Real-Time Vehicle Detection and Counting Using YOLOv8 and ByteTrack Multi-Object Tracking on Surveillance Cameras

[1] Filantropi Yusuf Aji Cahyono, [2] Raden Arief Setiawan, [3] Angger Abdul Razak
[123] Department of Electrical Engineering, Universitas Brawijaya, Malang, Indonesia
[1]filantropiy@gmail.com, [2]rarief@ub.ac.id, [3]aarazak@ub.ac.id

Abstract - Vehicle counting in tourist area parking facilities is typically performed manually, leading to counting errors and the inability to provide real-time capacity updates. This study proposes an automated real-time vehicle detection and counting system integrating YOLOv8s and ByteTrack Multi-Object Tracking on surveillance camera footage. The dataset consists of 1,377 images across two vehicle classes prepared through a data-leakage-free pipeline. Evaluation results show that the YOLOv8s model achieved precision of 0.904, recall of 0.961, F1-score of 0.930, and mAP@0.5 of 0.969. Vehicle counting evaluation over a 30-minute test video yielded a counting accuracy of 96.875%, MAE of 2.25, and MAPE of 3.125%. The system operated at an average processing speed of 37.82 FPS, exceeding the real-time threshold of 25–30 FPS. These results indicate that the proposed system has the potential to serve as a technically feasible prototype for automated real-time parking capacity monitoring under controlled daytime conditions.

Keywords — ByteTrack, Multi-Object Tracking, Parking Management, Surveillance Camera, Vehicle Detection, YOLOv8s.

I. Introduction

The tourism sector in Indonesia has experienced significant growth from year to year. The increasing number of tourists has directly impacted the growing volume of vehicles in tourist areas. According to data from the Central Statistics Agency (BPS), motor vehicle ownership in Indonesia grew at an average rate of 4.23% per year during the 2021–2022 period [1]. This condition has made the need for adequate and well-managed parking facilities increasingly urgent.

A common problem in tourist areas is the imbalance between parking capacity and the number of arriving vehicles. The parking index can approach full capacity during peak visiting hours, potentially causing long queues and visitor discomfort [2]. Manual parking management has proven to have several weaknesses, including recording errors and delays in communicating parking availability information to visitors [3].

Advances in computer vision and deep learning technology have opened significant opportunities to address these problems. Camera-based object detection systems can automatically recognize and count vehicles without human intervention [4]. The You Only Look Once (YOLO) method, particularly YOLOv8 released by Ultralytics in January 2023, has proven efficient for real-time applications with high mAP values and excellent inference speed [5].

A primary challenge in vehicle counting systems is the double-counting problem caused by the system's inability to recognize object identities across video frames. ByteTrack is a Multi-Object Tracking (MOT) algorithm that addresses this issue through a two-stage association mechanism that utilizes all detection boxes including low-confidence ones, resulting in more accurate tracking [6].

Several previous studies have been conducted in this field. Liang et al. (2022) proposed an improved YOLOv5 for real-time vehicle detection achieving competitive accuracy, but without an integrated tracking algorithm to prevent double counting [7]. Zuraimi and Kamaru Zaman (2021) used YOLOv4 combined with DeepSORT achieving AP50 of 82.08%; however, DeepSORT requires an additional re-identification model that increases computational load [8]. Rofii et al. (2021) achieved 93% precision and 94% recall using YOLOv4, but performance degraded under nighttime conditions [9]. Huyen et al. (2026) applied YOLOv8 for vehicle identification and classification in urban environments, achieving accuracy of 93–98%; however, the study focused solely on identification and classification tasks without incorporating a vehicle counting or parking capacity management mechanism [10]. Bilous et al. (2024) demonstrated the superiority of YOLOv8 with mAP@0.5 of 0.88 and 48 FPS but without tracking integration [11]. Sharma et al. (2023) used YOLOv8 combined with DeepSORT but focused on parking time violations rather than capacity management [12].

Based on the review of previous studies, several research gaps can be identified. Studies employing YOLOv8 have demonstrated strong detection performance but lack an integrated tracking mechanism, leaving them vulnerable to double counting errors. Meanwhile, tracking-based approaches such as DeepSORT introduce additional computational overhead due to their re-identification modules, limiting their suitability for real-time deployment. Furthermore, none of the reviewed studies addressed real-time parking capacity management as their primary objective. To address these gaps, this study proposes an integrated system combining YOLOv8s as the object detector and ByteTrack as a computationally lightweight yet accurate tracking algorithm. Unlike DeepSORT-based approaches,

ByteTrack achieves robust multi-object tracking without requiring a separate re-identification model, making it more suitable for real-time applications. The proposed system further incorporates a real-time parking capacity monitoring mechanism, distinguishing it from prior works that focused solely on vehicle detection or classification tasks.

II. Research Method

2.1 Research Design

The research methodology was carried out in three main stages as illustrated in Figure 1. The first stage began with a literature study and data understanding. The second stage was the pre-processing stage, which included data collection, data sorting and cleaning, dataset splitting into training data (70%), validation data (20%), and testing data (10%), data augmentation applied exclusively to the training subset to prevent data leakage, data annotation using LabelImg, and configuration through a data.yaml file. The third stage involved testing and evaluating the YOLOv8s model using precision, recall, F1-score, and mAP metrics.

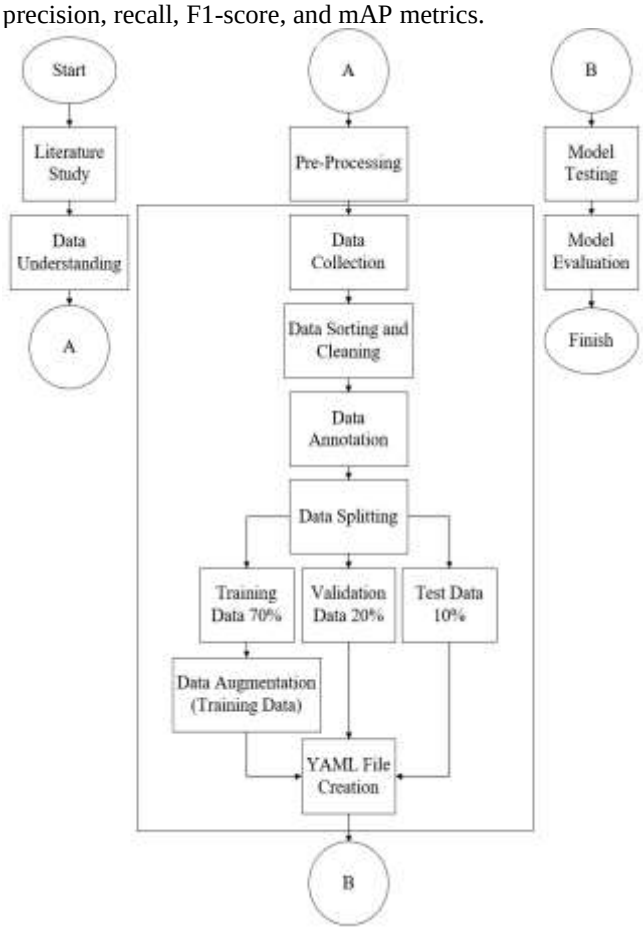

Figure 1. Research Flow Diagram

2.2 Dataset and Data Collection

The research dataset consists of two object classes, namely cars and motorcycles. Images were collected from both CCTV footage and smartphone recordings simulating a surveillance camera viewpoint. Both sources were pooled into a single dataset prior to splitting, with no distinction made between sources during the training, validation, or testing subset assignment. Data were collected through photography and video recording using a smartphone camera positioned to simulate a surveillance camera (CCTV) viewpoint. The original dataset comprised 390 car images and 420 motorcycle images, totaling 810 images. To prevent data leakage, the entire original dataset was first split into three subsets prior to any augmentation: training data (567 images, 70%), validation data (162 images, 20%), and testing data (81 images, 10%). Horizontal flip augmentation was then applied exclusively to the training subset to increase data variation and improve model generalization, doubling the training set from 567 to 1,134 images. The final dataset composition consisted of 1,134 training images, 162 validation images, and 81 testing images, for a total of 1,377 imagesIt is acknowledged that the original dataset of 810 images was collected from a single location under controlled daytime conditions, which may limit the generalizability of the trained model to other environments, camera angles, or lighting conditions. This limitation should be addressed in future work by expanding the dataset to include more diverse locations, weather conditions, and times of day.

2.3 Dataset Annotation

Dataset annotation was performed using LabelImg, a graphical annotation tool that supports the YOLO annotation format. Each image was manually annotated by drawing bounding boxes around vehicle objects and assigning the corresponding class labels. The annotation results were saved in text file format (.txt), containing the class number, bounding box center coordinates (x_center, y_center), and normalized width and height values in the range of 0 to 1.

2.4 Model Training Parameters

The YOLOv8s model was trained using a dataset configured through a data.yaml file. Training was conducted for 50 epochs with a batch size of 16, meaning each training iteration processed 16 images simultaneously before updating the model weights. The input image size was set to 640 × 640 pixels in accordance with the YOLOv8 standard input specification. The training process utilized GPU acceleration on an NVIDIA GeForce RTX 4050 via CUDA 12.1, significantly reducing training time compared to CPU-based processing. The training parameters used in this study are summarized in Table 1.

Table 1. YOLOv8s Model Training Parameters

Parameter	Value
Epochs	50
Batch Size	16
Image Size (imgsz)	640 × 640 pixel
Framework	Ultralytics YOLOv8 v8.3.91
Hardware Acceleration	NVIDIA RTX 4050 (CUDA 12.1)

2.5 Research Tools and Materials

This study utilized a number of hardware and software components to support the data collection process, model training, and implementation of the vehicle detection and counting system. The details of the tools and materials used are presented in Table 2.

Table 2. Research Tools and Materials

No.	Category	Name	Specification/ Version
1.	Hardware	Laptop	Intel Core i5-13500HX
2.	Hardware	GPU	NVIDIA GeForce RTX 4050
3.	Hardware	RAM	24 GB
4.	Hardware	Camera	CCTV & Smartphone
5.	Software	Python	3.8.0
6.	Software	Ultralytics YOLOv8	8.3.91
7.	Software	OpenCV	4.11.0
8.	Software	Jupyter Notebook	-
9.	Software	LabelImg	-
10.	Software	Operating System Windows 11	-

2.6 ByteTrack Implementation

ByteTrack was integrated into the YOLOv8s inference pipeline via the Ultralytics built-in tracking API using the default bytetrack.yaml configuration file. The key parameter values of the default bytetrack.yaml configuration used in this study are summarized in Table 3.

Table 3. ByteTrack Default Configuration Parameters

Parameter	Value	Description
track_high_thresh	0.25	Minimum confidence for high-confidence detections in first association stage
track_low_thresh	0.1	Minimum confidence for low-confidence detections in second association stage
new_track_thresh	0.25	Minimum confidence threshold to initialize a new track
track_buffer	30	Number of frames to buffer before removing a lost track

The Linear Assignment Problem (LAP) solver library (lap v0.5.12) was used as ByteTrack's Hungarian Algorithm component. ByteTrack operates through a two-stage association process: the first stage matches high-confidence detection boxes (track_high_thresh = 0.25) with active tracklets using a Kalman Filter, and the second stage utilizes low-confidence detection boxes (track_low_thresh = 0.1) to recover temporarily occluded objects. New tracks are initialized when detections do not match any existing tracks (new_track_thresh = 0.25), with a track buffer of 30 frames to determine track removal timing. Each detected vehicle is assigned a persistent unique ID by ByteTrack, stored in a dictionary-based tracking memory (track_memory) to consistently monitor centroid movement across frames. ByteTrack has been shown to achieve superior tracking performance compared to Re-ID-based methods while maintaining lower computational cost, making it well-suited for real-time integrated detection and counting pipelines [13].

2.7 Virtual Counting Line and Counting Logic

The virtual counting line was fixed at 70% of the frame height (line_y = frame_h × 0.7). The 70% position was selected based on visual observation of vehicle movement patterns within the recorded frames, where this position consistently captured complete vehicle crossing events for both entry and exit directions. This selection was not derived from a systematic sensitivity analysis across multiple line positions, and is therefore acknowledged as a limitation of the current study. Future work should include an ablation study evaluating the effect of varying line positions on counting accuracy to determine the optimal threshold empirically. A vehicle is counted as entering when its centroid moves from above to below the line (prev_y < line_y and cy ≥ line_y), and counted as exiting when the opposite transition occurs. Each vehicle ID is counted at most once per line crossing event, effectively preventing double counting. The remaining parking capacity is calculated using the following equation:

$$Remaining\ Capacity = Initial\ Capacity - (Vehicles\ Entered - Vehicles\ Exited) \quad (1)$$

2.8 Evaluation Metrics

System performance was evaluated using the following metrics:

a. Object Detection Metrics:

Precision measures the proportion of correct positive detections out of all positive detections made by the model, calculated as follows:

$$Precision = TP / (TP + FP) \quad (2)$$

Recall measures the proportion of actual positive objects that were successfully detected by the model, calculated as follows:

$$Recall = TP / (TP + FN) \quad (3)$$

F1-Score represents the harmonic mean of Precision and Recall, providing a balanced measure between the two metrics:

$$\text{F1-Score} = 2 \times (Precision \times Recall) / (Precision + Recall) \quad (4)$$

b. Counting Accuracy Metrics:

Counting Accuracy measures the percentage of correctly counted vehicles per category, averaged across all categories:

$$CA = (N_{GT} - |N_{GT} - N_{Sys}|) / N_{GT}) \times 100\% \quad (5)$$

Mean Absolute Error (MAE) measures the average absolute difference between ground truth and system count per category:

$$MAE = (1/n)\ \Sigma\ |N_{GT} - NSys| \quad (6)$$

Mean Absolute Percentage Error (MAPE) measures the average percentage error between ground truth and system count per category:

$$MAPE = (1/n)\ \Sigma\ (|NGT - NSys| / NGT) \times 100\% \quad (7)$$

where N_{GT} is the ground truth count, N_{Sys} is the system count, and n is the number of vehicle categories.

III. Results and Discussion

3.1 Model Training Results

The training process over 50 epochs demonstrated a consistent decrease in both training loss and validation loss values, as shown in Figure 2. On the training data, box_loss decreased from approximately 0.67 to 0.20, while cls_loss decreased significantly from 1.50 to approximately 0.20, indicating the model successfully learned to distinguish between car and motorcycle classes. The dfl_loss also decreased steadily from 1.05 to 0.80, reflecting improved bounding box regression precision. On the validation data, val/box_loss stabilized at approximately 0.25, val/cls_loss converged to approximately 0.40, and val/dfl_loss remained stable at approximately 0.80 toward the end of training. No significant increase in validation loss was observed throughout the training process, confirming that the model converged optimally without overfitting on the dataset used.

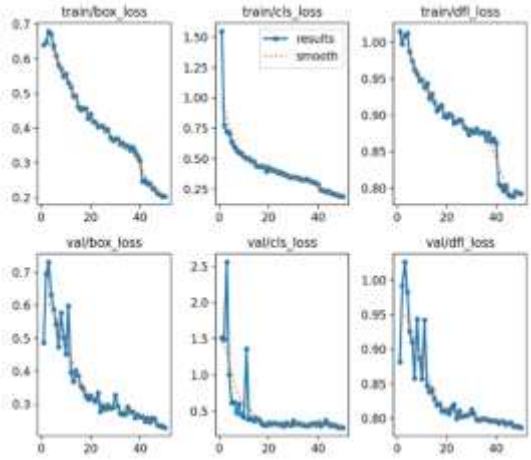

Figure 2. Training and Validation Loss Graphs over 50 Epochs

3.2 Evaluation Metrics Development

Figure 3 shows the development of evaluation metrics across 50 training epochs. During the first 10 epochs, precision, recall, mAP@0.5, and mAP@0.5-0.95 exhibited moderate fluctuation, consistent with early-stage weight adjustment. After epoch 15, all four metrics stabilized and showed consistent improvement until the final epoch. Precision stabilized at approximately 0.90, recall converged at approximately 0.93–0.96, mAP@0.5 reached approximately 0.97, and mAP@0.5-0.95 stabilized at approximately 0.95, reflecting successful model convergence toward the end of training. This pattern of early fluctuation followed by stable convergence is consistent with typical YOLOv8 training behavior reported in previous studies [14].

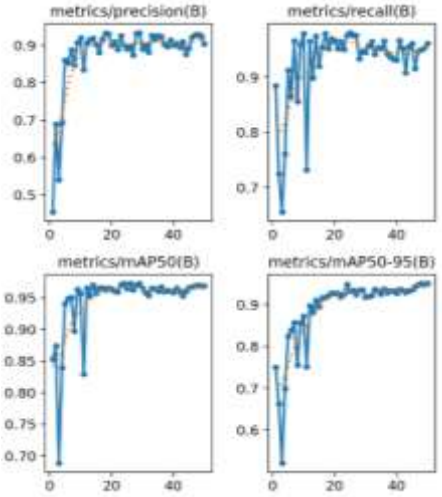

Figure 3. Evaluation Metrics Development Graph (Precision, Recall, mAP@0.5, mAP@0.5-0.95)

3.3 Detection Performance Results

The final detection performance of the YOLOv8s model evaluated on the test set is presented in Table 4.

Table 4. YOLOv8s Model Detection Performance per Class

Class	Precision	Recall	mAP@0.5	mAP@0.5-0.95
Car	0.943	1.000	0.995	0.992
Motorcycle	0.865	0.922	0.943	0.908
All Classes	**0.904**	**0.961**	**0.969**	**0.950**

The model achieved overall precision of 0.904 and recall of 0.961, with mAP@0.5 of 0.969 and mAP@0.5-0.95 of 0.950. The car class achieved near-perfect detection with recall of 1.000 and mAP@0.5 of 0.995, indicating that all car objects in the test set were successfully detected without any missed detections. The motorcycle class achieved slightly lower recall of 0.922 and mAP@0.5 of 0.943, attributed to the smaller physical size and greater viewing angle variation of motorcycles compared to cars. Similar trends of reduced detection performance for smaller vehicle classes in camera-based surveillance systems have also been reported in previous counting studies [15].

Figure 4 shows the Precision-Recall curve for both vehicle classes. The car class consistently occupies the upper-right region with mAP@0.5 of 0.995, while the motorcycle class yields mAP@0.5 of 0.943. The large area under both curves confirms strong detection capability across varying confidence thresholds.

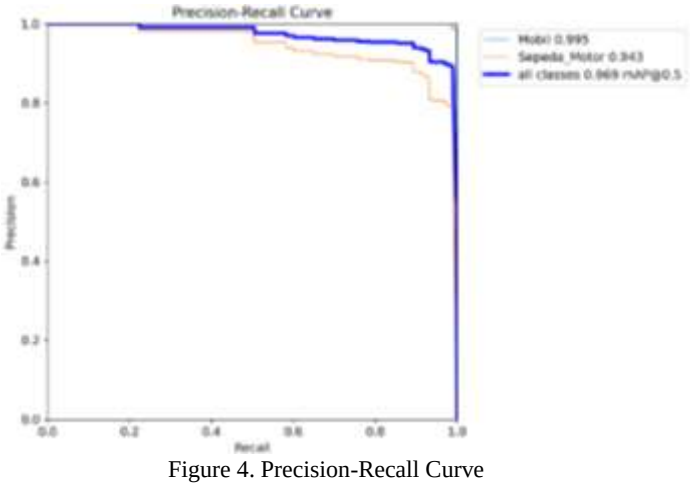

Figure 4. Precision-Recall Curve

Figure 5 shows the Precision-Confidence curve for both vehicle classes. The car class maintained consistently high precision above 0.85 across the full confidence range, reaching near-perfect precision of 1.00 at high confidence thresholds. The motorcycle class showed slightly lower but stable precision across most of the confidence range, with precision values converging toward the car class at higher confidence levels. The overall model achieved precision of 1.00 at a confidence threshold of 0.979 across all classes.

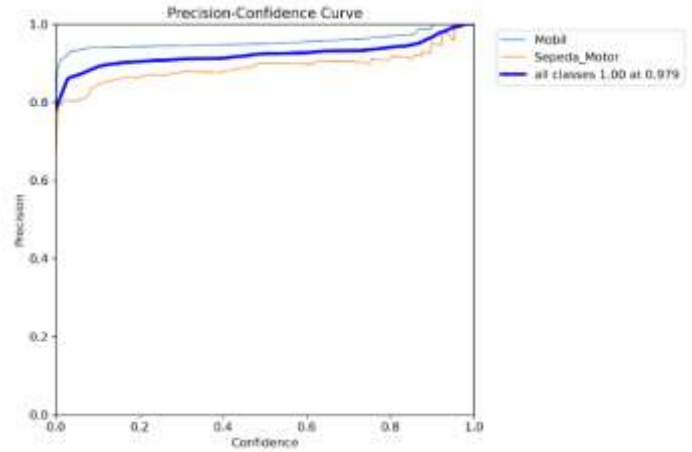

Figure 5. Precision-Confidence Curve

Figure 6 shows the F1-Confidence curve, illustrating the relationship between the F1-score and the confidence threshold. The model achieved the highest overall F1-score of 0.93 at an optimal confidence threshold of 0.473, indicating an optimal balance between precision and recall at this threshold. The car class exhibited a higher F1 curve compared to motorcycles across almost the entire confidence range, consistent with the per-class evaluation results in Table 4.

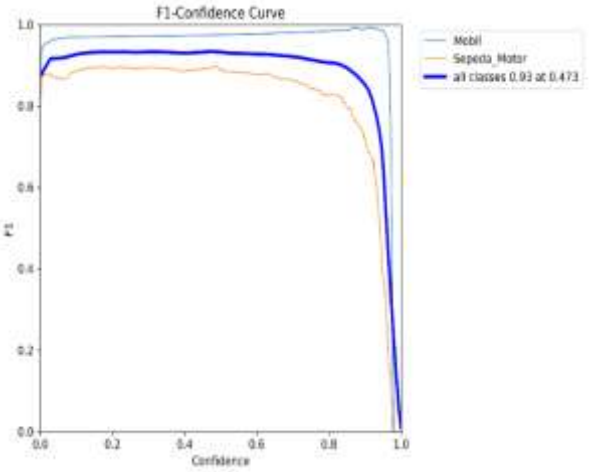

Figure 6. F1-Confidence Curve

Figure 7 shows the confusion matrix of the YOLOv8s model. The results show that 78 out of 78 car objects (100%) were correctly classified with zero misclassifications between classes. For motorcycles, 94 out of 103 objects (91.3%) were correctly classified, while 9 instances were misclassified as background. Additionally, 4 background instances were falsely predicted as cars and 14 as motorcycles, attributable to background objects with visual similarity to vehicles such as walls, poles, and stationary objects.

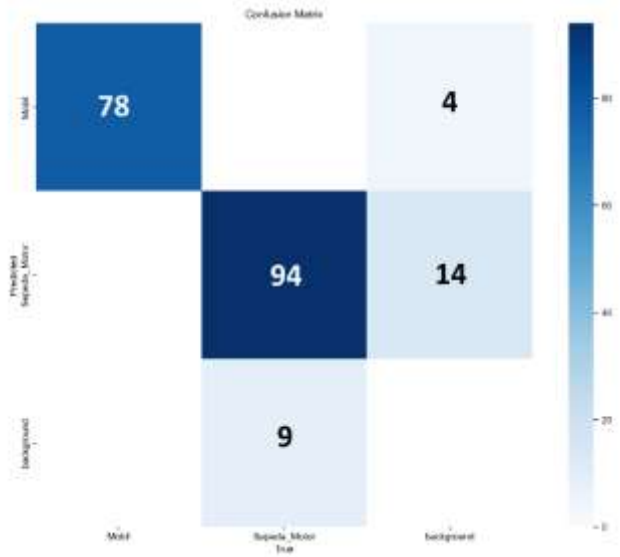

Figure 7. Confusion Matrix

3.4 Scenario Testing Results

The system was evaluated across six controlled testing scenarios to assess functional performance under isolated conditions. The quantitative results for each scenario are presented in Table 5.

Table 5. Scenario-wise Quantitative Results

No	Scenario	GT	System Count	False Count	Missed Count	Avg FPS
1.	No vehicle	0	0	0	0	37.82
2.	Single motorcycle entering	1	1	0	0	37.82
3.	Motorcycle simultaneous entry and exit	2	2	0	0	37.82
4.	Car and motorcycle simultaneous entry and exit	2	2	0	0	37.82
5.	Single car exiting	1	1	0	0	37.82
6.	Single car entering	1	1	0	0	37.82
Total		**7**	**7**	**0**	**0**	**37.82**

Across all six controlled testing scenarios, the system achieved perfect counting accuracy with zero false counts, zero missed counts, and zero ID switches, demonstrating reliable performance under isolated single-event conditions. The average processing speed of 37.82 FPS was consistent across all scenarios, confirming stable real-time performance. The consistent average FPS of 37.82 across all scenarios was computed as the mean processing speed over the entire video duration. The reported value reflects the stabilized processing speed after GPU warm-up, which explains the consistency across scenarios. Short-duration measurements may yield lower FPS values due to the initial GPU warm-up period before processing speed stabilizes.

These results indicate that the system correctly handles all basic vehicle movement patterns including single vehicle entry, single vehicle exit, and simultaneous multi-vehicle movements. However, it is acknowledged that these scenarios contain only seven total crossing events under controlled conditions, which is insufficient to statistically justify the reliability of the system under real-world operational settings. These controlled scenarios are therefore intended solely to verify the functional correctness of individual system components, including entry counting, exit counting, and simultaneous multi-vehicle handling, rather than to demonstrate field robustness. A more comprehensive evaluation involving larger-scale real-world testing is needed in future work. Figure 8 to Figure 13 illustrate representative frames from each testing scenario, showing bounding boxes, ByteTrack unique IDs, the virtual counting line, and real-time parking capacity overlay.

Figure 8. No-vehicle Condition (system stability test)

Figure 9. Motorcycles Entering

Figure 10. Motorcycle Simultaneous Entry and Exit

Figure 11. Car and Motorcycle Simultaneous Entry and Exit

Figure 12. Single Car Exiting

Figure 13. Single Car Entering

Figure 14. Vehicle Counting Accumulation Display

3.5 Vehicle Counting Results

The vehicle counting evaluation was conducted using a 30-minute test video, with results presented in Table 6.

Table 6. Vehicle Counting Results: Ground Truth vs System

Vehicle	Direction	Ground Truth	System Count	Absolute Error
Car	Entering	7	7	0
Car	Exiting	11	11	0
Motorcycle	Entering	72	79	7
Motorcycle	Exiting	72	70	2
Total		**162**	**167**	**9**

Table 7. Counting Accuracy Metrics

Metric	Value
Counting Accuracy	96.875%
MAE (Mean Absolute Error)	2.25
MAPE (Mean Absolute Percentage Error)	3.125%

The system achieved a counting accuracy of 96.875% with MAE of 2.25 and MAPE of 3.125% over a 30-minute test video. Cars were counted with perfect accuracy (0% error) for both entry and exit directions, consistent with the model's perfect recall of 1.000 for the car class. Motorcycles exhibited higher counting errors, with 7 false counts in the entering direction and 2 missed counts in the exiting direction. The false counts in motorcycle entering were primarily caused by pedestrians occasionally misclassified as motorcycles due to visual similarity at certain camera viewing angles. No pedestrian exclusion mechanism was implemented in the current system. This represents a notable limitation, as the absence of such a mechanism directly contributes to false counts in high-pedestrian environments. Future work should consider incorporating a pedestrian detection class or exclusion filter to mitigate this issue. The missed counts in motorcycle exiting were caused by temporary occlusion near the virtual counting line when a motorcycle was obscured by an adjacent car. The MAPE value of 3.125% remains within the accurate category (below 5%), confirming the system's viability as a prototype for real-time parking capacity monitoring under controlled daytime conditions. It should be noted that although the test video was recorded on a different day from the training data collection, both were captured at the same location and under similar daytime lighting conditions. Therefore, these results should be interpreted as performance within a controlled single-location daytime scope, and generalizability to other environments, camera angles, or lighting conditions remains to be validated in future work.

IV. Conclusion

The following conclusions are drawn from the results of this research:

1. A real-time vehicle detection and counting system was successfully designed and implemented by integrating YOLOv8s as the object detector and ByteTrack as the Multi-Object Tracking algorithm on surveillance camera footage. The system was built through a data-leakage-free pipeline in which dataset splitting preceded augmentation, and utilized a virtual counting line at 70% of the frame height combined with ByteTrack's persistent unique ID mechanism to prevent double counting while displaying real-time parking capacity information.
2. The YOLOv8s model achieved strong detection performance with precision of 0.904, recall of 0.961, F1-score of 0.930 at an optimal confidence threshold of 0.473, mAP@0.5 of 0.969, and mAP@0.5-0.95 of 0.950. The car class achieved near-perfect detection with recall of 1.000

and mAP@0.5 of 0.995, while the motorcycle class achieved recall of 0.922 and mAP@0.5 of 0.943.

3. Vehicle counting evaluation over a 30-minute test video yielded a counting accuracy of 96.875%, MAE of 2.25, and MAPE of 3.125%, confirming the system's viability as a technically feasible prototype for automated real-time parking capacity monitoring under controlled daytime conditions.
4. For future research, the system should be evaluated under nighttime and adverse lighting conditions, expanded with a more diverse dataset covering varied camera angles and vehicle densities, and deployed directly at actual tourist area parking gates to validate performance under real operational conditions. Integration of pedestrian exclusion mechanisms and license plate recognition could further enhance the system's capability for comprehensive parking management.

V. References

[1] Kepolisian Republik Indonesia, "Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis (Unit), 2021-2022," Badan Pusat statistik. Accessed: Mar. 05, 2024. [Online]. Available: https://www.bps.go.id/id/statistics-table/2/NTcjMg==/perkembangan-jumlah-kendaraan-bermotor-menurut-jenis--unit-.html

[2] M. Alymani, L. A. Almoqhem, D. A. Alabdulwahab, A. A. Alghamdi, H. Alshahrani, and K. Raza, "Enabling smart parking for smart cities using Internet of Things (IoT) and machine learning," PeerJ Comput. Sci., pp. 1–27, 2025, doi: 10.7717/peerj-cs.2544.

[3] A. Saadeldin, M. M. Rashid, A. A. Shafie, and T. F. Hasan, "Real-time vehicle counting using custom YOLOv8n and DeepSORT for resource-limited edge devices," vol. 22, no. 1, pp. 104–112, 2024, doi: 10.12928/TELKOMNIKA.v22i1.25096.

[4] D. S. Rajesh, T. Vaibhavi, T. Sreeja, V. Gayathri, and S. Srivalli, "Computer vision application : Vehicle counting and classification system from real time videos," Turkish J. Comput. Math. Educ., vol. 14, no. 03, pp. 79–89, 2023.

[5] Z. Huang, L. Li, G. C. Krizek, and L. Sun, "Research on Traffic Sign Detection Based on Improved YOLOv8," pp. 226–232, 2023, doi: 10.4236/jcc.2023.117014.

[6] Y. Zhang et al., "ByteTrack : Multi-Object Tracking by Associating Every Detection Box," in European Conference on Computer Vision (ECCV), Cham: Springer, 2022.

[7] Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, and X. Guo, "Real-Time Vehicle Detection Based on Improved YOLO v5," Sustainability, vol. 14, 2022, doi: 10.3390/su141912274.

[8] M. A. Bin Zuraimi and F. H. Kamaru Zaman, "Vehicle detection and tracking using YOLO and DeepSORT," ISCAIE 2021 - IEEE 11th Symp. Comput. Appl. Ind. Electron., pp. 23–29, 2021, doi: 10.1109/ISCAIE51753.2021.9431784.

[9] S. A. Rofii Faqih, Priyandoko Gigih, Fanani M. Ifan, "Vehicle Counting Accuracy Improvement By Identity Sequences Detection Based on Yolov4 Deep Neural Networks," Teknik, vol. 42, no. 2, pp. 169–177, 2021, doi: 10.14710/teknik.v42i2.37019.

[10] Q. Ly Thi Huyen, C., Pham Quyet, C., & Thi Nguyen, "Vehicle Identification and Classification Using YOLO Algorithm," IIUM Eng. J., vol. 27, no. 1, pp. 251–262, 2026, doi: 10.31436/iiumej.v27i1.3694.

[11] N. Bilous, V. Malko, M. Frohme, and A. Nechyporenko, "Comparison of CNN-Based Architectures for Detection of Different Object Classes," AI, vol. 5, no. 4, pp. 2300–2320, 2024, doi: 10.3390/ai5040113.

[12] N. Sharma, S. Baral, M. P. Paing, and R. Chawuthai, "Parking Time Violation Tracking Using YOLOv8 and Tracking Algorithms," Sensors, vol. 23, pp. 1–14, 2023.

[13] Y. Wang and V. Y. Mariano, "A Multi object Tracking Framework Based on YOLOv8s and Bytetrack Algorithm," IEEE Access, vol. PP, p. 1, 2024, doi: 10.1109/ACCESS.2024.3450370.

[14] R. Varghese and M. Sambath, "YOLOv8 : A Novel Object Detection Algorithm with Enhanced Performance and Robustness," 2024 Int. Conf. Adv. Data Eng. Intell. Comput. Syst., pp. 1–6, 2024, doi: 10.1109/ADICS58448.2024.10533619.

[15] M. Majumder and C. Wilmot, "Automated Vehicle Counting from Pre-Recorded Video Using You Only Look Once (YOLO) Object Detection Model," J. Imaging, vol. 2, p. 19, 2023, doi: https://doi.org/10.3390/jimaging9070131.