performance of object detection however, also saves the energy consumption of edge devices.**

Index Terms—**AdaBoost, Internet of Vehicle (IoV), Latency Reduction, Haar-like Traits.**

I. Introduction

Object detection demonstrates a vital part in monitoring the environment and tracking various activities in surveillance system [1]. The emergent technologies like FC provides us with breakthrough mechanisms to deal with massive amount of video data. The recent state-of-the-art have significantly achieved better results in static scenarios, though it is still complex to deal with dynamic scenarios. Object detection is affected by varied factors comprising location, density of the traffic, static or moving state, light conditions etc. Moreover, IoV multimedia applications in real time surveillance require time-sensitive data-driven decision making along with faster processing and communication to aid in visual object detection , sentiment identification and analysis [2]. Conventional methods fail to achieve these objectives due to wide requirements of storage and computing resources. Therefore, it is essential to utilize the novel machine learning models for accurate object detection and implement enhanced training process on deployed edge nodes [3]. Recent state-of-the-art techniques usually focusses on fog-based vision systems for the IoAV environment from the object detection perspective. Blaschke et al. [4] suggested a template matching technique which was semi-automatic that accepts input seed points of the user for generation of a rectangular templates for reference. In earlier investigations, Ranft and Stiller [5] studied techniques of machine vision for varied tasks of smart AVs, including isolating image and mapping, understanding the driving scenario and classifying objects. Mukhtar et al. [6] investigated various techniques of 2D object detection for Driver Support Systems focusing on features like appearance and motion utilizing a conventional pipeline. Therefore, motivated by the challenges highlighted above, we propose an scalable low latency object detection algorithm on fog-based AV system for better localization of objects. The contributions of this study are summarized as follows:

- We proposed a low-latency scalable object detection method for AV using an enhanced Haar Cascade classification technique. We have implemented a coefficient and frame difference matching technique for filtering out the duplicate frames and obtaining the key video frames for enhanced object recognition.
- In our experiments, we utilize a renowned traffic data set to assess our proposed algorithm. Our algorithm attains the highest performance as opposed to the obsolete methods in the existing state-of-the-art.

II. System Model

A. Significant Video Frames Detection Techniques

It is imperative to choice out frames which have strong correlation between them [7]. For this purpose, we formulate a novel key frame selection algorithm which first localizes the object in 3d space and then does the comparison from two similar detected frames. This process intends to find the similarity score of the highly correlated frames for its consideration in the further steps. Figure 1 shows an illustration of the proposed system model of the object detection representing all the steps in the process.

1) Relevant object detection algorithm: With the aim to achieve useful video frames out of the long video data, our proposed object selection algorithm is essential. The selection of objects in a frame is very subjective

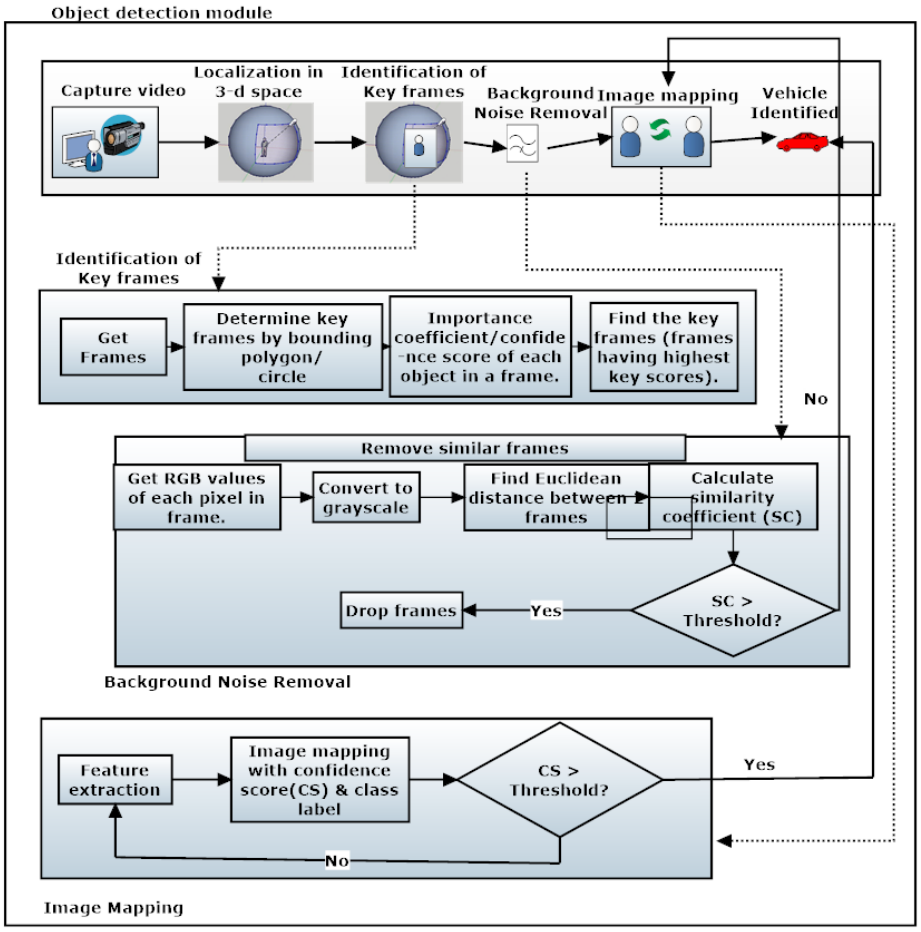

Object detection module
Capture video
Localization in 3-d space
Identification of Key frames
Background Noise Removal
Image mapping
Vehicle Identified
Identification of Key frames
Get Frames
Determine key frames by bounding polygon/ circle
Importance coefficient/confide-nce score of each object in a frame.
Find the key frames (frames having highest key scores).
No
Remove similar frames
Get RGB values of each pixel in frame.
Convert to grayscale
Find Euclidean distance between 2 frames
Calculate similarity coefficient (SC)
SC > Threshold?
Yes
Drop frames
Background Noise Removal
Feature extraction
Image mapping with confidence score(CS) & class label
CS > Threshold?
Yes
No
Image Mapping

Figure 1: System Design of Object Detection mechanism

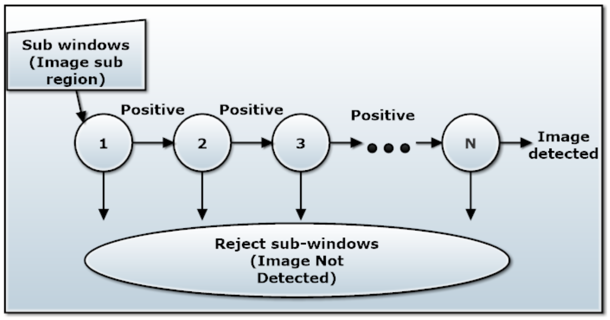

Sub windows (Image sub region)
Positive
Positive
Positive
1
2
3
N
Image detected
Reject sub-windows (Image Not Detected)

Figure 2: A schematic diagram of cascade classifier

to the quantity of the objects present in the video data. To make this idea more discreet, locating the object of interest in the video can be done with the thought as follows: Image which is appearing more prolonged, the greater coverage area it occupies in the frame, and therefore subjectively it will linger more centrally regarding it as an important object coming in the foreground. Keeping in this view, we take two parameters to conclude our relevant objects in the video frame. First parameter is based on the calculation of center point of the image while the second parameter is area coverage of the object in the image. The importance of the object is decided on the basis of its closeness to the center point of the image. The closer the object in the data is to the center point of the image, more weight it holds. To accomplish this task, we determine the weight of an object and subsequently perform the key score calculation of every object in the video. This will determine the relevant frames to be considered in accordance with the key score. **Procedure:** In a sample image, consider the center point of an object of interest O_I as $O_I(x_{OI}, y_{OI})$ and center point of the image as $I(x_i, y_i)$. We consider width & height of the image as w_d and h_t respectively. The weight coefficient of this object is labelled as W_c. The center point of the frame of the video has a negative correlation with the center of the object. The weight coefficient can be calculated with the following equation 1:

$$W_c = 1 - \frac{\sqrt{(x_{OI}-x_i)^2+(y_{OI}-y_i)^2}}{\sqrt{w_d^2+h_t^2/2}} \qquad (1)$$

From (1), we can interpret that range of weight coefficient is [0,1], maximum limit being 1. Here, maximum limit 1 signifies the image center & minimum limit 0 signifies the four corners of the picture. Subsequently, the key score calculation can be done as shown in equation 2 as follows: -

$$\sum_{i=0}^{N}\sum_{j=0}^{M}[W_c(i,j)*w_d(i,j)*h_t(i,j)] \qquad (2)$$

where parameters are defined as:
N: sum of all video frames
M: sum of all objects identified in j frames
$W_c(i,j)$: weight coefficient
$w_d(i,j)$: width of the object (i,j) $h_t(i,j)$: height of the object (i,j) The percentage of every identified object in the k number of objects can be formulated in equation 3 as follows:

$$Per = \frac{Key_{score}}{\sum_{i=0}^{k} Key_{score}(i)} \qquad (3)$$

The algorithm for recognizing key frames in a video chunk calculates Percentage, KeyScore, WeightCoefficient. The procedure Percentage calculates the key score proportion. Out of k objects, each detected object's proportion is calculated from this function. The proportion of each object detected in the frame is evaluated with the above Per formula. Then the procedure KeyScore determines the2 score of each object in the frame. It considers the center of the video frame and also the center of the object. The distance between both the parameters is calculated to obtain foreground objects. Subsequently, the procedure WeightCoefficient concludes the object's importance based on key score and percentage calculation. The output of the algorithm will be the set of important frames in the video.

2) Pixel difference comparison algorithm: For depicting the correlation between various pixels in an image, the image is transformed into bigger dimensional data set of pixels. Then, this high dimensional pixel data set is visualized in n-dimension space, i.e. Euclidean space. Algorithm 1 depicts the extent of difference of pixel values of two frames can be determined by equating the distance between two images in the n-dimensional space.To obtain the similarity index between the two images, we use Euclidean distance as a parameter for distance calculation in the following equation:-

$$dist(a,b) = \sqrt{(a_1-b_1)+(a_2-b_2)+...+(a_n-b_n)}$$

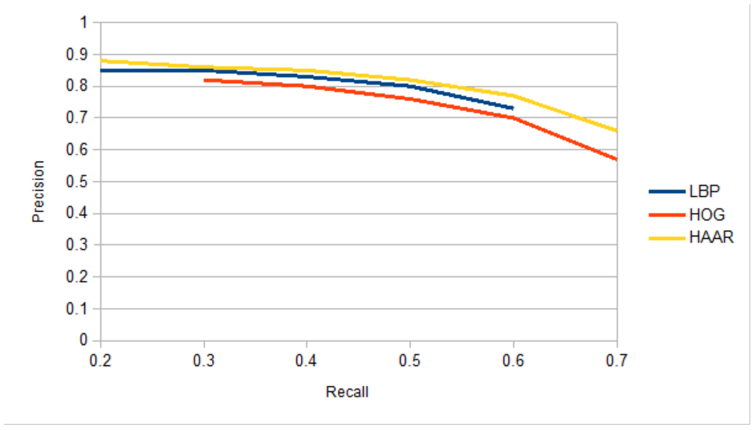
The maximum value of Euclidean distance can be determined by comparing the full white image with the full black image and therefore the Euclidean distance range can be evaluated as follows: The domain of Euclidean distance can be shown as $(0, bits_{max}\sqrt{(w_d*h_t)}$ where ,
$bits_{max}$: maximum value of bits per channel
w_d: breadth of picture
h_t: height of the picture

The normalization of distance is inversely correlated with similarity coefficient Sim_{coeff} shown in equation 4, therefore,

$$Sim_{coeff} = 1 - \frac{dist(a,b)}{bits_{max}\sqrt{w_d*h_t}} \quad (4)$$

B. Haar-like feature based object detection algorithm

A robust algorithm named Haar-like feature can be used to detect the objects [8]. It studies successively contiguous rectangular regions in a particular location inside a detection window. The intensity of these pixels in these adjacent regions are summed up for every



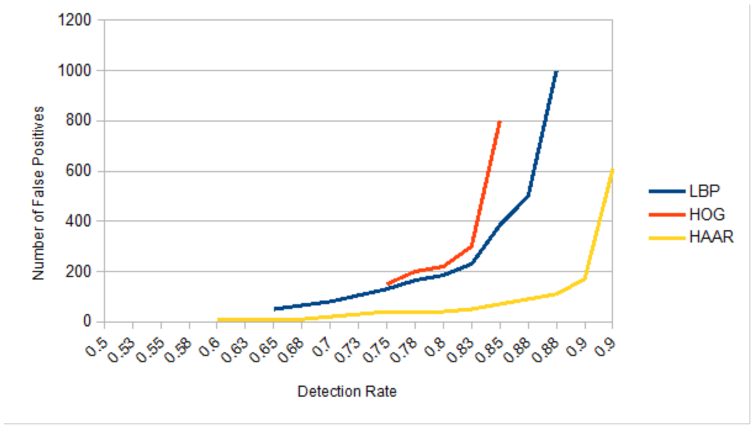


Figure 3: (a)Precision vs Recall, (b) Detection Rate vs Number of false positives

region. Computational design based on fast haar-like features is produced on the basis of two positions: basic erected rectangle and 45 degree rotated rectangles.

1) Haar-like traits algorithm: The prompt way to create features of haar-like attributes [9] is utilizing rectangular zones in various angles. An upright rectangle and a rectangle rotated at 45 degrees inside the window is used for generation of haar-like traits. The pseudocode of the algorithm is shown in Algorithm 2. The existence of an object in a frame can be determined by a window of measure (w_d, h_t) pixels. The intermediate haar-attribute set is the collection of total possible traits which can be expressed as follows in equation 5:
$Attribute_m =$

$$\sum_{i\epsilon m=1...N} RS(t) \qquad (5)$$

2) Multiple Cascading Classifiers: In our proposed algorithm we have considered the recognition rate of our model to be an important parameter for accurate object detection. We have constructed a robust classifier with relatively higher detection rate. It integrates multiple classifiers successively which is shown in the schematic diagram of cascade classifier structure in Figure 2. The training process ensures that the trained classifier

Algorithm 1: Pixel difference comparison

```
Input: Two Images $I_1$ and $I_2$
Output: A Boolean value stating if $I_1$ and $I_2$ are similar
Procedure IsSimilar(Image $I_1$, Image $I_2$):
    Load Images I1 and $I_2$;
    $Sim_{coeff}$, $w_d$, $h_t$, $bits_{max}$ ← Similarity Coefficient, width of I1, height of I1, maximum value of bits per channel respectively;
    // I1 and I2 are of same dimensions
    $list1$ ← GetPixelValue($I_1$);
    $list2$ ← GetPixelValue($I_2$);
    $gList1$ ← RGBtoGray($list1$);
    $gList2$ ← RGBtoGray($list2$);
    // now compute the Euclidean distance between the two images
    $N$ ← size of gList1 ;
    // $gList1$ and $gList2$ are of same size
    $d \leftarrow 0$;
    for ( $i = 0;\ i < n-1;\ i = i+1$ ) {
        $d \leftarrow d + \sqrt{gList1_i - gList2_i}$;
    }
    $dist(norm) \leftarrow dist(a,b)/bits_{max} \cdot \sqrt{w_d * h_t}$;
    $Sim_{coeff} \leftarrow 1 - dist(norm)$;
    if $Sim_{coeff}$ is very high then
        return true;
Procedure GetPixelValue(Image $I$):
    $N$ ← number of pixels in $I$
    $result$ ← an empty list of size $N$
    for ( $i = 0;\ i < n-1;\ i = i+1$ ) {
        $rgbval$ ← an empty tuple
        Store the RGB values of $pixel_i$ in $rgbval$;
        Append $rgbval$ to $result$;
    }
    return $result$;
Procedure RGBtoGray(List $L$):
    $N$ ← size of $L$
    $grayList$ ← empty list of size $N$
    for ( $i = 0;\ i < n-1;\ i = i+1$ ) {
        $grayList_i$ ← 0.299 * $L_{i0}$ + 0.587 * $L_{i1}$ + 0.114 * $L_{i2}$;
    }
    return $grayList$;
```

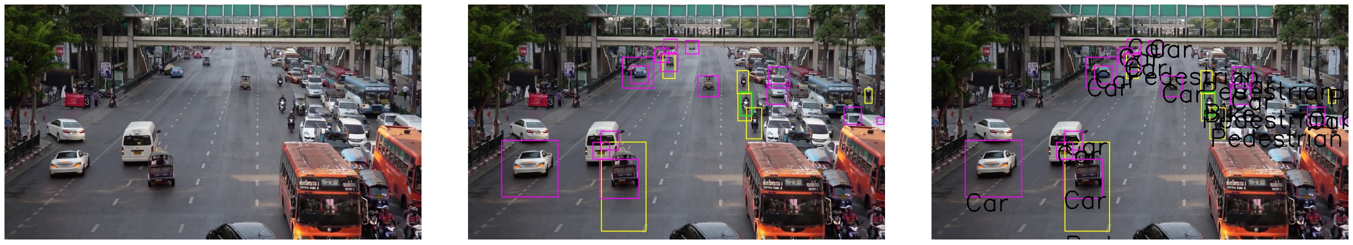
navigates the window and finds the object on the image in a sequential manner.

III. Implementation and Results

In this section, we conduct certain experiments using KITTI [10] dataset. All the experiments are conducted in the state-of-the-art computer vision library i.e. OpenCV of version 4.5.4 recognized for various video and image processing functionalities. We used the Ubuntu ver-

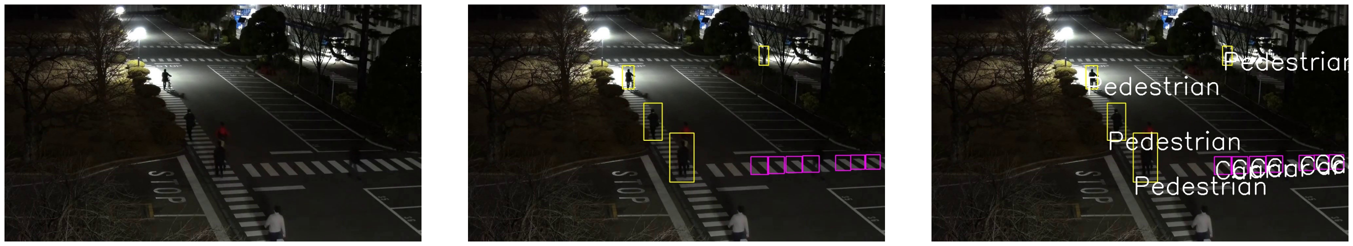
Algorithm 2: Haar-like trait algorithm

```
Input: Auxiliary gray scale image
Output: Sum of pixels of a rectangular sub-region of the image
Procedure RS(Tuple $t$, Image $AI$):
    $degree \leftarrow t{\cdot}degree$;
    if $degree = 0$ then
        return SumOfUprightRec ($t$, $AI$) ;
    else
        return SumOfRotatedRec ($t$, $AI$) ;
    end
Procedure SumOfUprightRec(Tuple $t$, Image $AI$):
    $p \leftarrow t{\cdot}p$, $q \leftarrow t{\cdot}q$, $w_d \leftarrow t{\cdot}\ w_d$ , $h_t \leftarrow t{\cdot}h_t$;
    $TA$ ← a 2-D matrix with no of rows and columns as $numRows$ and $numCols$ respectively;
    for ( $i = 0;\ i < numRows;\ i = i+1$ ) {
        for ( $j = 0;\ j < numCols;\ j = j+1$ ) {
            $TA(p, q) \leftarrow TA(p, q\text{-}1) + TA(p\text{-}1, q) + AI(p, q) - TA(p\text{-}1, q\text{-}1)$;
        }
    }
    $RS(t) \leftarrow TA(p\text{-}1, q\text{-}1) + TA(p{+}w_d\text{-}1, q{+}h_t\text{-}1) - TA(p\text{-}1, q{+}h_t\text{-}1) - TA(p{+}w\text{-}1, q\text{-}1)$;
    return $RS$ ;
Procedure SumOfRotatedRec(Tuple $t$, Image $grayscale_img$):
    $p \leftarrow t{\cdot}p$, $q \leftarrow t{\cdot}q$, $w_d \leftarrow t{\cdot}\ w_d$ , $h_t \leftarrow t{\cdot}h_t$;
    $numRows$, $numCols$ ← no of rows and columns in $grayscale\_img$ respectively;
    $RotTA$ ← a 2-D matrix with no of rows and columns as $numRows$ and $numCols$ respectively;
    // first pass from left to right and from top to bottom
    for ( $i = 0;\ i < numRows;\ i = i+1$ ) {
        for ( $j = 0;\ j < numCols;\ j = j+1$ ) {
            $RotTA(p, q) \leftarrow RotTA(p, q\text{-}1) + RotTA(p\text{-}1, q) + RotAI(p, q) - RotTA(p\text{-}1, q\text{-}1)$;
        }
    }
    // second pass from right to left and bottom to top
    for ( $i = 0;\ i < numRows;\ i = i+1$ ) {
        for ( $j = 0;\ j < numCols;\ j = j+1$ ) {
            $RotTA(p, q) \leftarrow RotTA(p, q) + RotTA(p\text{-}1, q{+}1) - RotTA(p\text{-}2, q)$ ;
            $RS(t) \leftarrow RotTA(p{+}w_d, q{+}w_d) + RotTA(p\text{-}h_t, q{+}h_t) - RotTA(p, q) - RotTA(p{+}w_d\text{-}h_t, q{+}w{+}h_t)$;
        }
    }
```



(a) Raw image (b) Object Localization (c) Object Detection

Figure 4: Scenario 1



(a) Raw image (b) Object Localization (c) Object Detection

Figure 5: Scenario 2

sion 20.04 having VGA compatible controller : Intel CorporationUHD Graphics 630.Training of the model was done by the positive images and testing is done on 10 samples of image.Figure 4 and 5 depicts results of object detection carried out by implementing our method. It shows three phases, the first phase showing the raw images of the KITTI dataset followed by second phase of object localization and lastly the third phase of object detection. We observed our method in different scenarios with varying illumination conditions.

1) Training with HOG: The training is conducted utilizing three classifiers types namely HOG, LBP, Haar and the respective results of detection are compared [**?**]. First the cascading object detector learns the HOG type feature. In this descriptor, the direction distribution of these particular gradients are utilized as features. This ensures capturing the overall shape of the object.

2) Training with LBP : Subsequently, the same dataset is utilized for AdaBoost training where parameters are trained using Local Binary Patterns (LBP) feature descriptor. Each pixel generates an 8-digit binary number on the basis of varying intensity values between the neighboring and the center pixel [11]. Further the binary patterns calculate the feature vectors where the feature indexes are the binary patterns and feature values are the quantity of the pixels holding that value. LBP method captures the localized patterns.

3) Haar Training: Lastly the Haar features are used as the feature type by the AdaBoost cascade training. They are termed as single value gained by the difference between pixel summation of black rectangle and pixel summation of white rectangle. Many features of Haar are generated which makes this method more time consuming as compared to former two feature types.

The mean average precision (mAP) is calculated for different objects over three methods such as Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG) and our proposed method Haar cascade algorithm. After the evaluation, Table III observes that Haar cascade coupled with AdaBoost cascading operation performs better than all three methods as we see it produces better precision results.

Furthermore, the detection rate test assesses the recall and precision of the classifier on different scale factor values. Table I shows that the classifier is tested over 10 image samples and 4 values of the scale factor. Every scale factor gave different detection rate. For best value of the detection rate, optimal scale factor value resulting in best performance of the classifier should be determined. According to the table scale factor= 1.1 value gives the best performance for our classifier. Table II depicts this value being utilized in the subsequent vehicle counting table.The Figure 3(a) depicts the improved precision value of the Haar-like trait algorithm in comparison to the other methods. Moreover,in Figure 3(b) the improvement of the detection rate is also witnessed with the quantity of false positives significantly reduced. The rate of detection is also signified as sensitivity or recall.

IV. Conclusion

In this paper, we suggested an improved and scalable object detection method for effective object detection. After observing the results obtained, we observe that that our proposed Haar-like traits algorithm significantly reduced the latency requirement in edge IoV system. Moreover, we aim to utilize the energy constraint devices by deploying edge devices so that timely service with adequate energy conservation can b done. mAP is used as a statistical measure for assessing the performance of our method which computed as 79.9% in our case. Experimental outcomes demonstrate that

Table I: Results of detection rate testing

Image	Scale Factor	Actual total count	True Positive	False Positive	False Negative	Recall	Precision
61	1.1	26	17	4	9	0.65	0.81
10	1.1	21	13	4	8	0.62	0.76
168	1.1	18	7	4	9	0.44	0.64
337	1.1	17	12	5	5	0.71	0.71
402	1.1	33	18	6	15	0.55	0.75
463	1.1	35	24	10	11	0.69	0.71
537	1.1	26	17	3	9	0.65	0.85
475	1.1	25	15	2	10	0.6	0.88
601	1.1	6	3	7	3	0.5	0.3
97	1.1	9	7	0	2	0.78	1
			Average:			**0.62**	**0.74**
61	1.3	26	4	1	22	0.15	0.8
10	1.3	21	8	2	13	0.38	0.8
168	1.3	18	3	1	15	0.17	0.75
337	1.3	17	5	1	12	0.29	0.83
402	1.3	33	10	3	23	0.3	0.77
463	1.3	35	7	2	28	0.2	0.78
537	1.3	26	12	1	14	0.46	0.92
475	1.3	25	7	0	18	0.28	1
601	1.3	6	2	4	4	0.33	0.33
97	1.3	9	1	0	8	0.11	1
			Average:			0.27	0.8
61	1.5	26	2	2	24	0.08	0.5
10	1.5	21	4	1	17	0.19	0.8
168	1.5	18	1	1	17	0.06	0.5
337	1.5	17	2	1	15	0.12	0.67
402	1.5	33	6	1	27	0.18	0.86
463	1.5	35	3	3	32	0.09	0.5
537	1.5	26	8	1	18	0.31	0.89
475	1.5	25	6	1	19	0.24	0.86
601	1.5	6	2	4	4	0.33	0.33
97	1.5	9	0	0	9	0	0
			Average:			0.16	0.59
61	1.7	26	1	4	25	0.04	0.2
10	1.7	21	1	0	20	0.05	1
168	1.7	18	0	1	18	0	0
337	1.7	17	2	0	15	0.12	1
402	1.7	33	2	0	31	0.06	1
463	1.7	35	3	0	32	0.09	1
537	1.7	26	6	0	20	0.23	1
475	1.7	25	4	0	21	0.16	1
601	1.7	6	1	0	5	0.17	1
97	1.7	9	0	0	9	0	0
			Average:			0.09	0.72

Table II: Results of vehicle counting in best image samples from Table I.

Image No.	TP	FP	FN	Recall	Precision
61	17	4	9	0.65	0.81
10	13	4	8	0.62	0.76
168	7	4	9	0.44	0.64
337	12	5	5	0.71	0.71
402	18	6	15	0.55	0.75
463	24	10	11	0.69	0.71
537	17	3	9	0.65	0.85
475	15	2	10	0.6	0.88
601	3	7	3	0.5	0.3
97	7	0	2	0.78	1
73	5	3	6	0.45	0.63
128	4	1	5	0.44	0.8
129	3	8	2	0.6	0.27
179	2	1	0	1	0.67
187	1	0	2	0.33	1
503	2	0	1	0.67	1
602	1	0	0	1	1
607	1	0	0	1	1
718	5	0	12	0.29	1
1708	1	1	5	0.17	0.5
	Average:			0.61	0.76

Table III: Mean average precision for different components over three methods

Classes	Haar %	LBP%	HOG %
Car	73.1%	68.9%	65%
Bus	88.8%	80%	82%
Pedestrian	65%	67%	61.3%
Bikes	93%	88.5%	85.4%
mAP	**79.9%**	76%	73.4%

our technique could outpace all recent methods and the proposed technique is effective in identifying the key frames for object detection. Our future investigation will focus on the study of deep-learning technologies with the integration of edge computing in the cloud-edge framework.

References

[1] Cheng Dai, Xingang Liu, Weiting Chen, and Chin-Feng Lai. A low-latency object detection algorithm for the edge devices of iov systems. *IEEE Transactions on Vehicular Technology*, 69(10):11169–11178, 2020.

[2] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868, 2019.

[3] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.

[4] Thomas Blaschke, Geoffrey J Hay, Qihao Weng, and Bernd Resch. Collective sensing: Integrating geospatial technologies to understand urban systems—an overview. *Remote Sensing*, 3(8):1743–1776, 2011.

[5] Benjamin Ranft and Christoph Stiller. The role of machine vision for intelligent vehicles. *IEEE Transactions on Intelligent vehicles*, 1(1):8–19, 2016.

[6] Amir Mukhtar, Likun Xia, and Tong Boon Tang. Vehicle detection techniques for collision avoidance systems: A review. *IEEE transactions on intelligent transportation systems*, 16(5):2318–2338, 2015.

[7] Asma Azim and Olivier Aycard. Layer-based supervised classification of moving objects in outdoor dynamic environment using 3d laser scanner. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1408–1414. IEEE, 2014.

[8] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. What could move? finding cars, pedestrians and bicyclists in 3d laser data. In *2012 IEEE International Conference on Robotics and Automation*, pages 4038–4044. IEEE, 2012.

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[11] Ashwin Arunmozhi and Jungme Park. Comparison of hog, lbp and haar-like features for on-road vehicle detection. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*, pages 0362–0367. IEEE, 2018.