Observation-Centric with Appearance Metric for Computer Vision-Based Vehicle Counting

Allysa Kate Brillantes [1,2,*], Edwin Sybingco [1,2,3], Robert Kerwin Billones [2,3,4], Argel Bandala [1,2], Alexis Fillone [2,3,5], and Elmer Dadios [2,3,4]

[1] Department of Electronics and Computer Engineering, De La Salle University, Manila, Philippines;
Email: edwin.sybingco@dlsu.edu.ph (E.S.), argel.bandala@dlsu.edu.ph (A.B.)
[2] Center for Engineering and Sustainable Development Research, Manila, Philippines;
Email: robert.billones@dlsu.edu.ph (R.K.B.), alexis.fillone@dlsu.edu.ph (A.F.), elmer.dadios@dlsu.edu.ph (E.D.)
[3] Intelligent Systems Innovation (ISI, Inc.), Philippines
[4] Department of Manufacturing Engineering and Management, De La Salle University, Manila, Philippines
[5] Department of Civil Engineering, De La Salle University, Manila, Philippines
*Correspondence: allysa_brillantes@dlsu.edu.ph (A.K.B.)

***Abstract*—Tracking objects in video sequences is a key step for applications involving computer vision like traffic monitoring and security systems. Occlusion is a frequent problem in object tracking which can result in the tracker losing track of the occluded object or misidentifying it with the occluding object. Moreover, the limited memory and computing power of traffic analysis systems presents a scaling problem, especially in object tracking applications. This paper aims to improve object tracking performance by minimizing data association errors in low frame rate tracking applications. Reducing frame rates alleviates memory and computing power limitations, and utilizing a tracker that can handle occlusion can address occlusion-related issues in object tracking. The proposed tracking method, Mask-OCSORT, uses the observation-tracking method with cosine similarity, intersection-over-union, and velocity consistency metrics for the association problem. The paper analyzes the effect of using bounding box and mask predictions of deep learning models in generating tracks. This study uses evaluation metrics like HOTA, MOTA, and IDF1 to assess the proposed tracking method, and employs evaluation metrics such as precision, recall, and F-score to assess the counting based on generated IDs from the tracking method. The study applied the Mask-OCSORT tracking for vehicle counting application and achieved an F-score of 87.18% at 5 frames per second (fps), and 75% at 1 fps.**

***Keywords*—mask-OCSORT, object tracking, instance segmentation, traffic surveillance systems, low framerate, data association**

I. Introduction

The rising number of vehicles on the roads has resulted in near-maximum capacity for existing transportation networks that causes congested roads in numerous countries [1, 2]. To address this issue, many researchers studied Intelligent Transportation Systems (ITS). ITS uses a range of communication, control, and electronic technologies to monitor and manage traffic flow, reduce congestion, provide efficient routes for travelers, improve productivity, and save time and money [3]. ITS provides essential data for transportation planning, urban management, and infrastructure maintenance. A traffic monitoring system is a vital part of any ITS project since it can provide information such as traffic index and traffic density, and it can be used in traffic analysis to optimize the operation of road systems. ITS can also predict future transportation requirements, and improve transportation safety [1]. Vehicles as indicators of the performance of transport systems became possible because of the constant monitoring of traffic parameters from static cameras. Computer vision-based vehicle counting is being studied by many ITS researchers [4, 5].

Most vision-based vehicle counting techniques are composed of detection, tracking, and counting processes [4–6]. The detection stage is the method of classifying and localizing vehicles in video frames. The tracking stage, on the other hand, is the process of preserving the vehicles' identities and extracting further their trajectories data to ensure that each vehicle is only counted once [5].

Tracking vehicles at low frame rates is necessary for applications such as visual surveillance and embedded systems because of factors such as the cost of hardware and the size of source data. However, lowering the frame rate of the video or skipping some video frames from the traffic video is equivalent to sudden or abrupt motion that may affect the accuracy of vehicle tracking and counting. Most existing tracking approaches, with only a few exceptions, cannot be easily implemented because they are vulnerable to motion and appearance discontinuity from the low frame rate data [7, 8].

Urban traffic monitoring systems may observe the occlusion issue in vehicle tracking. Simple Online and Realtime Tracking (SORT) [9] was utilized by some researchers for vehicle tracking for traffic monitoring but experienced missing and multiple counts [4, 10]. SORT uses linear estimation and is more appropriate in high

Manuscript received April 13, 2023; revised June 19, 2023; accepted July 19, 2023; published November 24, 2023.

frame rate videos and may produce errors when occlusions and non-linear object motion were encountered [11]. On the other hand, Observation-Centric SORT, i.e., OCSORT was introduced by Cao *et al*. [11] which is more robust over occlusion and nonlinear motion. However, experiments conducted on OCSORT show that it has limitations on tracking objects in low frame-rate videos or tracking fast-moving objects like cars in the KITTI dataset [12]. OCSORT finds it difficult to match objects by only using the Intersection Over Union (IOU) and direction consistency of tracks and observations [11].

This study aims to decrease data association errors during the tracking of vehicles in low-frame-rate videos by utilizing appearance features from the instance segmentation algorithm. The method proposed for localizing the vehicle is an instance segmentation method. This paper proposes Mask-OCSORT, a modification of the OCSORT tracker that uses mask features and predictions from the instance segmentation for tracking vehicles in a traffic video. Mask-OCSORT combined the cosine similarities of tracked features and new observation features with IOU and velocity consistency metrics.

II. LITERATURE REVIEW

A. Instance Segmentation

Instance segmentation performs both detection and semantic segmentation within an image. Object detection identifies the classification and coordinates of objects in the image, while semantic segmentation gives labels to each picture element of an image. Instance segmentation identifies and segments pixels belonging to each object instance. It is crucial for many computer vision tasks because of its applications, including scene understanding, autonomous driving, agricultural analysis, and medical image analysis [13].

According to Ref. [14], detection quality is necessary for vehicle tracking and counting and the use of deep learning-based detectors can enhance the tracking performance. Previous studies introduced quality detections in tracking to lessen the false negatives. Most tracking algorithms use the detection precision provided by Faster Region-based Convolutional Neural Networks (R-CNN) [15] or Single Shot MultiBox Detector (SSD) [16] but other methods that utilize different detection algorithms like You Only Look Once (YOLO) [17] series, which includes YOLO9000 [18], YOLOv3 [19], also exist.

In Ref. [20], instead of using appearance features inside the bounding box region for affinity computation of data association, the appearance features were extracted from the mask using a network based on Mask R-CNN [21] for pedestrian tracking. Chang *et al.* [22] utilizes the instance segmentation masks from YOLACT [23] for autonomous vehicles. The embedding vectors that represent the object information will not be accurate because the bounding-box region from the detectors contains background information or other objects [22]. This study aims to utilize the instance segmentation masks for appearance features in counting vehicles in low frame rate video.

B. Multiple Object Tracking

Multiple Object Tracking (MOT) is responsible for classifying and localizing objects, preserving their identifications, and extracting their trajectories from a video [24]. Fig. 1 shows the framework of the MOT system where detections are produced by the object detection module from the video frames. The tracks are generated by matching the visual and motion features of the detections from prior frames and the present frame using the data association algorithm. The output of the MOT system is unique identifications of objects on the video frames [25].

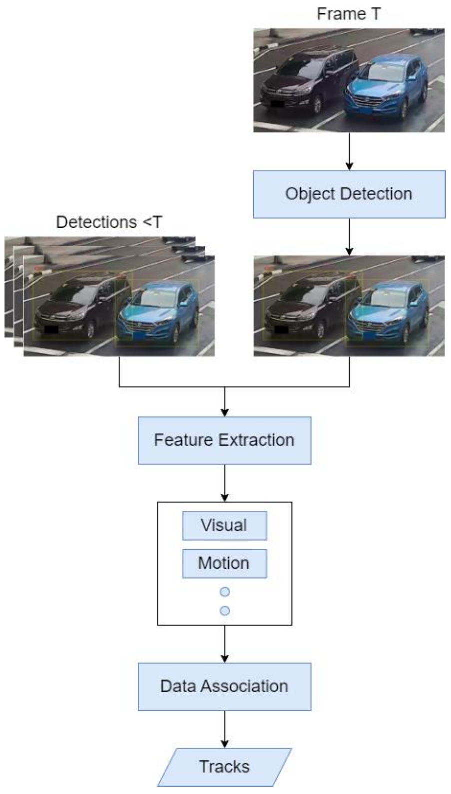


Figure 1. MOT general framework [25].

MOT systems can be classified based on their primary components. Appearance models are utilized for spatial information extraction from detections and similarity computation. An end-to-end MOT method was proposed by Wan *et al.* [26] that filters and refines the search region for association using template information from previous frames and detections from the current frame. Chen *et al.* [27] designed a reference search component for data association that uses past tracks as the reference for each track's current state based on visual temporal data. A graph tracker was proposed by Hyun *et al.* [28] that uses relational features to associate previous detections with the detections of the current frame. Dai *et al.* [29] utilized self-attention and cross-attention mechanisms in learning the features of each track and in modeling similarities between tracks and detections. According to the research of Gad *et al.* [25], these appearance-based studies rely only on visual features and did not perform well in terms of accuracy.

However, motion models are utilized for predicting the next locations of the detections. For the MOT problem, SORT [9] presented that utilizing Faster R-CNN [15] for object detection, the Kalman filter [30] for motion estimation and the Hungarian method [31] for the association of objects can improve the tracking performance.

OCSORT was introduced by Cao *et al*. [11] which is an observation-centric motion-based tracking algorithm capable of handling non-linear movements. According to Ref. [11], SORT accumulates state noises when no new detection matches with tracks caused by non-linear motion or occlusion. Also, SORT tracker relies on Kalman filter state's estimation.

According to a study on traffic monitoring by Khazukov *et al*. [10], vehicle counting errors caused by prolonged occlusions when utilizing a SORT tracker can be solved by using re-identification based on appearance cues. Also, Gad *et al*. [25] noted that utilizing both motion features and appearance features can achieve better accuracy [32]. Intending to count vehicles, this study proposes a modification to the OCSORT tracker by adding appearance features from instance segmentation as an association metric. Aside from the IOU and direction consistency of tracks and observations, the proposed method used appearance features from the cropped vehicle image or prediction mask of the instance segmentation as an association metric.

III. Proposed Method

This paper proposes a framework for vehicle counting in low-frame rate video using a modified OCSORT called Mask-OCSORT. As illustrated in Fig. 2, Mask-OCSORT adds appearance features from the instance segmentation into the association cost of the OCSORT tracker. The major components are instance segmentation and modified OCSORT Tracker.

The aim of the instance segmentation is to create mask features and locations of vehicles in the segmentation mask with corresponding classification and confidence scores. The bounding box can be generated from the segmentation mask to be used by Mask-OCSORT in state estimation and data association. Mask-OCSORT uses the bounding boxes, classifications, confidence scores, and appearance features as its inputs. This study proposes a combination of metrics: cosine similarity measure of appearance features, IOU, and the velocity consistency in the data association. The tracker outputs track identifications, classes, and vehicle locations. The number of generated track identifications from the tracking module is equivalent to the vehicle count in the traffic video.

To analyze the effect of using appearance features inside the bounding boxes and segmentation masks in generating tracks, the study compared Mask-OCSORT to OCSORT with a deep appearance descriptor (OCSORT-DA) as shown in Fig. 3. The difference between these methods is the appearance feature that they used. Mask-OCSORT directly used the output of the instance segmentation while OCSORT-DA's appearance features come from the output of a feature extraction model with the cropped vehicle as the input.

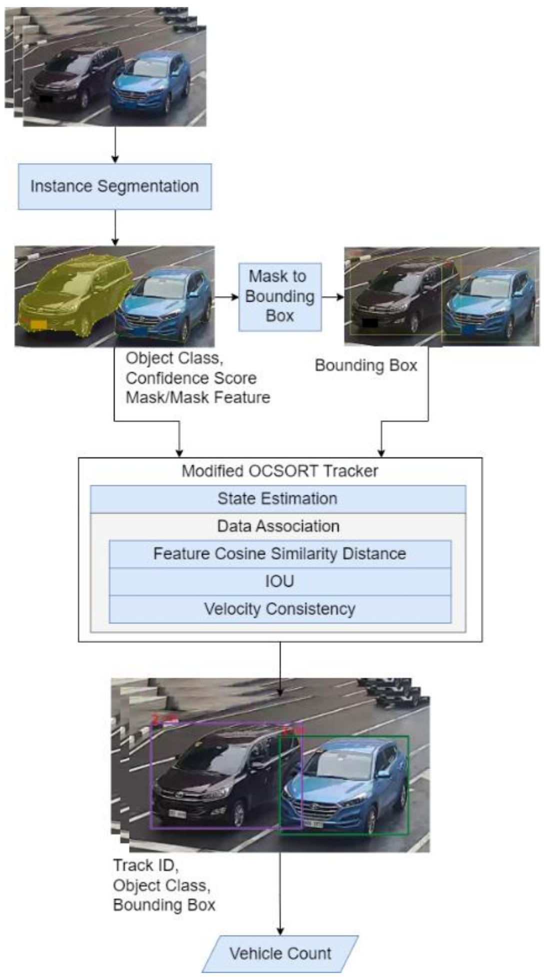


Figure 2. Vehicle counting framework using Mask-OCSORT.

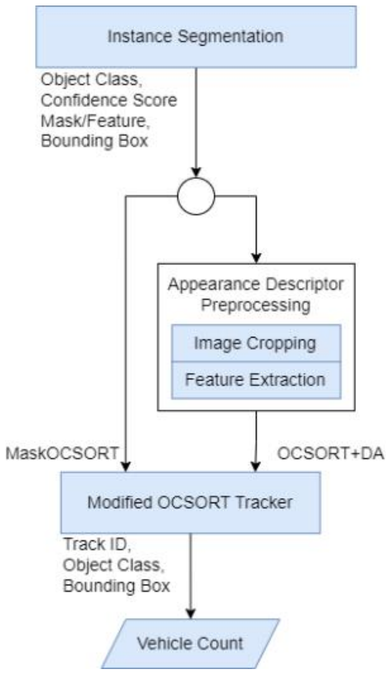

Instance Segmentation
Object Class, Confidence Score Mask/Feature, Bounding Box
Appearance Descriptor Preprocessing
Image Cropping
Feature Extraction
MaskOCSORT
OCSORT+DA
Modified OCSORT Tracker
Track ID, Object Class, Bounding Box
Vehicle Count

Figure 3. Difference of Mask-OCSORT with OCSORT+DA.

A. Vehicle Detection Using Instance Segmentation

Instance segmentation identifies the position of objects, marks the boundary of single instances, and distinguishes the category of each instance. In this study, the implementation of the proposed method used three known instance segmentation methods: (1) Mask R-CNN, (2) SOLOv2, and (3) YOLOv7.

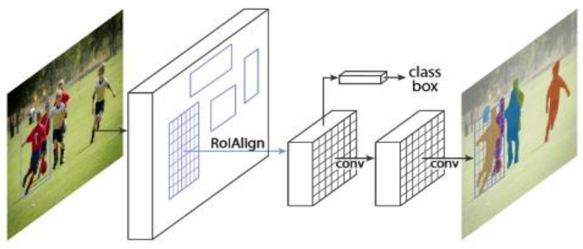


Figure 4. Mask R-CNN [21].

The first instance segmentation algorithm that this study used is the Mask R-CNN [21]. As shown in Fig. 4, in parallel with the classification and box prediction, Mask R-CNN performs the mask prediction from each region proposal using a Fully Connected Network (FCN) [30]. The study utilizes the segmentation masks, classifications, and scores from the output of the Mask R-CNN for the input of the proposed tracking method. The mask feature of size 256×14×14 received from the output of the mask head of Mask R-CNN was reduced to 256×4×4 using max-pooling for the data association of Mask-OCSORT. Table I shows the hyperparameters used in the Mask R-CNN implementation.

TABLE I. HYPERPARAMETERS OF MASK RCNN MODEL

Name	Value
Anchor Generator Sizes	[32, 64, 128, 256, 512]
Anchor Generator Sizes	[0.5, 1, 2]
RPN Top scoring RPN before NMS	1000
RPN Top scoring RPN after NMS	1000
RPN NMS Threshold	0.7
ROI BoxHead Pooler Resolution	7
ROI MaskHead Pooler Resolution	14
ROI BoxHead Number of FC Layers	2
ROI MaskHead Number of Convolutions	4
ROIHead IOU Threshold	0.5
ROIHead Score Threshold	0.05
ROIHead NMS Threshold	0.5
ROIHead Number of Classes	80
Maximum Detections per Image	100
Solver Base Learning Rate	0.02

Another instance segmentation that this study used is the Segmenting Objects by Locations (SOLO) [33] which is shown in Fig. 5. SOLO generates grids *S*×*S* leading to S^2 center location classes and predicted masks from the input image. It utilizes a backbone network and Feature Pyramid Network (FPN) [34] to produce features used in the category and mask branches. With the backbone and FPN, SOLOv2 predicted the kernel at each pyramid level. To construct the mask feature, SOLOv2 utilized all FPN levels to represent the mask feature by merging FPN feature maps. SOLOv2 used the normalized pixel coordinates by the deepest FPN level before convolutions and up-sampling layers to have accurate position data.

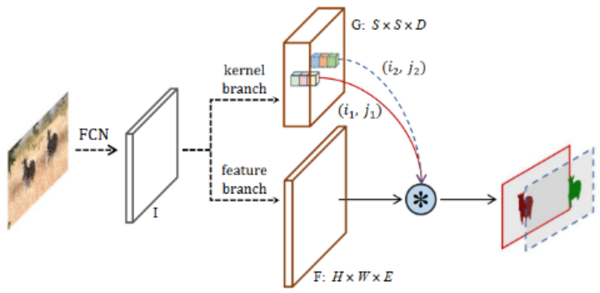


Figure 5. SOLOv2 [35].

Table II shows the hyperparameters used in the SOLOv2 implementation.

TABLE II. HYPERPARAMETERS OF SOLOV2 MODEL

Name	Value
Anchor Generator Sizes	[32, 64, 128, 256, 512]
Anchor Generator Sizes	[0.5, 1, 2]
RPN Top scoring RPN before NMS	1000
RPN Top scoring RPN after NMS	1000
RPN NMS Threshold	0.7
ROI BoxHead Pooler Resolution	7
ROI MaskHead Pooler Resolution	14
ROI BoxHead Number of FC Layers	2
ROI MaskHead Number of Convolutions	4
ROIHead IOU Threshold	0.5
ROIHead Score Threshold	0.05
ROIHead NMS Threshold	0.5
ROIHead Number of Classes	80
Maximum Detections per Image	100
Solver Base Learning Rate	0.02

Same with the implementation of tracking with Mask R-CNN, the bounding boxes from the segmentation masks were used for the motion estimation. The masks with the size of 192×336 from SOLOv2 were reduced to 192×28 using max-pooling for the data association.

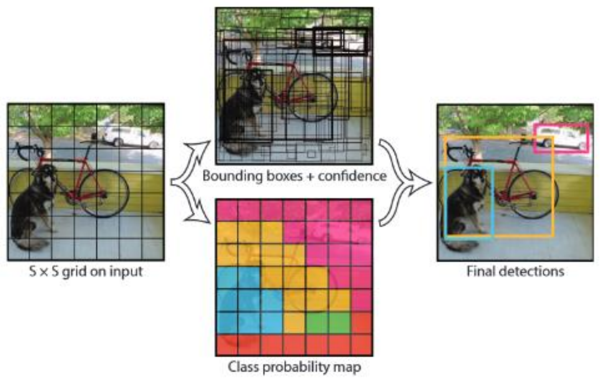


Figure 6. YOLO [17].

The last instance segmentation method that this study used is based on the YOLOv7 model [36]. Fig. 6 shows the YOLO model which can predict bounding boxes with confidence scores and class probabilities directly by also dividing the image into a grid [17, 37]. Table III shows the hyperparameters used in the YOLOv7 implementation.

Wang *et al.* [36] proposed a new re-parameterization module and scaling methods to enhance the accuracy of YOLO. This study utilized the instance segmentation implementation of YOLOv7.

TABLE III. HYPERPARAMETERS OF YOLOV7 MODEL

Name	Value
Optimizer Initial Learning Rate	0.01
Optimizer Final One Cycle Learning Rate	0.1
Optimizer Momentum	0.937
Optimizer Weight Decay	0.0005
Warmup Epochs	3.0
Warmup Initial Momentum	0.8
Warmup Initial Learning Rate	0.1
Box Loss	0.05
Classification Loss	0.3
Classification Binary Entropy (BCE) Loss Positive Weight	1
Objectness Loss	0.7
Objectness BCE Loss Positive Weight	1
Mask Loss	0.05
Mask BCE Loss Positive Weight	1
PointRend Loss	0.05
IOU Training Threshold	0.2
Anchor Multiple Threshold	4
Anchors per Output Layer	3
Attention Resolution	14
Mask Resolution	56

B. Vehicle Tracking

This study adopts OC-SORT and performs some modifications for tracking vehicles at low frame rates. OC-SORT proposed an observation-centric tracker that is more robust over occlusion and nonlinear motion. It conducts online smoothing over the Kalman filter's parameters back to the time of being untracked t_1 through a virtual trajectory $\mathrm{Traj}_{\mathrm{virtual}}$ of observations to prevent error accumulation due to lost detection or occlusion as shown in Eq. (1). In this equation, the last detection or recognition before being lost is represented as z_{t1} and the reassociated detection is represented as z_{t2}. OSS starts backchecking the Kalman filter's parameters from t_1 by alternating between its prediction and update stages when a lost track is reassociated along this virtual trajectory [11].

$$\widehat{z_t} = \mathrm{Traj}_{virtual}(z_{t1}, z_{t2}, t), t_1 < t < t_2 \qquad (1)$$

OCSORT smoothing is effective when the object is associated again after being untracked. A problem occurs when the tracker failed to reassociate the object and assigned it a different track identification. This can be observed when the video has a low frame rate since OCSORT's association depends on the IOU of the previous and current boxes.

DeepSORT [38] overcomes the issue of occlusion in SORT by combining motion and appearance data for its association metric. DeepSORT integrated a Convolution Neural Network (CNN) model for generating features used as a deep association metric. According to the research of Perera *et al.* [39], DeepSORT's feature extractor is not suitable for vehicles since the CNN model was trained using a large-scale person reidentification dataset [40]. With this, instead of using the existing network of DeepSORT that was trained for person reidentification, AlexNet was used by Perera *et al.* [39] to extract the appearance features of each detected object or vehicle. This study extracted the appearance feature with a size of 4096×1 from the second FCN layer of AlexNet for the implementation of OCSORT-DA.

With this, the study proposed to utilize appearance features to associate new observations with the previous tracks. The authors modified the association cost of OCSORT to integrate the appearance information in the data association process. The proposed method computed the cosine similarities of previously tracked features and new observation features. The cosine similarity measure reflects the appearance information to associate identities when motion is less discriminative. Mask-OCSORT combined the IOU and the cosine distances of previous tracks and new observations using a weighted sum for the association problem. Eq. (2) shows the modified association cost.

$$C(\hat{X}, Z) = \lambda_1 C_{IOU}(\hat{X}, Z) + (1 - \lambda_1) C_{COS}(\hat{X}, Z) + C_V(\hat{X}, Z, V)\ ) \qquad (2)$$

The $(\hat{X}, Z)$ are the estimated object states and observations while V is composed of directions of existing tracks from two previous observations. $C_{IOU}(\cdot,\cdot)$ computes the IOU, $C_{COS}(\cdot,\cdot)$ calculates the cosine similarity measure, and $C_V(\cdot,\cdot)$ determines the consistency of directions of tracks and the direction of the track's past observations and new observations.

Inspired by Wojke *et al.* [38], the influence of IOU and cosine similarity metrics on the association metric can be managed by using a hyperparameter. A hyperparameter IOU weight λ_1 was used as a multiplier for the IOU matrix and $(1 - \lambda_1)$ for the cosine similarity matrix, while velocity direction weight (λ_2) is the weighting factor of the velocity consistency term of the original OCSORT association cost. For the vehicle tracking application, vehicles tend to look the same so setting the λ_1 equal to 0 is not advisable.

IV. RESULTS AND DISCUSSION

A traffic video with a resolution of 1920×1080 and a frame rate of 25 was used to evaluate the proposed method. When processing the video, some frames were skipped to simulate low frame-rate videos. This paper implemented the attributes used by Wen *et al.* [41] in describing the test video. As shown in Table IV, the authors used a one-minute traffic video, with a single vehicle type, taken during the daytime with sunny illumination for evaluation. Despite the wet road that indicates previous rainfall, the weather on the recorded test video is sunny. Partial occlusions were observed in the test video which is based on occlusion ratio; occlusion ratio Is measured by computing the IOU of vehicles with each other. Moreover, the vehicles' scales can be classified into medium and large scales, where scale can be defined as the square root of the area in pixels [41].

TABLE IV. ATTRIBUTES OF TEST VIDEO

Attribute Name	Classification/Value
Vehicle Type	1 (car)
Illumination	Sunny
Scale	Medium (50–150 pixels), Large (above 150 pixels), 81–610 pixels
Occlusion Ratio	No occlusion, Partial occlusion 0–22%

The frame rates of the test video are 25 frames per second (fps), 5 fps, and 1 fps. The test video at 25 fps has 1500 frames, at 5 fps has 300 video frames, and at 1 fps has 60 video frames.

The study utilized pre-trained models of Mask R-CNN and SOLOv2 with ResNet-50 and a pre-trained model of YOLOv7 on MS COCO [42] for instance segmentation. The pre-trained models have 80 classes, and only the five vehicle classes (bicycle, car, motorcycle, bus, and trucks) were used during the evaluation. This study conducted experiments on the proposed tracking algorithm using Nvidia GeForce RTX 2070 GPU. Based on OCSORT, the tracking threshold is set to 0.3, and velocity direction weight (λ_2) is 0.2 while the and the hyperparameter IOU weight (λ_1) is set to 0.3.

The study's main objective is to count the vehicles in traffic videos at low frame rates. The authors visually checked the True Positive (TP), False Positive (FP), and False Negative (FN) to evaluate the proposed counting system, on the traffic video. TP indicates that an Identification (ID) was assigned to a vehicle, FN denotes no ID was assigned to a vehicle from the start until the vehicle is out of the frame and FP implies multiple counts of an object. For instance, a vehicle has three unique IDs, with one counted as TP and the remaining two IDs are considered FPs. There are also instances where an ID was reassigned to a new vehicle after the original vehicle was out of frame. The ID for the new vehicle is considered FP. The mathematical equations below evaluate the counting performance of the proposed method.

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

$$F-score = \frac{2\times Precision\times Recall}{Precision+Recall} \tag{5}$$

As shown in Tables V and VI, the proposed counting system was tested on different instance segmentation algorithms: Mask R-CNN, SOLOv2, and YOLOv7. The frame rates used for evaluation are 1 fps and 5 fps, and the appearance features either come from the bounding box region or instance segmentation mask. Mask-OCSORT tracker obtained the best results at the evaluation of vehicle counting, at both frame rates. The proposed tracker with the appearance features from SOLOv2 achieved the highest F-scores, but the processing time is longer than other instance segmentation. YOLOv7 can attain a lower processing time compared to using the other instance segmentation methods. Considering the results of the evaluation at 1 fps and 5 fps, the study's implementation of SOLOv2 with the Mask-OCSORT is advisable if the processing time is not necessary. However, if high accuracy with lower processing time is needed, Mask-OCSORT with Mask R-CNN is recommended.

TABLE V. VEHICLE COUNTING RESULTS AT 1 FPS

Tracker	Method	Appearance Feature	True Positive	False Positive	False Negative	Precision (%)	Recall (%)	F-Score (%)	Total Time (second)
DeepSORT	Mask R-CNN	Deep Appearance Descriptor	7	7	11	38.889	50	43.750	13.176
	SOLOv2		10	5	13	43.478	66.667	52.632	19.480
	YOLOv7		8	5	16	33.333	61.538	43.243	4.923
OCSORT	Mask R-CNN	None	10	8	4	71.429	55.556	62.5	11.963
	SOLOv2		9	9	4	69.231	50.0	58.065	18.223
	YOLOv7		9	9	7	56.250	50.0	52.941	3.497
OCSORT-DA	Mask R-CNN	Deep Appearance Descriptor	9	4	9	50	69.231	58.065	14.554
	SOLOv2		9	3	11	45	75	56.250	21.585
	YOLOv7		8	9	2	80	47.059	59.259	6.192
Mask-OCSORT	Mask R-CNN	Mask	12	6	2	85.714	66.667	**75.0**	13.045
	SOLOv2		11	7	3	78.571	61.111	*68.750*	19.184
	YOLOv7		10	8	3	76.923	55.556	64.516	3.787

Note: The best score is highlighted in **bold** while the second and third best results are presented in italics and underlined, respectively. This also applies to other tables.

TABLE VI. VEHICLE COUNTING RESULTS AT 5 FPS

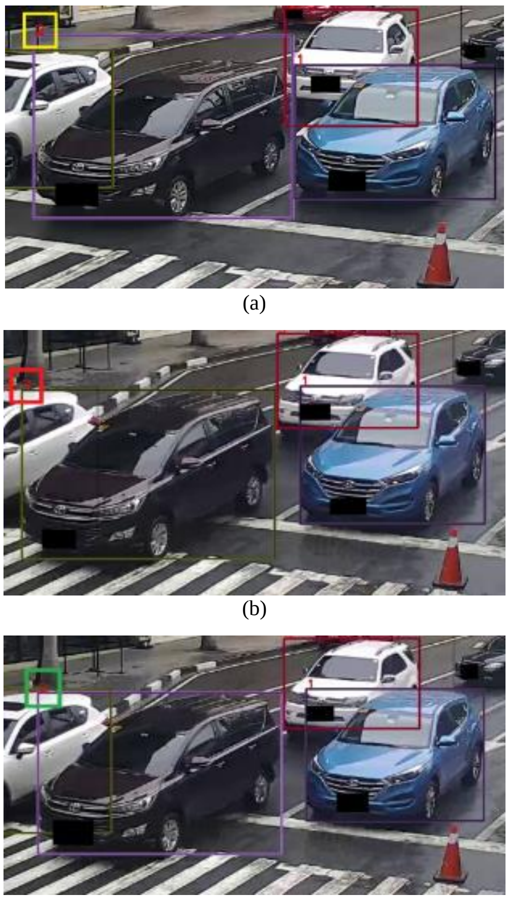
Tracker	Method	Appearance Feature	True Positive	False Negative	False Positive	Precision (%)	Recall (%)	F- Score (%)	Total Time (second)
DeepSORT	Mask R-CNN	Deep Appearance Descriptor	18	0	11	62.069	100	76.596	67.470
	SOLOv2		18	0	7	72	100	83.271	109.213
	YOLOv7		17	1	11	60.714	94.444	73.913	26.399
OCSORT	Mask R-CNN	None	18	0	6	75	100	*85.714*	59.407
	SOLOv2		17	1	6	73.913	94.444	82..927	99.217
	YOLOv7		17	1	7	70.833	94.444	80.952	16.536
OCSORT-DA	Mask R-CNN	Deep Appearance Descriptor	18	0	6	75.0	100	*85.714*	72.684
	SOLOv2		10	0	14	41.667	100	58.824	111.692
	YOLOv7		17	1	6	73.913	94.444	82.927	28.474
Mask-OCSORT	Mask R-CNN	Mask	18	0	6	75.0	100	*85.714*	62.755
	SOLOv2		17	1	4	80.952	94.444	**87.179**	101.507
	YOLOv7		17	1	8	68.0	94.444	79.070	17.504

Since the counting result is based on the tracking method, the study used the open-source toolkit of MOT Challenge [43, 44] to generate the performance metrics like MOTA, IDF1, and HOTA [45] for the tracking algorithm. Table VII shows that at different frame rates, the Mask-OCSORT with appearance features from SOLOv2 obtained the highest, MOTA, IDF1, and HOTA, which supports the high F-score achieved in the counting evaluation.

TABLE VII. TRACKING RESULTS OF PROPOSED METHOD VS PREVIOUS METHODS

Frame Rate	Metric	DeepSORT			OCSORT			OCSORT + DA			Mask-OCSORT		
		Mask R-CNN	SOLOv2	YOLOv7	Mask R-CNN	SOLOv2	YOLOv7	Mask R-CNN	SOLOv2	YOLOv7	Mask R-CNN	SOLOv2	YOLOv7
25	HOTA	51.467	54.936	54.813	51.8	55.955	50.884	34.95	42.756	41.551	56.904	**62.607**	*56.426*
	MOTA	57.223	54.895	47.423	51.895	52.395	44.836	49.663	51.847	46.23	*57.396*	**63.907**	50.529
	IDF1	57.081	61.797	64.196	61.067	63.958	57.324	39.066	46.892	48.566	*67.336*	**70.531**	66.501
5	HOTA	54.731	*58.368*	51.499	51.304	56.864	46.254	39.376	52.186	49.533	57.423	**63.215**	55.254
	MOTA	53.635	50.987	46.413	53.154	52.046	43.572	45.306	49.976	47.857	*56.957*	**60.279**	49.831
	IDF1	63.303	68.331	62.971	59.716	65.846	53.764	44.724	61.451	57.998	*66.785*	**73.194**	65.563
1	HOTA	43.068	42.735	42.885	46.157	50.256	46.238	47.126	47.392	38.674	*55.465*	**59.661**	51.295
	MOTA	38.929	33.577	31.144	40.389	41.119	31.873	40.146	36.983	27.251	*48.905*	**49.392**	42.336
	IDF1	50.76	51.994	51.371	49.664	55.433	49.595	51.092	51.54	42.93	*64.821*	**69.035**	57.521

The Mask-OCSORT tracker can associate tracks better compared to OCSORT. Fig. 7 shows that the Mask-OCSORT maintained the track ID (ID 2) of the object while OCSORT assigned a different ID (ID 7) to the same vehicle.


(a)

(b)

(c)

Figure 7. Association of OCSORT and Mask-OCSORT. (a) Previous Frame (b) Next Frame (OCSORT) (c) Next Frame (Mask-OCSORT).

Moreover, partial occlusion was observed in the test video. As shown in Fig. 8, the proposed tracker was able to retain the vehicle's track ID (ID 5). The vehicle was partially occluded by other vehicles in the previous frames, despite experiencing partial occlusion and changes in appearance in subsequent frames.

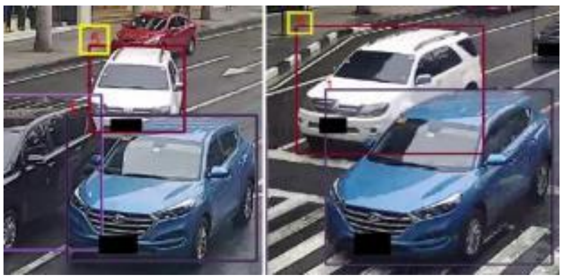


Figure 8. Partial occlusion of vehicles.

Similar-looking vehicles are common in traffic videos. For instance, in Fig. 9, two vehicles are similar-looking and after some time the second vehicle will take the position of the first vehicle. The proposed tracker ensured that the second vehicle will not take the track ID of the first vehicle by considering the computed values for IOU, appearance feature, and vehicle direction during data association.

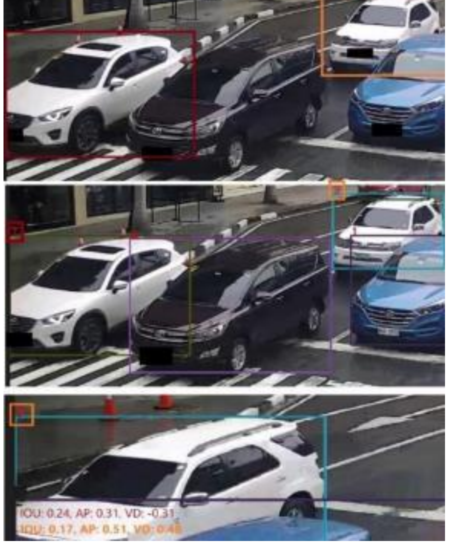


Figure 9. Similar-looking vehicles.

A. Instance Segmentation

To further investigate the result of tracking, the study evaluated the detection results of instance segmentation methods. Padilla *et al.* [46] utilizes an open-source toolkit to compute the detection's average precision. The average precision of the detection results from instance segmentation at different frame rates are presented in Table VIII.

TABLE VIII. DETECTION RESULTS OF PRE-TRAINED MODELS OF INSTANCE SEGMENTATION

Frame Rate	**No. of Cars (GT)**	**Average Precision (%)**		
		Mask R-CNN	**SOLOv2**	**Yolov7**
25	411	95.03	**95.64**	94.64
5	2077	94.79	**95.26**	94.18
1	10398	94.88	**95.08**	93.60

As shown in Table VIII, SOLOv2 has the highest average precision at all frame rates. This supports the result of evaluation on trackers, that the highest HOTA, MOTA, and IDF1 utilized the SOLOv2 model.

The processing speed of the vehicle counting method depends on the processing speed of the detection and tracking modules. Runtime (in frames per second) was the metric used by Chen *et al.* [47] to represent the processing speed of its tracker excluding the speed of the detector. In this study, both detection runtime and tracker runtime were obtained to see how the instance segmentation methods and the tracking algorithms affect the processing speed of the vehicle counting system. The detection and tracking runtime for each method was shown in Table IX. Among the instance segmentation methods, YOLOv7 has the highest runtime followed by Mask R-CNN. For the tracking methods, OCSORT achieved the highest runtime since this tracker did not use appearance features. The proposed tracker, Mask-OCSORT achieved the second-highest runtime and was faster compared to other trackers that utilized both motion and appearance features.

TABLE IX. DETECTION AND TRACKING TIME PER FRAME

Module	**Method Name**	**Runtime (frame/second)**
Instance Segmentation	Mask R-CNN	*5.057*
	SOLOv2	3.128
	YOLOv7	**22.009**
Tracker	DeepSORT	23.932
	OCSORT	**152.462**
	OCSORT+DA	21.640
	Mask-OCSORT	*65.517*

B. Single-Class Multi-object Tracking

Since the trackers utilized pre-trained models of instance segmentation methods, the authors investigated the effect of the number of classes in the tracking and counting of vehicles in a traffic video. Instead of using the five vehicle classes, the authors restricted the classification to one category, specifically cars, when performing vehicle tracking and counting. This approach was applied in the test video using SOLOv2.

Comparing the results from Tables VII and X, single-classification tracking obtained better results. Fig. 10 shows that there is an error in multi-class tracking when a vehicle has multiple classifications which results in multiple identifications. This can be attributed to the false positives observed in Tables V and VI.

TABLE X. SINGLE-CLASSIFICATION TRACKING RESULTS

Frame Rate	**Metric**	**Deep SORT**	**OC SORT**	**OCSORT-DA**	**MASK-OCSORT**
25	HOTA	87.066	**87.614**	23.366	**87.614**
	MOTA	83.679	**83.982**	30.86	**83.982**
	IDF1	91.537	**92.305**	18.906	**92.305**
	Count	27/18	23/18	12/18	23/18
5	HOTA	86.402	**88.323**	26.247	**88.323**
	MOTA	82.717	**85.446**	28.078	**85.446**
	IDF1	91.179	**92.895**	21.569	**92.895**
	Count	24/18	19/18	10/18	19/18
1	HOTA	74.588	80.984	38.089	**83.846**
	MOTA	70.213	74.468	27.052	**78.116**
	IDF1	81.62	85.857	37.542	**88.169**
	Count	12/18	11/18	11/18	12/18

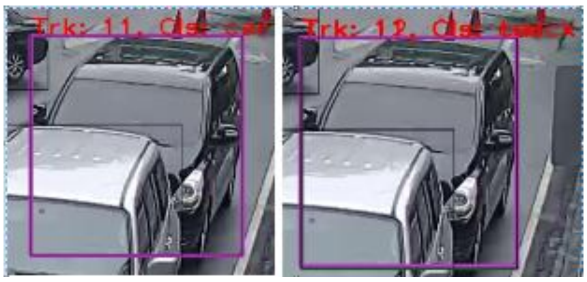


Figure 10. Single-class vs multi-class tracking.

C. Evaluation on Congested Video

Also, this research evaluated Mask-OCSORT and other trackers on a video where the movement of vehicles is slow due to congestion and stoplight, as shown in Fig. 11. The attributes of this video are the same as the previous video except for the occlusion ratios that range from 0 to 74% where heavy occlusion can be observed.

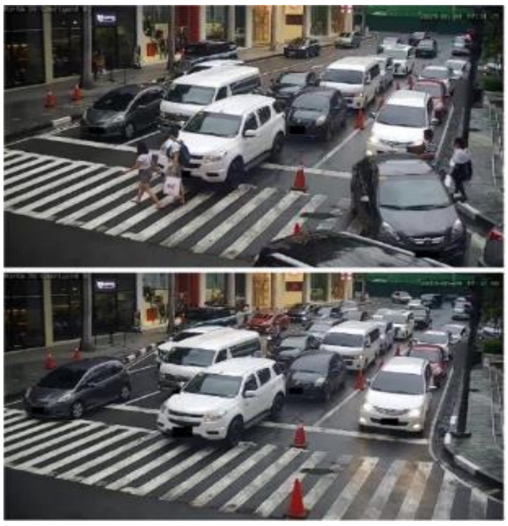


Figure 11. Congested traffic video.

Table XI shows that at different frame rates, the Mask-OCSORT with appearance features from SOLOv2 and OCSORT obtained the same metric values: HOTA, MOTA, IDF1, and count (ground truth of 18 vehicles).

Since the movement of vehicles is slow, utilizing IOU is enough to associate vehicles with their previous tracks.

TABLE XI. TRACKING RESULTS OF MASK-OCSORT AND PREVIOUS METHODS ON CONGESTED VIDEO

Frame Rate	Metric	Deep SORT	OC SORT	OCSORT-DA	MASK-OCSORT
25	HOTA	80.382	**82.101**	22.876	**82.101**
	MOTA	83.691	**83.982**	33.636	**83.982**
	IDF1	91.542	**92.304**	18.968	**92.304**
	Count	21	23	27	23
5	HOTA	58.150	60.699	19.247	**60.742**
	MOTA	37.219	38.269	1.720	**38.385**
	IDF1	63.690	68.921	15.779	**68.989**
	Count	17	32	31	32
1	HOTA	55.919	**60.127**	18.828	**60.127**
	MOTA	32.692	**39.793**	0.343	**39.793**
	IDF1	63.024	**68.425**	19.329	**68.425**
	Count	15	22	23	22

D. Hyperparameters

TABLE XII. EFFECTS OF HYPERPARAMETERS IN NORMAL-FLOW VIDEO

fps	Parameters	HOTA	MOTA	IDF1	CNT
1	λ_1=0.3	78.386	78.116	88.169	12
	λ_1=0.5	78.153	78.116	87.844	12
	λ_1=0.7	76.983	76.596	87.744	12
	M_A =10	78.175	78.723	87.884	12
	M_A =30	78.386	78.116	88.169	12
	M_A =50	78.386	78.116	88.169	12
	M_H =1	77.966	80.243	86.446	20
	M_H =3	78.175	78.723	87.884	12
	M_H =5	77.391	77.204	87.375	11
	λ_2=0.2	78.175	78.723	87.884	12
	λ_2=0.5	78.175	78.723	87.884	12
	λ_2=0.8	78.175	78.723	87.884	12
5	λ_1=0.3	82.62	85.446	92.895	19
	λ_1=0.5	82.62	85.446	92.895	19
	λ_1=0.7	82.62	85.446	92.895	19
	M_A =10	82.62	85.446	92.895	19
	M_A =30	82.62	85.446	92.895	19
	M_A =50	82.62	85.446	92.895	19
	M_H =1	82.102	84.172	92.406	19
	M_H =3	82.62	85.446	92.895	19
	M_H =5	82.986	86295	93.225	19
	λ_2=0.2	82.62	85.446	92.895	19
	λ_2=0.5	82.62	85.446	92.895	19
	λ_2=0.8	82.62	85.446	92.895	19
25	λ_1=0.3	82.063	83.994	92.31	21
	λ_1=0.5	82.063	83.994	92.31	21
	λ_1=0.7	82.063	83.994	92.31	21
	M_A =10	82.101	83.982	92.304	23
	M_A =30	82.063	83.994	92.31	21
	M_A =50	82.069	84.006	92.322	20
	M_H =1	82.026	83.824	92.226	24
	M_H =3	82.063	83.994	92.31	21
	M_H =5	82.163	84.237	92.297	20
	λ_2=0.2	82.063	83.994	92.31	21
	λ_2=0.5	82.063	83.994	92.31	21
	λ_2=0.8	82.063	83.994	92.31	21

Since this research proposed appearance features together with the IOU results for data association, this study introduces the IOU weight (λ_1) with a default value of 0.3. The authors experimented with different values for IOU weight (λ_1), velocity direction weight (λ_2), tracker's minimum hits (M_A), and tracker's maximum age (M_A). The default weights from OCSORT are M_A= 30, M_H =0.3, and λ_2=0.2.

It was shown in Table XII that processing a normal traffic video at 1 fps using λ_1 of 0.3 produced better results compared to the default value of 0.5 and higher value of IOU weight. At 5 fps and 25 fps, changing the value λ_1, while keeping other hyperparameters at default, does not affect the tracking result.

For the congested video, it was observed in Table XIII that changing the λ_1 and λ_2 does not affect the tracking accuracy at different frame rates. It can also be observed at the vehicle count of 1 fps, decreasing the value of M_H produces better results. However, it should be noted that decreasing the minimum hits of the tracker can cause more false positives. It can also be observed that increasing the value M_A at higher frame rates increases the tracking results.

TABLE XIII. EFFECTS OF HYPERPARAMETERS IN CONGESTED VIDEO

fps	Parameters	HOTA	MOTA	IDF1	CNT
1	λ_1=0.3	60.127	39.793	68.425	22
	λ_1=0.5	60.127	39.793	68.425	22
	λ_1=0.7	60.127	39.793	68.425	22
	M_A =10	60.127	39.793	68.425	22
	M_A =30	60.187	39.793	68.474	22
	M_A =50	60.187	39.793	68.474	22
	M_H =1	60.881	38.34	68.904	25
	M_H =3	60.127	39.793	68.463	22
	M_H =5	59.813	41.124	68.495	21
	λ_2=0.2	60.127	39.793	68.425	22
	λ_2=0.5	60.127	39.793	68.425	22
	λ_2=0.8	60.127	39.793	68.425	22
5	λ_1=0.3	60.742	38.385	68.989	32
	λ_1=0.5	60.742	38.385	68.989	32
	λ_1=0.7	60.742	38.385	68.989	32
	M_A =10	60.742	38.385	68.989	32
	M_A =30	60.715	38.327	68.987	27
	M_A =50	60.715	38.385	69.094	25
	M_H =1	60.989	37.802	68.989	35
	M_H =3	60.742	38.385	68.989	32
	M_H =5	60.577	38.823	68.962	30
	λ_2=0.2	60.742	38.385	68.989	32
	λ_2=0.5	60.742	38.385	68.989	32
	λ_2=0.8	60.742	38.385	68.989	32
25	λ_1=0.3	59.192	37.918	64.407	50
	λ_1=0.5	59.192	37.918	64.407	50
	λ_1=0.7	59.192	37.918	64.407	50
	M_A =10	59.192	37.918	64.407	50
	M_A =30	59.841	37.953	65.238	39
	M_A =50	60.554	37.942	66.999	33
	M_H =1	59.09	37.651	64.065	61
	M_H =3	59.255	37.918	64.407	50
	M_H =5	59.192	38.116	64.596	45
	λ_2=0.2	59.192	37.918	64.407	50
	λ_2=0.5	59.192	37.918	64.407	50
	λ_2=0.8	59.192	37.918	64.407	50

It can be seen in Tables V and VI that there are more false negatives when the frame rate is 1 fps compared to 5 fps. Fig. 12 shows the tracks of an untracked vehicle. The change in its scale and the distance it traveled per frame may affect its association score during tracking.

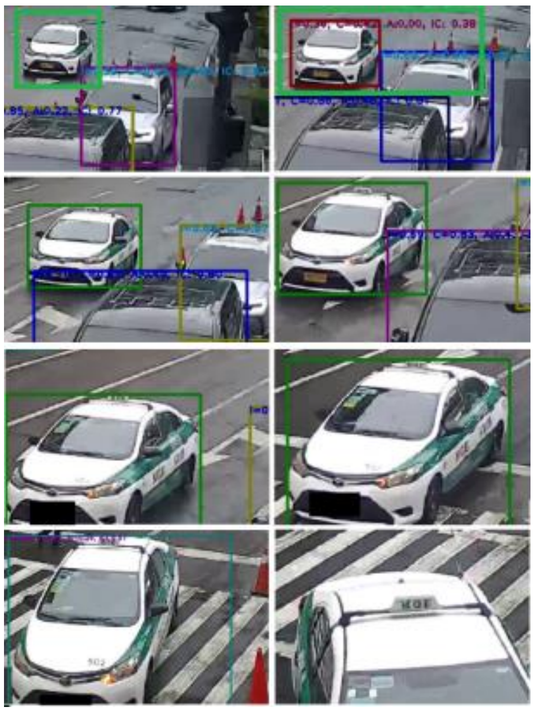


Figure 12. False negative.

An ID should be generated for the vehicle if the vehicle's association costs on these frames are greater than the tracking threshold. Fig. 13 shows that IDs were assigned to the vehicle when the value of M_H is 1.

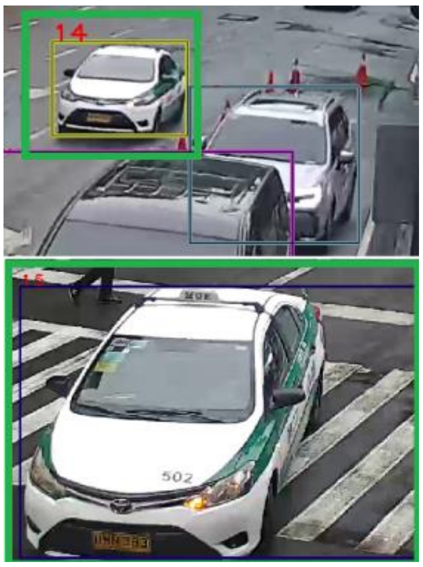


Figure 13. IDs assigned to a vehicle when MH = 1.

E. *Evaluation on UADetrac*

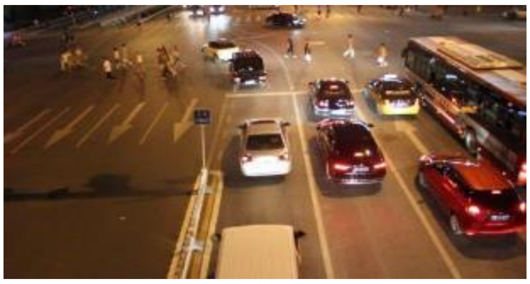
Table XIV shows the combined tracking and counting results of the proposed method evaluated on test videos from UA-Detrac [41] at different frame rates.

TABLE XIV. COMBINED TRACKING AND COUNTING RESULTS OF UA-DETRAC

Video	HOTA	MOTA	IDF1	COUNT
MVI_39031	53.467	−1.575	63.64	161/63
MVI_39051	63.779	45.576	74.583	317/173
MVI_39211	28.61	−255.899	26.296	116/80
MVI_39271	56.819	−19.7	58.497	189/153
MVI_39311	44.619	−24.911	44.651	301/212
MVI_39361	35.574	−292.292	32.337	50/38
MVI_39371	59.633	19.201	65.364	267/87
MVI_39401	44.764	−57.488	52.466	247/250
MVI_39501	37.325	−55.838	34.918	281/153
MVI_39511	40.589	−25.34	40.407	398/360
MVI_40701	55.634	−4.697	58.678	239/204
MVI_40711	37.839	−101.789	42.163	282/92
MVI_40712	64.801	21.954	70.267	229/207
MVI_40714	70.802	52.967	71.674	448/217
MVI_40742	65.885	31.873	61.837	269/119
MVI_40743	50.665	−24.317	56.4	171/133
MVI_40761	38.134	39.441	35.938	196/192
MVI_40762	54.943	24.26	55.97	246/179
MVI_40771	45.269	−65.534	50.312	208/140
MVI_40772	73.215	46.734	77.431	160/92
MVI_40773	76.263	58.507	81.5	461/388
MVI_40774	64.804	−14.346	61.698	359/161
MVI_40775	59.816	−9.056	61.18	158/192
MVI_40792	57.314	−64.742	53.965	186/152
MVI_40793	64.267	29.296	66.059	241/219
MVI_40851	53.219	−38.721	54.539	263/197
MVI_40852	61.02	19.205	67.042	393/197
MVI_40853	64.209	19.942	66.02	306/250
MVI_40854	62.865	15.95	66.162	68/88
MVI_40855	59.599	26.408	60.81	433/192
MVI_40863	44.336	−13.472	37.302	268/250
MVI_40864	52.887	30.666	53.767	254/204
MVI_40891	62.37	25.562	66.523	113/36
MVI_40892	57.621	20.124	63.033	196/95
MVI_40901	66.106	27.057	67.687	209/144
MVI_40902	44.562	−60.904	45.043	289/144
MVI_40903	60.241	39.426	64.012	205/182
MVI_40904	48.47	−45.088	46.283	167/170
MVI_40905	66.895	29.844	67.823	94/82

Fig. 14 shows the sample view of traffic videos from UA-Detrac. The first image shows a sample frame of MVI_40773, a dataset from UA-Detrac, that achieved the highest combined HOTA. The second image shows a sample frame from MVI_39211 that obtained the lowest combined HOTA.

Table XV shows tracking and counting results of the proposed method evaluated on test data from UA-Detrac [41]. It was shown that the proposed method has better tracking and counting accuracy at 25 fps and 5 fps compared to 1 fps.


(a)

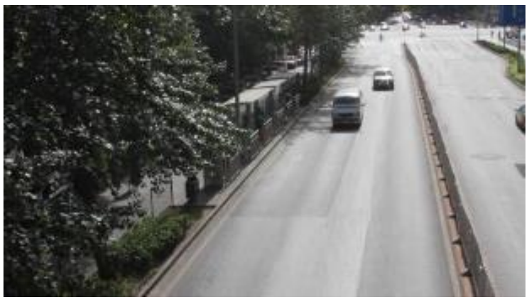

(b)

Figure 14. Sample Frames from UA-Detrac. (a) MVI_40773 (highest HOTA) (b) MVI_39211 (lowest HOTA).

TABLE XV. EVALUATION OF UA-DETRAC TEST VIDEO AT DIFFERENT FRAME RATES

fps	HOTA	MOTA	IDF1	COUNT
25	76.919	59.168	82.099	40/28
5	75.353	58.062	81.033	36/28
1	60.02	44.309	64.234	18/26

V. CONCLUSION

The paper proposes a vehicle counting method using the Mask-OCSORT, a modification of the OCSORT tracker that uses the mask prediction of instance segmentation to associate previously tracked objects with new observations. Mask-OCSORT combines the weighted sum of the metrics cosine similarity measure and IOU with the velocity consistency metric for data association. The study evaluated the proposed tracking method using the performance metrics from MOT Challenge such as HOTA, MOTA, and IDF1 Score. The evaluation shows that introducing appearance features in the data association and applying mask features produces better tracking results compared to previous trackers evaluated. The study also utilized metrics such as precision, recall, and F-score to assess the vehicle count using the proposed tracker. Compared with the previous trackers, employing Mask-OCSORT in vehicle counting can achieve better counting results.

The proposed method is dependent on the accuracy of the instance segmentation method and the authors utilized pre-trained models that limit the vehicle classification. Also, the proposed method was evaluated on a dataset with partial occlusions and medium to large scales. For future work, researchers can train their models to accommodate other vehicle classes and utilize other instance segmentation methods to improve the accuracy of tracking. Mask feature extraction methods can also be explored to improve the data association and processing time. Moreover, different camera viewpoints, vehicle speeds, levels of scales, and occlusions can be studied to improve tracking performance in more complex environments.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

A.K.B. conducted the research and wrote the paper; E.S., R.K.B., A.B., A.F., and E.D. helped draft the paper; all authors approved the final version.

ACKNOWLEDGMENT

This work is supported by the Engineering for Research and Development for Technology (ERDT) of the Department of Science and Technology (DOST) in this study.

REFERENCES

[1] M. Won, “Intelligent traffic monitoring systems for vehicle classification: A survey,” *IEEE Access*, vol. 8, pp. 73340–73358, 2020. doi: 10.1109/ACCESS.2020.2987634

[2] M. Won, T. Park, and S. H. Son, “Toward mitigating phantom jam using vehicle-to-vehicle communication,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1313–1324, May 2017. doi: 10.1109/TITS.2016.2605925

[3] B. Singh and A. Gupta, “Recent trends in intelligent transportation systems: A review,” *J. Transp. Lit.*, vol. 9, pp. 30–34, Apr. 2015. doi: 10.1590/2238-1031.jtl.v9n2a6

[4] B. Zhang and J. Zhang, “A traffic surveillance system for obtaining comprehensive information of the passing vehicles based on instance segmentation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 7040–7055, Nov. 2021. doi: 10.1109/TITS.2020.3001154

[5] Z. Kadim, K. Mohd. Johari, D. Fairol, Y. S. Li, and H. W. Hon, “Real-time vehicle counting in complex scene for traffic flow estimation using multi-level convolutional neural network,” *Int. J. Adv. Technol. Eng. Explor.*, vol. 8, no. 75, pp. 338–351, Feb. 2021. doi: 10.19101/IJATEE.2020.762128

[6] I. J. C. Valencia, E. P. Dadios, A. M. Fillone, J. C. V. Puno, R. G. Baldovino, and R. K. C. Billones, “Vision-based crowd counting and social distancing monitoring using tiny-YOLOv4 and DeepSORT,” in *Proc. 2021 IEEE International Smart Cities Conference (ISC2)*, Sep. 2021, pp. 1–7. doi: 10.1109/ISC253183.2021.9562868

[7] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, “Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1728–1740, Oct. 2008. doi: 10.1109/TPAMI.2008.73

[8] G. Lee, R. Mallipeddi, and M. Lee, “Tracking multiple moving vehicles in low frame rate videos based on trajectory information,” in *Proc. 2013 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 2013, pp. 3615–3620. doi: 10.1109/SMC.2013.616

[9] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *Proc. 2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 3464–3468. doi: 10.1109/ICIP.2016.7533003

[10] K. Khazukov *et al.*, “Real-time monitoring of traffic parameters,” *J. Big Data*, vol. 7, no. 1, 2020. doi: 10.1186/s40537-020-00358-x

[11] J. Cao, X. Weng, R. Khirodkar, J. Pang, and K. Kitani, “Observation-centric SORT: Rethinking SORT for robust multi-object tracking,” arXiv preprint, arXiv.2203.14360, 2022

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013. doi: 10.1177/0278364913491297

[13] W. Gu, S. Bai, and L. Kong, "A review on 2D instance segmentation based on deep neural networks," *Image Vis. Comput.*, vol. 120, 104401, Apr. 2022. doi: 10.1016/j.imavis.2022.104401
[14] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, Mar. 2020. doi: 10.1016/j.neucom.2019.11.023
[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," arXiv preprint, arXiv.1506.01497, 2016.
[16] W. Liu *et al.*, *SSD: Single Shot Multibox Detector*, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2
[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
[18] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.
[19] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint, arXiv.1804.02767, 2018.
[20] Z. Zhou, J. Xing, M. Zhang, and W. Hu, "Online multi-target tracking with tensor-based high-order graph matching," in *Proc. 2018 24th International Conference on Pattern Recognition (ICPR)*, Aug. 2018, pp. 1809–1814. doi: 10.1109/ICPR.2018.8545450
[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," arXiv preprint, arXiv.1703.06870, 2018.
[22] X. Chang, H. Pan, W. Sun, and H. Gao, "YolTrack: Multitask learning based real-time multiobject tracking and segmentation for autonomous vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5323–5333, Dec. 2021. doi: 10.1109/TNNLS.2021.3056383
[23] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proc. 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019, pp. 9156–9165. doi: 10.1109/ICCV.2019.00925
[24] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artif. Intell.*, vol. 293, 103448, April 2021. doi: 10.1016/j.artint.2020.103448
[25] A. Gad, T. Basmaji, M. Yaghi, H. Alheeh, M. Alkhedher, and M. Ghazal, "Multiple object tracking in robotic applications: Trends and challenges," *Appl. Sci.*, vol. 12, no. 19, Art. no. 19, Jan. 2022. doi: 10.3390/app12199408
[26] J. Wan *et al.*, "DSRRTracker: dynamic search region refinement for attention-based Siamese multi-object tracking," arXiv preprint, arXiv.2203.10729, 2022.
[27] M. Chen, Y. Liao, S. Liu, F. Wang, and J.-N. Hwang, "TR-MOT: Multi-object tracking by reference," arXiv preprint, arXiv.2203.16621, 2023.
[28] J. Hyun, M. Kang, D. Wee, and D.-Y. Yeung, "Detection recovery in online multi-object tracking with sparse graph tracker," in *Proc. 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, IEEE, Jan. 2023, pp. 4839–4848. doi: 10.1109/WACV56688.2023.00483
[29] P. Dai, Y. Feng, R. Weng, and C. Zhang, "Joint spatial-temporal and appearance modeling with transformer for multiple object tracking," arXiv preprint, arXiv.2205.15495, 2022.
[30] T. Basar, "A new approach to linear filtering and prediction problems," in *Control Theory: Twenty-Five Seminal Papers (1932-1981)*, IEEE, 2001, pp. 167–179. doi: 10.1109/9780470544334.ch9
[31] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logist. Q.*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955. doi: 10.1002/nav.3800020109
[32] L. Chen, H. Ai, C. Shang, Z. Zhuang, and B. Bai, "Online multi-object tracking with convolutional neural networks," in *Proc. 2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 645–649. doi: 10.1109/ICIP.2017.8296360
[33] X. Wang. (2022). SOLO: Segmenting objects by location. [Online]. Available: https://github.com/WXinlong/SOLO
[34] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," arXiv preprint, arXiv.1612.03144, 2017.
[35] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic and fast instance segmentation," arXiv preprint, arXiv.2003.10152, 2020.
[36] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," arXiv preprint, arXiv.2207.02696, 2022.
[37] C.-W. Chuang and C.-P. Fan, "Deep-learning based joint iris and sclera recognition with YOLO network for identity identification," *J. Adv. Inf. Technol.*, vol. 12, no. 1, pp. 60–65, 2021. doi: 10.12720/jait.12.1.60-65
[38] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," arXiv preprint, arXiv.1703.07402, 2017.
[39] I. Perera *et al.*, "Vehicle tracking based on an improved deepsort algorithm and the YOLOv4 framework," in *Proc. 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*, Aug. 2021, pp. 305–309. doi: 10.1109/ICIAfS52090.2021.9606052
[40] L. Zheng *et al.*, *MARS: A Video Benchmark for Large-Scale Person Re-identification*, Oct. 2016, pp. 868–884. doi: 10.1007/978-3-319-46466-4_52
[41] L. Wen *et al.*, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," arXiv preprint, arXiv.1511.04136, 2020.
[42] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," arXiv preprint, arXiv.1405.0312. 2015.
[43] P. Dendorfer *et al.*, "MOTChallenge: A benchmark for single-camera multiple target tracking," *Int. J. Comput. Vis.*, vol. 129, no. 4, pp. 845–881, Apr. 2021. doi: 10.1007/s11263-020-01393-0
[44] J. Luiten. (2022). TrackEval. [Online]. Available: https://github.com/JonathonLuiten/TrackEval
[45] J. Luiten *et al.*, "HOTA: A higher order metric for evaluating multi-object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 548–578, Feb. 2021. doi: 10.1007/s11263-020-01375-2
[46] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, Art. no. 3, Jan. 2021. doi: 10.3390/electronics10030279
[47] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1840–1852, Mar. 2021. doi: 10.1109/TITS.2020.3025687