ISSN No:-2456-2165

Class Wise Object Tracking Using Simple Online Real-Time Tracking

Amit Kumar^, Arpana Alka*, Jyoti Prakash Sahoo^, Nimai Chand Das Adhikari*
*Analytic Labs
^Eupep Technologies Pvt. Ltd

Abstract:- One of the most important problems every country is facing to solve is the immense addition of the traffic to increase burden if smooth management. This research revolves around solving and maintaining the traffic solution like congestion, idle road, emergency etc. by using the help of traffic surveillance camera. In this research, we are trying to make an efficient object tracking solution called as Class Wise Object Tracking Algorithm (CWOT). The conventional SORT (Simple Online Realtime Tracking) does not take class IDs of detected objects which the proposed methodology takes into account. This methodology describes the detection, accelerating speed of the detection, propagating object states into the future frames and then associating current detections with the existing objects using SORT (Simple Online Realtime Tracking). Then generated IDs via SORT are tagged to object detection class IDs using this algorithm. YOLOv3 is used for the different object detection in real-time. The algorithm performed fairly well on the dataset used from the BSCL (Bhubaneswar Smart City Limited, Odisha, India) and the accuracy attained was reported to be 98.95%. This CWOT achieved good accuracy and gave more insights than SORT usually does.

Keywords:- *YOLOv3, SORT, MOT, Object Detection, Multi-Object Tracking, Smart Traffic Management Systems.*

I. INTRODUCTION

Traffic is one of the most important words that make a commuter tense whether commuting to the office or for any urgent work. This has become one of the most important factors of any government body to take care of this existing and ever-increasing problem of traffic congestion. This has called for the implementation of the smart trafficking systems which will not only try to optimize the flow of the traffic in one particular direction by controlling the time of the green light rather to check for the history of the number of vehicles being allowed to pass in the part history with the very minimal congestion. From the statistica.com, the total number of vehicles being sold in the year 2016 has increased by 111% in comparison to the year 2015 and this is likely to follow a similar trend. Apart from the new vehicles that are present on the roads of the city Bhubaneswar, there are a lot of unfit vehicles which also present. According to an article published by the govt. of Odisha (India), in the year 2016, there were approximately 17,322 vehicles that were playing without the fitness certificate. This demands analysis of real-world data from traffic cameras. Despite the necessity, there are a limited number of researches that aim to analyze the real-world data from CCTV systems. Some of the works are the studies [1] and [8]. Bhubaneswar as being one of the growing smart cities in India, calls for the management of the traffic control has become a most important measure. In this research paper, we will be dealing with the controlling of the Bhubaneswar traffic system, a smart city of Odisha, India. We shall attempt not only to estimate the total traffic flow but more importantly analyze and track various classes of vehicles. With the growth of the economy and the population, the impact of the urbanization and modern living has increased the level of the annual cost that is bearded through the traffic jams. This not only impacts on the low efficiency on the transportation networks rather it results with immense loss of time, fuel and increase the burden of air pollution [13]. This, as already discussed has urged to create a more intelligent traffic forecasting and optimization system which not only deals with the traffic management rather also optimizes it to create a better environment. Hence, once of the major impact that is created is by the intelligent transport management [32].

II. LITERATURE REVIEW AND EXISTING WORK

In this section, we provide an overview of recent advances in the research topics related to our work, namely, object detection and multiple object tracking (MOT). Most of the object detectors are based on convolutional neural networks (CNN) and can be broadly divided into two classes: single-stage detectors and two-stage detectors. The single-stage detectors are normally fast and predict bounding boxes around the object together with classes within a single network pass. Some common examples of single-stage detectors are YOLO [6] and SSD [9]. These architectures work genuinely well where target objects take a considerable portion of frame or image. One example dataset is KITTI dataset [14], one of the most famous dataset on vehicles. Up until 2018, YOLOv2 architecture [15] provided good speed-precision trade-off and specialized in vehicle detection via anchors clustering, multi-layer feature fusion strategy and additional loss normalization. But thereafter YOLOv3 [16] even better results and the architecture was a bit larger than YOLOv2. A Darknet 53 was used for YOLOv3. A paper was written on YOLOv3 detection performed on KITTI and PascalVOC datasets by Moran Ju, Haibo Luo, Zhongbo Wang, Bin Hui and Zheng Chang[12]. In contrast to single-stage detectors, two-stage detectors first predict regions and then try to refine and classify each of them during the second stage. Early R-CNNs [17] used to utilize a straight forward approach: regions were generated using a selective search algorithm and then fed to the classification CNN. The speed of R-CNN is slow due to the selective search compute time and requirement to run heavy classifier per each region. To overcome the limitation Fast-RCNN was proposed [10]. Ross Girshick provided the whole image to the CNN instead of running CNN on every region. After that, he pooled regions of interest (ROI) from the last feature map. After replacing selective search in the Faster R-CNN [18] with tiny CNN, it further boosted the precision and speed of the detector. These were called region proposal networks.

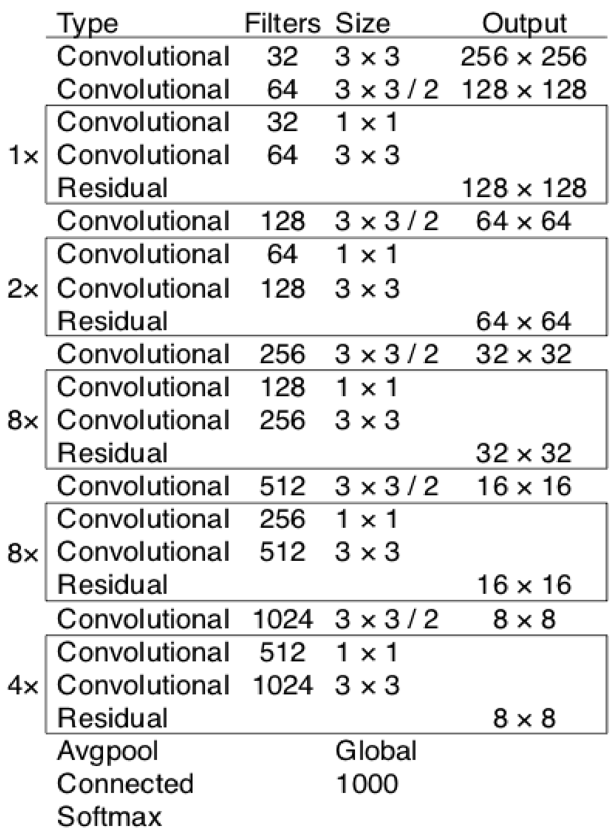
A review of the main advantages and disadvantages of single-stage detectors and two-stage detectors is provided in the work [11]. An estimation of traffic flow using Mask-RCNN and SORT was described in [19]. Conventionally Multiple Object Tracking MOT has been solved either using Multiple Hypothesis Tracking (MHT) [7] or the Joint Probabilistic Data Association (JPDA) filters [20, 2], which introduce a delay in making difficult decisions when there is high uncertainty over the object assignments. An appearance model for each target to prune the MHT graph was used by Kim et al. [3] to achieve state-of-the-art performance. A lot of online tracking methods aim at building appearance models either from the individual objects themselves [21, 22, 25] or a global model[23, 24, 4, 5] via online learning. On top of appearance models, motion is often incorporated to enhance associating detections to previously generated IDS [1, 23, 24, 4]. If one considers only one-to-one correspondences modelled as bipartite graph matching, globally optimal solutions such as the Hungarian algorithm [30] can be beneficial [26, 29]. The recent SORT algorithm [27] uses a simple Hungarian Algorithm along with Kalman filter and achieves good performance and speed in real-time applications. Deep SORT [28] further adds a pre-trained neural network to generate features for the objects, so that association could be made based on feature similarity. Some added advantages include minimal occlusion issue and re-discovering of objects.

III. METHODOLOGY

The proposed methodology describes about detection, accelerating speed of detection, propagating object states into future frames and associating current detections with existing objects using SORT, tagging generated IDs via SORT with YOLO classes.

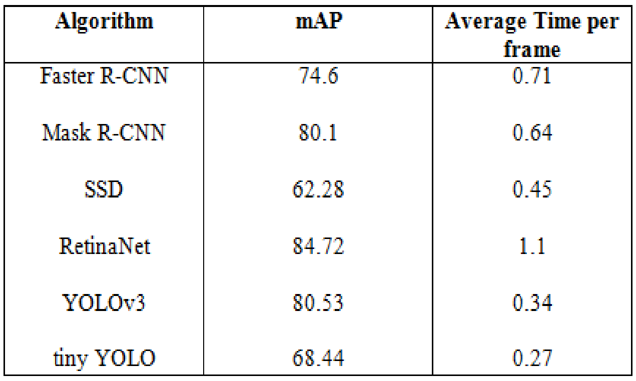
➢ *Detection*

The problem statement we are trying to solve demanded traffic flow estimation through a particular junction and also the estimation of each class of vehicle from that junction. The dataset of operation was the video data gathered from traffic surveillance overhead cameras placed by esteemed Bhubaneswar Smart City Limited (BSCL). Before testing on a given dataset, a simple experiment was done where all standard pre-trained algorithms on KIITI dataset. The results are mentioned in Table 1. Out of all standard algorithms, the selected algorithm was YOLOv3, the reason being the speed of algorithm was of utmost concern since the data was huge and a quick algorithm was required for estimation of traffic flow. Algorithms such as RetinaNet offered very good accuracy but proved to be extremely slow whereas algorithms like tiny YOLO performed very fast but with poor accuracy. The only algorithm which seemed to have a balanced trade-off was YOLOv3. YOLOv3 has some improvements in comparison to YOLOv2. One such instance is that it has a better feature extractor, the Darknet-53 [16] with some shortcut connections. Therefore it uses a much deeper network, 53 convolution layers. Architecture of Darknet-53 model is shown in Figure 1. It also has a better object detector with feature map up-sampling and concatenation. Re-training was done on BSCL dataset because the pre-trained YOLOv3 model cannot detect Indian vehicles such as auto-rickshaws, motor-bikes, etc. For our model to detect those objects, the annotation was done on new images and retraining was performed. Retraining was done on a Tesla K80 GPU. A lot of classes were detected byYOLOv3 but for experiment and calculation purpose, classes namely, “car”, “motor-bike”, “truck”, “bus” and “auto-rickshaw” were taken into account.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig 1:- Darknet-53 model [Source: medium.com]



Algorithm	mAP	Average Time per frame
Faster R-CNN	74.6	0.71
Mask R-CNN	80.1	0.64
SSD	62.28	0.45
RetinaNet	84.72	1.1
YOLOv3	80.53	0.34
tiny YOLO	68.44	0.27

Table 1:- Performance of various pre-trained algorithms on KITTI dataset

➢ *Speed of Detection of the Objects*

The inbuilt deep learning neural network module of OpenCV does not support GPU acceleration. Without GPU acceleration it was almost impossible to traverse through the vast magnitude of data. Hence PyTorch library was used. We can think PyTorch as Numpy arrays with strong GPU acceleration, but just instead of Numpy arrays, these are PyTorch tensors. PyTorch allows dynamic graph computation which can immensely affect the network behaviour. If CUDA is available on the system, PyTorch’s GPU implementation is quite possible. As part of the output, we get a tensor of dimension D x 8, where D is the true detection in all frames. The output is written on a CSV file and therefore, every detection corresponds to one row in the CSV file and every row has 8 columns. The 8 columns are namely index of the frame, 4 corner coordinates of the bounding box, the score of the class that has the maximum confidence and lastly, the index of the class.

➢ *Object Tracking*

For object tracking, the algorithm used was SORT (Simple Online Real Tracking),

- SORT uses a pragmatic approach to multiple object tracking (MOT) where the main focus is to associate objects efficiently for real-time applications [27]. If given the bounding box information for an object ID in frame 1, then normally there are two common methods to assign IDs in subsequent frames:
- Centroid based assignment: This is a simple form of assignment where IDs are assigned by looking into centroids. This is done by calculating centroids for each bounding box in frame 1. In frame 2, the new centroids are observed and based on the distance from previous centroids IDs are assigned by looking at relative distance. One obvious assumption in this method is that frame to frame centroids shall move a little distance. Drawback arises when objects in the frame are closely placed and therefore centroids of different bounding boxes are closely placed.
- Kalman filter: Kalman filter [31] is an improvement over previously discussed centroid based assignment. Kalman filter can be used in any place where there is **uncertain information** about some dynamic system, and an **educated guess** can be made about what the system is going to do next. Kalman filter allows tracking based on position and velocity using Gaussian. When it receives a new reading, it can assign measurement using probability and update itself. It is quite fast to run. Since it takes into account, both position and velocity of motion, it has better results than centroid based assignment.

SORT [27] uses the later, Kalman filter [31] for its object tracking. SORT approximates the inter-frame displacements of each object with a linear constant velocity which is independent of other objects and camera motion. The state of every target is modelled as follows:

$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T$

(1)

where u and v are horizontal and vertical pixel location of the centre of the target, s and r are the area or scale, and the aspect ratio of the target's bounding box. The aspect ratio is considered to be constant. When a detection is associated with one target, the detected bounding box is used to update the target state where the velocity components are solved using the Kalman filter framework. If there is no associated detection to a target, its state is simply predicted without correction using the velocity model. The correction information is obtained from associating objects in two adjacent frames. For instance, if A is detected within frame t, B is detected within frame t+1, then A and B are associated as the same object using IOU (Intersection Over Union) criteria. Kalman filter here uses the location of B as a new measurement for A to update the states. If no association is found then the corresponding ID is considered as lost. Now, the intermediate CSV file formed contains coordinates of bounding boxes which is parsed for application of SORT on detected objects. As discussed, SORT only uses the coordinates of the bounding boxes for tracking of an object. So, if we draw a line of interest in video frames and start tracking detected object, then with a little bit of coding we can count the total number of objects crossing the line of interest distinctively. In other words, if the ID of a detected object before the line of interest is same as that of the ID after the line of interest, then counter counting number of objects crossed increases by 1. But how to get the class of the detected object that crossed the line of interest?

➢ *Tagging Generated IDs with YOLOv3 Classes*

The generic SORT [27] algorithm assigns a unique ID by taking bounding box information as input as discussed earlier but leaves out a crucial attribute, i.e., the class labels generated by object detection. To get around this limitation a dictionary is maintained that maps SORT generated IDs to corresponding class labels from YOLOv3 [16]. Key in this dictionary is a unique ID generated by SORT and the value is a class label. To create this data structure we begin with an empty dictionary, let us call this ‘Lables_dict'. Object detection by YOLOv3 gives us a matrix whose rows represent bounding boxes with class names as discussed in previous sections. Let us call the matrix as 'YOLOv3_matrix'. We save this information before passing the YOLOv3_matrix to SORT tracker. The SORT tracker, in turn, returns a matrix, let us call it as 'SORT_matrix'. The SORT_matrix has data similar to YOLOv3_matrix but it does not have class labels, instead, it has unique IDs for all bounding boxes. We also maintain a list of unique IDs currently in use and IDs used in the previous frame. For each frame, we find out which IDs have been discarded between consecutive frames by comparing the current and previous unique ID lists. The IDs that are found to be discarded are then used to remove corresponding keys from the Lables_dict since they are no longer relevant. So, we loop over the SORT_matrix and try to do one of the two following options:

- Check if the unique ID exists in the Labels_dict and if it does return the id.
- In case the label does not exist in the dictionary, try to match the SORT_matrix row in YOLOv3_matrix. We found the unique ID and class label to Labels_dict.

The flow chart depiction of the algorithm is provided in Figure 2.

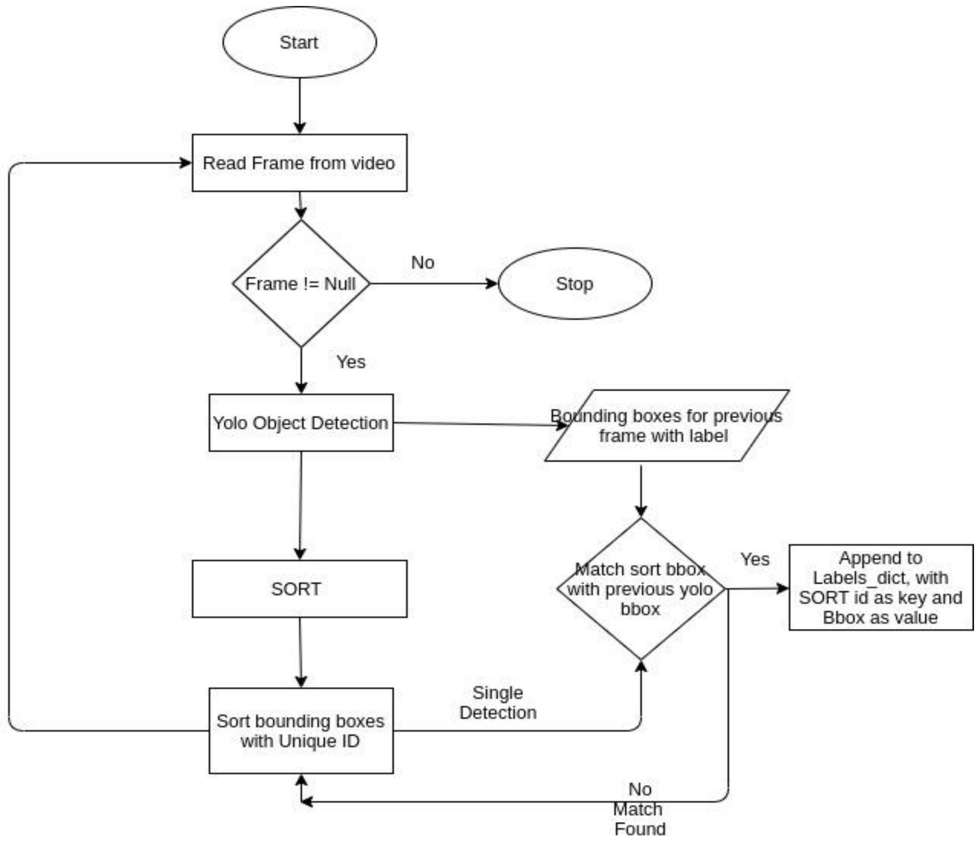

Start
Read Frame from video
Frame != Null
No
Stop
Yes
Yolo Object Detection
Bounding boxes for previous frame with label
SORT
Match sort bbox with previous yolo bbox
Yes
Append to Labels_dict, with SORT id as key and Bbox as value
Sort bounding boxes with Unique ID
Single Detection
No Match Found

Fig 2:- Algorithm for tagging SORT IDs to Class IDs

➢ *Experiments*

First of all, a line of interest was drawn on video frames just before the junction. In this case the line of interest was placed at ((0.1*W),(0.45*H)),((0.73*W),(0.45*H)). PyTorch implementation of YOLOv3 was applied on the video data and an intermediate CSV was generated where each row signified detection and each row had 8 columns. The CSV was parsed to SORT for tracking of detected objects. Two sets of output data were obtained. One data was obtained that gave the total number of vehicles which crossed the line of interest. This is that data which is obtained by aggregating all those detected objects whose ID is the same before and after the lime of interest. This data does not require tagging of class IDs to unique IDs Let us call it as $total_{vehicle}$. The other set of data eliminates the unique ID and maps it to detected class. Then aggregation of class-wise data is obtained. Let us call it as $count_i$. The accuracy and error of this computation is defined by:

$$accuracy = \frac{1}{total_{vehicles}} \sum_{i=0}^{n} count_i \qquad (2)$$

$$err = \frac{total_{vehicles} - \sum_{i=0}^{n} count_i}{total_{vehicles}} \qquad (3)$$

here, i represents the class numbers.

The experiment was done on various lengths of video and results are shown in Table 2. We can see in Figure 3, before the application of the proposed algorithm, SORT assigns a unique ID to detected objects irrespective of class IDs. So estimating the total number of vehicles passing the line of interest is possible but the estimate of various classes of vehicles is not possible. Whereas in Figure 4, 5 and 6, we can see class names being assigned as well to the detected objects after including discussed algorithm along with SORT. This allows us to analyze distinctly the number vehicles belonging to different classes crossing the line of interest. The counter on top in both figures represents the total number of vehicles crossing the line.

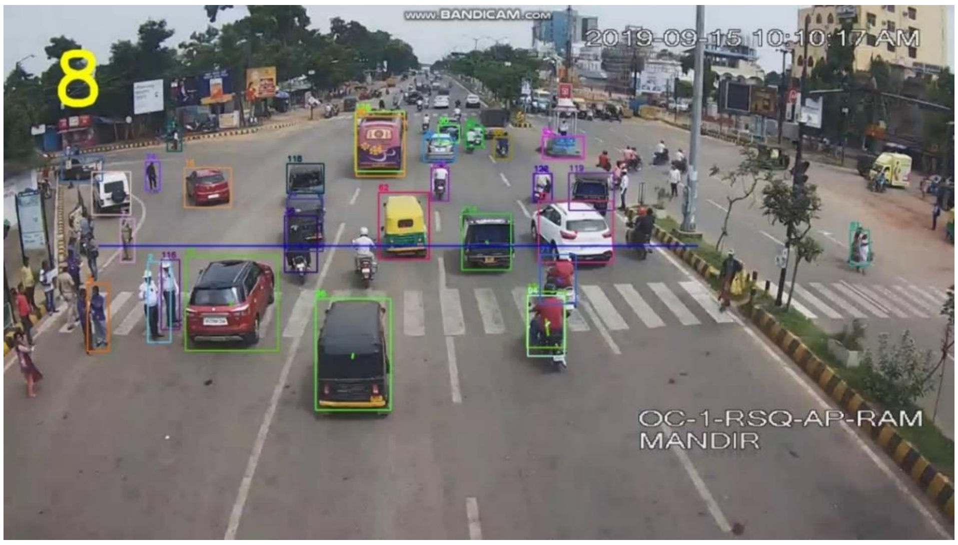


Fig 3:- Unique ID being assigned by SORT, courtesy: Bhubaneswar Smart City Limited

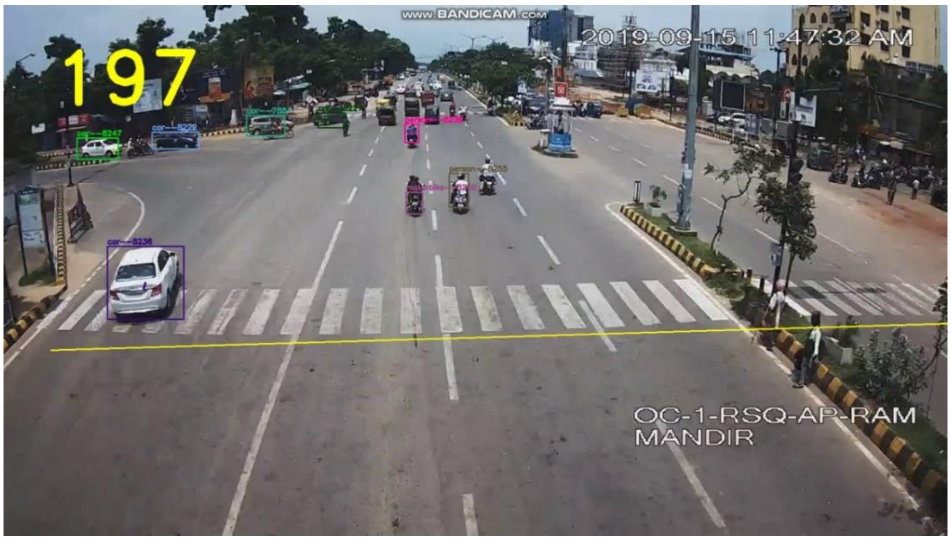


Fig 4:- Parsing class names using the proposed algorithm, courtesy: Bhubaneswar Smart City Limited

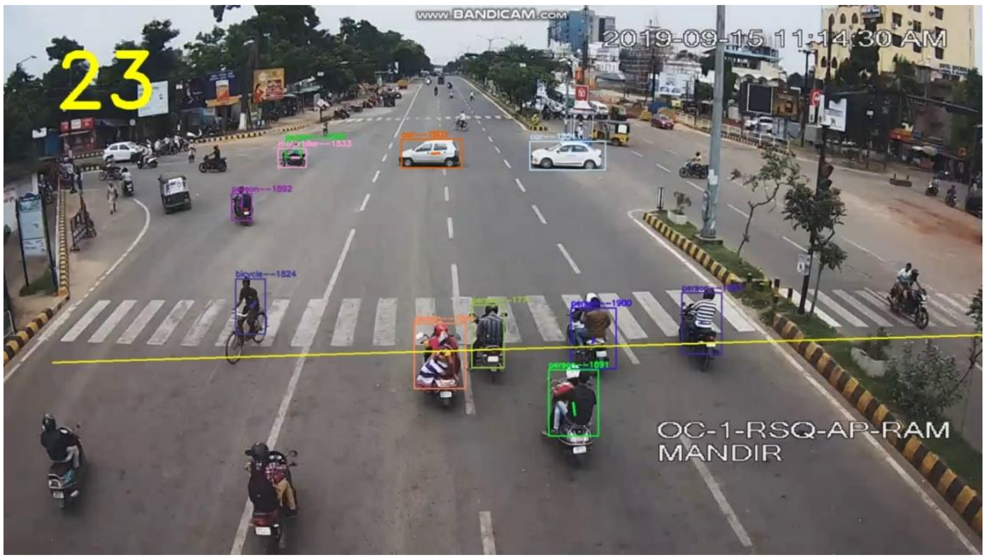


Fig 5:- Parsing class names using the proposed algorithm, courtesy: Bhubaneswar Smart City Limited

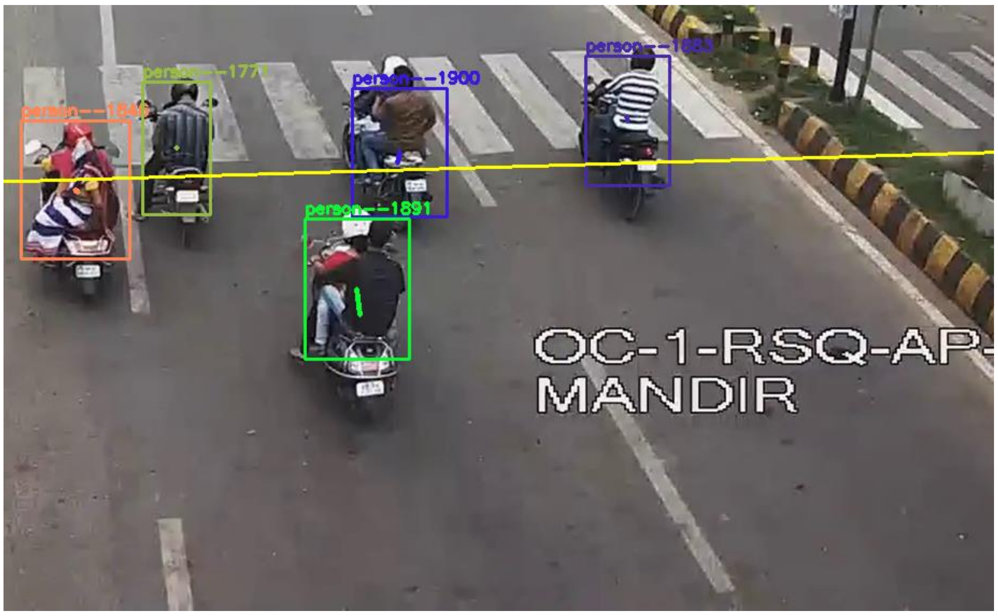


Fig 6:- Zooming in on Figure 3, courtesy: Bhubaneswar Smart City Limited

IV. RESULTS AND DISCUSSION

The speed of detection of objects was boosted by using PyTorch on a Tesla K80 GPU and resultant speed of processing was at 15 fps. After performing the proposed experiment on various lengths of videos, namely 30 seconds, 60 seconds, 120 seconds, 300 seconds and 600 seconds, we observe the discussed algorithm performed fairly well. The average accuracy found was 98.95%. Graphical Analysis between actual vehicle count and observed vehicle count on various video lengths is shown in figure 7. We can observe that there is not a lot of variation between actual vehicle count and vehicle count obtained by applying our algorithm. Plot for respective percentage errors is shown in figure 8. In real-world applications such as our case, traffic cameras record entire day's activity in the duration of five, ten, twenty or thirty minutes. Owing to this fact the small error rate is well within tolerable limits and gives fairly good insights into individual classes of detected objects. This proved out to be an important feature for our model because it helps us in counting and analyzing vehicles on Indian roads with a great deal of accuracy. Some important observations and notes are as follows:

- The approach described takes at least one frame to start assigning class labels since initially the Labels_dict, discussed in previous sections, is empty. So, if we start analyzing a video in the first frame we will get only IDs in the first frame. No mapping would be present.
- Another scenario where to consider is when our object detection algorithm misses an object for even a single frame. If objects are missed for even a single frame it means it will take at least 2 more frames to assign a class label in a best-case scenario. If objects are missed in multiple frames then assigning class labels becomes more and more difficult.
- One practical scenario is that when the video has glitches. That makes assignment of class labels a bit difficult. In real life scenario, such as the BSCL Dataset, glitches are almost inevitable.

Sl No.	**Duration in seconds**	**Car**	**Motorbike**	**Bus**	**Truck**	**Auto-Rickshaw**	**Total Vehicles (Actual)**	**Error (%)**
1	30	3	4	0	0	3	10	0
2	60	8	6	1	0	7	22	0
3	120	7	26	2	0	12	48	2.08
4	300	38	78	1	1	20	140	1.43
5	600	154	184	3	0	53	401	1.74

Table 2:- Results with the proposed algorithm on various lengths of videos

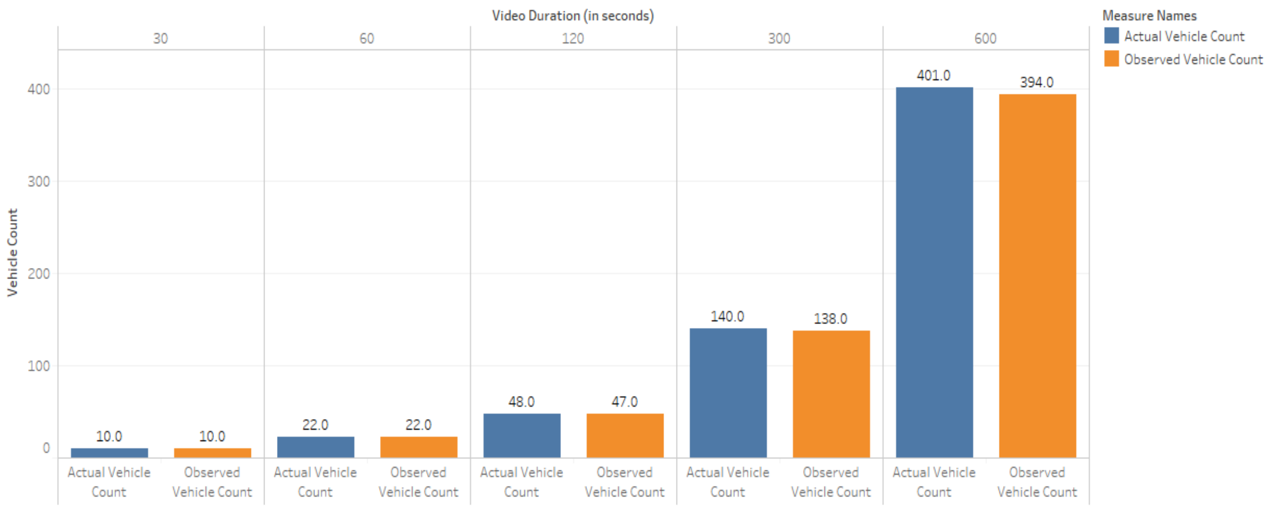


Fig 7:- Analysis between actual and observed vehicle count.

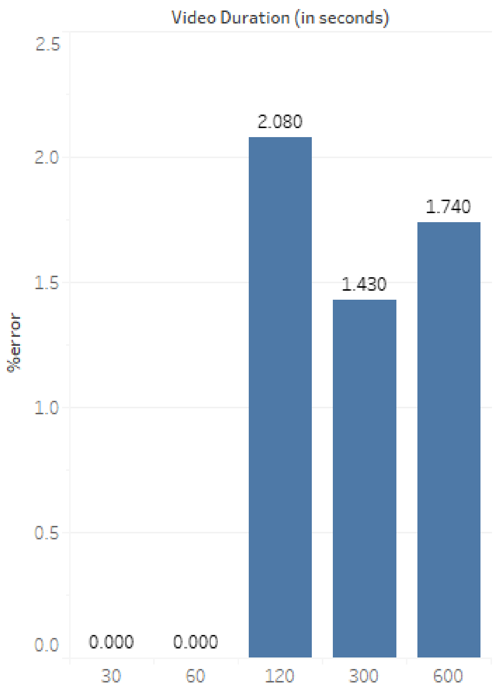


Fig 8:- Percentage Error at different video lengths.

V. CONCLUSION

In this paper, YOLOv3 was implemented using PyTorch for detection and SORT was used for tracking. A simple and easy to implement algorithm was discussed which takes class IDs or class names into account while tracking. This helps to get analytics about which type of vehicle crosses how many times in a given time frame especially in the given dataset. This algorithm provides a base structure and modifications can be done on it in future. The algorithm should work well with any tracker that takes bounding box information as input and gives bounding box information as well as a unique tracking ID as output. As part of our future work, we can try to implement this algorithm in other tracking frameworks like DeepSORT (SORT with a deep association metric) [28] which have minimum occlusion and object re-identification occurs. Also, we would work on reducing the number of frames that our algorithm takes to start operating to one frame which is currently two frames.

REFERENCES

[1]. Zhang S, Wu G, Costeira JP, Moura JM. FCN-rLSTM: Deep Spatio-temporal neural networks for vehicle counting in city cameras. In: Proceedings of the IEEE international conference on computer vision 2017-Oct, 2017.

[2]. S. H. Rezatofighi, A. Milan, Z. Zhang, A. Dick, Q. Shi, and I. Reid, “Joint Probabilistic Data Association Revisited,” in International Conference on Computer Vision, 2015.

[3]. C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, “Multiple Hypothesis Tracking Revisited,” in International Conference on Computer Vision, 2015.
[4]. J. H. Yoon, M. H. Yang, J. Lim, and K. J. Yoon, “Bayesian Multi-Object Tracking Using Motion Context from Multiple Objects,” in Winter Conference on Applications of Computer Vision, 2015.
[5]. A. Bewley, L. Ott, F. Ramos, and B. Upcroft, “ALExTRAC: Affinity Learning by Exploring Temporal Reinforcement within Association Chains,” in International Conference on Robotics and Automation. 2016, IEEE.
[6]. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). 2016, p. 779–88. IEEE.
[7]. D. Reid, “An Algorithm for Tracking Multiple Targets,” Automatic Control, vol. 24, pp. 843–854, 1979.
[8]. Peppa MV, Bell D, Komar T, Xiao W. Urban traffic flow analysis based on deep learning car detection from CCTV image series. Int Arch Photogramm Remote Sens Spat Inf Sci, 2018.
[9]. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC. SSD: single shot multibox detector. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics) 9905 LNCS. 2016.
[10]. Girshick R. Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision 2015 Inter, 2015.
[11]. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K. Speed/accuracy trade-offs for modern convolutional object detectors. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR).
[12]. Moran Ju , Haibo Luo, Zhongbo Wang, Bin Hui, Zheng Chang : The Application of Improved YOLO V3 in Multi-Scale Target Detection, MDPI 2019
[13]. Du, Shengdong, et al. "A hybrid method for traffic flow forecasting using multimodal deep learning." *arXiv preprint arXiv:1803.02099* (2018).
[14]. Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun, “Vision meets Robotics: The KITTI Dataset” in The Intesrnational Journal of Robotics Research, 2013
[15]. Sang J, Wu Z, Guo P, Hu H, Xiang H, Zhang Q, Cai B. : An improved YOLOv2 for vehicle detection. Sensors. 2018;18(12):4272.
[16]. Joseph Redmon, Ali Farhadi : YOLOv3: An Incremental Improvement, Cornell University. 2018.
[17]. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2012
[18]. Ren S, He K, Girshick R, Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell. 2017.
[19]. Fedorov Aleksandr, Nikolskaia, Kseniia, Ivanov Sergey, Shepelev Vladimir, Minbaleev Alexey. In: Traffic flow estimation with data from a video surveillance camera, Springer.2019
[20]. Y. Bar-Shalom, Tracking and data association, Academic Press Professional, Inc., 1987.
[21]. S. H. Bae and K. J. Yoon, “Robust Online Multi-Object Tracking based on Tracklet Confidence and Online Discriminative Appearance Learning,” Computer Vision and Pattern Recognition, 2014.
[22]. Y. Min and J. Yunde, “Temporal Dynamic Appearance Modeling for Online Multi-Person Tracking,” oct 2015.
[23]. A. Bewley, V. Guizilini, F. Ramos, and B. Upcroft, “Online Self-Supervised Multi-Instance Segmentation of Dynamic Objects,” in International Conference on Robotics and Automation. 2014, IEEE.
[24]. W. Choi, “Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor,” in International Conference on Computer Vision, 2015.
[25]. Y. Xiang, A. Alahi, and S. Savarese, “Learning to Track : Online Multi-Object Tracking by Decision Making,” in International Conference on Computer Vision, 2015.
[26]. A. Perera, C. Srinivas, A. Hoogs, and G. Brooksby, “Multi-Object Tracking Through Simultaneous Long Occlusions and Split-Merge Conditions,” in Computer Vision and Pattern Recognition. 2006, IEEE.
[27]. Bewley A, Ge Z, Ott L, Ramos F, Upcroft B. Simple online and realtime tracking. In: 2016 IEEE international conference on image processing (ICIP)
[28]. Nicolai Wojke, Alex Bewley, Dietrich Paulus, “Simple online and realtime tracking with a deep association metric,” in 2017, IEEE
[29]. A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, “3D Traffic Scene Understanding from Movable Platforms,” Pattern Analysis and Machine Intelligence, 2014.
[30]. H. W. Kuhn, “The Hungarianmethod for the assignment problem,” Naval Research Logistics Quarterly, vol. 2, 1955.
[31]. R. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” Journal of Basic Engineering, vol. 82, no. Series D, pp. 35–45, 1960
[32]. Vlahogianni, Eleni I., John C. Golias, and Matthew G. Karlaftis. "Short-term traffic forecasting: Overview of objectives and methods." *Transport reviews* 24.5 (2004): 533-557.