**Comparative Evaluation of Vehicle Direction and Motion Detection Methods for Multi-layer Contiguous Virtual Layer (MCVL)**

\*Manipriya Sankaranarayanan, P. Rupesh, S.V.S Apparao, and O Khadhar BashaDepartment of Computer Science and Engineering, Indian Institute of Information Technology Sri City, Chittoor, Andhra Pradesh, India  
[\*](mailto:*manipriya.s@iiits.in)[manipriya.s@iiits.in](mailto:*manipriya.s@iiits.in), [rupesh.p21@iiits.in](mailto:rupesh.p21@iiits.in), [apparao.s21@iiits.in](mailto:apparao.s21@iiits.in), khadarbasha.o21@iiits.in

\*Corresponding Author

**Abstract.** Accurate vehicle detection from traffic videos is essential for efficient traffic management, enabling real-time monitoring, congestion analysis, and traffic flow optimization. This research work introduces a novel framework, Multi-layer Contiguous Virtual Layer (MCVL), employing heuristic techniques to enhance detection accuracy and operational efficiency in urban mobility solutions. To evaluate MCVL's effectiveness in vehicle detection, this paper conducts a comparative analysis of three distinct methods: optical flow, blob tracking, and YOLO sort, for detecting vehicle movements in traffic videos. While existing research predominantly relies on deep learning for motion boundary detection, these approaches often entail extensive training on large datasets, leading to high computational costs and increased complexity. To mitigate these challenges, this paper proposes three alternative algorithms that leverage existing methods, avoiding the need for intensive training phases while maintaining robust performance in real-world traffic scenarios.

**Keywords:** Vehicle detection**,** Multi-layer Contiguous Virtual Layer**,** Direction of motion, Heuristic Techniques.

**1 Introduction**

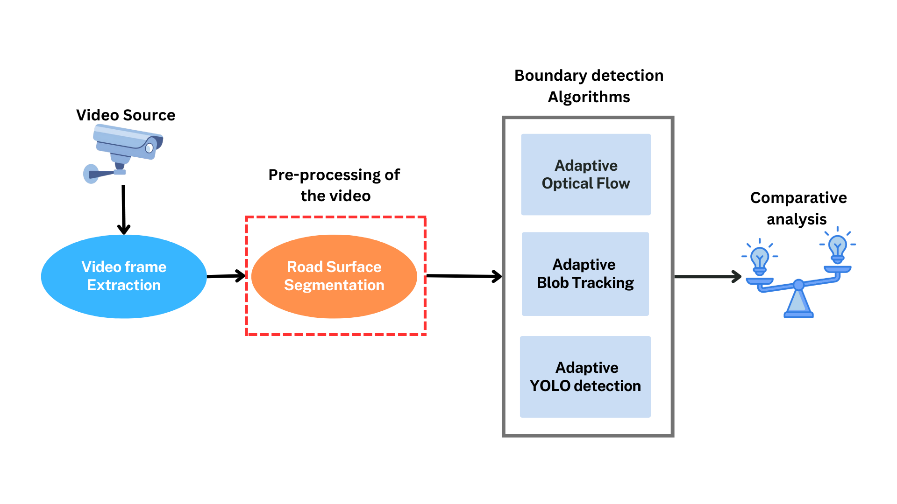
The detection and tracking of vehicle movements are fundamental to a wide variety of applications, such as traffic management, autonomous driving, and urban planning. Accurate vehicle tracking facilitates real-time traffic monitoring and control, enhancing road safety. In autonomous driving, it is critical for navigation and collision avoidance, ensuring smooth and safe vehicle operations. Urban planners rely on detailed vehicle movement data to design and optimize road networks, reduce congestion, and improve overall urban mobility. Furthermore, effective vehicle tracking supports the development of intelligent transportation systems, which integrate various technologies to create smarter and more responsive traffic solutions. The importance of these applications highlights the necessity for more advanced methods in this area to ensure effectiveness and efficiency.  
  
Traditional methods for vehicle detection frequently depend on deep learning techniques, which are known for their high accuracy. However, these methods demand for computational resources, and powerful hardware to process the complex algorithms. Additionally, they require extensive training datasets to learn effectively, which involves collecting, processing vast amounts of data. This process also consumes significant storage space to manage these large datasets and the resulting models. These requirements pose substantial challenges, especially in real-time scenarios where quick processing is essential and in resource-constrained environments where computational power and storage are limited. As a result, implementing deep learning-based vehicles movement detection in such settings becomes difficult, necessitating the exploration of more efficient alternatives. Given the complexities inherent in traffic environments, robust vehicle detection frameworks face significant challenges. Addressing these limitations is crucial for effective traffic management. Hence, this work introduces a novel approach, the Multiple Contiguous Virtual Layer (MCVL), designed to overcome the diverse challenges present in traffic video scenes. Building upon earlier research [1, 2], this paper extends and enhances the MCVL framework, significantly improving detection accuracy and overall application performance. The adaptability and efficacy of MCVL can be dynamically adjusted based on the positioning and configuration of frameworks within traffic videos, underscoring its versatility. To further enhance understanding and practical application, this work rigorously evaluates three distinct algorithms Optical Flow, Blob Tracking, and YOLO—implemented using the OpenCV library. These all are pre-existing algorithms and are chosen due to the advantages they provide in solving the problem statement.

Optical flow [7] is widely used method for motion estimation. This method is conceptually straightforward and easy to implement, which makes it accessible for many applications. Also, it is computationally less expensive and more robust to local variations compared with other algorithms like Horn-Schunck Method [18], Farneback Method (Dense optical flow) [19]. Blob tracking [14] can be less sensitive to noise in the image compared to feature-based methods and require less computational power compared to more sophisticated tracking methods like deep learning-based trackers [20]. YOLO [9] is extremely fast because it predicts bounding boxes and class probabilities directly from full images in a single evaluation, unlike methods like R-CNN [21] which require multiple evaluations. SORT [11] is designed to be fast and efficient, making it suitable for real-time applications. Despite its simplicity, SORT provides a strong baseline performance for multi-object tracking tasks. All these algorithms don’t directly solve the problem statement. So, adaptive approaches are implemented in a heuristic way by using these existing methods to approach the solution of the problem.  
These algorithms specifically address challenges in vehicle movement and direction detection, offering insights into their comparative effectiveness within the MCVL framework. Unlike traditional deep learning approaches, these algorithms do not require extensive training phases, thereby significantly reducing computational complexity and memory usage. This innovative approach aims to provide efficient and practical solutions that can be readily applied across various domains within intelligent transportation systems.

The subsequent sections discuss the following: Section 2 discusses the detail of the task done for vehicle direction and motion detection. The subsections also include the details of Multi-layer Contiguous Virtual Layer (MCVL), pre-processing modules and the three different techniques used for MCVL. Section 3 presents the comparative analysis on benchmark datasets followed by conclusion.

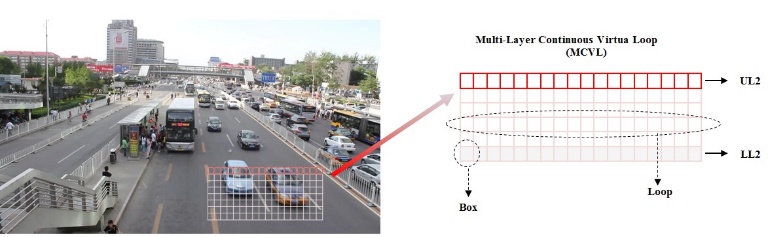
**2 Vehicle Direcion and Motion Detection Methods for MCVL**

This section describes the methods and heuristic techniques used for implementing and analyzing approaches to vehicle movement detection. The detail of the entire work flow for the analysis is presented in Fig.1. According to Fig. 1, the video data is first inputted. Individual frames are then extracted from the video. Next, the road surface area is extracted using road surface segmentation by evaluating the initial number of frames. A binary mask of the road surface area is generated, which helps remove the static background and focus on the road region, reducing computational complexity. Afterward, three different approaches are implemented using the binary road mask to detect road boundaries for different directions of motions directions. Finally, the predictions are evaluated by manually calculating the actual boundaries and generating error rates to compare the approaches, which is discussed in the results section.

  
**Fig. 1.** Overall Workflow for Comparative Evaluation of Vehicle Direction and Motion Detection

The entire work for vehicle direction and motion detection are proposed for using it in Multi-layer Contiguous Virtual Layer (MCVL). The videos captured by traffic cameras undergo processing using an innovative framework known as Multiple Contiguous Virtual Layer (MCVL) [22]. This framework integrates image processing methodologies and techniques that achieve a balance between precision and computational overhead, all while ensuring robustness.

MCVL comprises multiple layers of contiguous segments arranged in a grid-like structure. Each segment is delineated by a bounding box, forming a continuous loop of interconnected boxes. Pixelwise processing is executed across all individual grids within the MCVL. Given the extensive coverage of road traffic surveillance cameras, videos typically encompass multiple lanes and diverse directions of traffic flow. Using MCVL, the complexity of the scene is reduced, thereby optimizing processing efficiency and potentially lowering false positives [22]. Fig. 2 illustrates a detailed schematic of the MCVL design and its operational framework.



**Fig. 2** Multi-Layer Contiguous Virtual Layer

The initial task to implementing MCVL is to determine the boundaries of the region of interest to solve the problem of detection. The comparative analysis in this paper helps identify the best method for determining the boundaries of the region of interest, advancing the problem-solving process. The detail of the process are discussed in the forthcoming modules and presented in Fig.3.

**2.1 Video Frames Extraction**

To implement all these approaches the first step is to extract the frames from the given highway or traffic video. The video is first captured using the function VideoCapture(),which is inbuilt function is OpenCV library. Later the frames are extracted using the read() function and used for further processing **2.2 Road Surface Segmentation- Pre-processing Module**

Road surface segmentation focuses on extracting the road surface from few initial video frames by excluding background elements. Gaussian blur [6] is first applied to reduce noise and enhance image clarity. Frame differencing [5] and thresholding are then used to highlight changes between consecutive frames, identifying moving objects. The Background Subtractor MOG2 [3,4] algorithm further refines this by separating moving objects from the static background. Combining these methods with bitwise-AND operations generates accurate binary masks, which are aggregated using a bitwise-OR operation to produce a binary road surface mask [1,2].

**2.3 Boundary Detection Algorithms for Vehicle Directions**

Three different adaptive algorithms including optical flow, blob tracking and YOLO sort are discussed in this section for identifying the boundaries of different directions of motion of traffic video for MCVL.

**2.3.1 Adaptive Optical Flow Algorithm for MCVL**

Optical flow [8] is a powerful method for estimating motion by analyzing the apparent movement of objects between consecutive frames. This section outlines modified algorithm of classic Lucak Kanade optical flow for tracking vehicles using optical flow, highlighting key steps such as key point estimation, consistency checks, trajectory formation, and direction vector analysis.

Road surface segmentation [1,2] to the initial 250 frames to get the binary road surface mask is applied. The corner points in the initial frame using *cv2.goodFeaturesToTrack()*, serving as keypoints are detected. The keypoint locations in the current frame using Lucas-Kanade optical flow *(cv2.calcOpticalFlowPyrLK())* are estimated [7]. Later, verification of  consistency, by selecting points with small error values and stable status, forming trajectories with points at a consistent distance are done. This ensures smooth motion tracking. Finally, estimate vehicle direction using the last two points of each trajectory to determine movement direction. The details of the adaptive optical flow algorithm is presented in Algorithm 1.  
 **Algorithm 1: Adaptive Optical Flow**

**input** : Video frames  
 **output** : Boundaries of different directions of motions  
   
 *Set Lucas-Kanade optical flow parameters* (winSize, maxLevel etc.)  
 *Set feature detection parameters*(maxCorners, qulaitylevel, mindistance etc.)  
 *Initialisation of variables for trajectory storage and processing* ***Road Surface Segmentation****(video)*  
 **while** *True* **do**:  
 *suc, frame = cap.read()* (*if* *frame capture is unsuccessful, exit loop*)  
 *frame\_gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)* **if** *trajectories not empty* : *(compute optical flow)*  
 *p0,p1,p0r values using* ***cv2.calcOpticalFlowPyrLK()*** *Update Trajectories:* ***Append*** *new points to trajectories if the displacement is significant* ***Remove*** *trajectory[0]* **if** *len(tranjectory) > tracjectory\_len* **for** *trajectory in trajectories:  
 x1,y1 = trajctory[-2]  
 x2,y2 = trajctory[-1]  
 direction\_vector = np.array([x2-x1,y2-y1])  
 ( Normalize the direction vector)  
 angle = int(np.arctan2(direction\_vector[1],direction\_vector[0]))   
  
 (Finding the limits based on coordinates)* **for each** *trajectory :   
 if x<left\_limit:  
 x=left\_limit  
  
 if x>right\_limit:  
 x=right\_limit*

*(Iterate all tracjectories to get the final limits of the   
 frame for a particular direction)*

**At** *specified intervals, detect new features to track  
 p =* ***cv2.goodFeaturesToTrack****(frame\_gray, mask=mask, \*\*feature\_params)* (*initialize new trajectories using these features)  
 cv2.imshow('Optical Flow', img) (Visulalize the detections)  
 prev\_gray = frame\_gray (Update the prev\_frame)* **end**  
  
  
**2.3.2 Adaptive Blob Tracking and SORT for MCVL**  
Algorithm 2 begins by selecting the first 250 frames of the video for road surface extraction [1,2]. After the segmentation, Blob tracking [14] using OpenCV's SimpleBlobDetector was used to detect moving vehicles within the segmented road surface. Initialize the SORT [11] tracker to manage multiple objects, using Kalman filtering [13] and the Hungarian algorithm for data association. Update the SORT tracker with detected vehicle positions, assigning unique IDs and determining movement direction by comparing coordinates in consecutive frames. Assigning different colors to vehicles based on direction and visualizing their trajectories to analyze traffic patterns and flow.  
 **Algorithm 2 : Adaptive Blob tracking**

**input** : Video frames  
 **output** : Boundaries of different directions of motions  
   
 *Import libraries   
 Initialize vehicle dictionary and variables for boundaries and direction counts* ***Road Surface Segmentation****(video)*  
 **while** *True* **do**:  
 *frame = cap.read() (if frame capture is unsuccessful, exit loop)*

*(Apply background subtraction and thresholding)*

*fg\_mask = cv2.createBackgroundSubtractorMOG2().apply(frame)*

*fg\_mask = cv2.threshold(fg\_mask, 230, 255, cv2.THRESH\_BINARY)[1]*

*fg\_mask = cv2.erode(fg\_mask, None, iterations=3)*

*fg\_mask = cv2.dilate(fg\_mask, None, iterations=2)*

**for** *each contour in contours:*

*x, y, w, h = cv2.boundingRect(contour)*

***Append*** *[x, y, x+w, y+h, 1] to detections*

**if** *detections not empty:*

*tracked\_objects = tracker.update(detections)  
 current\_ids = set()*

**for** *obj in tracked\_objects:*

*obj\_id = int(obj[4])*

*bbox = obj[:4]*

*current\_ids.add(obj\_id)*

**if** *obj\_id in vehicle\_dict:*

***Update*** *direction for obj\_id*

*vehicle\_dict[obj\_id]['bbox'] = bbox*

***Append*** *bbox to vehicle\_dict[obj\_id]['path']*

**else:**

*vehicle\_dict[obj\_id] = {'id': obj\_id, 'bbox': bbox, 'direction': None*  
 *obsolete\_ids = set(vehicle\_dict.keys()) - current\_ids*

**for** *obj\_id in obsolete\_ids:*

***Remove*** *obj\_id from vehicle\_dict*

***Record*** *frame boundaries*

*min\_left, max\_left, min\_right, max\_right = get\_min\_max\_coordinates()*

***Append*** *[frame\_count, min\_left, max\_left, min\_right, max\_right] to   
 frame\_boundaries*

*Draw bounding boxes and paths on the frame*

**for** *vehicle in vehicle\_dict.values():*

*Draw rectangle on frame for vehicle['bbox']*

*Draw vehicle ID and coordinates on frame*

**if***path length > 1:*

*Draw path on frame*

***Display*** *frame*

*Save boundaries and direction counts to CSV*

*Release video capture and writer resources*

**end**

**2.3.3 Adaptive YOLO and SORT Algorithm for MCVL**

The latest version of YOLO [10] to detect vehicles is applied for obtaining bounding boxes and confidence scores. The SORT tracker to manage and update vehicle positions based on YOLO detections is initialized. Each vehicle's direction by comparing its current and previous coordinates: a higher y-coordinate indicates downward movement, while changes in x-coordinates indicate horizontal movement are determined. The trajectories and movement directions are visualized by drawing boundaries based on the tracked positions. Algorithm 3 shown the details of the method used for adaptive YOLO and SORT.

**Algorithm 3: Adaptive YOLO and SORT**

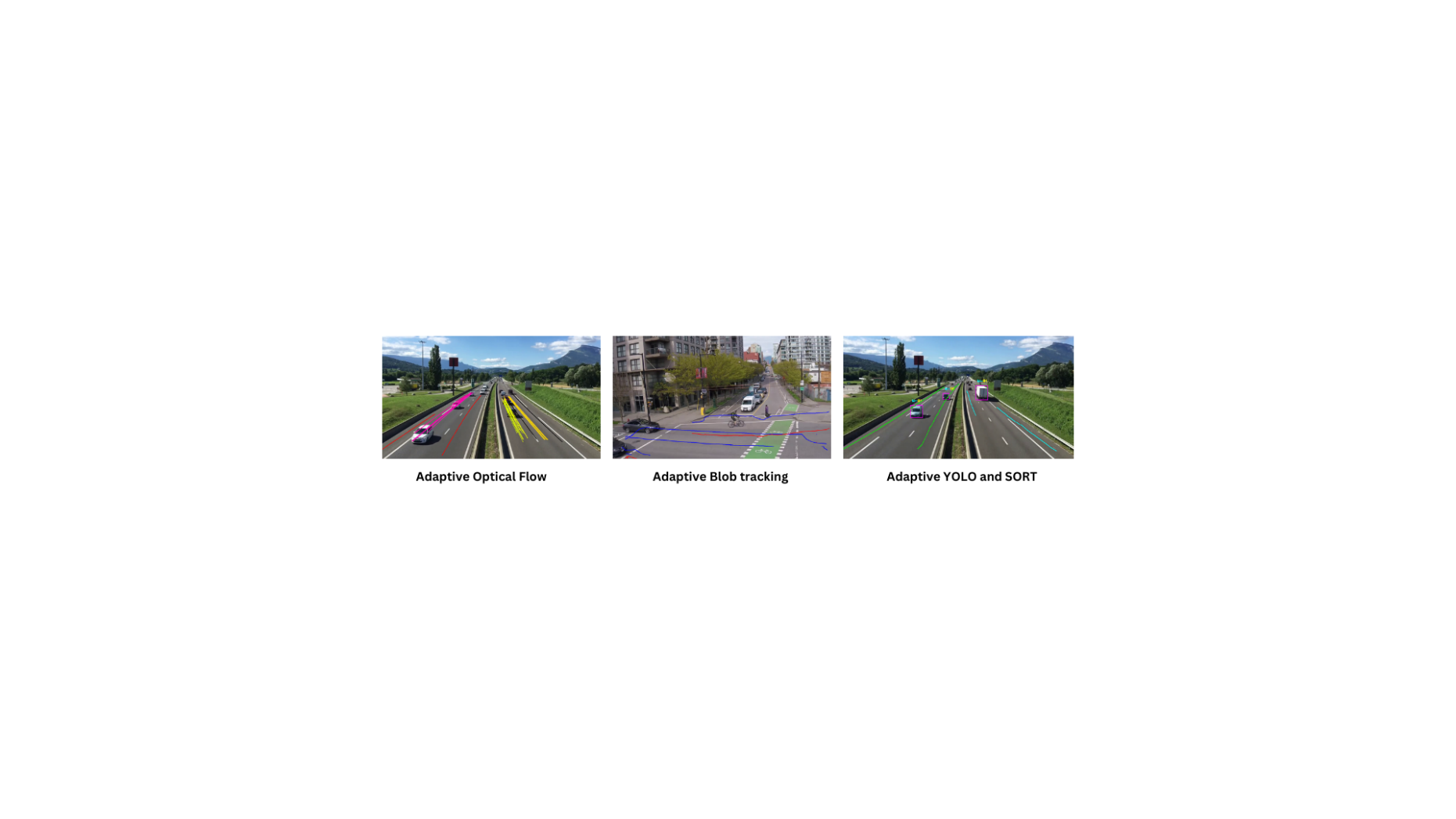
**input** : Video frames  
 **output** : Boundaries of different directions of motions  
  
 *Initialise an empty tracking\_obj* ***Road Surface Segmentation****(video)* *def* ***intialize\_y\_direction():* while** *frame\_count* **< *k(e.g. 300)***:  
 *suc, frame = cap.read()* (*if* *frame capture is unsuccessful, exit loop*)  
 *frame\_count+=1* *results = YOLO\_model(frame\_copy)  
 Using results get the detections array  
 (Use SORT tracker to get the ID’s using the detections array)* **for** *result in resultTracker:  
 Get the Tracking ID and coordinates of each vehicle* **if** *ID in tracking\_obj:  
 prev\_pt = tracking\_obj[tracking\_id]  
 curr\_pt = (x1, y1, w, h)* **if** *curr\_pt[1] >= prev\_pt[1]+2: (Top to bottom movement of vehicles)  
 Store the coordinates of that vechile in a variable and   
 compare with next vehcile coordinates* **else** *: (Bottom to top movement of vehicles)  
 Store the coordinates of that vechile in a variable and   
 compare with next vehcile coordinates* **end** *return ID’s*   
 *import the* ***intialize\_y\_direction()*** *function.*  
 *Initialize video capture from the input footage.*  
 *Load the YOLO model with pre-trained weights.  
 Initialize the SORT tracker with specified parameters.*  
 *Get the vehcile ID’s which acts as boundaries using* ***intialize\_y\_direction()\_\_\_\_(1)***  
  
 **while** *True* **do**:  
 *suc, frame = cap.read()* (*if* *frame capture is unsuccessful, exit loop*)  
 *results = YOLO\_model(frame\_copy)* **for** *result in results:  
 boxes = result.boxes* **for** *box in boxes:  
 Get the detections using coordinates obtained from detstructuring box  
 Using them get the detections  
  
 resultsTracker = SORT\_tracker.update(detections)* **for** *result in resultTracker:  
 Get the Tracking ID and coordinates of each vehicle  
   
 If those ID’s fectched in* ***(1)*** *are present in these ID’s:  
 Track the trajectory points of vechicle  
 It becomes the boundary in that direction* ***cv2.imshow****(‘Detection’, frame\_copy) (Visulalize the detections)   
 Combine each trajectory using* ***cv2.line()*** **end**

**3 Results and Discussion**

A demanding, real-world benchmark for multi-object tracking and detection is called UA-DETRAC. Images of typical and interstate road scenes are included in this dataset. The screenshot of few videos are shown in Fig.3.

**Fig. 3.** Screenshots from the **DETRAC** dataset

In this research, the implemented approaches focus on prediction tasks for directions of motions boundary detection. To assess the performance of these approaches, we utilize standard evaluation metrics such as Mean Square Error (MSE) and Root Mean Square Error (RMSE) and their respective computational time are analyzed.

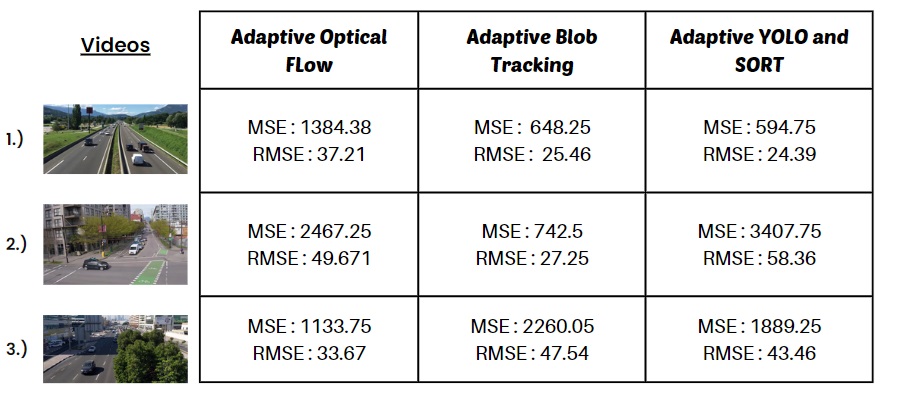


**Fig.4** Boundary of Vehicle Motion Detection

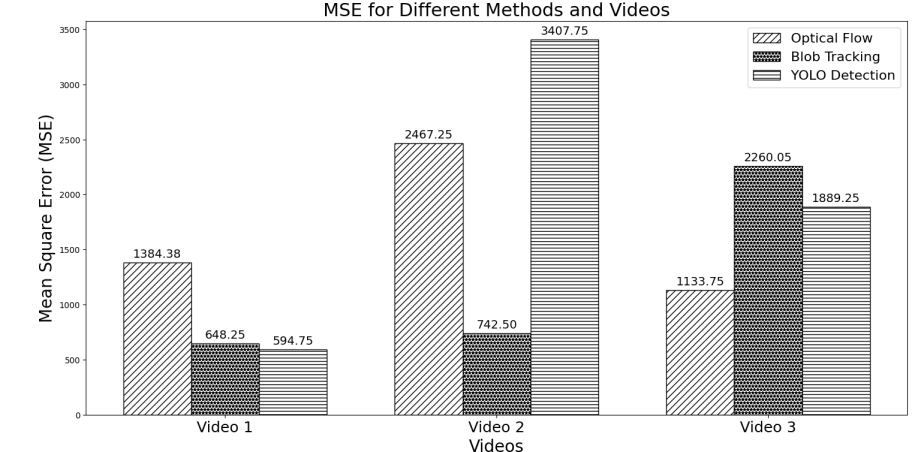
The workflow of the boundary detection using adaptive Optical flow, adaptive blob tracking and adaptive YOLO for a sample traffic video are shown in **Fig.4.**

The results are compared using the error rates discussed along with the time complexities**.** Actual boundaries for the videos in the DETRAC dataset are calculated manually. Later, the error is computed by comparing these manually established boundaries with the predicted boundaries for vehicles motion directions. Table 1. represents the comparison of the results on different videos using the three approaches mentioning their MSE and RMSE values. The graphical representation for few videos of MSE and RMSE are shown in Fig 5 and 6 respectively.

**Table 1.** Comparison of Adaptive Approaches on different videos

****

It is observed that the blob tracking approach performs well among the three methods evaluated. However, there are instances where the optical flow method outperforms the others. Each approach has its unique advantages and disadvantages. Additionally, variations in the error values were noted when different videos were analyzed.

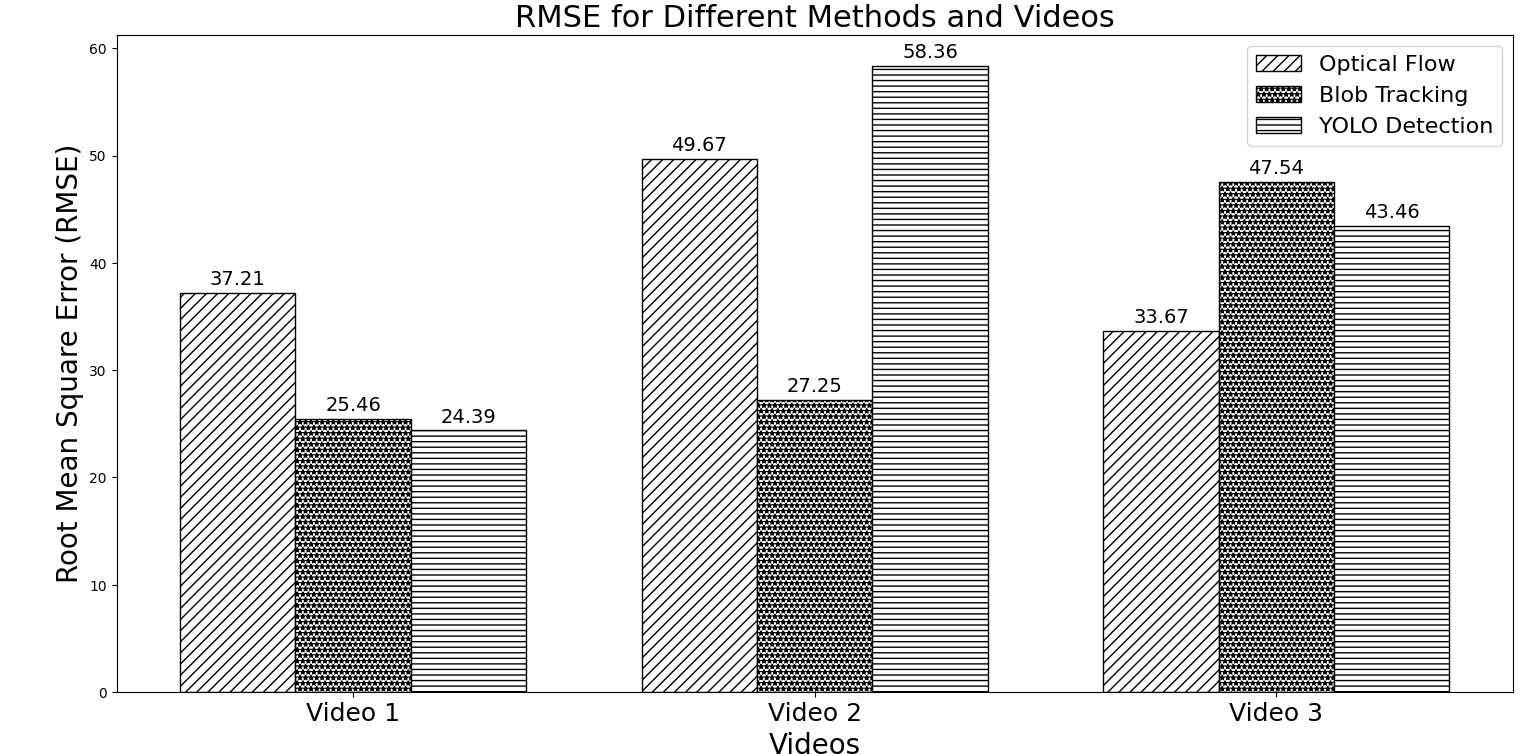


**Fig.5.** Graphical Representation of MSE

One reason the blob tracking approach records minimal error is that the bounding boxes, or blobs, adjust as the vehicle moves farther from the camera. This adjustment allows the predicted boundary to closely match the actual value. In the optical flow approach, the boundary is detected using trajectory points derived from feature detection. As the car moves away from the camera, it becomes smaller, and fewer features are detected.In turn, the road boundary is not accurately detected, affecting the error.   
 **Table 2.** Average MSE of adaptive Optical flow, Block tracking and YOLO Approaches

|  |  |  |
| --- | --- | --- |
| **Optical Flow** | **Blob tracking** | **YOLO** |
| 1673.7234 | 1221.734 | 1969.167 |

In the YOLO approach, the size of the bounding box remains almost fixed unless the vehicle size changes significantly. This rigidity causes the bounding box to sometimes exceed the vehicle's actual size, impacting the boundary detection and increasing the error.



**Fig.6.** Graphical Representation of RMSE  
  
For better visualization, we created a graphical comparison of RMSE and MSE errors for the three approaches. Additionally, we compiled their average RMSE and MSE values into Table 2 and Table 3, respectively. All these approaches involve analyzing some initial number of frames and extracting the required information from those frames to solve the problem statement. Some initial time is consumed to fulfil this step for all three approaches.

**Table 3.** Average RMSE of adaptive Optical flow, Block tracking and YOLO Approaches

|  |  |  |
| --- | --- | --- |
| **Optical Flow** | **Blob tracking** | **YOLO** |
| 40.91 | 34.953 | 44.38 |

**Table 4.** Average time taken (in seconds) per frame to implement adaptive Optical flow, Block tracking and YOLO Approaches

|  |  |  |
| --- | --- | --- |
| **Optical Flow** | **Blob tracking** | **YOLO** |
| 0.2029 | 0.25 | 1.84 |

Table 4 represents average time taken (in seconds) per frame to implement of all three adaptive approaches. Out of all the three algorithms, the adaptive YOLO method required more time per frame to implement whereas adaptive Optical flow only took 0.2029 seconds per frame to perform the same, which was the least time taken among the three methods. The adaptive blob tracking consumed around 0.25 seconds per frame to implement the heuristic algorithm.  
  
**4 Conclusion**This research work introduces a novel framework which can be used to evaluate Multi-layer Contiguous Virtual Layer (MCVL) effectiveness in vehicle detection; this paper conducted a comparative analysis of three distinct heuristic methods which are designed using the existing algorithms like optical flow, blob tracking, and YOLO sort, for detecting vehicle movements in traffic videos. The existing research relied on deep learning approaches; but faces some major drawbacks. To over- come these challenges, this paper proposed three alternative adaptive approaches along with the comparative evaluation using evaluation metrics like Mean Square Error (MSE), Root Mean Square Error (MSE). In addition to that a brief explanation of each approach along with visual interpretation of results is also provided for easy comprehension.

##### **Acknowledgment**

This publication is an outcome of the R\&D work undertaken in the project under TiHAN Faculty Fellowship of NMICPS Technology innovation Hub on Autonomous Navigation Foundation being implemented by Department of Science and Technology National Mission on Interdisciplinary Cyber-Physical Systems (DST NMICPS) at IIT Hyderabad.

**References**

1. Hadi Ghahremannezhad, Hang Shi, Chengjun Liu, “Automatic Road Detection in Traffic Videos”, *IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS) pp., 2021.*
2. Huansheng Song, Haoxiang Liang\* , Huaiyu Li, Zhe Dai and Xu Yun (2019) “Vision-based vehicle detection and counting system using deep learning in highway scenes”. *European Transport Research Review*
3. Salman Qasim, Kaleem Nawaz Khan, Miao Yu; Muhammad Salman Khan "Performance Evaluation of Background Subtraction Techniques for Video Frames". *2021 International Conference on Artificial Intelligence (ICAI)*
4. Radhakrishnan, M (2013).Video object extraction by using background subtraction techniques for sports applications. *Digital Image Processing, 5(9),91–97.*
5. “Moving Object Detection and Segmentation using Frame Differencing and Summing Technique” Gopal Thapa, Kalpana Sharma, M.K.Ghose, *September 2014 International Journal of Computer Applications 102(7):20-25*
6. Estevao Gedraite, M.Hadad January 2011 “Investigation on the effect of a Gaussian Blur in image filtering and segmentation”. *ELMAR, 2011 Proceedings*
7. H.Yedjour, January 2021 Optical Flow Based on Lucas-Kanade Method for Motion Estimation. In : *Artificial Intelligence and Renewables Towards an Energy Transition (pp.937-945)*
8. Dhara Patel, Saurabh Upadhyay, Volume 61– No.10, January 2013 Optical Flow Measurement using Lucas kanade Method. *International Journal of Computer Applications (0975 – 8887)*
9. Real-Time Vehicle Detection Based on Improved YOLO v5 Yu Zhang, Zhongyin Guo, Jianqing Wu. *September 2022 Sustainability 14(19):12274*
10. “Vehicle Detection and Tracking using YOLO and Deep SORT” Muhammad Azhad Bin Zuraimi; Fadhlan Hafizhelmi Kamaru Zaman *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*
11. Mahmoud Abouelyazid (2023) “Comparative Evaluation of SORT, DeepSORT, and ByteTrack for Multiple Object Tracking in Highway Videos”. *Vol. 8 No. 11 (2023): IJSICS-November*
12. Karl Leyven Leonida, Karla Veronica Sevilla, Cyrel O. Manlises "A Motion-Based Tracking System Using the Lucas-Kanade Optical Flow Method". *2022 14th International Conference on Computer and Automation Engineering (ICCAE)*
13. Y. Ng, Y. Latif, T.-J. Chin, and R. Mahony, “Asynchronous kalman filter for event-based star tracking,” in Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel*, October 23–27, 2022, Proceedings, Part I. Springer, 2023, pp. 66–79*
14. A. Chaturvedi and A. Shukla, "Automatic detection of satellite images using blob detection and boundary tracking techniques", *2020.*
15. S. Li, J. Chen, W. Peng, X. Shi and W. Bu, "A vehicle detection method based on disparity segmentation", *Multimedia Tools Appl.*, vol. 82, no. 13, pp. 19643-19655, May 2023.
16. A. A. Kumar et al., "A Comparative Study of Various Filtering Techniques", *ICOEI, 2021.*
17. N. M. Al-Shakarji, F. Bunyak, G. Seetharaman and K. Palaniappan, "Multi-object tracking cascade with multi-step data association and occlusion handling", 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1-6, 2018.
18. Enric Meinhardt-Llopis1 , Javier S´anchez2 , Daniel Kondermann3 “Horn–Schunck Optical Flow with a Multi-Scale Strategy”, *Image Processing On Line on 2013–07–19*
19. K.C Hari; Sushil Shrestha; Manish Pokharel,"Video Object Motion Tracking using Dense Optical Flow Techniques", *2023 International Conference on Informatics and Computing (ICIC)*
20. Shuman Guo,Shichang Wang"A Review of Deep Learning-Based Visual Multi-Object Tracking Algorithms for Autonomous Driving", Applied Sciences 12(21):10741,October 2022
21. Chakradhara Panda,"Object Detection and Tracking using Faster R-CNN", September 2019,International Journal of Recent Technology and Engineering (IJRTE)
22. Sankaranarayanan, M., Mala, C., Mathew, S. “Efficient vehicle detection for traffic video-based intelligent transportation systems applications using recurrent architecture”. Multimed Tools Appl 82, 39015–39033, 2023.