PARKING TIME VIOLATION TRACKING USING DEEP LEARNING MODELS

NABIN SHARMA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ROBOTICS AND COMPUTATIONAL INTELLIGENCE SYSTEMS

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2023

KMITL-2023-EN-M-407-174

Thesis Title	Parking Time Violation Tracking Using Deep Learning Models
Student	Nabin Sharma
Student ID	64601008
Degree	Master of Engineering
Program	Robotics and Computational Intelligence Systems
Year	2023
Thesis Advisor	Asst. Prof. Dr. Rathachai Chawuthai

ABSTRACT

The proposed approach in this paper aims to address the primary issue of time violations in parking areas in Thailand. The current solution involves CCTV surveillance cameras and human monitoring, but this paper suggests a new and affordable system that combines CCTV, deep learning models, and object tracking algorithms. The system utilizes YOLOv8, a state-of-the-art object detection model, along with the DeepSORT/OC-SORT algorithm for efficient detection and tracking of vehicles. By incorporating these advanced techniques, the system can set a timer and track time violations accurately.

The integration of deep learning models and tracking algorithms within a single framework contributes to improved system performance. The paper's results section demonstrates the effectiveness of both tracking algorithms, achieving impressive MOTA scores (Multiple Object Tracking Accuracy) across four different surveillance datasets. For DeepSORT, the scores were (1.0, 1.0, 0.96, 0.90), and for OC-SORT, the scores were (1.0, 0.76, 0.90, 0.83). Overall, this proposed approach presents a promising solution for addressing time violations in parking areas by leveraging advanced object detection, tracking algorithms, and deep learning techniques. Its affordability and potential for accurate monitoring make it a compelling alternative to the existing methods.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Assistant Professor Dr. Rathachai Chawuthai, for his invaluable advice, guidance, and support throughout the entire duration of this study. His expertise and insightful counseling have played a pivotal role in helping me progress towards my research goals. I am truly grateful for his exceptional support throughout the research process. In addition, I would like to thank the office of Information Technology of School of Engineering, KMITL for the video footage.

I would also like to extend my appreciation to Mr. Sushish Baral and Dr. May Phu Paing for their unwavering support and encouragement throughout this journey. Their valuable input and encouragement have been instrumental in shaping the direction of my research. Their presence and assistance have been a constant source of support whenever I required it. Finally, I want to express my appreciation to my parents, family members, relatives, and friends who have supported and encouraged me throughout this journey. Their unwavering support and motivation have been invaluable. I am deeply grateful to everyone who directly or indirectly contributed to the successful completion of this research.

Table of Contents

INTRODUCTION **8**

1.1 BACKGROUND 8

1.2 PROBLEM DESCRIPTION 9

1.3 RESEARCH OBJECTIVE 9

1.4 SCOPE OF STUDY 9

1.5 EXPECTED CONTRIBUTION 10

BACKGROUND AND LITERATURE REVIEW **11**

2.1 OBJECT CLASSIFICATION 11

2.1.1 Steps for Object Classification *11*

2.1.2 Evaluation Metrics *17*

2.2 OBJECT DETECTION 18

2.2.1 Steps for Object Detection *19*

2.2.2 Evaluation Metrics *23*

2.3 OBJECT TRACKING 24

2.3.1 Steps for Object Tracking *25*

2.3.2 Evaluation Metrics *27*

2.4 LITERATURE REVIEW 28

2.4.1 Object Detection *29*

2.4.2 Object Tracking *31*

METHODOLOGY **34**

3.1 OVERALL PROCESS 34

3.2 DATASET 35

3.3 OBJECT DETECTION 37

3.4 OBJECT TRACKING 42

3.5 TIME VIOLATION 44

3.6 EVALUATION METHOD 46

RESULT AND DISCUSSION **49**

4.1 EXPERIMENTAL SETUP 49

4.2 RESULT 50

4.3 DISCUSSION 53
CONCLUSION 56
REFERENCES 57
AUTHOR BIOGRAPHY 60
APPENDIX A 61
APPENDIX B 72

List of Figures

Figure 1.1 Time restriction of 15 min applied to parking for consumers....8
Figure 2.1 Example of Object Classification [4]....11
Figure 2.2 Max Pooling vs Average Pooling [6]....13
Figure 2.3 Residual Block Architecture [9] 14
Figure 2.4 AlexNet Architecture [10] 15
Figure 2.5 Inception Architecture[10] 17
Figure 2.6 Object Detection Steps [1] 18
Figure 2.7 Types of Object Detectors [11] 19
Figure 2.8 YOLO [12] 20
Figure 2.9 SSD Architecture [11] 20
Figure 2.10 Architecture of Regional Based - CNN [11] 22
Figure 2.11 Intersection over Union (IOU) [13] 24
Figure 2.12 Comparison of Tracking Algorithms 27
Figure 3.1 High Level Process of our proposed work 35
Figure 3.2 Sample Camera used for our dataset 36
Figure 3.3 Detailed Architecture of YOLOv8 37
Figure 3.4 Surveillience video from Site 1 [11] 41
Figure 3.5 Surveillience video from Site 2 [11] 41
Figure 3.6 Surveillience video from Site 3 [11] 41
Figure 3.7 Surveillience video from Site 4 [11] 42
Figure 3.8 Workflow of our Proposed Algorithm [11] 43
Figure 3.9 Pipeline of OC-SORT 44
Figure 3.10 Flowchart of Time Violation Tracking 45
Figure 3.11 Showing Vehicle in 3 Frames 47
Figure 4.1 Sample images of tracking by our model [11] 50
Figure 4.2 Performance of DeepSORT for Location 2 Dataset [11] 52
Figure 4.3 Performance of OCSORT for Location 2 Dataset [11] 53

List of Tables

Table 2.1 Results on the DanceTract test Set 33
Table 3.1 Definition and Description of TP,FP, TN,FN and IDS 47
Table 4.1 Description of FP, FN and IDS.....51
Table 4.2 Performance of YOLOv8 with DeepSORT 50
Table 4.3 Performance of YOLOv8 with OCSORT 52

Chapter 1

Introduction

1.1 Background

Parking violations encompass actions like parking a vehicle in a restricted area or exceeding time limits. Figure 1.1 illustrates an example of implementing a time-restricted parking rule at a mini-mart in Thailand. Many shopping malls and mini-marts utilize closed-circuit television (CCTV) systems in their parking lots to monitor time-related parking violations. Presently, parking violations are manually inspected by relevant authorities. However, these methods are costly and involve high labor expenses due to the constant surveillance required for tracking incoming and outgoing vehicles. To tackle the parking space issue, various existing parking solutions have integrated Internet of Things (IoT) devices to provide drivers with real-time parking information [2]. Nonetheless, employing sensors and hardware, while ensuring accuracy, demands continuous upkeep, rendering it an impractical solution for mini-marts. Thus, there remains a need for an affordable parking violation detection system.

The majority of current research efforts primarily focus on aiding drivers in locating available parking spaces. Numerous models have been developed to assist drivers in finding nearby parking spots in real time [3]. However, there is a scarcity of studies aimed at helping authorities efficiently manage parking spaces. Few investigations have delved into parking violations and time restrictions.

Figure 1.1 Time restriction of 15 min applied to parking for consumers

1.2 Problem Description

The issue of parking problems in Thailand, particularly in cities, is indeed a growing concern due to the increasing number of vehicular registrations [4] . In Thailand. Shopping malls and mini-marts, including popular ones like Tesco Express, 7-Eleven, and Mini Big C, are facing challenges in effectively managing their limited parking spaces. In a fast-paced environment like 7-Eleven, where parking facilities are limited, it becomes crucial to track the parking time of each vehicle in the parking lot. By monitoring parking durations, businesses can ensure fair usage of the available spaces and discourage parking violations such as occupying spots for an extended period. This issue has drawn the attention of mini-mart executives and the general public.

1.3 Research Objective

The research aims to achieve the following objectives in the context of parking time violation tracking:

- Utilize an object detection algorithm to identify vehicles through CCTV cameras in minimart parking lots.
- Employ an object tracking algorithm to continuously track the detected vehicles across multiple frames, assigning each vehicle a unique ID for tracking purposes.
- Implement a time violation tracker to classify the tracked cars into two categories: those that have violated parking time limits and those that have not violated them.

These objectives demonstrate a comprehensive approach to addressing parking time violations through the integration of object detection, object tracking, and time violation classification methods.

1.4 Scope of Study

This study concentrates on tracking parking time violations within minimarts using CCTV cameras. It employs the advanced YOLOv8 model for object detection and utilizes two distinct object tracking algorithms, namely DeepSORT and OCSORT. The study is limited to the use of a single CCTV camera. It specifically focuses on a small parking area associated with a

convenience store, thus excluding multi-camera detection and tracking. The research evaluates the performance of DeepSORT and OCSORT by comparing them across various metrics.

1.5 Expected Contribution

This research introduces an innovative approach to detect parking time violations in fast-paced environments like 7-11 and Tesco Express parking lots. This approach appears to be the first attempt to track violations based on time in parking lot scenarios. The goal is to develop a system capable of identifying parking time violations using predefined time thresholds, making it applicable in diverse parking lot settings.

Chapter 2

Background and Literature Review

This discussion will delve into the existing literature surrounding object detection and object tracking, both of which are crucial components of effective parking time violation tracking systems.

2.1 Object Classification

The task at hand is object classification, where the objective is to determine the class or object type of a given image from a predefined set of known classes. For instance, if we have a binary classifier designed to classify images into two classes such as elephant or giraffe, we can input an image to this classifier and inquire about the probability that the image belongs to the elephant class (or the giraffe class). This can be better understood through the example illustrated in Figure 2.1.

Figure 2.1 Example of Object Classification [5]

2.1.1 Steps for Object Classification

Due to the rapid advancement of Convolutional Neural Networks architecture, CNN is mostly used for Object Classification tasks. The general steps for object classification are divided into two basic steps:

2.1.1.1 Feature Extraction

In a Convolutional Neural Network (CNN), feature extraction refers to the process of automatically learning discriminative features from input images. This step is crucial for the success of CNNs in image classification tasks. In feature extraction, the input image is convolved with a set of learnable filters or kernels. Each filter performs a dot product between its weights and a local patch of the input image, producing a feature map that captures different patterns and textures.

2.1.1.2 Feature Selection

After the feature extraction in a Convolutional Neural Network (CNN), the feature maps are commonly passed through pooling operations to aggregate and down-sample the information. Two popular pooling techniques are Max Pooling and Average Pooling (Mean Pooling). Here's a brief explanation of these techniques:

- **Max Pooling:** Max Pooling divides the input feature map into non-overlapping regions and selects the maximum value within each region. By retaining the maximum value, it captures the strongest or most dominant features present in each local neighborhood. Max Pooling helps in achieving translation invariance, where the position of the feature in the region becomes less relevant.
- **Average (Mean) Pooling**: Average Pooling, as the name suggests, calculates the average value within each region of the input feature map. It computes the mean value, providing a summary of the overall strength or intensity of the features in the local neighborhood. Average Pooling can help in reducing noise and capturing more general information about the features.

Both Max Pooling and Average Pooling serve the purpose of down-sampling the feature maps, reducing the spatial dimensions, and summarizing the information in a more compact form. This down-sampling aids in reducing the computational complexity of subsequent layers and extracting higher-level features that are more robust and invariant to small spatial variations. Once feature extraction and selection has been performed, the features are converted to 1D vector and passed through Dense layers, the output of which in turn can be passed through

SoftMax layer to get the probabilities of the object in the image that belongs to each class. Figure 2.2 shows the working illustration of Max Pooling and Average Pooling.

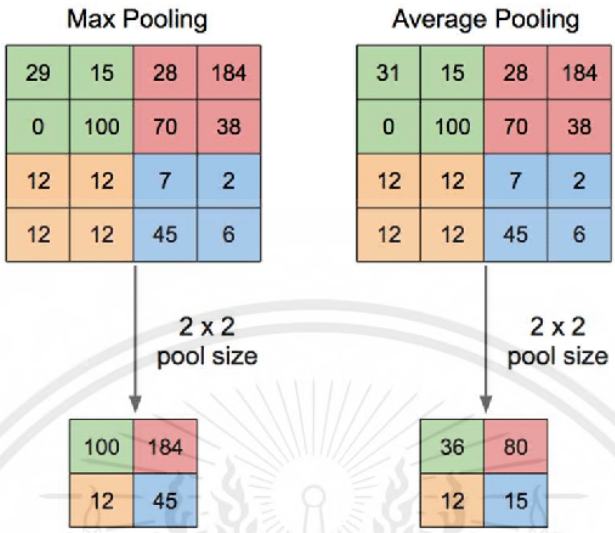


Figure 2.2 Max pooling vs Average Pooling [6].

ResNet: ResNet, short for Residual Network, is a deep neural network architecture that addresses the problem of vanishing gradients in very deep networks. It was introduced by Kaiming He et al. in 2015 [7] and has been widely adopted in various computer vision tasks, including image classification, object detection, and image segmentation. The key innovation of ResNet is the introduction of residual connections, also known as skip connections or shortcut connections. These connections allow the network to learn residual mappings, capturing the difference between the desired output and the current output of a layer. By propagating this difference through the network, ResNet enables the training of extremely deep networks without suffering from degradation in performance. Here are some key characteristics and components of the ResNet architecture:

1. Residual Blocks: The building blocks of ResNet are residual blocks, which consist of multiple convolutional layers with a residual connection that adds the input to the output of the block. This bypass connection allows the gradient to flow directly through the network, mitigating the problem of vanishing gradients.
2. Identity Shortcut: In ResNet, when the input and output of a residual block have the same dimensions, the shortcut connection is simply an identity mapping. This allows the network to learn residual functions instead of completely relearning the mapping from scratch.

3. Bottleneck Architecture: ResNet often uses a bottleneck architecture in its deeper variants, where each residual block contains a bottleneck layer with 1x1, 3x3, and 1x1 convolutions. This reduces the computational cost while maintaining the representational capacity of the network.

4. Pre-activation ResNet: The original ResNet introduced the concept of pre-activation, where batch normalization and ReLU activation are applied before the convolutional layers. This ordering helps in alleviating the problem of gradient degradation and improves the overall performance of the network.

ResNet has achieved state-of-the-art performance in various image classification benchmarks, such as ImageNet. Its deep architecture, aided by residual connections, allows the network to capture and learn complex hierarchical features effectively.

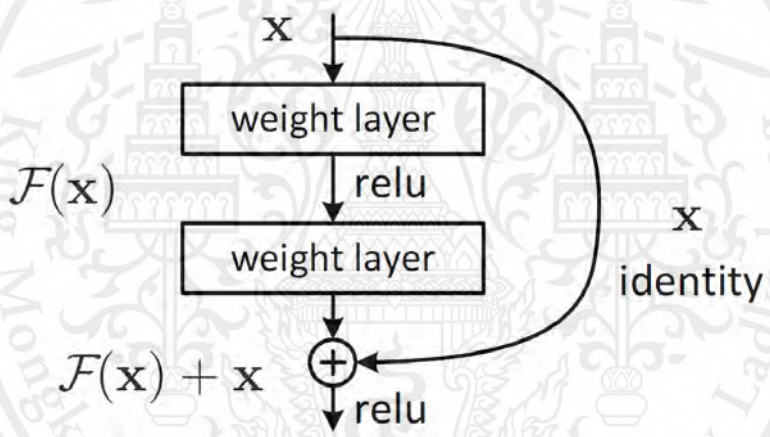


Figure 2.3 Residual Block Architecture [9].

AlexNet : AlexNet is a deep convolutional neural network architecture that played a significant role in advancing the field of computer vision. It was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton [8] . This architecture won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, significantly outperforming other competing approaches.The key features and components of AlexNet include:

1. Architecture: AlexNet consists of eight layers, including five convolutional layers followed by three fully connected layers. It has a large number of learnable parameters, contributing to its capacity to learn complex features.

2. Convolutional Layers: The first convolutional layer in AlexNet performs convolution with 96 filters of size 11x11, followed by max pooling. Subsequent convolutional layers use smaller

filter sizes, such as 5x5 and 3x3, with increased depth. These layers extract features at different levels of abstraction.

3. Activation Function: AlexNet uses the rectified linear unit (ReLU) activation function, which helps alleviate the vanishing gradient problem and speeds up training.

4. Local Response Normalization (LRN): LRN is applied after the ReLU activation in the early convolutional layers of AlexNet. It normalizes the responses within local neighborhoods to enhance the network's ability to generalize and respond to variations in input.

5. Overlapping Max Pooling: Max pooling is applied after each of the first two convolutional layers. Unlike traditional non-overlapping pooling, AlexNet employs overlapping pooling with a stride of 2, resulting in a denser feature map and reduced spatial dimensions.

6. Dropout: AlexNet incorporates a dropout regularization technique, specifically applied to the fully connected layers. Dropout randomly sets a fraction of the activations to zero during training, which helps prevent overfitting and improves generalization.

7. Fully Connected Layers: The last three layers of AlexNet are fully connected layers with a large number of neurons. The final fully connected layer produces the class probabilities using the SoftMax activation function.

AlexNet's success in the ILSVRC marked a breakthrough in using deep convolutional neural networks for image classification tasks. Its architecture, utilization of ReLU activations, overlapping pooling, dropout regularization, and large-scale training on GPUs contributed to its superior performance and paved the way for subsequent advancements in deep learning for computer vision.

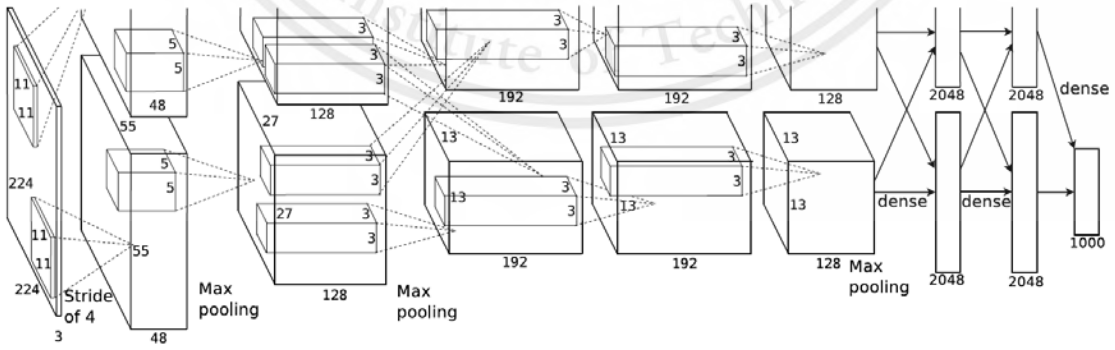


Figure 2.4 AlexNet Architecture [10].

Inception: Inception is a deep convolutional neural network architecture introduced by Christian Szegedy et al. in 2014 [9]. It was developed to address the challenge of designing

networks that are both deep and computationally efficient. Inception networks have been highly successful in various computer vision tasks, including image classification and object detection. The key idea behind the Inception architecture is the use of multiple parallel convolutional operations at each layer, allowing the network to capture different scales or levels of abstraction within the same layer. This multi-scale processing helps in efficiently learning features at various resolutions and helps the network adapt to objects of different sizes in the input image. Here are some key characteristics and components of the Inception architecture:

1. Inception Modules: The building blocks of Inception networks are Inception modules. An Inception module consists of parallel convolutional operations of different kernel sizes (e.g., 1x1, 3x3, 5x5) and max pooling operations. By concatenating the output feature maps of these parallel operations, the network captures information at multiple scales.
2. 1x1 Convolutions: Inception networks extensively use 1x1 convolutions within the Inception modules. These 1x1 convolutions serve two primary purposes: dimension reduction (reducing the number of input channels) and feature combination. By reducing the dimensionality before applying larger convolutions, the network reduces computational complexity.
3. Bottleneck Layers: To further reduce computational cost, Inception networks employ bottleneck layers, which use 1x1 convolutions to reduce the number of input channels before applying larger convolutions. This bottleneck layer helps in reducing the number of parameters and improves efficiency.
4. Auxiliary Classifiers: Inception networks often include auxiliary classifiers at intermediate layers during training. These auxiliary classifiers consist of additional convolutional and fully connected layers and provide additional gradients during backpropagation. They encourage the network to learn more general features and help in combating the vanishing gradient problem.
5. Grid-Reduction: To handle spatial reduction in the network, Inception networks use grid-reduction modules. These modules include max pooling and 1x1 convolutions to reduce spatial dimensions while maintaining feature richness.

By incorporating the Inception modules with parallel convolutions and 1x1 convolutions for dimension reduction, Inception networks strike a balance between depth and computational efficiency. This allows for efficient learning of complex features at different scales, making

them well-suited for tasks that involve objects of varying sizes or resolutions. Notable variants of Inception include Inception v1, v2, v3, and Inception-ResNet, which have shown impressive performance in various image classification benchmarks.

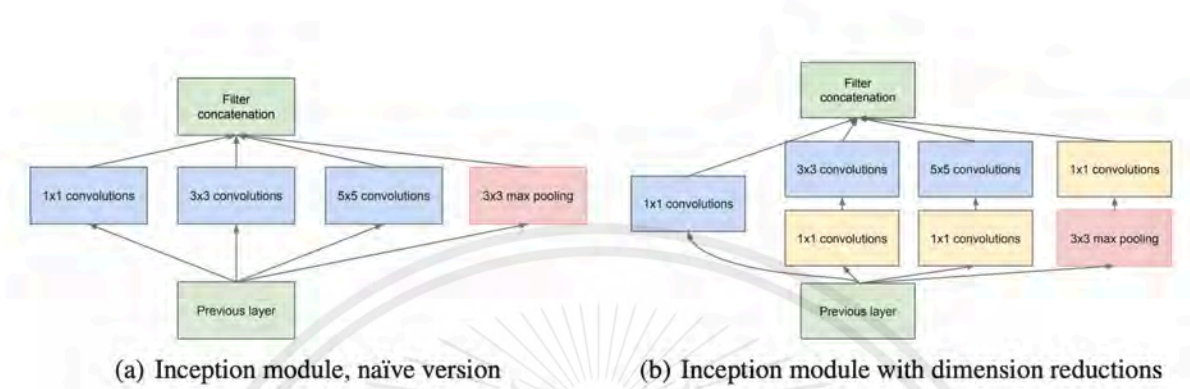


Figure 2.5 Inception Architecture [10]

2.1.2 Evaluation Metrics

When evaluating object classification models, several metrics are commonly used to assess their performance. Here are some widely used evaluation metrics for object classification:

1. Accuracy: Accuracy is the most basic and straightforward metric, representing the proportion of correctly classified samples out of the total samples. It is calculated by dividing the number of correctly classified samples by the total number of samples.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

2. Precision: Precision measures the proportion of correctly predicted positive samples out of all samples predicted as positive. It focuses on the correctness of positive predictions. It is calculated as the ratio of true positives (TP) to the sum of true positives and false positives (FP).

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

3. Recall (Sensitivity or True Positive Rate): Recall measures the proportion of correctly predicted positive samples out of all actual positive samples. It focuses on capturing all

positive samples without missing any. It is calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (FN).

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

4. F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of both precision and recall. It is calculated as:

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

These evaluation metrics help assess the performance of object classification models from different perspectives, considering both correct predictions and errors. The choice of metrics depends on the specific requirements and priorities of the classification problem at hand.

2.2 Object Detection

Object detection is a computer vision task that involves locating and classifying objects of interest within an image or a video. Unlike object classification, which focuses on determining the class of an entire image, object detection aims to identify and localize multiple objects within an image, often by drawing bounding boxes around them. Figure 2.6 shows the object detection steps.

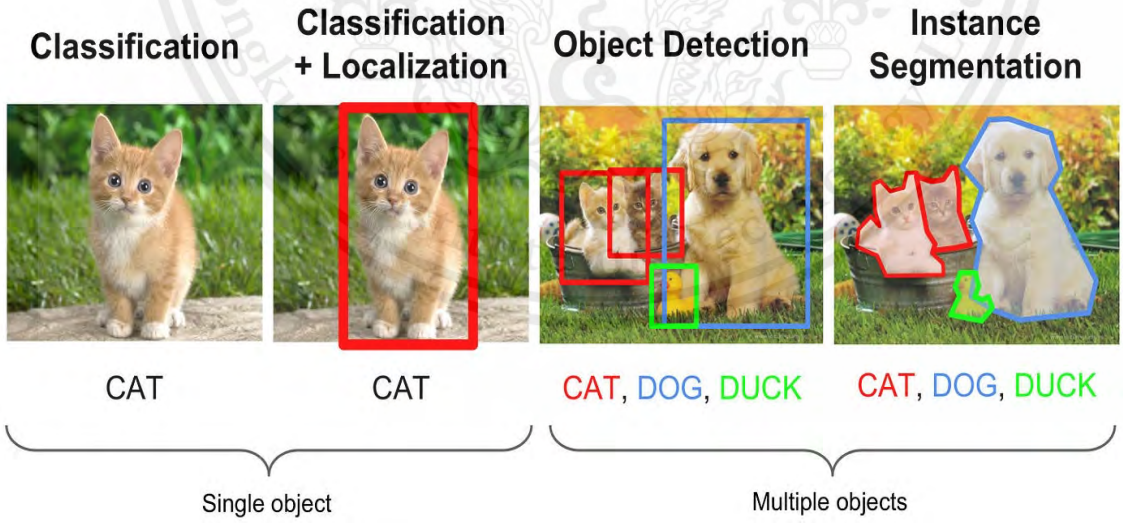


Figure 2.6 Object Detection Steps [1]

2.2.1 Steps for Object Detection

Over the past few years, object detection has gained lot of popularity in the field of Deep Learning. Hence, there are famously two types of object detectors i.e. One Stage Detectors and Two Stage Detectors. Figure 2.7 shows the types of object detection algorithm.

One Stage Detectors: One-stage detectors are a type of object detection model used in computer vision tasks. Unlike two-stage detectors, which involve a region proposal stage followed by object classification and refinement, one-stage detectors directly predict bounding boxes and class probabilities in a single pass. One-stage detectors are known for their simplicity and efficiency. They typically use a convolutional neural network (CNN) architecture, such as YOLO (You Only Look Once) or SSD (Single Shot MultiBox Detector), to process the input image and predict object bounding boxes and their corresponding class labels.

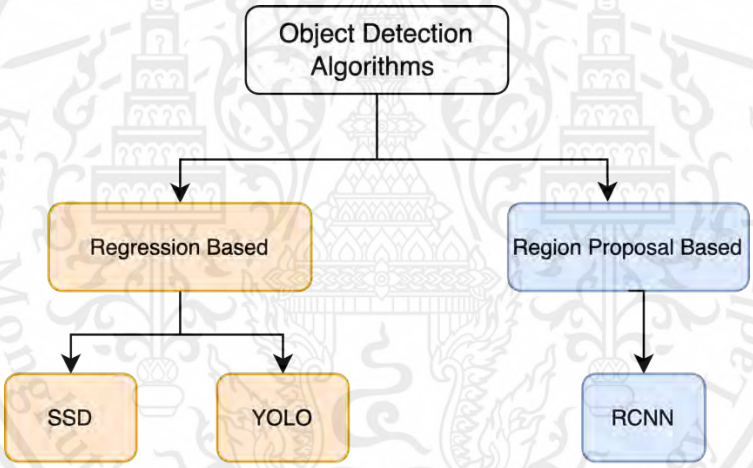


Figure 2.7 Types of Object Detectors [11]

YOLO is one of the popular one-stage detectors. It divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell. YOLO can handle multiple object classes simultaneously and is known for its real-time performance, making it suitable for applications that require fast object detection. Figure 2.8 shows the working method of YOLO. The detailed architecture of YOLOv8 is discussed in the next section.

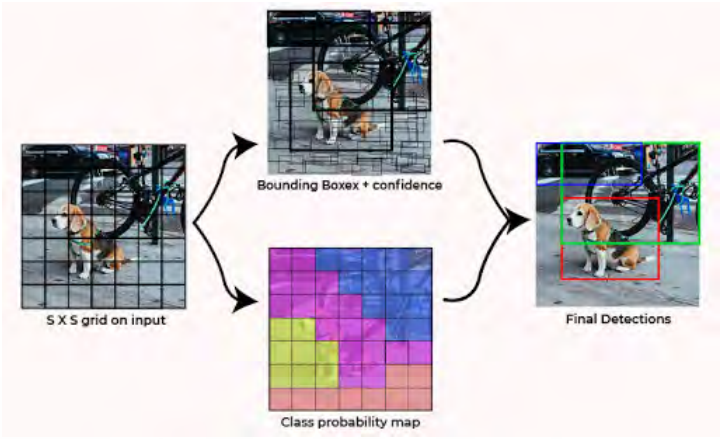


Figure 2.8 YOLO [12]

SSD is another widely used one-stage detector. It employs a series of convolutional layers with different scales to detect objects at various sizes. By using feature maps from multiple scales, SSD can detect objects with different aspect ratios and scales more effectively. Figure 2.9 shows the architecture of SSD. Here's an overview of how the SSD architecture works:

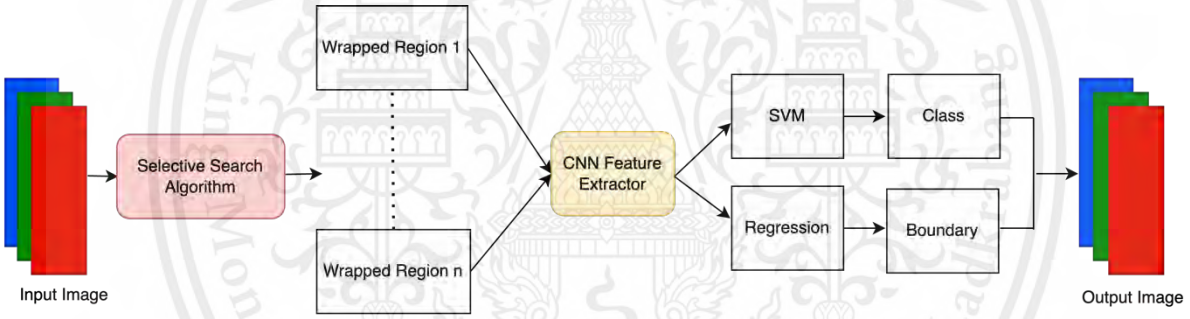


Figure 2.9 SSD Architecture [11]

1. Input Image: The SSD takes an input image of fixed size as its input.

2. Base Network: The input image is passed through a base network, such as VGG16 or ResNet, which extracts a set of feature maps at different scales. These feature maps capture rich spatial information at different levels of abstraction.

3. Convolutional Layers: The SSD architecture adds a series of convolutional layers on top of the base network. These layers are designed to further process the feature maps and generate a set of predictions at multiple scales.

4. Convolutional Predictors: At each spatial location in the feature maps, the SSD uses convolutional layers with different kernel sizes to predict the presence of objects and their corresponding class labels. These convolutional predictors output a set of confidence scores

for each class, indicating the likelihood of the presence of an object of that class at that location.

5. Default Boxes/Anchors: The SSD uses predefined default boxes, also known as anchors, at each spatial location in the feature maps. These default boxes have different aspect ratios and scales, allowing the model to handle objects of various shapes and sizes. The default boxes are used as reference templates for generating bounding box predictions.

6. Bounding Box Predictors: For each default box, the SSD predicts four values: the offsets for the box's center coordinates and the width and height relative to the default box's size and aspect ratio. These predictions are used to adjust and refine the positions and sizes of the default boxes to match the objects present in the image.

7. Multi-Scale Predictions: The SSD makes predictions at multiple scales using feature maps from different layers of the network. This enables the model to detect objects of different sizes and handle objects with varying aspect ratios effectively.

8. Non-Maximum Suppression (NMS): After obtaining the predicted bounding boxes and confidence scores, the SSD applies non-maximum suppression to filter out redundant detections. NMS selects the highest-confidence bounding boxes and suppresses overlapping detections that exceed a certain threshold.

The SSD architecture allows for efficient and real-time object detection by performing detection in a single pass through the network. Its multi-scale predictions and default boxes enable it to handle objects of various sizes and aspect ratios. By combining these components, the SSD achieves accurate object detection while maintaining computational efficiency.

One-stage detectors have advantages in terms of simplicity, speed, and real-time performance. They are often used in scenarios where fast object detection is required, such as autonomous driving, surveillance systems, and robotics applications. However, they may struggle with detecting small objects and achieving high accuracy compared to two-stage detectors, which typically have a region proposal stage for refining the detection results.

Two stage Detectors: Two-stage detectors are a type of object detection model used in computer vision tasks. Unlike one-stage detectors, which directly predict bounding boxes and class probabilities in a single pass, two-stage detectors involve a two-step process: region proposal and object classification/refinement. As shown in Figure 2.10, an overview of two stage detectors are discussed below:

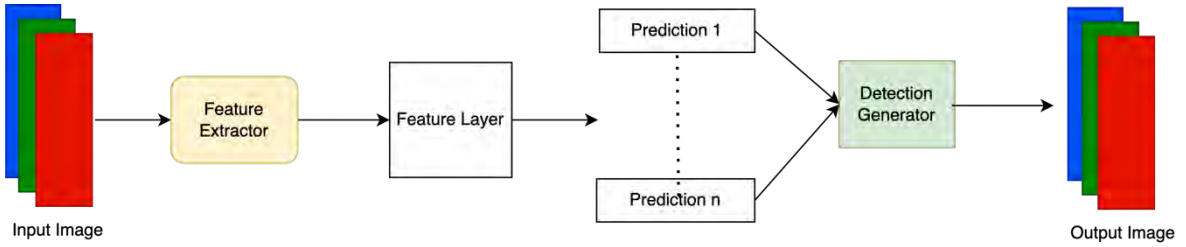


Figure 2.10 Architecture of Regional Based - CNN [11]

1. Region Proposal: The first stage of a two-stage detector is responsible for generating region proposals or candidate bounding boxes that potentially contain objects of interest. It aims to reduce the search space for object detection by proposing a set of regions that are likely to contain objects. Popular region proposal algorithms include Selective Search and Region Proposal Network (RPN).

2. Region of Interest Pooling: The proposed regions are cropped from the feature maps generated by a backbone network (e.g., VGG16, ResNet) using a technique called pooling. pooling resizes the proposed regions to a fixed size and aligns them with the spatial grid of the feature maps, ensuring consistent input sizes for subsequent processing.

3. Feature Extraction: The cropped regions are then passed through additional layers of the network to extract features specific to each region. This process typically involves convolutional layers and can include other operations like pooling or normalization.

4. Object Classification/Refinement: In the second stage, the extracted features from the ROI's are used for object classification and bounding box regression. The classifier predicts the class probabilities for each ROI, indicating the likelihood of different object categories being present. Simultaneously, the regressor refines the bounding box coordinates of each proposed region to better align with the ground truth object boundaries.

5. Non-Maximum Suppression (NMS): After obtaining the predicted bounding boxes and class probabilities, non-maximum suppression is applied to remove redundant detections. NMS selects the highest-confidence bounding boxes and suppresses overlapping detections that exceed a certain threshold.

Two-stage detectors excel at precise localization and achieving high accuracy in object detection. The initial region proposal stage helps to focus on potential object regions, reducing the computational burden of processing the entire image. By separating the region proposal and object classification/refinement steps, two-stage detectors can effectively handle objects

at different scales and achieve improved detection accuracy. However, they tend to be computationally more expensive than one-stage detectors. Popular two-stage detectors include Faster R-CNN, R-FCN, and Mask R-CNN.

2.2.2 Evaluation Metrics

Object detection models are evaluated using various metrics to assess their performance in accurately detecting and localizing objects in images. Here are some commonly used metrics for object detection:

1. Precision: Precision measures the proportion of correctly predicted positive detections (true positives) out of all positive detections (true positives + false positives). It indicates the model's ability to avoid false positives.
2. Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive detections (true positives) out of all ground truth positive instances (true positives + false negatives). It represents the model's ability to find all positive instances.
3. Average Precision (AP): Average Precision calculates the average precision value across different levels of recall by considering a range of confidence thresholds. It summarizes the precision-recall curve and provides a single value to measure overall detection performance.
4. Mean Average Precision (mAP): Mean Average Precision computes the average AP over multiple object classes. It is a widely used metric to evaluate object detection models, particularly in benchmark datasets such as Pascal VOC and COCO. mAP gives an overall performance measure, taking into account both precision and recall across multiple classes.
5. Intersection over Union (IoU): IoU measures the overlap between predicted bounding boxes and ground truth bounding boxes. It is calculated as the ratio of the intersection area between the two boxes to their union area. IoU is used to determine if a predicted bounding box is considered a true positive or a false positive. Figure 2.11 shows the formula for IOU.

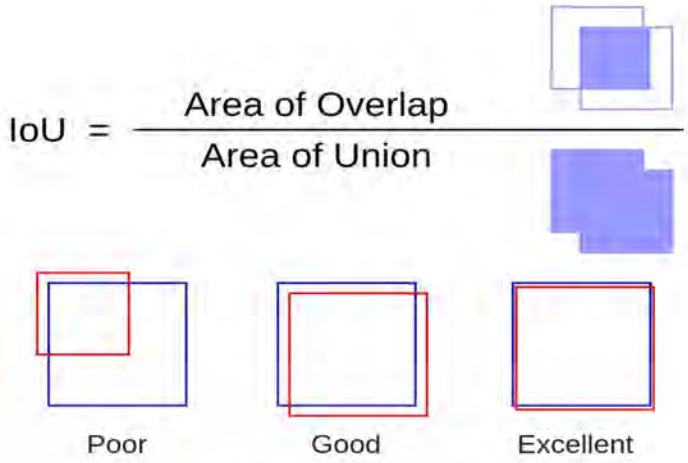


Figure 2.11 Intersection over Union (IOU) [13]

6. F1 Score: The F1 score combines precision and recall into a single metric. It is the harmonic mean of precision and recall, providing a balanced measure of both metrics. F1 score is useful when there is an uneven distribution between the number of positive and negative instances.

7. Localization Error: Localization error measures the accuracy of the predicted bounding boxes by calculating the average distance or overlap between the predicted bounding boxes and the ground truth bounding boxes.

8. Mean Average Precision at Different IoU thresholds (mAP@ [IoU threshold]): In some cases, mAP is reported at different IoU thresholds (e.g., mAP@0.5, mAP@0.75). It measures the average precision considering only detections with IoU values above a certain threshold. This metric provides insights into the model's performance at different levels of localization accuracy.

These metrics help evaluate the performance of object detection models and provide a quantitative assessment of their ability to accurately detect and localize objects in images. Different metrics serve different purposes, and the choice of metric depends on the specific requirements of the task and the evaluation dataset.

2.3 Object Tracking

Object tracking refers to the task of following and maintaining the identity of an object of interest over consecutive frames in a video or a sequence of images. It involves estimating the position, size, and other relevant attributes of the object in each frame, allowing for the continuous tracking of its movement and changes over time.

2.3.1 Steps for Object Tracking

Object tracking plays a crucial role in various applications, including surveillance systems, autonomous driving, action recognition, augmented reality, and human-computer interaction. It enables tasks such as object localization, behavior analysis, trajectory prediction, and anomaly detection. Object tracking algorithms typically follow a general pipeline that involves the following steps:

1. Initialization: The tracker is initialized by selecting or detecting the target object in the first frame. This step typically involves specifying a bounding box or a region of interest around the target object.
2. Object Representation: A representation or descriptor is extracted from the target object region in the initial frame. This representation can be based on color, texture, shape, or a combination of these features.
3. Motion Estimation: The motion of the target object is estimated by analyzing the displacement and appearance changes between consecutive frames. This can be achieved using techniques such as optical flow, feature matching, or correlation-based methods.
4.Object Localization: The estimated motion is used to predict the position and size of the target object in the current frame. This is typically done by applying a motion model and updating the initial bounding box or region of interest.
5. Appearance Model Update: The appearance model of the target object is updated to adapt to changes in appearance due to variations in lighting conditions, occlusions, or object deformations. This ensures robust tracking performance over time.
6. Occlusion Handling: Object tracking algorithms need to handle occlusions, where the target object may be partially or completely obscured by other objects. Various techniques, such as online learning, particle filtering, or graph-based methods, can be employed to handle occlusions and maintain accurate tracking.
7. Tracking Evaluation: The performance of the object tracker is evaluated based on metrics such as accuracy, robustness, speed, and stability. Common evaluation metrics include Intersection over Union (IoU), tracking precision, tracking recall, and the average number of false positives or false negatives.

Object tracking algorithms can be categorized into different types, including correlation-based methods, model-based methods, feature-based methods, and deep learning-based methods.

Each category employs different techniques and algorithms to address the challenges of object tracking. Overall, object tracking is a dynamic and evolving research field, with ongoing developments in both traditional and deep learning-based approaches. The goal is to achieve accurate, robust, and real-time tracking performance across various tracking scenarios and application domains. There are many types of object tracking algorithms. Some of the popular are ByteTracker, DeepSORT, StrongSORT, OCSORT etc. In our research we have implemented DeepSORT and OCSORT. The details of the architecture are discussed in the next section.

ByteTracker: ByteTracker is an object tracking algorithm introduced in the research paper titled " ByteTracker is an object tracking algorithm introduced in the research paper titled "ByteTracker: Learning to Track Objects from Bytes" by Fu-En Yang, Cheng-Chun Lee, and Yen-Yu Lin in 2021 [14]. ByteTracker is a deep learning-based tracking algorithm that operates directly on the binary representation of image data, i.e., the raw bytes, without utilizing any visual content.

The key idea behind ByteTracker is to learn discriminative features directly from the binary representation of images using a deep neural network. Instead of relying on pixel values or visual appearance, ByteTracker aims to capture semantic and structural information encoded in the raw bytes. This approach allows ByteTracker to track objects efficiently even in scenarios where visual appearance or texture-based methods may fail, such as when dealing with highly compressed or encrypted data.

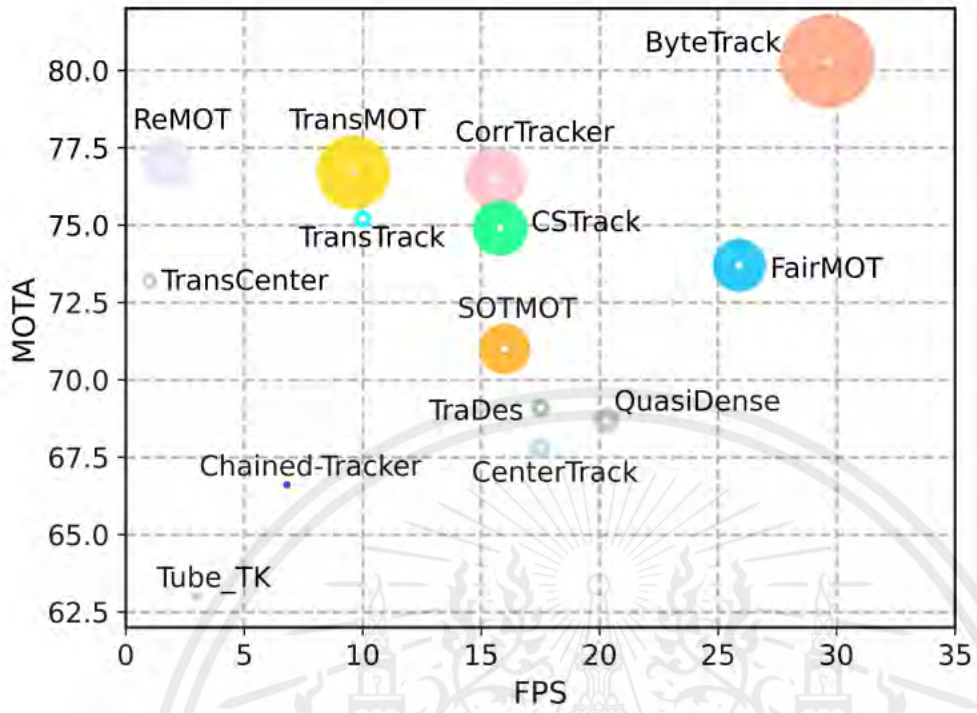


Figure 2.12 Comparison of Tracking Algorithms

ByteTracker consists of two main components: the Feature Extractor and the Tracker. The Feature Extractor is responsible for learning discriminative features from the binary representation of images. It utilizes a convolutional neural network (CNN) to extract feature representations from the raw bytes. The Tracker component employs an online tracking algorithm to associate and track objects based on the learned features. One of the advantages of ByteTracker is its ability to operate on data that cannot be visually observed, such as encrypted or compressed images. It shows promising results in tracking objects in such scenarios, where traditional visual-based methods may struggle. However, it is important to note that ByteTracker is still an active area of research, and its performance and limitations are still being explored and refined.

2.3.2 Evaluation Metrics

Object Tracking Algorithms have many evaluation metrics to validate the performance of the tracker. In our work, we have implemented MOTA object tracking algorithm. We discuss about MOTA in Chapter 3. Beside MOTA, HOTA and IDF1 are few other metrics for object tracking models.

HOTA (Higher Order Tracking Accuracy) is a set of object tracking metrics that provide a more comprehensive evaluation of tracking performance than traditional tracking metrics like Intersection over Union (IoU) or tracking precision and recall. HOTA takes into account various aspects of tracking accuracy and object identity preservation. The HOTA metrics include:

1.HOTA: HOTA is the primary metric and represents the overall tracking accuracy. It considers both localization accuracy (object position) and identity switches (ID switches) in the tracking results. It is a combined measure that considers both false positives (FP), false negatives (FN), localization errors (LocA), and identity switches (IDS).
2.Detection Accuracy: DetA measures the accuracy of object detection. It considers false positives (FP) and false negatives (FN) in object detection.
3.Localization Accuracy: LocA assesses how accurately the tracked object's position matches the ground truth. It is concerned with the spatial localization of the tracked object.
4.Identity Accuracy IdA quantifies the accuracy of preserving object identities throughout the tracking sequence. It measures identity switches (IDS) and false merges (FM).
5.False Merges: FM counts how many times different objects are merged into a single track. It's a measure of identity confusion in tracking.
6.Identity Switches: IDS quantifies the number of times an object changes its identity during tracking, reflecting the ability of the tracker to maintain correct object identities.
7. Fragmentation: Frag measures the fragmentation of the tracking results, which can occur when a single object track is divided into multiple shorter tracks.

HOTA is particularly useful in evaluating the performance of multi-object tracking algorithms, especially in scenarios where object identities are crucial, such as surveillance, autonomous driving, and object tracking in crowded environments. It provides a more comprehensive assessment of tracking quality by considering both object localization and identity preservation.

2.4 Literature Review

Deep Learning applications have garnered substantial attention thanks to ongoing research and development efforts. These methods have found utility in a wide range of tasks, including classification [15], object detection [16], object tracking [17], and healthcare. One specific

application of Deep Learning is in tracking parking time violations, particularly in settings like mini-marts where there are strict time constraints on parking. This task necessitates the integration of object detection and object tracking techniques.

2.4.1 Object Detection

In early developmental stages, the machine learning-based object detection pipeline was characterized by a fundamental two-step process, primarily encompassing the critical tasks of Region of Interest (ROI) extraction and subsequent object classification [18]. However, as the field of computer vision advanced, a more comprehensive framework for object detection was introduced by Zhao et al., which divided the process into three distinct phases: ROI selection, feature extraction, and final object classification [19].

The initial phase of this refined approach involves the meticulous selection of Regions of Interest (ROIs) within an image. Traditionally, one of the prevalent techniques employed for ROI extraction was the utilization of sliding windows, which systematically traversed the image at various scales. Nonetheless, it became evident that this exhaustive sliding window approach posed formidable challenges in practice due to its substantial processing complexity.In the contemporary landscape of object detection, Deep Learning algorithms have emerged as the cornerstone, substantially revolutionizing the field.. The first category includes pioneering approaches such as "You Only Look Once" (YOLO) and the "Single Shot MultiBox Detector" (SSD). These methods conceptualize object detection as a regression problem and are characterized as one-stage networks.

Conversely, the second category encompasses algorithms like the "Region-Based CNN" (R-CNN) , which adopt a more intricate approach. R-CNN, initially focuses on locating the regions of interest within an image, which are subsequently classified into specific object classes. This methodology involves the implementation of a selective search algorithm to determine the number of candidates bounding box object regions. The features extracted from these regions are then fed into a Convolutional Neural Network (CNN), serving as a feature extractor. Finally, Support Vector Machine (SVM) is employed to discern whether an object is indeed present within a given candidate region, utilizing the extracted features.

While R-CNN has exhibited commendable performance across various object detection tasks, it is essential to note that training these models demands a substantial investment of time

and computational resources. Furthermore, the speed of detection using R-CNN is inherently limited, as indicated by the constraints highlighted in prior research [20].

In the pursuit of accelerating both the training and inference processes, object detection witnessed the advent of one-stage detectors, exemplified by models like SSD (Single Shot MultiBox Detector) and YOLO (You Only Look Once). These innovations were introduced to significantly enhance the efficiency of object detection tasks, addressing the limitations of traditional two-stage detectors such as R-CNN.The cornerstone of SSD lies in the utilization of Convolutional Neural Network (CNN)-based feature extractors. These feature extractors, integrated into the model, play a pivotal role in the object detection process. At the culmination of feature extraction, the architecture of SSD incorporates convolutional feature layers that generate predictions at multiple scales. This multi-scale prediction mechanism is instrumental in achieving robust object detection results.

One notable distinction between one-stage detectors like SSD and their two-stage counterparts, such as R-CNN, lies in their approach to proposals. While traditional two-stage detectors heavily rely on region-based proposals for object localization, one-stage detectors like SSD dispense with this intermediary step. This elimination of region-based proposals leads to a notable enhancement in the speed of object detection. In essence, one-stage detectors directly predict object bounding boxes and class labels, eliminating the need for a separate proposal generation phase. In the specific research under consideration, YOLO (You Only Look Once) was chosen as the model of choice for the object detection task. Further elaboration on the YOLO model, including its intricacies and details, is expounded upon comprehensively in the Methodology section of this research work.Over the years, the field of parking lot monitoring has seen several noteworthy attempts to harness machine learning and image processing techniques for effective management and utilization of parking spaces. These endeavors have spanned a range of approaches and methodologies, each contributing to the evolution of parking space detection systems.

In an early exploration of parking space monitoring through machine learning, a pioneering effort dating back to 2002 leveraged color vector features in conjunction with a Support Vector Machine (SVM) classifier [21]. This approach aimed to discern parking spaces within a parking lot by analyzing color-based characteristics, marking an initial foray into the application of

machine learning for parking management.Advancing the state of the art, Huang et al. introduced a novel concept in the form of a day and night parking space detection system. This system, built upon a 3D model of parking spaces, relied on a Bayesian hierarchical framework to operate seamlessly in both daytime and nighttime conditions [22]. This innovative approach demonstrated the adaptability of machine learning techniques to various environmental settings.In addition to the adoption of deep learning models, several image processing techniques have played a significant role in addressing parking space detection challenges. For instance, Menéndez et al. devised a method that involved the temporal analysis of video frames from parking areas to identify vacant spaces [23]. This method encompassed background subtraction using Gaussian mixture models to detect and track vehicles within the parking lot. Furthermore, it incorporated the creation of a transience map to monitor the influx and egress of vehicles, offering a holistic view of parking space occupancy dynamics.

In a pursuit to curb illegal vehicle parking in a robust environment, Xie et al. proposed an optimized version of the Single Shot MultiBox Detector (SSD) [24]. This optimized SSD was tailored for real-time video stream analysis, achieving an impressive accuracy rate of 99%. By harnessing the power of machine learning and computer vision, this approach proved effective in identifying instances of unauthorized vehicle parking, thereby contributing to enhanced parking enforcement. These diverse approaches underscore the multifaceted nature of parking space detection challenges and highlight the continual evolution of methodologies and technologies in this domain, with machine learning and image processing playing pivotal roles in advancing parking management systems.

2.4.2 Object Tracking

Object tracking is a fundamental technique employed in computer vision and Deep Learning to detect and monitor objects across successive frames in videos or image sequences. This method relies on both spatial and temporal characteristics of objects to maintain their identities as they move through time. The core concept behind object tracking involves initiating a set of initial detections, assigning distinct IDs to these objects, and subsequently tracking their movement across consecutive frames. Over the past decade, object tracking methods have experienced a surge in popularity within the realms of Deep Learning and

computer vision. These techniques have found applications in various domains and have been particularly instrumental in solving complex problems in the following areas:

- **Single Object Tracking (SOT):** This category of object tracking focuses on monitoring and tracking a single object of interest as it moves through a sequence of frames.
- **Multiple Object Tracking (MOT):** In contrast, Multiple Object Tracking algorithms are designed to identify and track multiple objects within a single frame. These algorithms are tasked with not only detecting multiple objects but also assigning and preserving their unique identities as they traverse through different frames.

Object tracking has found practical applications across diverse domains, including but not limited to:

- **Pedestrian Tracking:** In fields like surveillance and autonomous vehicles, pedestrian tracking is crucial for ensuring the safety and navigation of individuals on foot. [25]

- **Vehicle Tracking:** The monitoring of vehicles is vital in transportation and traffic management systems, contributing to improved road safety and congestion management [26].

- **Player Tracking:** In sports analysis and entertainment industries, tracking the movement of players enhances game analysis and viewer experience [27].

For instance, Parico and Ahamed implemented the YOLOv4 object detection model for detecting pears and employed the DeepSORT multiple objects tracking algorithm to track and count pears [34]. Similarly, Hou et al. introduced DeepSORT along with a low-confidence search filter to mitigate false detections, thereby improving tracking accuracy [32]. Liu and Liu proposed a 3-D constrained multiple kernels approach, aided by Kalman filtering, to track vehicles detected by a YOLOv3 network [28]. In this particular article's context, the focus is on a closed-circuit television (CCTV)-based multiple object tracking technique, which allows for the identification and monitoring of multiple objects within a sequence of images or frames. The effectiveness of such a tracking method is contingent on various factors, including the

quality of object detection under diverse weather conditions, the degree of occlusion, and changes in illumination. Consequently, the selection of the most suitable object detection and tracking algorithm is of importance.

To address these challenges, state-of-the-art tracking algorithms like DeepSORT and OCSORT were implemented. Notably, the benchmark results on the DanceTrack dataset showcased in Table 2.1, showed the superior performance of the OC-SORT Algorithm, outperforming other tracking algorithms across various metrics, including HOTA, AssA, and IDF1. These results underscore the effectiveness and robustness of the OC-SORT Algorithm in the context of multiple object tracking, affirming its position as a high-performance tracking solution.

Table 2.1. Results on the DanceTract test Set

Tracker	**HOTA**	**DetA**	**AssA**	**MOTA**	**IDF1**
SORT [29]	47.9	72.0	31.2	91.8	50.8
DeepSORT [33]	45.6	71.0	29.7	87.8	47.9
ByteTrack [14]	47.3	71.6	31.4	89.5	52.5
OC-SORT [30]	54.6	80.4	40.2	89.6	54.6
OCSORT + Linear Interp [30]	55.1	80.4	40.4	92.2	54.9

Chapter 3

Methodology

In this chapter, we present a detailed explanation of our algorithm, focusing on the object detection and object tracking components. For object detection, we have chosen YOLOv8 as our algorithm of choice. YOLOv8 is a state-of-the-art object detection model known for its accuracy and efficiency in real-time applications. We describe the architecture and working principles of YOLOv8, highlighting its key features that make it suitable for our object detection needs. Moving on to the object tracking algorithm, we have implemented a comparative analysis between DeepSORT and OC-SORT. DeepSORT is a deep learning-based object tracking algorithm that combines object detection with data association techniques to achieve robust multi-object tracking. OC-SORT, on the other hand, is a variant of SORT (Simple Online and Real-time Tracking) algorithm that incorporates object occlusion handling capabilities. We provide a detailed explanation of the methodologies used in DeepSORT and OC-SORT, emphasizing their strengths and limitations.

To evaluate the performance of our algorithm, we conducted a comparative analysis on four different datasets. These datasets were carefully selected to represent a variety of tracking scenarios and challenges. We describe the characteristics of each dataset and explain how they were used for the evaluation. We then present the results of the comparative analysis, focusing on metrics such as MOTA (Multiple Object Tracking Accuracy), which measures the overall accuracy of the tracking algorithm. Throughout this chapter, we provide a comprehensive overview of our algorithm, its components, and the evaluation process. We discuss the advantages and limitations of our chosen object detection and tracking algorithms, and present the findings from the comparative analysis. The in-depth analysis and results provide valuable insights into the performance and suitability of our algorithm for object detection and tracking in various scenarios.

3.1 Overall Process

In this chapter, we provide an in-depth exploration of the holistic process of our proposed work. Our research encompasses a multifaceted approach, with key components including the

deployment of an object detection algorithm for precise vehicle identification within frames. This is seamlessly followed by the integration of an object tracking algorithm, to track vehicle movements across successive frames to establish their trajectories. A central focus of our work lies in the development and implementation of a dedicated time violation check algorithm. This algorithm diligently examines the duration for which vehicles occupy parking spaces, flagging instances of parking time violations. Figure 3.1 shows the high level process of our proposed work.

To convey actionable insights, our system is designed to display vital violation information, encompassing the amount of total time and a unique identifier (ID) linked to the offending vehicle. By integrating these algorithms, our research offers a comprehensive solution for vehicle detection, tracking, and the detection of parking time violations. In the subsequent sections of this chapter, we dive deeper into each of the algorithm along with their integration.

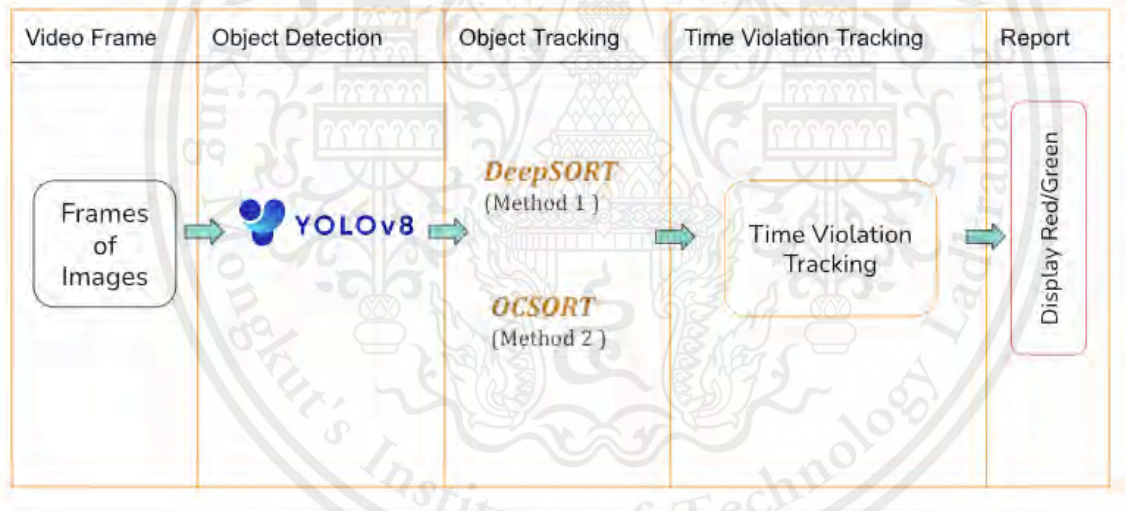


Figure 3.1 High Level Process of our proposed work

3.2 Dataset

In our pursuit of creating a robust dataset for our research on parking time violations, we encountered certain limitations and privacy concerns when it came to acquiring data from mini-marts. To overcome these challenges, we opted for an alternative approach and curated our dataset using video footage sourced from the CCTV cameras strategically placed at King Mongkut's Institute of Technology, Ladkrabang (KMITL).

The CCTV system implemented at KMITL boasts a state-of-the-art Panasonic V series, known for its exceptional performance. These cameras offer an impressive capture rate of 15–30 frames per second (FPS) along with 1080P resolution, ensuring high-quality video recordings. Notably, these cameras also possess an IP66 rating, providing them with resilience against the elements, including water and dust, thus ensuring the durability of our data source.

Our data collection process was meticulously executed using four of these advanced cameras, strategically positioned at various key locations across the sprawling university campus. Each camera maintained the same high-resolution 1080P output and a consistent frame rate of 15 FPS, ensuring uniformity and reliability throughout the dataset. The selection of these specific locations was done with careful consideration to incorporate a wide range of camera angles and viewpoints, mirroring real-world scenarios as closely as possible.The resulting dataset we have meticulously compiled comprises a rich diversity of video samples. These samples encompass a wide spectrum of lighting conditions, from bright daylight to challenging weather situations. This comprehensive representation of parking lot scenarios at KMITL is invaluable to our research, as it allows us to develop and test our parking time violation detection system under various real-world conditions. By making the most of the advanced capabilities of the Panasonic V series cameras and our thoughtfully selected dataset locations, we are confident that our research will yield meaningful insights and contribute to the development of effective solutions for managing parking time violations, especially in spaces with limited parking durations like mini-marts.

Figure 3.2 Sample Camera used for our dataset

3.3 Object Detection

There are plenty of object detection algorithms to choose from. For our research work, we chose YOLOv8. YOLO (You Only Look Once) is a popular choice for object detection algorithms due to its unique characteristics and advantages. One of the main reasons to choose YOLO is its exceptional speed. YOLO is specifically designed for real-time object detection, making it suitable for applications where fast processing is crucial. Unlike two-stage detectors, YOLO performs object localization and classification in a single pass over the image, enabling real-time performance even on resource-constrained devices

In addition to its speed, YOLO also offers good accuracy. While it may not achieve the highest accuracy compared to slower two-stage detectors, it still performs well in object detection tasks. Recent versions of YOLO, such as YOLOv4 and YOLOv5, have demonstrated significant improvements in accuracy, making them competitive choices for various applications. The detail architecture of YOLOv8 has been discussed in the below section.

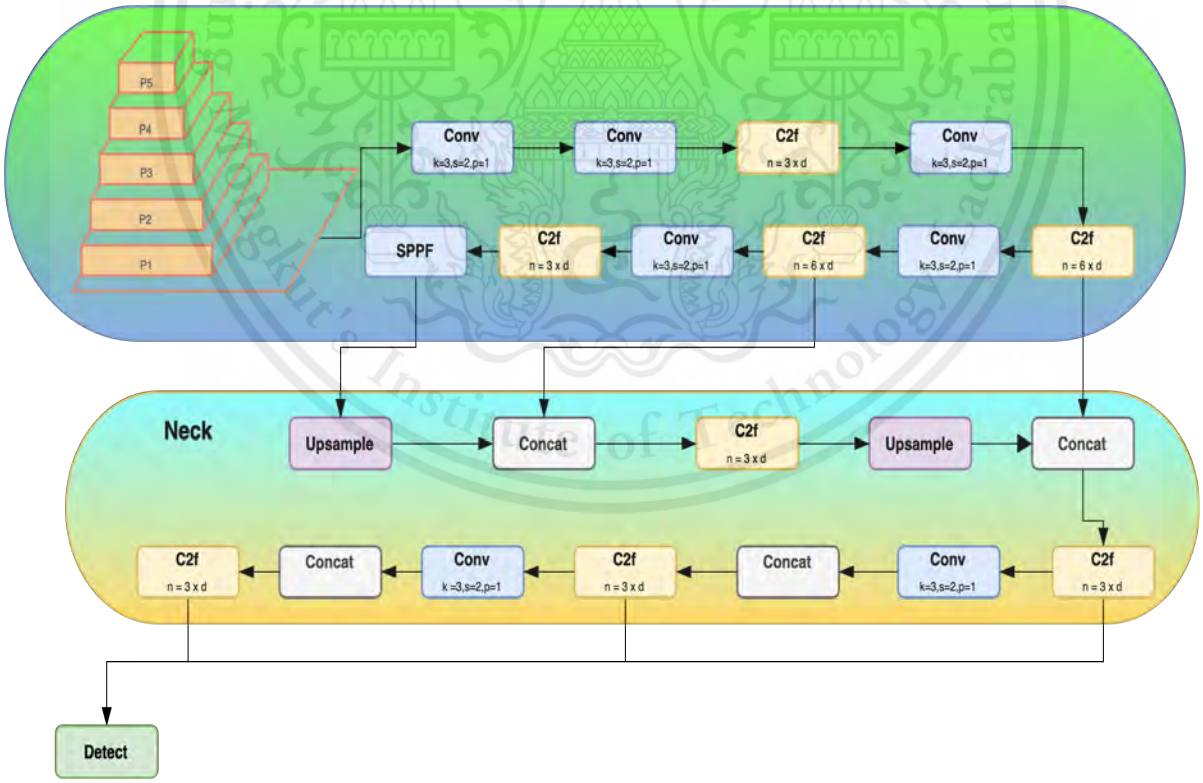


Figure 3.3 Detailed Architecture of YOLOv8

In the year 2015, a remarkable breakthrough in the field of computer vision occurred when Joseph Redmon and Ali Farhadi, distinguished researchers from the University of Washington, introduced the innovative object recognition algorithm known as YOLO, which stands for "You Only Look Once" [31]. YOLO quickly gained recognition for its exceptional performance, surpassing many of its predecessors in object detection, such as R-CNN (Region Convolutional Neural Network) and DPM (Deformable Parts Model) [31].One of the distinguishing features of YOLO is its holistic approach to image analysis. Unlike traditional methods that rely on sliding window-based or region proposal-based techniques, YOLO processes the entire image comprehensively during both training and testing phases. This unique approach allows YOLO to implicitly capture not only the appearance of objects but also their contextual information and relationships with other objects within the scene.

While the initial version of YOLO demonstrated the ability to swiftly recognize objects in images, it did face some challenges when it came to precisely locating smaller objects. However, it is important to note that this limitation primarily affected scenarios involving very small objects, which may not be relevant in certain problem domains. Therefore, for many practical applications, YOLO's capabilities are more than sufficient.A fundamental concept within the YOLO algorithm is the division of input images into a grid structure with dimensions M × M. Each grid cell is responsible for detecting objects whose centers fall within its boundaries. Within each grid cell, YOLO makes predictions regarding the bounding boxes of potential objects and assigns a confidence score to these predictions. This confidence score, in YOLO, is defined as the product of two factors: *Pr(Object) and IOU*.

- Pr(Object): This term represents the probability of an object's presence within the given grid cell.

- IOU (Intersection over Union): IOU measures the degree of overlap between the predicted bounding box and the ground truth bounding box.

Each grid cell generates a set of five predictions, including the x and y coordinates of the bounding box's center, its width (w) and height (h), and the associated confidence score. Additionally, YOLO's architecture produces conditional class probabilities (Pr(Class|Object)) for each grid cell. These class probabilities help determine the specific category or class of the object detected within that grid cell.

In essence, YOLO's unique characteristics and architecture have revolutionized the field of object recognition and have made it a widely adopted algorithm for a broad range of applications, from autonomous driving to surveillance systems and beyond. This holistic, grid-based approach to object detection, combined with its ability to handle contextual information efficiently, has solidified YOLO's position as a cornerstone in the realm of computer vision and image analysis.

$$Class \times IOU^{truth} = Class \mid Object \times Object \times IOU^{truth} \quad (1)$$

In the concluding layers of the YOLO algorithm, the network performs two critical tasks simultaneously. Firstly, it predicts the coordinates of bounding boxes that encapsulate the detected objects within the image. Secondly, it calculates the associated class probabilities for each of these predicted bounding boxes. To ensure consistency and compatibility with the subsequent stages of the algorithm, the predicted bounding box coordinates are normalized to fall within the range of 0 to 1. This normalization process is essential to facilitate the consistent representation of object locations within the image, regardless of its size or dimensions.

As the network progresses through its layers, it introduces increasing levels of non-linearity to enhance its capacity for capturing complex relationships within the data. This is primarily achieved by employing the leaky rectified linear activation function, as described in Equation (2). The leaky rectified linear activation function allows for the modeling of non-linearities in the data by introducing a small gradient for negative input values, which helps prevent the vanishing gradient problem and improves training stability.

However, it is noteworthy that the final layer of the YOLO network operates slightly differently from the preceding layers. Instead of using the leaky rectified linear activation function, the final layer employs a linear activation function. This choice is often made for the last layer to produce unbounded output values, which are particularly useful for tasks like regression, such as predicting bounding box coordinates. The linear activation function does not introduce non-linearity, allowing the network to output continuous values that represent the final bounding box coordinates accurately.

This strategic use of activation functions throughout the network architecture, culminating in the linear activation function in the final layer, ensures that YOLO is capable of both precise

object localization through bounding box predictions and accurate classification through class probability estimations. The combination of these elements contributes to the remarkable performance of YOLO in real-time object detection and recognition tasks.

$$F(x) = \begin{cases} x, & if\ x > 0. \\ 0.1x, & otherwise \end{cases} \quad (2)$$

In the years following its original release in 2015, the YOLO (You Only Look Once) object recognition framework has seen a series of notable advancements and iterations. Each subsequent version has aimed to enhance the model's capabilities and overall performance. Here's an overview of some of the key developments in the evolution of YOLO:

- YOLOv2 (2016): YOLOv2 marked a significant improvement over the original model. It introduced several key features, including batch normalization for more stable training, anchor boxes for better object localization, and dimension clustering to handle a wider range of object sizes.
- YOLOv3 (2018): YOLOv3 further refined the model's performance by incorporating a more efficient backbone network, featuring a feature pyramid for multi-scale object detection, and employing focal loss to prioritize hard-to-detect objects.
- Subsequent Versions: Building on the success of YOLOv3, a series of successors emerged, including YOLOv4, YOLOv5, YOLOv6, and YOLOv7, each with its own set of optimizations and improvements tailored to various use cases and requirements.
- YOLOv8 (2023): In 2023, Ultralytics released YOLOv8, the latest iteration in the YOLO series. YOLOv8 introduced the C2f module, which is based on Cross Stage Partial (CSP) architecture. This module enhances the model's learning capacity while reducing computational demands. The C2f module consists of two Conv Modules and n BottleNeck blocks, interconnected via Split and Concat operations. The remainder of the backbone architecture in YOLOv8 remains consistent with that of YOLOv5. Additionally, the SPPF Module is used in the final layer of the backbone to further improve performance.

These advancements in the YOLO series underscore the ongoing efforts to push the boundaries of object recognition and detection in the field of computer vision. YOLOv8, in particular, demonstrates a commitment to improving both the model's accuracy and its

computational efficiency, making it a valuable tool for a wide range of applications, from autonomous vehicles to surveillance systems and beyond. The evolution of YOLO continues to play a pivotal role in shaping the future of real-time object detection technology.

For YOLOv8, we configured the image-size parameter to 640, thereby resizing the longest dimension to 640 pixels while maintaining the aspect ratio. Consequently, the resized images closely approximated 640 × 360 pixels. We filtered the detected classes to include (2, 5, 7), which corresponded to car, bus, and truck classes, as our violation algorithm focused on larger vehicles. The confidence threshold was set at 0.5, determining the probability of a class occurring within a bounding box. Additionally, we set the maximum object detection parameter to 100 to ensure efficient processing of multiple objects.

Figure 3.4 Surveillience video from Site 1 [11]

Figure 3.5 Surveillience video from Site 2 [11]

Figure 3.6 Surveillience video from Site 3 [11]

Figure 3.7 Surveillience video from Site 4 [11]

3.4 Object Tracking

Once vehicle detection within a parking lot is done, it becomes possible to employ object tracking algorithms for vehicle tracking. There are many ways to implement object tracking. The latest version of YOLOv8 has introduced support for the ByteTrack object tracker. However, it's important to note that native support for trackers such as DeepSORT and OCSORT is currently lacking in YOLOv8 [31]. This absence poses a limitation as it prevents us from utilizing the native YOLOv8 code to enable these specific trackers. Furthermore, the built-in tracker in YOLOv8 operates at a surface level, limiting our ability to modify the tracker's architecture and crucial hyperparameters. Due to this constraint, the chosen method for object tracking was deemed unsuitable, as it does not allow for the necessary customization of the tracker's underlying components. The study encompassed the implementation of two tracking algorithms. The initial algorithm chosen was DeepSORT. The process flow of our algorithm is depicted in Figure 3.8. The DeepSORT algorithm was employed to trace individual vehicles across each frame. DeepSORT, an extension of SORT (Simple Online Realtime Tracking) [32],utilizes appearance descriptors to reduce identity switches, thereby enhancing tracking efficiency. In situations requiring predictions for temporal or time series data, the chosen algorithm is Kalman filtering.

To facilitate vehicle tracking, we employed the DeepSORT (Deep Simple Online and Realtime Tracking) and OC-SORT (Online and Realtime Sort) tracking algorithms. For the DeepSORT algorithm, we configured the max-age parameter to 10, which preserved the ID of a vehicle for a specified number of frames before removing the ID. Furthermore, we set the n_init parameter to 2, which dictated the number of objects detected before initializing the tracking

process. Max_cosine_distance was set to 0.3, serving as the threshold for determining vehicle similarity by the DeepSORT algorithm. We opted not to track the trajectory of vehicles over time, as indicated by the trajectory parameter being set to False. This selection was made based on our specific research requirements, where tracking the vehicle's path was unnecessary for the parking time violation detection task.

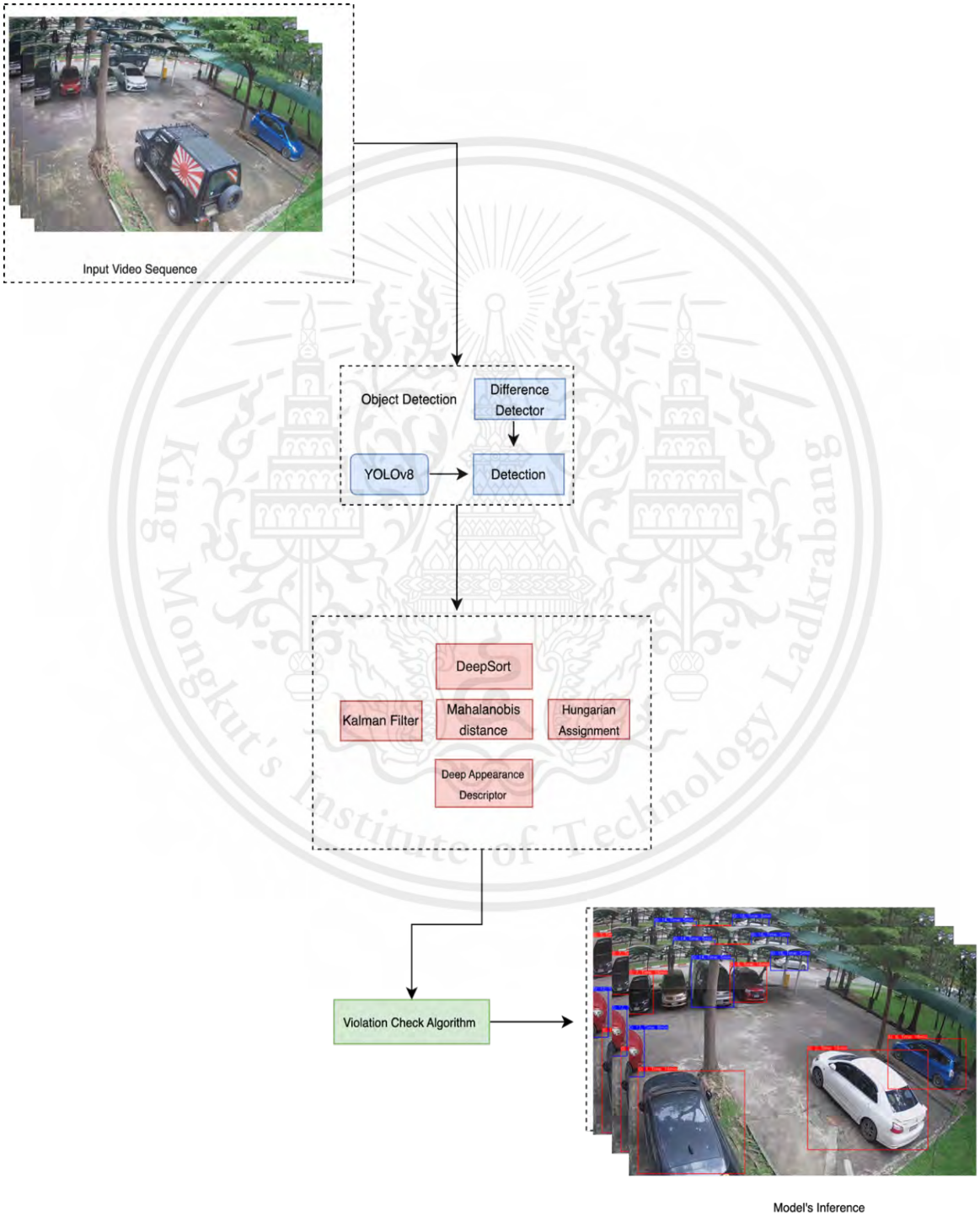


Figure 3.8 Workflow of our Proposed Algorithm [11]

The second tracking algorithm that was integrated into the system is referred to as Observation-Centric SORT (OC-SORT), as detailed in reference [36]. OC-SORT made its debut in the prestigious Computer Vision and Pattern Recognition (CVPR) conference in the year 2023. This algorithm has been rigorously evaluated across a diverse array of datasets, including but not limited to MOT17, MOT20, KITTI, head tracking tasks, and with a notable emphasis on the DanceTrack dataset. In scenarios characterized by highly non-linear object motions, OC-SORT consistently delivers state-of-the-art tracking results.

One of OC-SORT's distinguishing features is its focus on enhancing tracking accuracy and robustness, especially during periods of occlusion, where objects may be temporarily hidden from view. To achieve this, OC-SORT leverages object observations to calculate a virtual trajectory for the tracked objects. The pipeline for OC-SORT is visually represented in Figure 3.9,illustrating the algorithm's key stages and workflow.

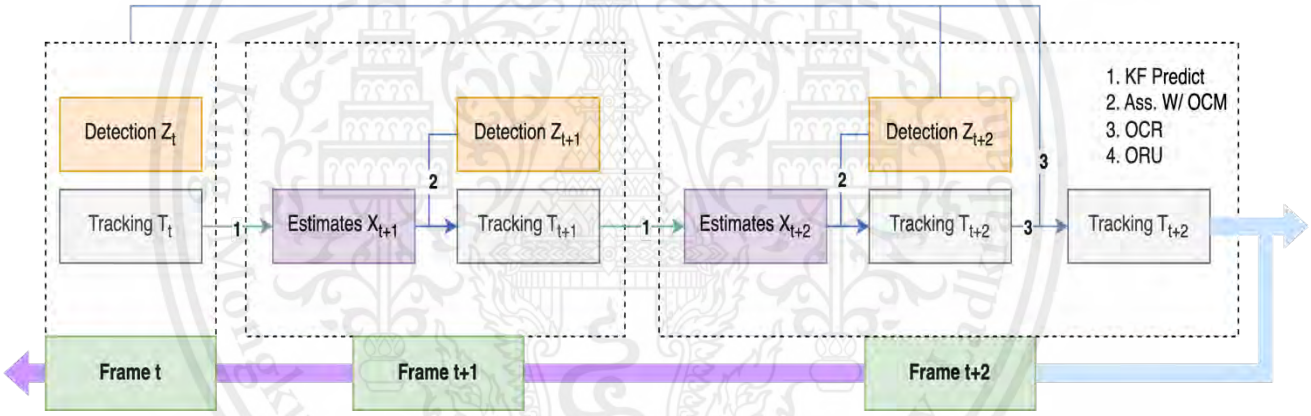


Figure 3.9 Pipeline of OC-SORT

3.5 Time Violation

Once vehicle detection and tracking were seamlessly integrated, with each unique vehicle assigned a distinct ID, a straightforward time violation algorithm was implemented to monitor all vehicles within the frame. This algorithm checked for the presence of each vehicle every minute. If a vehicle was absent in the subsequent 10 consecutive frames, its assigned ID was removed. Conversely, if the same ID persisted for 15 consecutive frames, it was indicative of a time restriction violation. In the accompanying figure, a blue label denoted no violation, while a red label signified a violation.

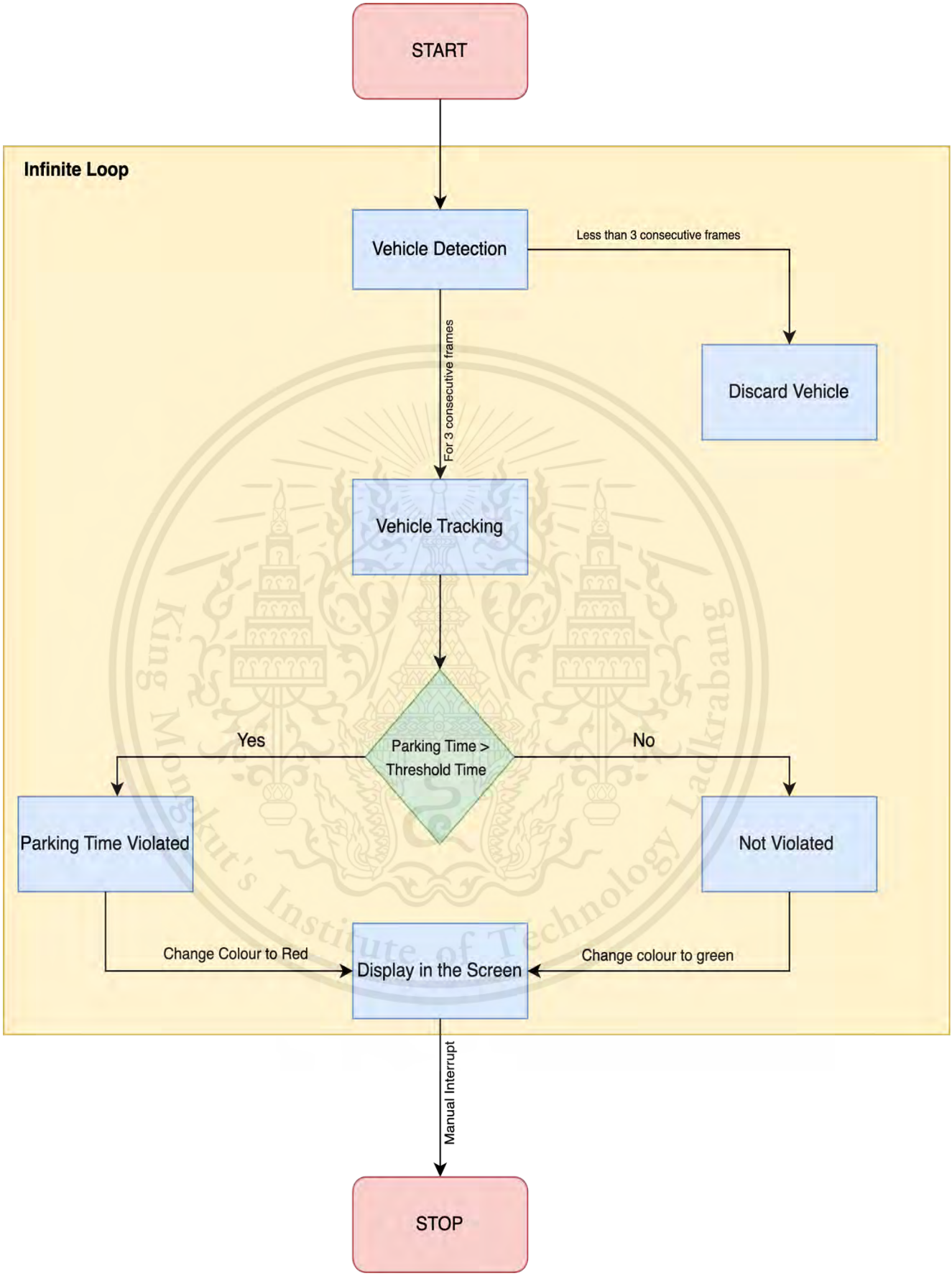


Figure 3.10 Flowchart of Time Violation Tracking

3.6 Evaluation Method

Object detection algorithms, like YOLOv8, rely on the Mean Average Precision (mAP) metric to assess their performance in accurately detecting objects within images. However, in the context of this research, which primarily focused on tracking vehicles across frames, it was imperative to validate the results using an object tracking algorithm. To achieve this, we employed the MOTA (Multiple Object Tracking Algorithm) metrics, which serve as a comprehensive measure of the accuracy of both detection and tracking algorithms.

In the process of evaluating using MOTA metrics, it was crucial to establish clear definitions for key terms, including True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). Table 3.1 provides detailed definitions for these terms to ensure a precise understanding of their implications within the context of our research.

- True Positive (TP): Instances where the algorithm correctly detected and tracked a vehicle, aligning with the ground truth.

- False Positive (FP): Occurrences where the algorithm falsely detected a vehicle in the absence of a corresponding ground truth object. For example, one FP case occurred when a vehicle's reflection was erroneously identified as a vehicle.

- False Negative (FN): Instances where the algorithm failed to detect and track a vehicle that existed in the ground truth. Various factors, such as distance from the camera, adverse weather conditions, varying illumination, etc., could contribute to FN cases.

- True Negative (TN): A scenario where neither the algorithm nor the ground truth identified or tracked a vehicle, indicating a correct absence of detection.

The MOTA metric is quantified using Equation (3), and it takes into account the various elements defined above. FN, in particular, highlights the importance of detecting and tracking objects accurately, even under challenging conditions. It underscores the impact of missed detections, which can occur due to factors beyond the algorithm's control.

In summary, MOTA metrics serve as a comprehensive evaluation tool for assessing the performance of both detection and tracking algorithms in the context of object tracking. This approach allows us to account for various real-world challenges, providing a more holistic perspective on the accuracy and effectiveness of our system.

Table 3.1 Definition and Description of TP,FP, TN and FN

Definition	Description
True Positive (TP)	Vehicle is present and the algorithm can track vehicle
False Positive (FP)	Vehicle is not present but the algorithm tracks vehicle.
True Negative (TN)	Vehicle is not present and the algorithm does not track vehicle
False Negative (FN)	Vehicle is present but the algorithm does not track vehicle

$$MOTA = 1 - \frac{\Sigma\, FN+FP+IDS}{\Sigma\, GT} \quad (3)$$

Figure 3.8 provides a visual representation of vehicle detection and tracking scenarios, employing YOLOv8 in conjunction with two distinct object tracking algorithms under varying lighting and weather conditions. GT refers to the total ground truth object.

Figure 3.11 Showing Vehicles in 3 Frames.

Let us consider the above picture as an example to understand MOTA calculation. “t” refers to the current time frame whereas “t+1” and “t+2” refers to the next 2 consecutive time frame. In the beginning, the ID of Blue, White and Black car are 1,2 and 3 respectively. For the frame “t+1” the ID is the same whereas for the “t+2” the ID of Blue car is swapped for id of White car. This increases the count of IDS i.e., ID Switch. False Negative, False Positive and ID Switch will be calculated for every frame of the video and finally, MOTA will be calculated.

$$\text{For t, FN} = 0, \text{FP} = 0 \text{ and IDS} = 0$$

$$\text{For t} + 1, \text{FN} = 0, \text{FP} = 0 \text{ and IDS} = 0$$

$$\text{For t} + 2, \text{FN} = 0, \text{FP} = 0 \text{ and IDS} = 1$$

Therefore,

$$\text{MOTA} = 1 - \frac{\Sigma\,(FN+FP+IDS)}{\Sigma\,GT} = 1 - \frac{(0+0+0)+(0+0+0)+(0+0+1)}{3} = 0.68$$

Chapter 4

Result and Discussion

In this section, we will delve into a comprehensive discussion regarding the performance of the model employed in our research. Furthermore, we will discuss the precise experimental setup that played a vital role in shaping the outcomes of our study. This comprehensive exploration will shed light on the intricacies of our research methodology and the factors that contributed to the results we achieved.

4.1 Experimental Setup

In this section, we explore into the experimental platform in detail, offering insights into the algorithm's workflow and the specific parameters selected for the experiment. Our experimentation took place on an Ubuntu Linux workstation, equipped with a robust hardware configuration. The workstation was powered by a 3.6 GHz Intel Xeon Quad-Core processor, supported by 8GB of RAM, and featured an NVIDIA Quadro P4000 graphics card. These specifications ensured that the experimental environment could efficiently handle the computational demands of our research.

For the crucial task of vehicle detection, we employed YOLOv8x, which emerged as the top-performing variant among the YOLO (You Only Look Once) versions. This choice was based on the algorithm's impressive Mean Average Precision (mAP) score, which reached an impressive 53.9. Notably, this outperformed all other YOLO versions we evaluated. YOLOv8x utilized a pretrained model that had been trained on the MS COCO dataset. The video feed's resolution was set at 1080P (1920 × 1080) with a frame rate of 15 frames per second (fps).

4.2 Result

In this section, we explore into the details about the results of our work and engage in a thorough discussion of the result .

Figure 4.1 Sample images of tracking by our model [11]

The comparative performance analysis reveals that YOLOv8 combined with the DeepSORT algorithm consistently outperformed the recently introduced OC-SORT algorithm. It's noteworthy that OC-SORT had previously demonstrated substantial improvements across multiple MOT datasets. The corresponding MOTA (Multiple Object Tracking Accuracy) scores for DeepSORT and OC-SORT, as displayed in Tables 4.2 and 4.3.

- DeepSORT: Achieved MOTA scores of (1.0, 1.0, 0.96, 0.90), indicating high tracking accuracy across different scenarios as shown in Table 4.2

- OC-SORT: Attained MOTA scores of (1, 0.76, 0.90, 0.8), with the highest accuracy in the first scenario, gradually improving in subsequent scenarios but with variations in tracking performance as shown in Table 4.3

The OC-SORT algorithm's performance on this particular dataset left much to be desired. It consistently grappled with a set of challenges, primarily centered around its ability to maintain consistent vehicle identification, a critical aspect of object tracking. These challenges were particularly pronounced in adverse weather conditions, specifically during rainy weather.

One of the prominent issues observed was the algorithm's propensity to frequently switch vehicle IDs during tracking. This behavior introduced significant inaccuracies in the tracking process, as the algorithm struggled to maintain a stable identity for the detected vehicles. As a result, the algorithm's tracking outputs were often marred by disruptions in the continuity of vehicle identification.

Table 4.1 Description of FP,FN and IDS

Definition	Description
False Positive (FP)	Vehicle is not present but the algorithm tracks vehicle.
False Negative (FN)	Vehicle is present but the algorithm does not track vehicle
Identity Switch (IDS)	The ID of vehicle is swapped with another ID of another vehicle.

This inconsistency in vehicle identification had a cascading effect on the overall tracking performance of OC-SORT. Accurate tracking hinges on the ability to correctly associate and follow objects over time, ensuring that the identity of each object remains stable throughout the tracking process. When the algorithm repeatedly changed vehicle IDs, it led to erroneous associations and, consequently, a reduced quality of tracking results.The impact of these tracking inaccuracies was most pronounced in scenarios with inclement weather, such as rainy conditions. Rain can introduce additional complexities in object detection and tracking, including occlusion due to water droplets on camera lenses and changes in object appearance. In such challenging conditions, the OC-SORT algorithm's struggles with maintaining vehicle identities were exacerbated, resulting in a suboptimal performance.

Table 4.2 Performance of YOLOv8 with DeepSORT

Model	Dataset	FP	FN	IDS	MOTA
YOLOv8 + DeepSORT	Location 4	0	0	0	1
	Location 3	0	0	0	1
	Location 2	1	0	0	0.96
	Location 1	0	3	0	0.90

This inconsistency in vehicle identification had a cascading effect on the overall tracking performance of OC-SORT. Accurate tracking hinges on the ability to correctly associate and follow objects over time, ensuring that the identity of each object remains stable throughout the tracking process. When the algorithm repeatedly changed vehicle IDs, it led to erroneous associations and, consequently, a reduced quality of tracking results.The impact of these tracking inaccuracies was most pronounced in scenarios with inclement weather, such as rainy conditions. Rain can introduce additional complexities in object detection and tracking, including occlusion due to water droplets on camera lenses and changes in object appearance. In such challenging conditions, the OC-SORT algorithm's struggles with maintaining vehicle identities were exacerbated, resulting in a suboptimal performance.

In summary, the OC-SORT algorithm's shortcomings on this dataset were particularly evident in its inability to ensure consistent vehicle identification, especially during adverse weather conditions like rain. The frequent switching of vehicle IDs led to inaccuracies in tracking, significantly affecting its overall performance in these scenarios.

Table 4.3 Performance of YOLOv8 with OCSORT

Model	Dataset	FP	FN	IDS	MOTA
YOLOv8 + OCSORT	Location 4	0	0	0	1
	Location 3	0	0	7	0.76
	Location 2	1	0	2	0.90
	Location 1	0	4	1	0.83

4.3 Discussion

In Figures 4.2, we have the opportunity to discuss into the tracking performance of both algorithms on the Location 2 dataset. Let's focus on a specific frame, namely "f23401," which is presented in both figures. In this particular frame, an intriguing contrast between the two tracking algorithms becomes apparent: DeepSORT successfully maintains accurate vehicle IDs, while OC-SORT faces notable challenges in this aspect.

Figure 4.2 Performance of DeepSORT for Location 2 Dataset [11]

The difficulties encountered by OC-SORT in consistently tracking vehicle IDs can potentially be attributed to a combination of factors, notably the adverse weather conditions and the presence of visually similar vehicles in close proximity. Rainy weather conditions can introduce complexities in object tracking, such as occlusion due to water droplets on camera lenses and changes in the visual appearance of vehicles. These challenges might have led to difficulties in distinguishing between vehicles, thereby resulting in the frequent switching of vehicle IDs.

In contrast, DeepSORT appears to exhibit a higher level of robustness in handling such challenging scenarios. This enhanced performance could be attributed to DeepSORT's utilization of deep appearance feature embedding, which allows it to create more robust and distinctive object representations. These representations enable DeepSORT to maintain reliable tracking, even when faced with challenging conditions like those encountered in the Location 2 dataset.

It's worth noting that the performance of both tracking algorithms is context-dependent. In Location 1, for instance, we observed a discrepancy in the count of False Negatives (FN) between DeepSORT (3) and OC-SORT (4). This discrepancy was primarily due to the presence

of reflections from vehicles on the windows, which interfered with the vehicle detection process. These reflections caused some vehicles to be falsely labeled as negatives, ultimately affecting the Multiple Object Tracking Accuracy (MOTA) scores.

In summary, Figures 4.2 and 4.3 shed light on the contrasting tracking abilities of DeepSORT and OC-SORT in the Location 2 dataset, with DeepSORT demonstrating better vehicle ID tracking in challenging conditions. Furthermore, in Location 1, reflections from vehicle windows introduced difficulties that affected both algorithms' performance, highlighting the nuanced nature of object tracking in real-world scenarios.

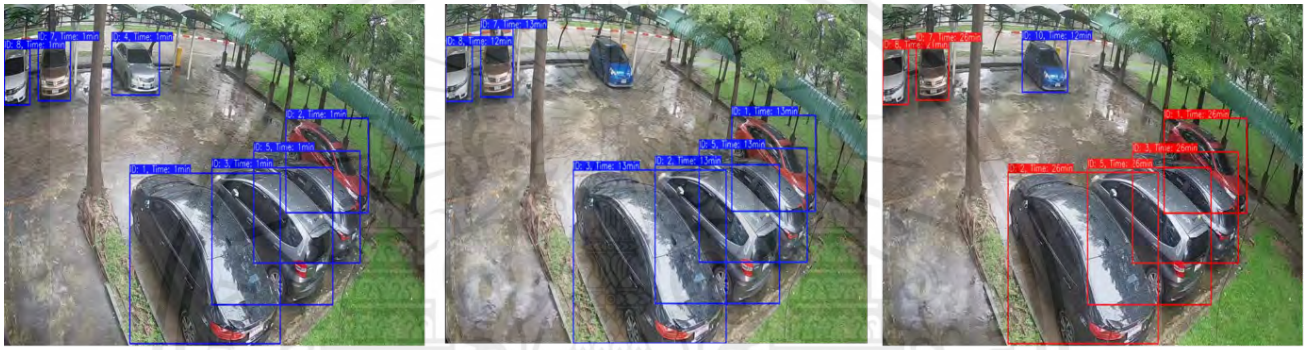


Figure 4.3 Performance of OCSORT for Location 2 Dataset [11]

The conducted experiment has yielded valuable insights into the performance of vehicle tracking algorithms, shedding light on the significant impact of factors such as camera position and weather variations, particularly when a single camera is used for inference. It becomes evident that achieving optimal tracking results requires careful consideration of these factors.One key takeaway from this experiment is the crucial role of camera positioning. Placing the camera at or near a top-view perspective emerges as a highly effective strategy for enhancing the performance of vehicle tracking algorithms. This strategic camera placement minimizes the occurrence of vehicle overlapping, which, in turn, leads to a considerable improvement in tracking accuracy.

The rationale behind this approach is rooted in the advantages of a top-view perspective. When the camera captures a top-down view of the monitored area, it provides a clear and unobstructed line of sight to the vehicles. This perspective reduces the chances of vehicles overlapping or obstructing each other, which can be a common challenge in tracking scenarios.

The unambiguous view offered by a top-view perspective enables the tracking algorithm to more accurately detect and track individual vehicles. Consequently, this results in a higher degree of reliability and precision in tracking outcomes.

In summary, the experiment underscores the significance of camera placement in optimizing the performance of vehicle tracking algorithms. Selecting a top-view perspective for camera placement is paramount as it effectively minimizes vehicle overlap and maximizes the overall effectiveness of the tracking algorithms. This insight can be particularly valuable in real-world applications where accurate and reliable vehicle tracking is of utmost importance, such as traffic management, surveillance, and autonomous vehicle navigation.

Chapter 5

Conclusion

Manually tracking parking time violations is impractical due to its reliance on manpower and high expenses. This research paper demonstrates a successful parking time violation tracking algorithm, offering the potential to cut down on labor costs. The algorithm is adaptable to various parking lot scenarios. Vehicle detection within the parking area utilizes the YOLOv8 algorithm, while vehicle tracking across frames employs DeepSORT and OC-SORT. DeepSORT achieved noteworthy MOTA scores of (1.0, 1.0, 0.96, 0.90) across diverse surveillance datasets, displaying precise and effective multi-object tracking. Meanwhile, OC-SORT obtained MOTA scores of (1.0, 0.76, 0.90, 0.83) across the same datasets, while still maintaining respectable scores, showing some divergence compared to DeepSORT, particularly in the second dataset where the MOTA score fell to 0.76 due to frequent ID switching. Both algorithms demonstrated their proficient object tracking capabilities across the surveillance data. These findings furnish valuable insights into the tracking performance of each algorithm, aiding in algorithm selection based on specific tracking requisites and dataset attributes. Although not obligatory, custom training of the detection model with vehicle data might enhance its performance, potentially bolstering the algorithm's robustness.

In upcoming research, researchers could delve into synchronizing the algorithm with multiple cameras. This synchronization would empower the algorithm to identify parking time violations in scenarios involving multiple CCTV cameras. Through the integration of camera feeds, the algorithm can amass a more comprehensive overview of the parking lot, ultimately refining violation detection accuracy. This adaptability is pivotal for extending the algorithm's relevance to expansive parking areas or contexts where multiple cameras are already in operation.

References

[1] A. Ouaknine, "Review of Deep Learning Algorithms for Object Detection," 5 2 2018. [Onlin Available: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852. [Accessed 10 06 2023].

[2] Jeffrey, PATIL, Roshan, NARAHARI, Y. Skanda & DIDAGI, J. Bapat and Das, "Wireless Sensor Network Based Smart Parking System," *Sensors & Transducers,* 2014.

[3] T. D. a. Kim, "A Novel Location-Centric IoT-Cloud Based On-Street Car Parking Violation Management System in Smart Cities," *MDPI Sensor,* 2016.

[4] CEIC DATA, "Thailand Number of Registered Vehicles," 2023. [Online]. Available: https://www.ceicdata.com/en/indicator/thailand/number-of-registered-vehicles. [Accesse 6 2023].

[5] Jaesung, "Create an Image Classifier Model," 2023. [Online]. Available: https://chic0815.gitbook.io/jaesung/wwdc-scholarship/swift-playground/framework-coreml/creating-an-image-classifier-model.

[6] M. Yani, "Application of Transfer learning using CNN Method for Early detection of Terry's Nail.," *Journal of Physics : Conference Series ,* 2019.

[7] K. H. Sun, Z. Xiangyu, R. Shaoqing and Jian, "Deep Residual Learning for Image Recognitio *CoRR,* 2015.

[8] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet Classification with Deep Convolution Neural Netowkr," *Neural Information Processing Systems,* 2012.

[9] C. Rabinovich, W. Liu and J. Yangqing, "Going Deeper with Convolutions," *arXiv,* 2014.

[10] S. Christian, L. Wei, J. Yangqing, S. Pierre, S. Reed and D. Anguelov, "Going Deeper with Convolutions," *CoRR,* 2014.

[11] N. Sharma, S. Baral and M. P. P. a. R. Chawuthai, "Parking Time Violation Tracking Using YOLOv8 and Tracking Algorithms," *Sensor MDPI,* 2023.

[12] G. Karimi, "Introduction to YOLO Algorithm for Object Detection," 15 April 2021. [Online]. Available: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-f object-detection/. [Accessed 3 July 2023].

[13] "Bounding Box Intersection over Union (IoU) Measure," [Online]. Available: http://www.gabormelli.com/RKB/Bounding_Box_Intersection_over_Union_%28IoU%29_M re. [Accessed 3 July 2023].

[14] Z. Yifu, S. Peize, J. Yi, Y. Dongdong, W. Fucheng, Y. Zehuan, L. Ping, L. Wenyu and W. Xinggang, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," *ArXiV,* 20

[15] S. B. a. M. P. P. Thinam Tamang, "Classification of White Blood Cells: A Comprehensive St Using Transfer Learning Based on Convolutional Neural Networks," *MDPI Diagnostics ,* 202

[16] M. P. Paing, "Adenoma Dysplasia Grading of Colorectal Polyps Using Fast Fourier Convolutional ResNet (FFC-ResNet)," *IEEE Access ,* vol. 11, 2023.

[17] M. P. Paing, "Histopathological Classification of Colorectal Polyps using Deep Learning," *IE Conference on Information Networking,* 2023.

[18] J. Janai, B. Fatm, G. Aseem and Andreas, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends in Computer Graphics Vision,* vol. 12, 2020.

[19] Z. Zhao, P. Zheng, S. tao Xu and X. Wu, "Object Detection with Deep Learning: A Review, 2016.

[20] Q. Mao, H. Sun, Y. Liu and R. Jia, "Mini-YOLOv3: Real-Time Object Detector for Embeddec Applications," *IEEE Access,* vol. 7, 2019.

[21] N. Dan, "Parking Management System and Methods," *US Patent,* 2003.

[22] C. Huang, Y. Tai and S. Wang, "Vacant Parking Space Detection Based on Plane-Based Bayesian Hierarchical Framework.," *IEEE Transaction,* vol. 9, 2015.

[23] J. Menéndez, C. Postigo and J. Torres, "Vacant parking area estimation through backgrour subtraction and transience map analysis.," *IET Intell. Transp. Syst,* vol. 9, 2015.

[24] X. Xie, C. Wang, S. Chen, G. Shi and Z. Zhao, "Real-Time Illegal Parking Detection System Based on Deep Learning," *CoRR ,* 2017.

[25] Z. Sun, J. Chen, L. Chao, W. Ruan and M. Mukherjee, "A Survey of Multiple Pedestrian Tracking Based on Tracking-by-Detection Framework.," *IEEE Transaction on Circuits and Systems,* vol. 31, 2021.

[26] X. Hou, Y. Wang and L. Chau, "Vehicle Tracking Using Deep SORT with Low Confidence Tr Filtering," *In Proceedings of the 2019 16th IEEE International Conference on Advanced Vi and Signal Based Surveillance (AVSS), Taipei, Taiwan,,* 2019.

[27] M. Buric, M. Ivasic-Kos and M. Pobar, " Player Tracking in Sports Videos.," *In Proceedings o the 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Sydney, Australia,* 2019.

[28] T. Liu and Y. Liu, "Deformable Model-Based Vehicle Tracking and Recognition Using 3-D Constrained Multiple-Kernels and Kalman Filter.," *IEEE Access,* vol. 9, 2021.

[29] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple Online and Realtime Tracking.," *arXiv ,* 2016.

[30] J. Cao, J. Pang, X. Weng, R. Khirodkar and K. Kitani, "Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking,," 2023.

[31] Ultralytics, "Tracker," 11 10 2023. [Online]. Available: https://github.com/ultralytics/ultralytics/tree/main/ultralytics/trackers.

[32] N. Wojke, A. Bewley and D. Paulus, " Simple Online and Realtime Tracking with a Deep Association Metric," *CoRR,* 2017.

[33] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Tim Object Detection," 2016.

[34] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *CoRR,* 2016.

[35] J. 41. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *CoRR,* 2018.

[36] G. Jocher, A. Chaurasia and J. Qiu, "YOLO By Ultralytics".

[37] J. Cao, J. Pang, X. Weng, R. Khirodkar and Kitani, ". Observation-Centric SORT: Rethinking S for Robust Multi-Object Tracking, 202," 2023.

Author Biography

Name – Surname	Nabin Sharma
Date of Birth	1998-11-19
Current Address	99/176 Soi Mu Baan Chonlada Suvarnabhumi, Ladkrabang 54
Permanent Address	Budanilkantha, Nilopool, Kathmandu, Nepal
Email Address	sharmanabin010@gmail.com
Contact Number	+66990724052, +9779841256688

Educational Background

Master of Engineering (Robotics and Computational Intelligence Systems), 2023, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand

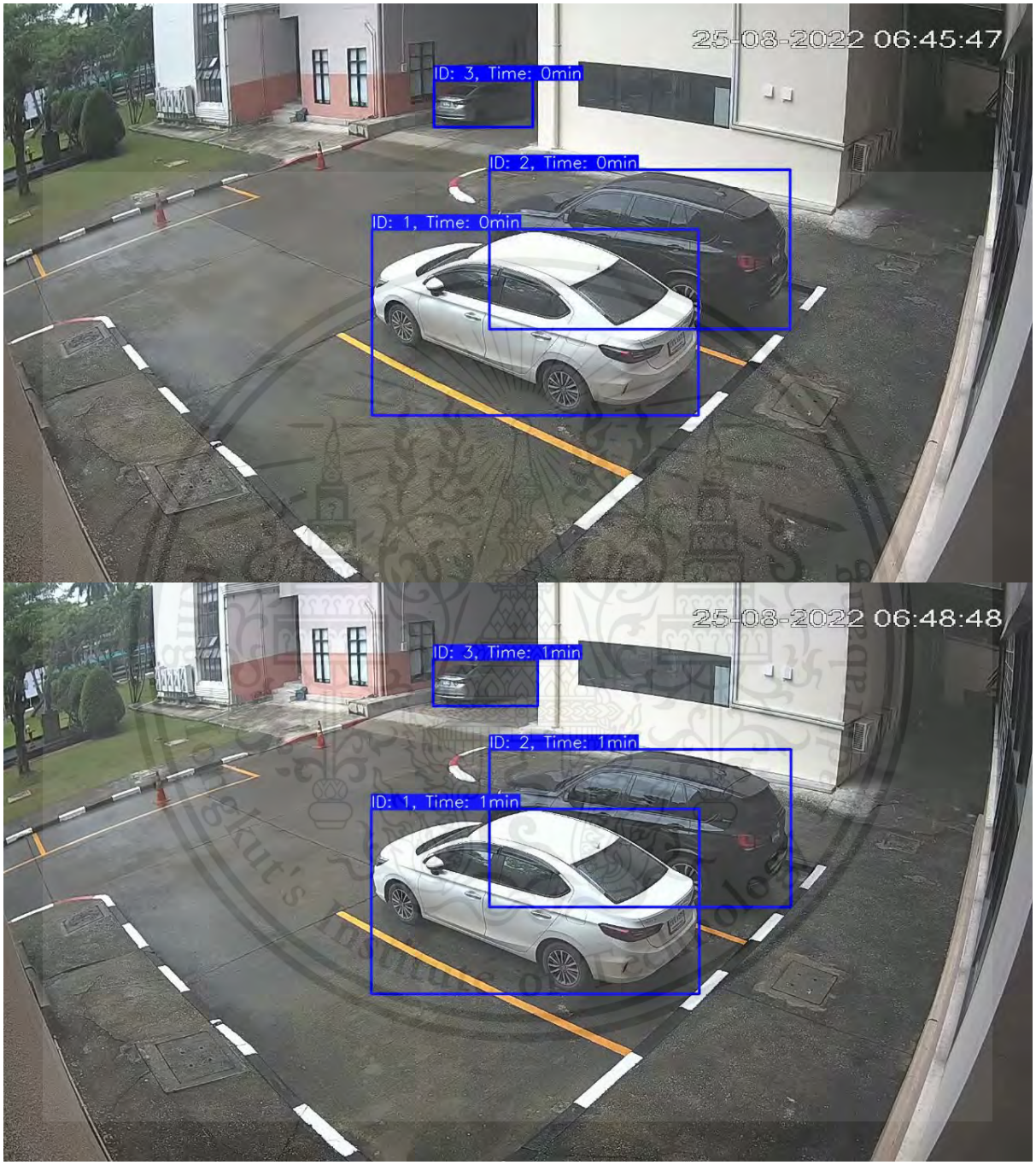
Bachelors in Computer Innovation Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand

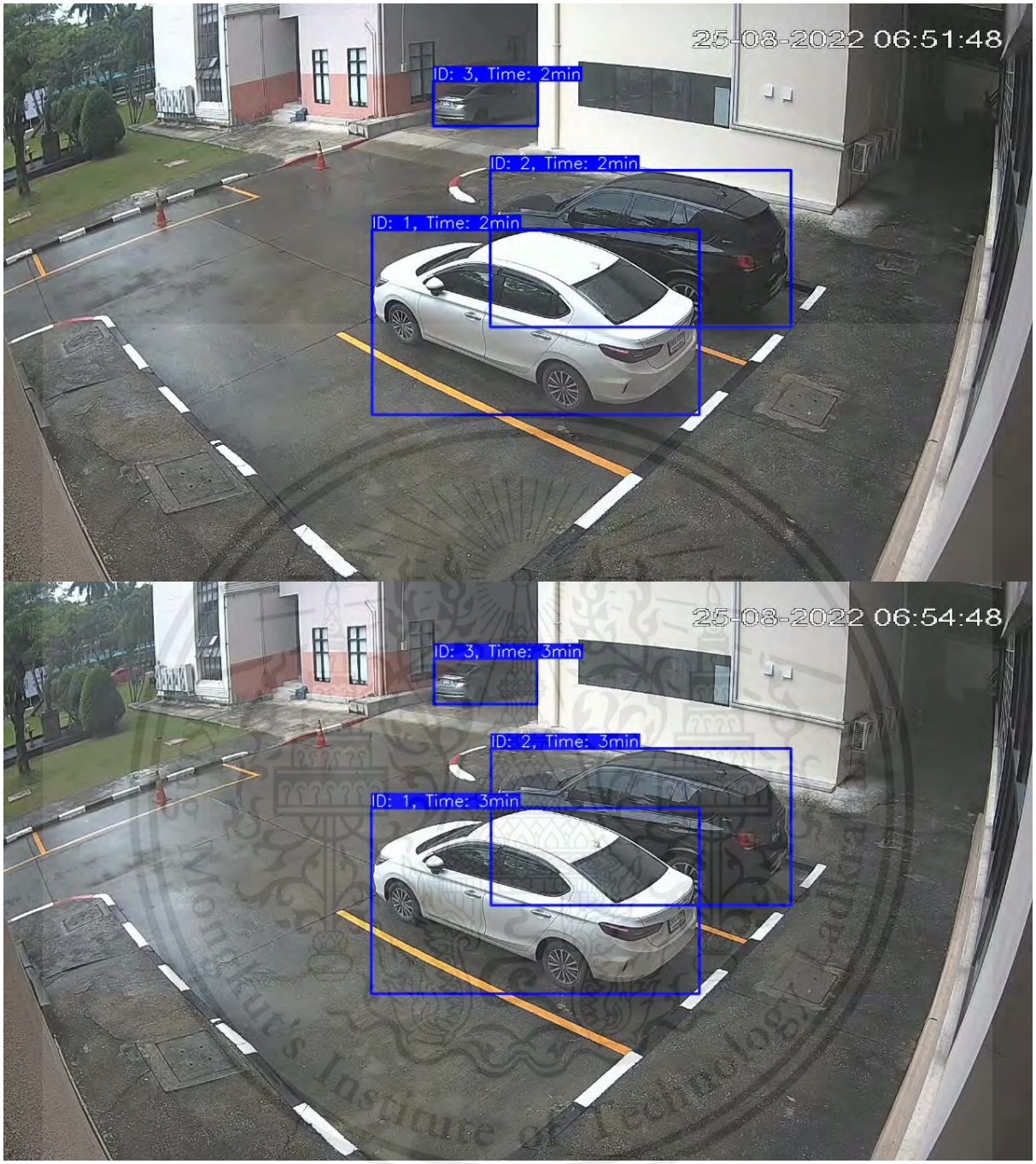
Publication

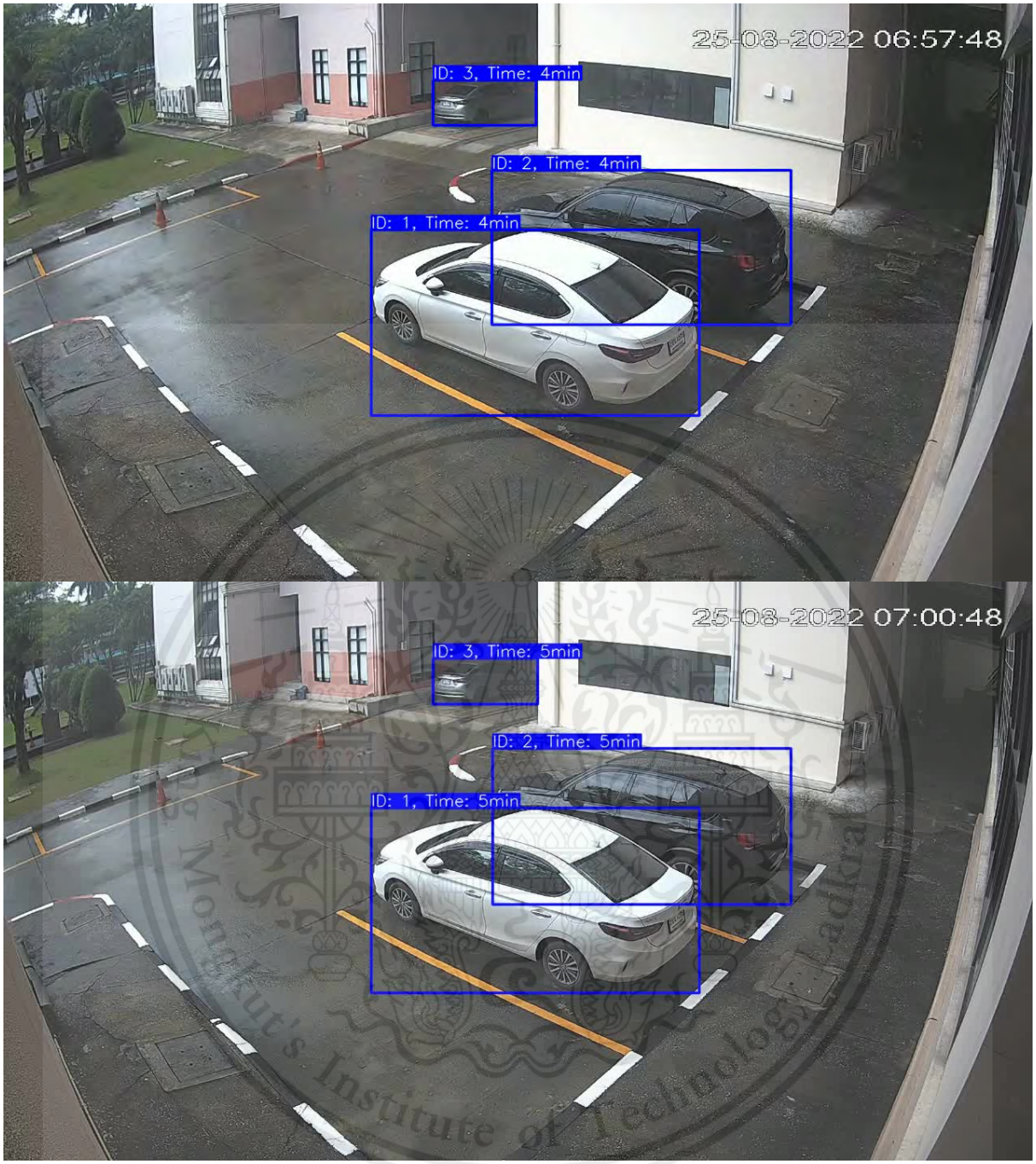
Sharma, N.; Baral, S.; Paing, M.P.; Chawuthai, R. Parking Time Violation Tracking Using YOLOv8 and Tracking Algorithms. *Sensors* **2023**, *23*, 5843. https://doi.org/10.3390/s23135843

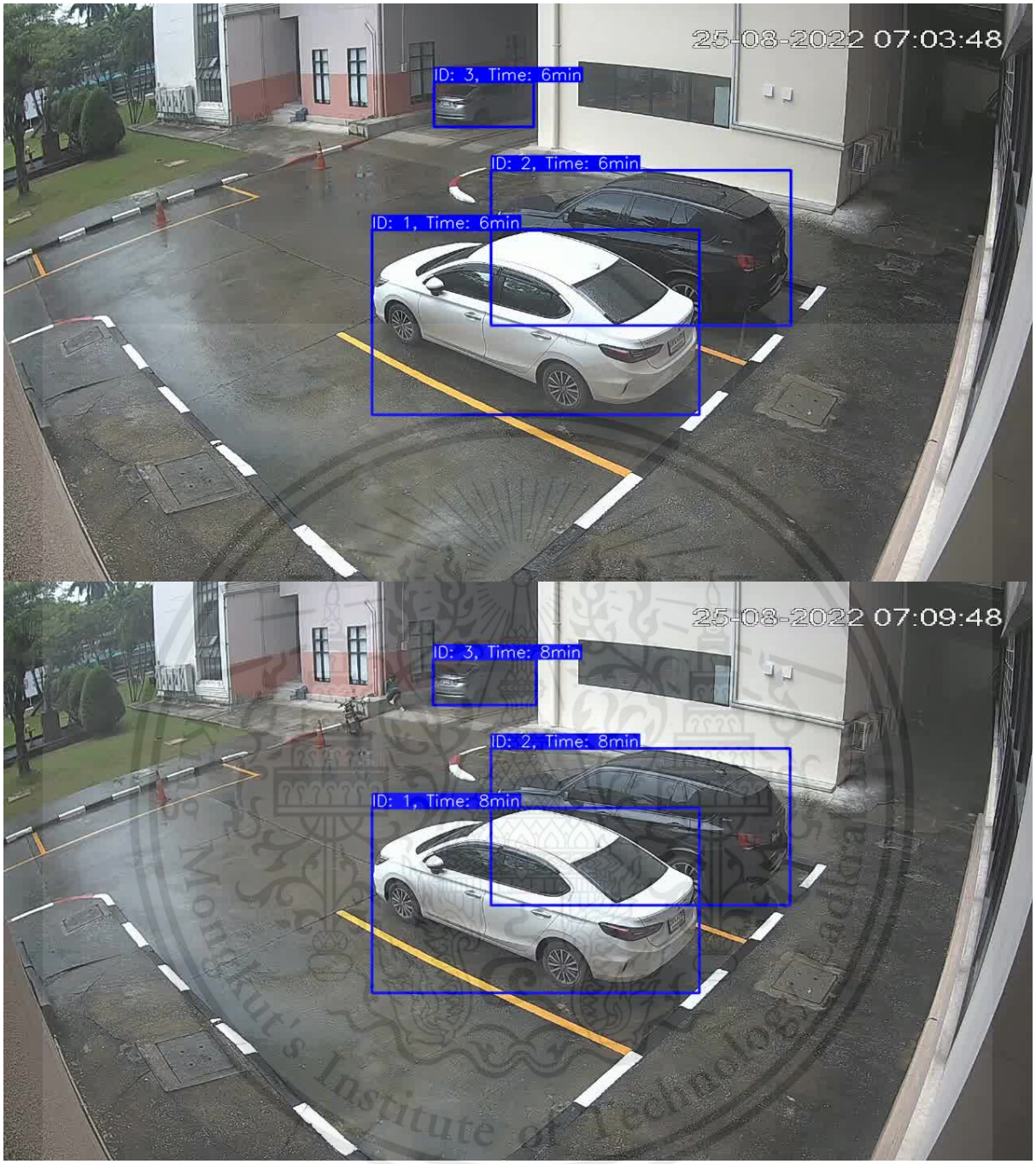
Appendix A

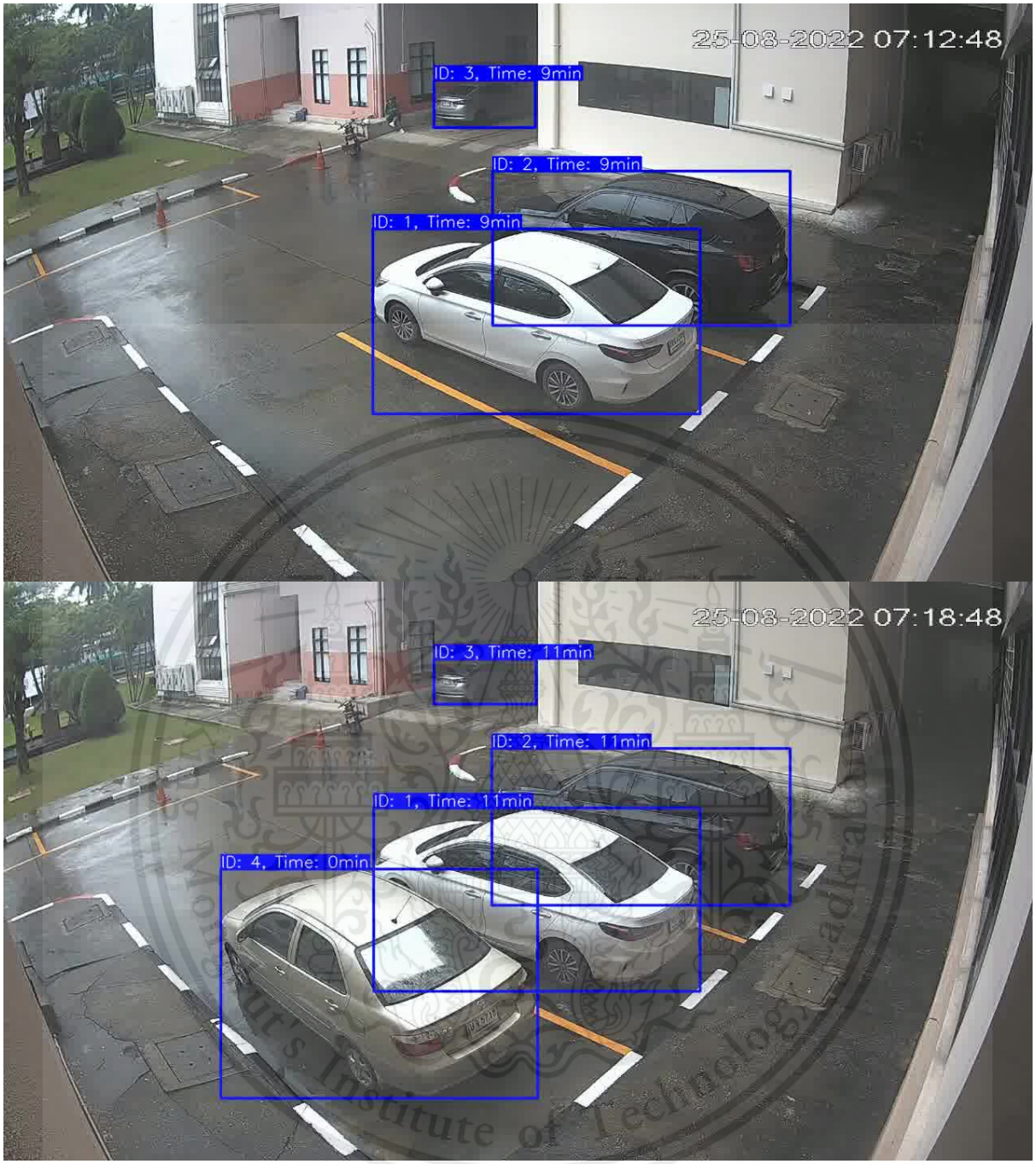
Screenshot of the performance of OCSORT is depicted for E12 building.

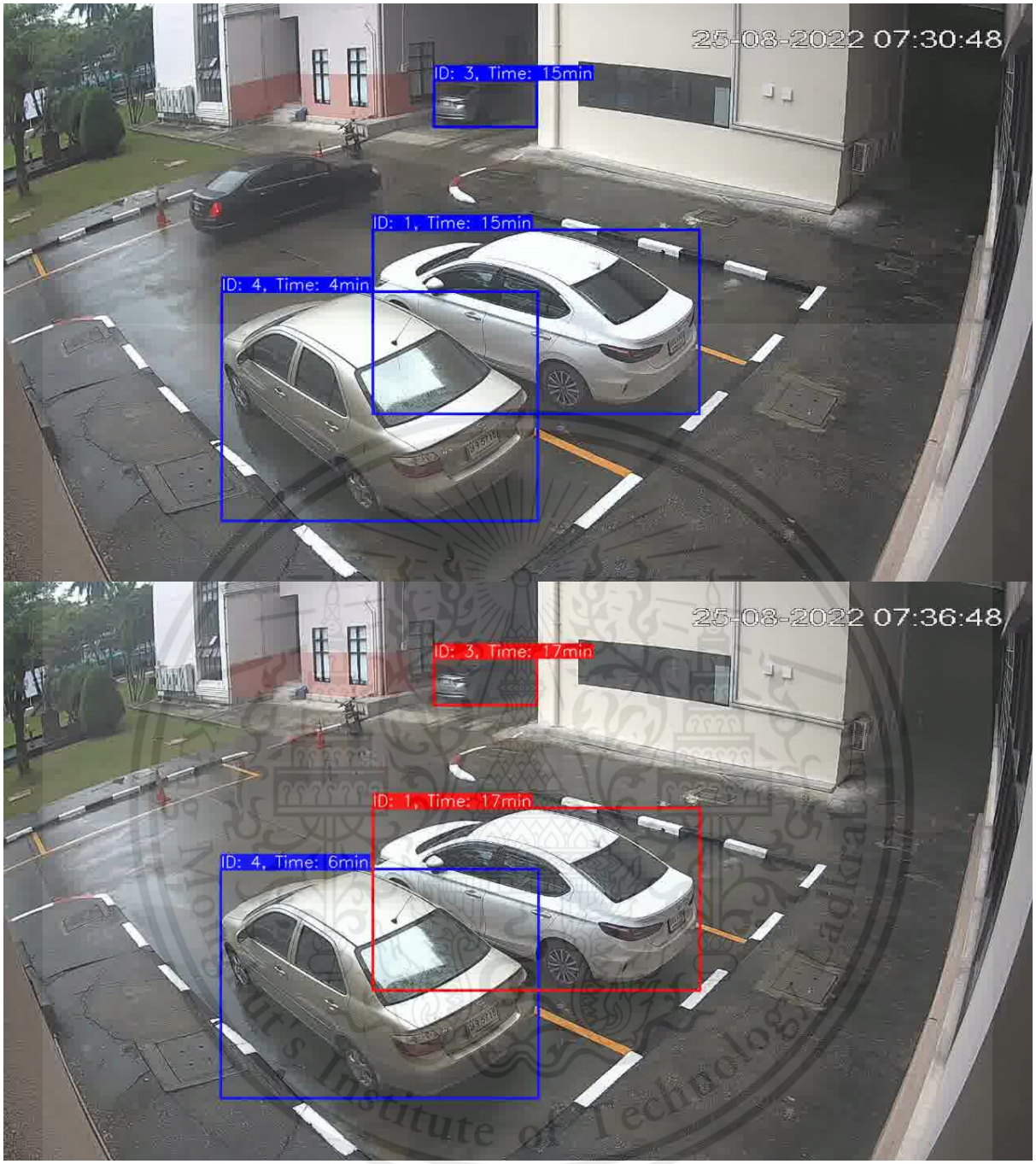


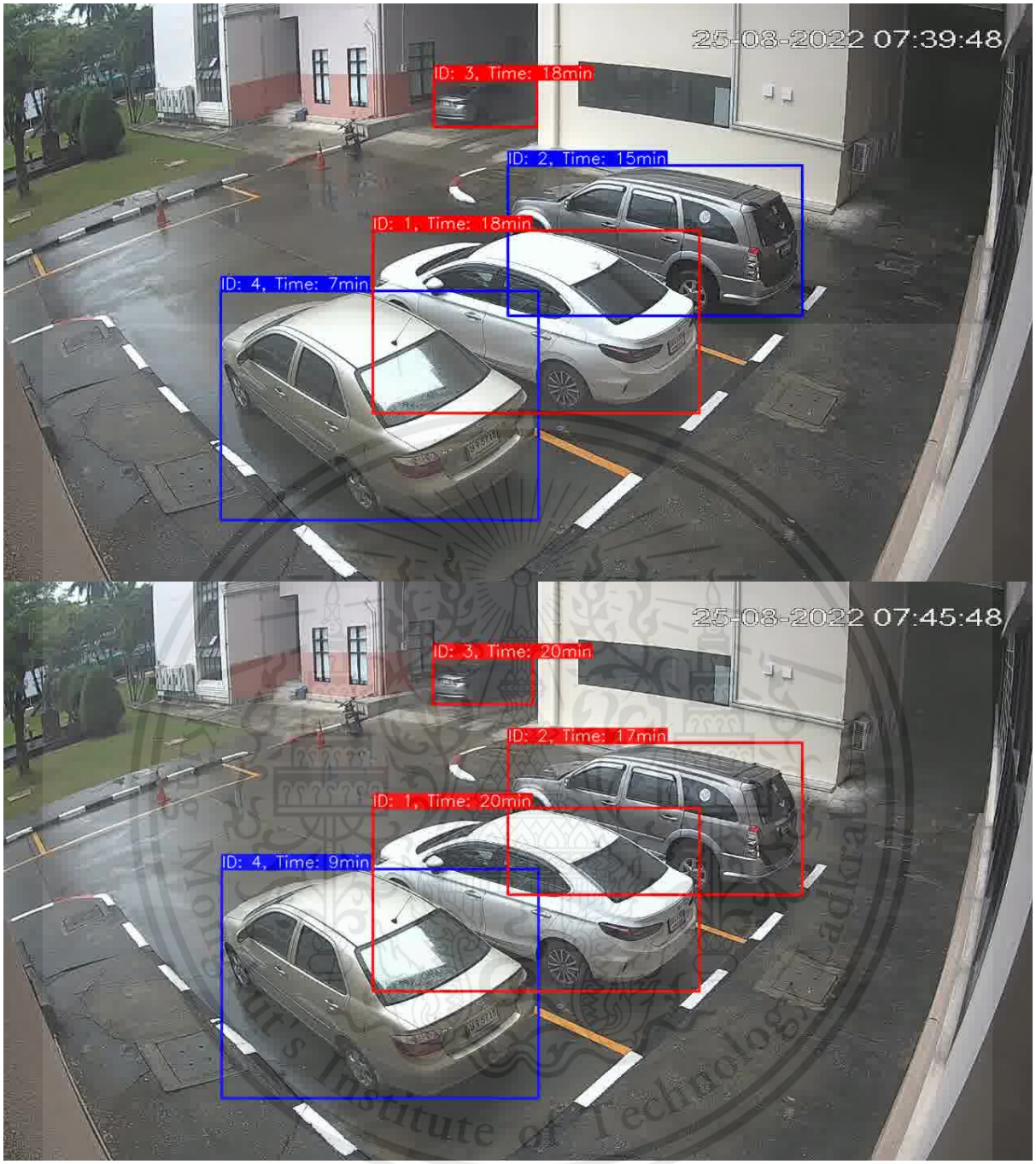


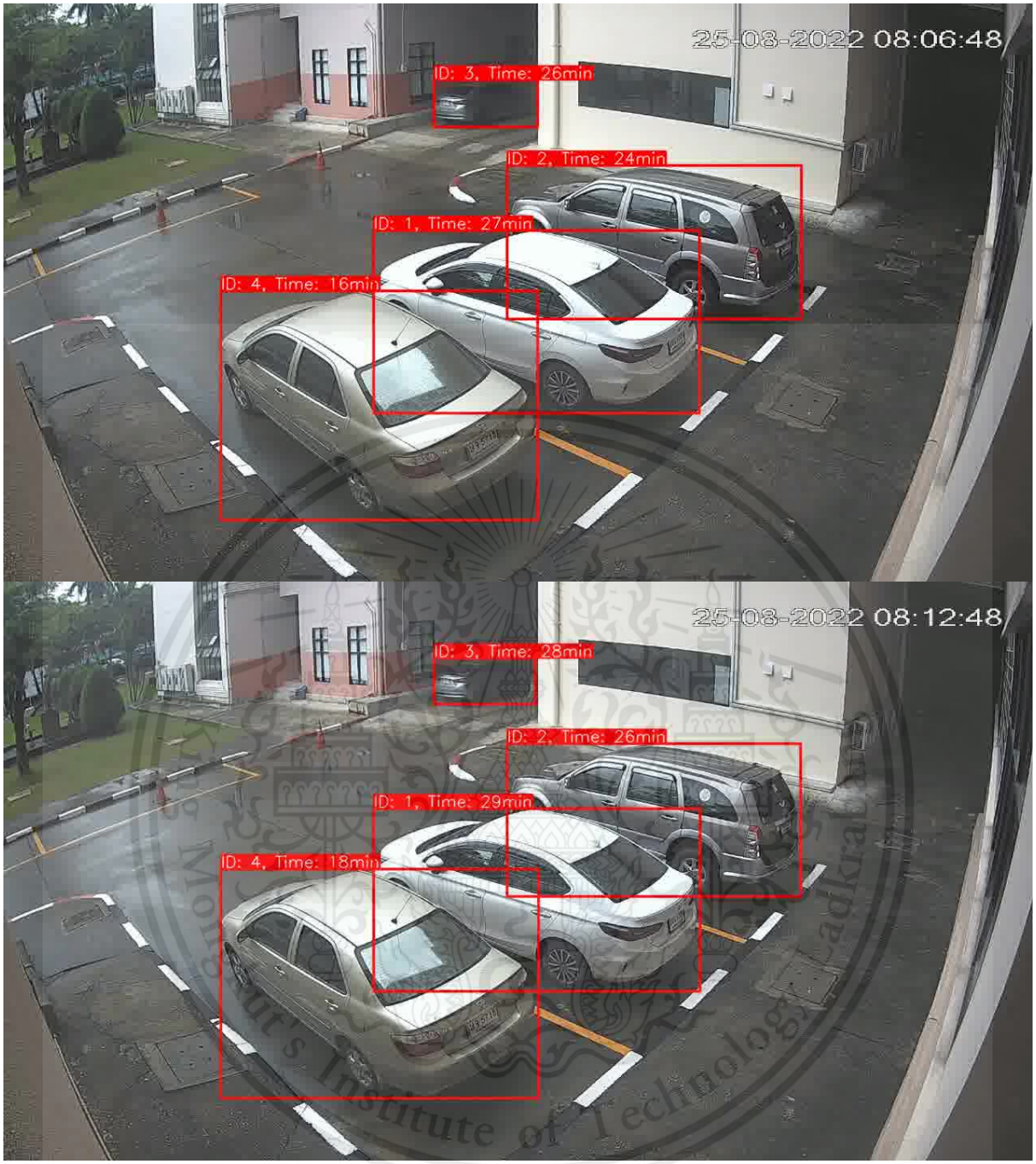












Appendix B

Publication

sensors

MDPI

Article

Parking Time Violation Tracking Using YOLOv8 and Tracking Algorithms

Nabin Sharma [1], Sushish Baral [1], May Phu Paing [2] and Rathachai Chawuthai [3,*]

[1] Department of Robotics and AI, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand; 64601008@kmitl.ac.th (N.S.); sushish.ba@kmitl.ac.th (S.B.)
[2] Department of Biomedical Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand; may.pa@kmitl.ac.th
[3] Department of Computer Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
* Correspondence: rathachai.ch@kmitl.ac.th

Abstract: The major problem in Thailand related to parking is time violation. Vehicles are not allowed to park for more than a specified amount of time. Implementation of closed-circuit television (CCTV) surveillance cameras along with human labor is the present remedy. However, this paper presents an approach that can introduce a low-cost time violation tracking system using CCTV, Deep Learning models, and object tracking algorithms. This approach is fairly new because of its appliance of the SOTA detection technique, object tracking approach, and time boundary implementations. YOLOv8, along with the DeepSORT/OC-SORT algorithm, is utilized for the detection and tracking that allows us to set a timer and track the time violation. Using the same apparatus along with Deep Learning models and algorithms has produced a better system with better performance. The performance of both tracking algorithms was well depicted in the results, obtaining MOTA scores of (1.0, 1.0, 0.96, 0.90) and (1, 0.76, 0.90, 0.83) in four different surveillance data for DeepSORT and OC-SORT, respectively.

Keywords: DeepSORT; OC-SORT; object detection; tracking algorithm; vehicle tracking; YOLOv8

Citation: Sharma, N.; Baral, S., Paing, M.P.; Chawuthai, R. Parking Time Violation Tracking Using YOLOv8 and Tracking Algorithms. *Sensors* **2023**, *23*, 5843. https://doi.org/10.3390/s23135843

Academic Editor: Ikhlas Abdel-Qader

Received: 8 May 2023
Revised: 5 June 2023
Accepted: 16 June 2023
Published: 23 June 2023

1. Introduction

In Thailand, due to the growing number of vehicular registrations, parking has become a serious issue and has introduced an unsolved challenge [1]. The rapid increase in car registrations in Thailand has led to parking problems in cities throughout Thailand. In order to meet the need for parking spaces, shopping malls and mini-marts are trying to effectively manage the limited number of parking spaces that are accessible to the public. In Thailand, mini-marts such as Tesco Express, 7-Eleven, and Mini Big C face a substantial parking problem due to the limited availability of parking spaces. In a fast-paced environment such as 7-Eleven with limited parking facilities, it is essential to track the parking time of each vehicle in the parking lot. This issue of parking violations has attracted the attention of many mini-mart executives and the general public.

Parking violations include activities such as parking a vehicle in a restricted area and time limit violations. Figure 1 shows an example of the implementation of a time restriction parking rule in a mini-mart in Thailand. Most shopping malls and mini-marts use closed-circuit television (CCTV) in parking lots to observe parking time violations. Currently, parking violations are manually checked by the relevant authorities. However, these methods are expensive and incur high labor costs, due to the constant monitoring required to track incoming and outgoing vehicles. To address the parking space problem, various existing parking solutions have used IoT devices to provide drivers with real-time parking details [2]. However, using sensors and hardware, while providing accuracy, also

requires constant maintenance, making it an unideal solution for mini-marts. A low cost parking violation detection system is still required.

The majority of currently available research focuses on assisting drivers to locate parking spots. Numerous models have been developed to assist drivers with nearby parking spaces in real time [3]. However, there are very few studies that have been conducted to help authorities to manage parking spots effectively. Very few studies have been conducted regarding parking violations and time restrictions.

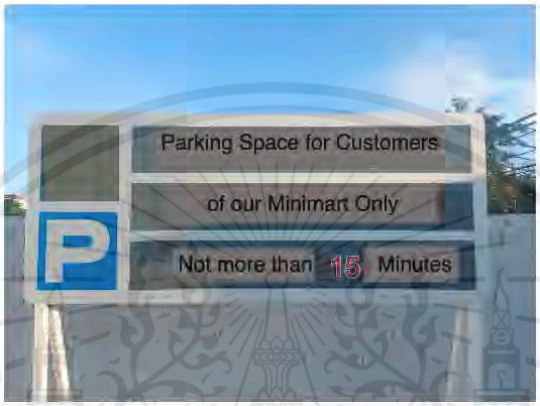


Figure 1. Representative parking restriction image, imposing a parking time limit of 15 min for consumers.

Over the past few years, mini-marts in Thailand have adopted the method of allocating a parking time of 15 min per vehicle outside their stores, in order to keep up with parking demand. However, this method has not been effective, as additional manpower is required to record the arrival time and departure time of each car.

Recently, considerable effort has been made to use CCTV cameras to identify illegal parking practices. Many research studies have employed a Gaussian Mixture Model-based image segmentation approach to retrieve vehicle information [4,5]. Akhawaji et al. [4] further utilizing a Kalman filter to eliminate false alarms and enhance efficient vehicle tracking. However, the effectiveness of this strategy might be affected if the lighting conditions in the operating area change. Apart from image processing algorithms, many Deep Learning algorithms have been deployed to detect parking violations. For instance, networks such as the Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLO) have been implemented to identify parking violations [6–8].

This paper introduces a real-time parking time violation tracking algorithm using closed-circuit cameras and DL models with a tracking algorithm to persist the information from one frame to its subsequent frame. The algorithm leverages the state-of-the-art object detection technique, YOLOv8, to identify vehicles within a parking lot. Each detected vehicle is then assigned a unique ID in the parking lot using DeepSORT/OC-SORT, enabling efficient monitoring of parking time violations. To ensure good performance across varying levels of illumination, weather conditions, and short-term obstructions, optimal parameters were carefully selected for the algorithms. An important advantage of this algorithm is that it does not require prior knowledge about the Region of Interest (ROI), making it adaptable to various parking lot scenarios. This research paper presents the implementation of parking violation detection in a time-series context. The paper's major contributions can be summarized as follows:

(1) The proposed algorithm demonstrates effective vehicle detection and tracking across various natural conditions, exploring the SOTA object detection and tracking models;

(2) The methodology exhibits an accurate and reliable solution to the parking time violation problem;

(3) This paper introduces a frame-by-frame evaluation with respect to time, which is a new approach to the problem domain.

The remaining sections of this paper are organized as follows. Section 2 reviews the background and literature related to object detection and object tracking. Section 3 describes the details of the proposed parking time violation process, including the experimental setup and experimental settings of parameters for the algorithms. Section 4 presents the results and discussion.

2. Related Work

Deep Learning applications have received significant traction due to their extensive research and development. Deep Learning-based methods have been applied in many tasks, such as classification [9], object detection [10], object tracking [11], and healthcare [12–14]. The objective of parking time violation tracking is to identify parking time violations by vehicles, especially in mini-marts which have a limited parking time. The task involves implementation of object detection and object tracking. Therefore, the current literature on object detection and object tracking is discussed accordingly.

2.1. Object Detection

In its early years, the machine learning-based object detection pipeline consisted of two primary steps: ROI extraction and object classification [15]. Based on Zhao et al. [16], object detection models consist of three phases: selection of region of interest, extraction of features, and, finally, classification of objects. One common ROI extraction technique uses sliding windows that move various scales across an image. Exhaustive sliding is, however, very challenging to apply in practice due to its great processing complexity.

Deep Learning algorithms are extensively implemented for object detection tasks. They can mainly be categorized into two types, as shown in Figure 2. You Only Look Once (YOLO) [17] and Single Shot MultiBox Detector (SSD) [18] consider detection tasks as a regression problem and are one stage networks. On the other hand, algorithms such as Region-Based CNN (RCNN) [10] firstly locate the region of interest, which is further classified into classes. As depicted in Figure 3, The R-CNN model uses a selective search algorithm to determine the number of candidates for bounding box object regions, and then features are fed into CNN, which acts as a feature extractor. Support Vector Machine (SVM) is used to classify whether the object is present within that candidate region by utilizing the retrieved features. R-CNN can perform well in various object detection tasks, but training these models takes significant time, and the speed of detection is limited [19].

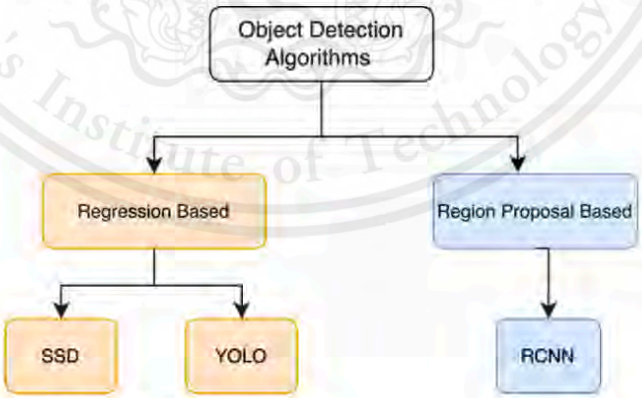


Figure 2. Object detection algorithms.

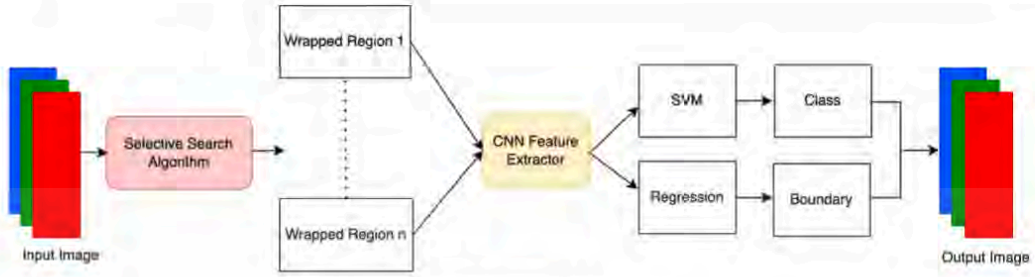


Figure 3. Flowchart of region-based CNN.

To improve the speed of training and inference, one-stage detectors such as SSD and YOLO have been introduced for object detection tasks. Figure 4 shows the architecture of SSD Models. In SSD, CNN-based feature extractors are used. At the end of feature extraction, convolutional feature layers generate predictions at multiple scales. Since SSD does not use region based proposals, SSD models have increased detection speed, as opposed to two-stage detectors such as R-CNN. In this research work, YOLO was implemented for the object detection task. The details of the YOLO model are discussed accordingly in the Methodology section.

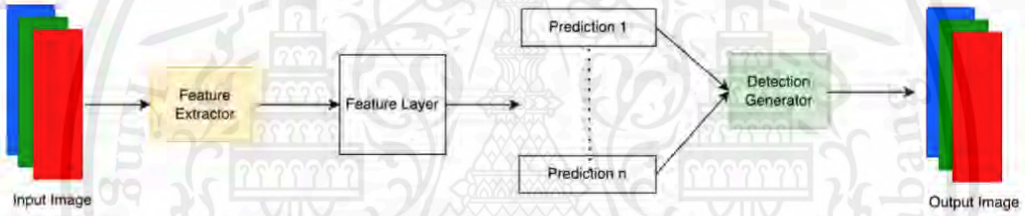


Figure 4. Flowchart of Single Shot MultiBox Detector.

One of the first attempts to tackle the issue of parking lot monitoring using machine learning utilized color vector features on an SVM classifier to distinguish parking spaces inside a parking lot in 2002 [20]. Based on a 3D model of parking spaces, Huang et al. [21] proposed a day and night operating parking space detection system using a Bayesian hierarchical framework. Apart from Deep Learning models, many image processing techniques have been implemented to solve parking space detection problems. For instance, Menéndez et al. [22] proposed temporal analysis of parking area video frames to identify vacant spaces. This method includes the process of background subtraction using the Gaussian mixture to identify and track vehicles in the parking lot. In addition, a transience map was created to observe incoming and outgoing vehicles. Xie et al. [7] proposed an optimized SSD to detect illegal vehicle parking from a video stream in a robust environment, achieving 99% accuracy.

Recently, Patel and Meduri [23] presented an automatic parking space detection algorithm that comprised two steps. The first step included vehicle detection with Faster R-CNN and YOLOv4, while vehicle tracking was used to differentiate between moving and stationary vehicles. Based on Patel and Meduri's research [23], this technique reduced the amount of human effort required by up to 90%. Grbić and Koch [24] proposed occupancy classification and a parking space detection algorithm, wherein parking spaces were determined as occupied or empty using a trained ResNet34 deep classifier; they extensively evaluated this approach on publicly known parking datasets such as PKLot [25] and CNRPark+EXT [26]. Chen et al. [27] implemented an enhanced SSD (Single Shot MultiBox Detector) algorithm for the quick recognition of vehicles in traffic scenarios. For feature

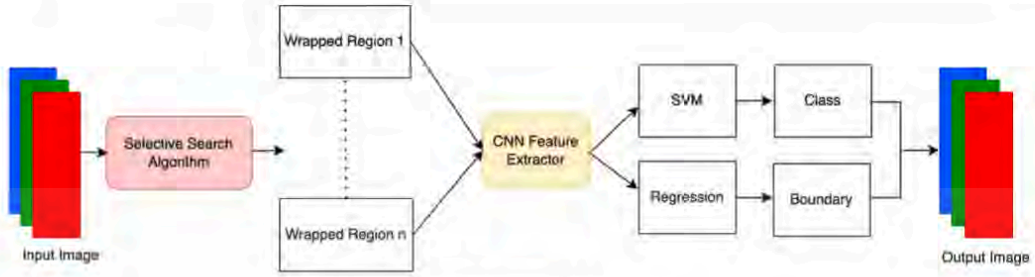


Figure 3. Flowchart of region-based CNN.

To improve the speed of training and inference, one-stage detectors such as SSD and YOLO have been introduced for object detection tasks. Figure 4 shows the architecture of SSD Models. In SSD, CNN-based feature extractors are used. At the end of feature extraction, convolutional feature layers generate predictions at multiple scales. Since SSD does not use region based proposals, SSD models have increased detection speed, as opposed to two-stage detectors such as R-CNN. In this research work, YOLO was implemented for the object detection task. The details of the YOLO model are discussed accordingly in the Methodology section.

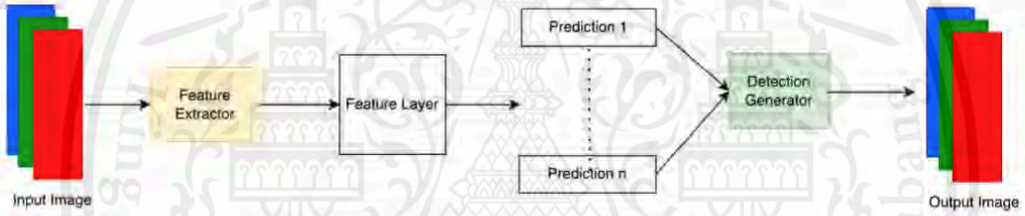


Figure 4. Flowchart of Single Shot MultiBox Detector.

One of the first attempts to tackle the issue of parking lot monitoring using machine learning utilized color vector features on an SVM classifier to distinguish parking spaces inside a parking lot in 2002 [20]. Based on a 3D model of parking spaces, Huang et al. [21] proposed a day and night operating parking space detection system using a Bayesian hierarchical framework. Apart from Deep Learning models, many image processing techniques have been implemented to solve parking space detection problems. For instance, Menéndez et al. [22] proposed temporal analysis of parking area video frames to identify vacant spaces. This method includes the process of background subtraction using the Gaussian mixture to identify and track vehicles in the parking lot. In addition, a transience map was created to observe incoming and outgoing vehicles. Xie et al. [7] proposed an optimized SSD to detect illegal vehicle parking from a video stream in a robust environment, achieving 99% accuracy.

Recently, Patel and Meduri [23] presented an automatic parking space detection algorithm that comprised two steps. The first step included vehicle detection with Faster R-CNN and YOLOv4, while vehicle tracking was used to differentiate between moving and stationary vehicles. Based on Patel and Meduri's research [23], this technique reduced the amount of human effort required by up to 90%. Grbić and Koch [24] proposed occupancy classification and a parking space detection algorithm, wherein parking spaces were determined as occupied or empty using a trained ResNet34 deep classifier; they extensively evaluated this approach on publicly known parking datasets such as PKLot [25] and CNRPark+EXT [26]. Chen et al. [27] implemented an enhanced SSD (Single Shot MultiBox Detector) algorithm for the quick recognition of vehicles in traffic scenarios. For feature

extraction, the authors used the MobileNet v2 network, which helped to improve the detection accuracy of the algorithm. Experimental results showed that the proposed algorithm achieved 82.59% and 84.83% accuracy for the BDD100K and KITTI datasets, respectively. By introducing asymmetric convolution and a global point tracking module, Li et al. [28] proposed YOLO–GCC based on the YOLO algorithm and achieved an average accuracy of 83.0% on the TSD–MAX dataset. Jung et al. [29] proposed classification and localization of vehicles in traffic monitoring, using ResNet50 for vehicle classification, adding an optimized drop-down convolutional neural network (DropCNN) to improve the performance of classification.

2.2. Vehicle Tracking

Object tracking is a technique for detecting objects across frames by utilizing their spatial and temporal characteristics. In its simplest form, the method consists of obtaining the initial set of detections, giving them distinct IDs, and following them over frames, which is the essence of object tracking. Over the last decade, object tracking methods have gained popularity in the fields of Deep Learning and computer vision.

Single Object Tracking (SOT) and Multiple Object Tracking (MOT) are the two categories into which object tracking can be subdivided [30]. A Multiple Object Tracking algorithm's primary responsibility is to identify multiple objects in a frame, assign and preserve their identities, and follow the object's trajectory across input frames.

Object tracking has been used in many domains such as pedestrian tracking [31], vehicle tracking [32], and player tracking [33]. Parico and Ahamed [34] implemented YOLOv4 for pear detection and a multiple object tracking algorithm, DeepSORT, for pear tracking and counting. Hou et al. [32] proposed a tracking algorithm called DeepSORT and a low-confidence search filter that reduced false detections produced by the original DeepSORT algorithm. To track vehicles detected by a YOLOv3 network, Liu and Liu [35] proposed 3-D constrained multiple kernels aided by Kalman filtering. In this article, we focus on a closed-circuit television (CCTV)-based multiple object tracking technique, which allows identification of multiple objects in a sequence of images. The effectiveness of such a tracking method depends on the quality of detection in various weather settings, the proportion of occlusion, and illumination change. Therefore, it is crucial to choose the best object detection and tracking algorithm. Aware of our objective, state-of-the-art tracking algorithms such as DeepSORT and OCSORT were implemented. Table 1 shows the benchmark results on the DanceTrack dataset. OC-SORT outperforms all the existing tracking algorithm in many metrics, such as HOTA, AssA and IDF1. This result shows strong evidence of the high performance of the OC-SORT Algorithm.

Table 1. Results on the DanceTrack test set.

Tracker	HOTA	DetA	AssA	MOTA	IDF1
SORT [36]	47.9	72.0	31.2	91.8	50.8
DeepSORT [11]	45.6	71.0	29.7	87.8	47.9
ByteTrack [37]	47.3	71.6	31.4	89.5	52.5
OC-SORT [38]	54.6	80.4	40.2	89.6	54.6
OCSORT + Linear Interp [38]	55.1	80.4	40.4	92.2	54.9

3. Proposed Parking Time Violation Algorithm

The proposed algorithm utilizes a state-of-the-art object detection algorithm called YOLOv8 [39] for detecting vehicles. For tracking vehicles, two object tracking algorithms were implemented, Deep SORT (Simple Online Real Tracking) [11] and Observation-Centric SORT (OC-SORT) [38]. In-depth discussion of the algorithm is included in the subsection below.

3.1. Dataset

Due to limitations and privacy concerns, acquiring mini-mart data was not possible. As such, the dataset was comprised of video footage obtained from CCTV cameras installed at King Mongkut's Institute of Technology, Ladkrabang (KMITL). The CCTV model utilized at KMITL is the Panasonic V series, which offers a 15–30 frames per second (FPS) capture rate and a 1080P resolution. These cameras are IP66 rated, providing resistance to water and dust.

Data collection was conducted using 4 cameras placed at different locations within the university campus, all with a resolution of 1080P and a frame rate of 15 FPS. The selection of dataset locations was carefully carried out to incorporate diverse camera angles and viewpoints. The video samples within the dataset encompassed various daylight and weather settings, offering a comprehensive representation of the parking lot scenarios at KMITL.

3.2. Vehicle Detection

In 2015, Joseph Redmon and Ali Farhadi from the University of Washington created the cutting-edge object recognition algorithm called YOLO (You Only Look Once) [17]. YOLO outperforms many other object detection algorithms such as R-CNN and DPM [17]. YOLO sees the entire image during training and testing, in contrast to sliding window- and region proposal-based approaches, implicitly capturing contextual information about classes as well as their appearance. The initial version of YOLO had the ability to quickly recognize objects in images, but it had trouble locating smaller objects precisely. Since the data in our problem domain does not contain very small images, there is no problem using YOLO for this study. YOLO divides input images into a grid of dimensions M × M. For instance, a grid cell is in charge of detecting an object if its center falls inside that grid cell. Each grid cell predicts the bounding box and offers a confidence score for the associated boxes. In YOLO, confidence is described as *Pr(Object)* × *IOU* where *Pr(Object)* represents the probability of the presence of an object, and IOU represents the Intersection over Union (IOU), i.e., the overlap area between inference and ground truth. Each grid cell generates five predictions (x, y, w, h, and a confidence score). Additionally, each grid produces p conditional class probabilities, expressed as *Pr(Class | Object)*. Equation (2) below demonstrates the method to obtain class-specific confidence scores for each box during the test phase.

$$\text{Class}_i \times \text{IOU}_{\text{pred}}^{\text{truth}} = \text{Class}_i \mid \text{Object} \times \text{Object} \times \text{IOU}_{\text{pred}}^{\text{truth}} \tag{1}$$

The final layers predict both the coordinates of their bounding boxes and their associated class probabilities. Then, the bounding boxes are normalized to fall between 0 and 1. All further layers increase non-linearity by using the leaky rectified linear activation function, as described in Equation (2), with an exception of the final layer, which employs a linear activation function:

$$f(x) = \begin{cases} x, & \text{if } x > 0. \\ 0.1x, & \text{otherwise.} \end{cases} \tag{2}$$

In 2016, YOLOv2 was released, improving on the original model by including batch normalization, anchor boxes, and dimension clusters [40]. Similarly, YOLOv3 was released in 2018, improving the model's performance by employing a more efficient backbone network, incorporating a feature pyramid, and employing focal loss [41]. The initial model of YOLO had many successors, including YOLOv4, YOLOv5, YOLOv6, and YOLOv7. In 2023, YOLOv8 was released by Ultralytics [39]. Figure 5 shows the detailed architecture of YOLOv8. Figure 6–9 show images of the dataset.

In the backbone architecture, the C2f module, based on Cross Stage Partial (CSP), is used in YOLOv8, as opposed to the C3 Module used in YOLOv5. The architecture of CSP enhances the learning capacity of CNN and decreases the computational effort of the model. The C2f module comprises two Conv Modules and n BottleNeck, connected through Split

and Concat. The remainder of the backbone park is the same as that in YOLOv5. At the final layer of the backbone, the SPPF Module is used.

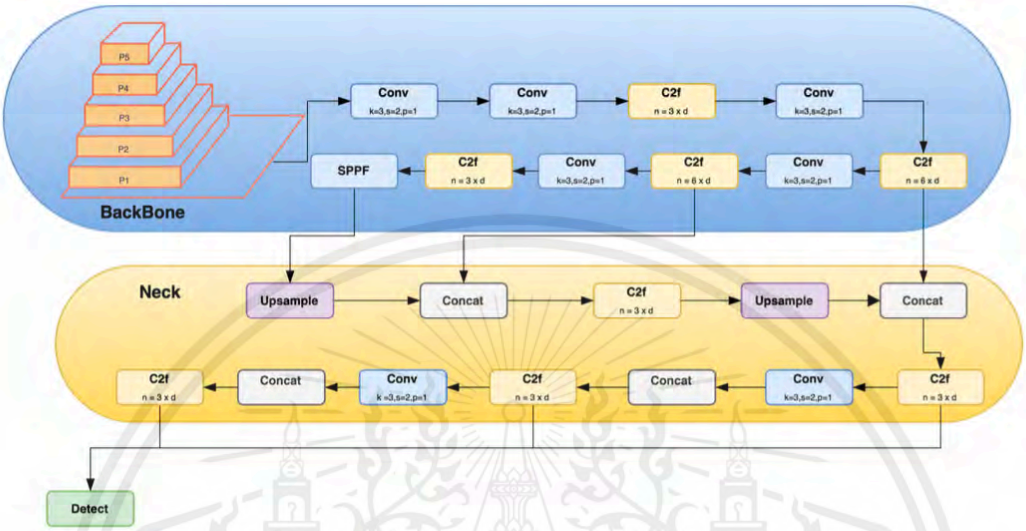


Figure 5. YOLOv8 architecture.

Figure 6. CCTV footage from Location 1.

Figure 7. CCTV footage from Location 2.

Figure 8. CCTV footage from Location 3.

Figure 9. CCTV footage from Location 4.

3.3. Movement Tracking

Once successfully able to detect vehicles in a parking lot, object tracking algorithms could be deployed to track the vehicles. Two tracking algorithms were implemented in the research. The first algorithm was DeepSORT. Figure 10 shows the workflow of our algorithm. The DeepSORT algorithm was implemented to track each vehicle throughout the frame. DeepSORT is an extension of SORT (Simple Online Realtime Tracking) [11]. DeepSORT uses appearance descriptors to minimize identity shifts, increasing the effectiveness of tracking. For problems involving the prediction of temporal or time series data, Kalman filtering is the used algorithm.

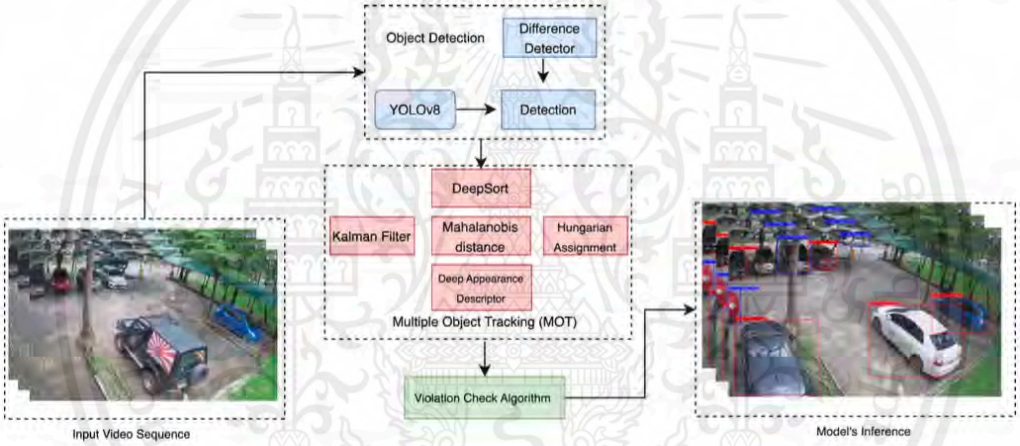


Figure 10. Workflow of our proposed work.

The second tracking algorithm that was implemented was Observation-Centric SORT (OC-SORT) [38]. OC-SORT was published in CVPR 2023. On numerous datasets, including MOT17, MOT20, KITTI, head tracking, and, particularly, DanceTrack, where the object motion is very non-linear, OC-SORT produces state-of-the-art results. To improve the accuracy and robustness during a period of occlusion, OC-SORT implements object observations to compute a virtual trajectory. Figure 11 shows the pipeline of OC-SORT.

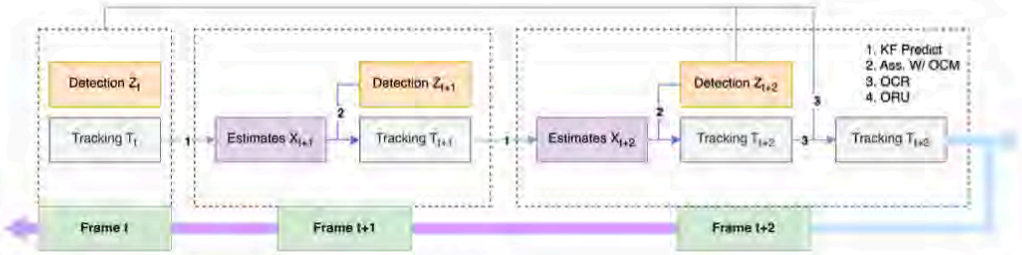


Figure 11. Pipeline of OC-SORT.

3.4. Time Violation

After successful integration of vehicle detection, followed by vehicle tracking that assigns a unique ID to each unique vehicle, a simple time violation algorithm was deployed to track every car in the frame. Every minute, the presence of the vehicle was checked. If the vehicle did not exist in the next 10 consecutive frames, then the ID of the vehicle was discarded. On the other hand, if the ID was present in 15 consecutive frames, the vehicle was determined to have violated the time restriction. In the figure, a blue label is associated with no violation, while a red label is associated with violation.

3.5. Experimental Setup

This section discusses the experimental platform, providing details of the in-depth flow of the algorithm, including the chosen parameters. The workstation used for the experiment was Ubuntu Linux. All experiments were conducted in a 3.6 GHz Intel Xeon Quad-Core processor with 8GB RAM and an NVIDIA Quadro P4000 graphics card.

A Mean Average Precision (mAP) of 53.9 was achieved by YOLOv8x, which was higher than all other YOLO versions. Therefore, YOLOv8x was chosen for vehicle detection. A pretrained model was used, as the class was already trained on the MS COCO dataset. The resolution of the video feed was 1080P (1920 × 1080), with a frame rate of 15 fps. For YOLOv8, the image-size parameter was set to 640. The model thus resized the longest dimension to be 640, i.e., size 1920 became 640, while maintaining the aspect ratio. Therefore, the resized images were close to 640 × 360. Classes were filtered to be (2, 5, 7) i.e., car, bus, and truck, as the violation algorithm was checking for big vehicles. The confidence threshold was set to 0.5. The probability of the class occurring in the bounding box was evaluated using the confidence score. The maximum object detection parameter was set to 100.

For tracking vehicles, DeepSORT/OC-SORT tracking algorithms were used. The max-age parameter was set to 10. For DeepSORT algorithm, Max Age preserves the ID of a vehicle for a threshold number of frames before deleting the ID. The n_init parameter was set to 2. The n_init parameter refers to the number of objects detected before initializing the tracking of the object. max_cosine_distance was set to 0.3. Max cosine distance was the threshold to identify vehicle similarity by the DeepSORT algorithm. The trajectory parameter was set to False, as the trajectory of the vehicle over time was not required.

3.6. Validation Criteria

Object detection algorithms such as YOLOv8 use Mean Average Precision (mAP) to validate the performance of object detection. Since the objective of this work was to track vehicles across a frame, it was essential to validate the results with an object tracking algorithm. MOTA (Multiple Object Tracking Algorithm) metrics were used to validate the results. MOTA metrics measure the accuracy of both detection and tracking algorithms. Firstly, it was essential to define True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). Table 2 shows the detailed definitions. The formula for MOTA is shown in Equation (3). FN refers to no match for the ground truth object. This can occur

because of various reasons, such as the vehicle being too far away from the camera, weather, different illumination, etc. FP refers to no ground truth object but a match by the algorithm. There was one instance of FP where the reflection of the vehicle was considered as a vehicle. False IDs refer to identification switches between the tracked vehicles. GT refers to total ground truth object.

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDS_t}{\sum_t GT_t} \quad (3)$$

Table 2. Definition and description of TP, TN, FP, and FN.

Definition	Description
True Positive (TP)	Vehicle is present and the algorithm can track vehicle.
True Negative (TN)	Vehicle is not present and the algorithm does not track vehicle
False Positive (FP)	Vehicle is not present but the algorithm tracks vehicle.
False Negative (FN)	Vehicle is present but the algorithm does not track vehicle

4. Results and Discussion

Figure 12 illustrates samples of vehicle detection and tracking, using YOLOv8 with two different object tracking algorithms across various daylight and weather settings.The performance of YOLOv8 with DeepSORT proved to be superior for the dataset in contrast to the recently introduced OC-SORT algorithm, which had exhibited notable advancements across numerous MOT datasets. MOTA scores of (1.0, 1.0, 0.96, 0.90) and (1, 0.76, 0.90, 0.8) were achieved for DeepSORT and OC-SORT tracking algorithms as shown in Table 3. The OC-SORT algorithm did not exhibit satisfactory results on this specific dataset. It consistently encountered challenges in maintaining consistent vehicle identification, particularly in the presence of rainy conditions. The algorithm frequently switched vehicle IDs, leading to inaccuracies in tracking which impacted the overall performance. In Figures 13 and 14, the tracking of each algorithm on the Location 2 dataset can be observed. In the provided sample frame (f23401) from both Figures 13 and 14, it is observed that the DeepSORT algorithm successfully tracked the vehicle IDs, while the OC-SORT algorithm struggled to accurately track the IDs. The challenges faced by OC-SORT in correctly tracking the IDs could potentially be attributed to factors such as rainy weather conditions and the presence of visually similar vehicles in close proximity. These factors may have led to difficulties in distinguishing between vehicles, resulting in ID switching. The robustness of DeepSORT in handling such scenarios could be attributed to its utilization of deep appearance feature embedding, which allows for more reliable tracking, even in challenging conditions. In the case of Location 1, the count for False Negatives (FN) was observed to be 3 and 4 for DeepSORT and OC-SORT, respectively, due to the presence of reflections from vehicles on the windows. These reflections interfered with the vehicle detection process, causing some vehicles to be falsely labeled as negatives which, thus, affected the MOTA scores.

Overall, based on the experiment conducted, it is clear that the performance of the vehicle tracking algorithm is greatly influenced by factors such as camera position and weather variation, particularly when using a single camera for inference. To achieve optimal tracking results, positioning the camera at or near a top-view perspective can greatly enhance the performance of the vehicle tracking algorithm. This camera placement minimizes the occurrence of vehicle overlapping, leading to improved tracking accuracy. By capturing a top-down view of the monitored area, the camera can provide a clear and unobstructed line of sight to the vehicles, reducing the chances of them overlapping or obstructing each other. This unambiguous view allows the tracking algorithm to more accurately detect and track individual vehicles, ensuring reliable and precise tracking results. Thus, selecting a top-view perspective for camera placement is crucial in minimizing vehicle overlap and maximizing the effectiveness of the vehicle tracking algorithms.

Figure 12. Sample images of detection and tracking by our model.

Table 3. Performance comparison of the two tracking algorithms.

Model	Dataset	FP	FN	IDS	MOTA
YOLOv8 + DeepSORT	Location 4	0	0	0	1
	Location 2	0	0	0	1
	Location 3	1	0	0	0.96
	Location 1	0	3	0	0.90
YOLOv8 + OC-SORT	Location 4	0	0	0	1
	Location 2	0	0	7	0.76
	Location 3	1	0	2	0.90
	Location 1	0	4	1	0.83

Figure 13. Tracking of DeepSORT on the Location 2 dataset.

Figure 14. Tracking of OC-SORT on the Location 2 dataset.

5. Conclusions

Manual parking time violation tracking is unideal, due to its requirements for manpower and high cost. In this research paper, a parking time violation tracking algorithm was successfully demonstrated; this has the potential to reduce additional manpower and labor costs. The algorithm can be implemented in many parking lot settings. The YOLOv8 algorithm was used to detect vehicles in the parking lot, while DeepSORT and OC-SORT were used to track the vehicles throughout the frame. For the DeepSORT algorithm, the MOTA scores were recorded as (1.0, 1.0, 0.96, 0.90) across four different surveillance datasets. This indicates a high level of accuracy and effectiveness in tracking multiple objects. On the other hand, the OC-SORT algorithm achieved MOTA scores of (1.0, 0.76, 0.90, 0.83) across the same datasets. While still achieving relatively high scores, there is a slight variation compared to DeepSORT, particularly in the second dataset, where the MOTA score dropped to 0.76 due to frequent ID switching. Overall, both algorithms showcased their ability to effectively track objects in the given surveillance data. These results provide valuable insights into the tracking performance of each algorithm and can aid in selecting the most suitable algorithm based on specific tracking requirements and dataset characteristics. The performance of the detection model might be increased by customized training of the model with vehicles. This is not necessary, but may help the algorithm to become more robust.

In future studies, researchers can explore the synchronization of the algorithm with two or more cameras. This synchronization would enable the algorithm to detect parking time violations in situations where multiple CCTV cameras are deployed. By integrating the feeds from multiple cameras, the algorithm can gather a more comprehensive view of the parking lot, improving the accuracy of violation detection. This scalability is important for expanding the algorithm's applicability to larger parking areas or scenarios where multiple cameras are already in use.

Author Contributions: Conceptualization, N.S., S.B. and R.C.; Methodology, N.S., R.C. and S.B.; Validation, N.S., M.P.P. and R.C.; Analysis, S.B.; Investigation, S.B.; Resources, N.S.; Data collection, R.C., N.S. and S.B.; Draft Preparation, N.S.; Review and Editing, R.C., N.S. and S.B.; Visualization, N.S.; Supervision, R.C., S.B. and M.P.P.; Project Administration, R.C., S.B. and M.P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research did not receive any external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from KMITL to use the dataset for research purposes.

Data Availability Statement: The data presented in the research paper are available on request, as they represent private parking lot data from KMITL.

Acknowledgments: We would like to acknowledge our sincere gratitude to the office of Information Technology of the School of Engineering, King Mongkut's Institute of Technology, Ladkrabang, for video data.

Conflicts of Interest: The authors declare no conflict of interests.

References

1. CEIC Flex. *Thailand Number of Registered Vehicles*; CEIC: Bangkok, Thailand, 2023.
2. Dinh, T.; Kim, Y. A Novel Location-Centric IoT-Cloud Based On-Street Car Parking Violation Management System in Smart Cities. *Sensors* **2016**, *16*, 810. [CrossRef]
3. Joseph, J.; Patil, R.; Narahari, S.; Didagi, Y.; Bapat, J.; Das, D. Wireless Sensor Network Based Smart Parking System. *Sensors Transducers* **2014**, *162*, 5–10.
4. Akhawaji, R.; Sedky, M.; Soliman, A.H. Illegal Parking Detection Using Gaussian Mixture Model and Kalman Filter. In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 30 October–3 November 2017; pp. 840–847. [CrossRef]
5. Sarker, M.M.K. Detection and recognition of illegally parked vehicles based on an adaptive gaussian mixture model and a seed fill algorithm. *J. Inf. Commun. Converg. Eng.* **2015**, *13*, 97–204. [CrossRef]
6. Chin Kit, N.; Cheong, S.; Yap, W.; Foo, Y.L. Outdoor Illegal Parking Detection System Using Convolutional Neural Network on Raspberry Pi. *Int. J. Eng. Technol.* **2018**, *7*, 17. [CrossRef]
7. Xie, X.; Wang, C.; Chen, S.; Shi, G.; Zhao, Z. Real-Time Illegal Parking Detection System Based on Deep Learning. *CoRR* **2017**, abs/1710.02546. Available online: http://xxx.lanl.gov/abs/1710.02546 (accessed on 15 April 2023).
8. Tang, H.; Peng, A.; Zhang, D.; Liu, T.; Ouyang, J. SSD Real-Time Illegal Parking Detection Based on Contextual Information Transmission. *Comput. Mater. Contin.* **2019**, *61*, 293–307. [CrossRef]
9. Tamang, T.; Baral, S.; Paing, M.P. Classification of White Blood Cells: A Comprehensive Study Using Transfer Learning Based on Convolutional Neural Networks. *Diagnostics* **2022**, *12*, 2903. [CrossRef]
10. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014. Available online: http://xxx.lanl.gov/abs/1311.2524 (accessed on 15 April 2023).
11. Wojke, N.; Bewley, A.; Paulus, D. Simple Online and Realtime Tracking with a Deep Association Metric. *CoRR* **2017**, abs/1703.07402, Available online: http://xxx.lanl.gov/abs/1703.07402 (accessed on 20 April 2023).
12. Paing, M.P.; Pintavirooj, C. Adenoma Dysplasia Grading of Colorectal Polyps Using Fast Fourier Convolutional ResNet (FFC-ResNet). *IEEE Access* **2023**, *11*, 16644–16656. [CrossRef]
13. Paing, M.P.; Cho, O.S.; Cho, J.W. Histopathological Classification of Colorectal Polyps using Deep Learning. In Proceedings of the 2023 International Conference on Information Networking (ICOIN), Bangkok, Thailand, 11–14 January 2023; pp. 472–477. [CrossRef]
14. Keakultanes, R.; Paing, M.P.; Pintavirooj, C. Automatic Cardiopulmonary Resuscitation System. In Proceedings of the 2022 14th Biomedical Engineering International Conference (BMEiCON), Songkhla, Thailand, 10–13 November 2022; pp. 1–5. [CrossRef]
15. Janai, J.; Güney, F.; Behl, A.; Geiger, A. Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art. *arXiv* **2017**, arXiv:1704.05519. Available online: http://xxx.lanl.gov/abs/1704.05519 (accessed on 15 April 2023).
16. Zhao, Z.Q.; Zheng, P.; tao Xu, S.; Wu, X. Object Detection with Deep Learning: A Review. 2019. Available online: http://xxx.lanl.gov/abs/1807.05511 (accessed on 29 April 2023).
17. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. 2016. Available online: http://xxx.lanl.gov/abs/1506.02640 (accessed on 15 April 2023).
18. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision – ECCV 2016*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 21–37. [CrossRef]
19. Mao, Q.C.; Sun, H.M.; Liu, Y.B.; Jia, R.S. Mini-YOLOv3: Real-Time Object Detector for Embedded Applications. *IEEE Access* **2019**, *7*, 133529–133538. [CrossRef]
20. Dan, N. Parking Management System and Methods. U.S. Patent 10/066,215, 31 July 2003.
21. Huang, C.C.; Tai, Y.S.; Wang, S.J. Vacant Parking Space Detection Based on Plane-Based Bayesian Hierarchical Framework. *IEEE Trans. Circuits Syst. Video Technol.* **2013**, *23*, 1598–1610. [CrossRef]
22. Menéndez, J.M.; Postigo, C.; Torres, J. Vacant parking area estimation through background subtraction and transience map analysis. *IET Intell. Transp. Syst.* **2015**, *9*, 835–841. [CrossRef]
23. Patel, R.; Meduri, P. Car detection based algorithm for automatic parking space detection. In Proceedings of the 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 14–17 December 2020; pp. 1418–1423.
24. Grbić, R.; Koch, B. Automatic vision-based parking slot detection and occupancy classification. *Expert Syst. Appl.* **2023**, *225*, 120147. [CrossRef]
25. de Almeida, P.R.L.; Oliveira, L.S.; Britto, A.S., Jr.; Silva, E.J., Jr.; Koerich, A.L. PKLot—A Robust Dataset for Parking Lot Classification. *Expert Syst. Appl.* **2015**, *42*, 4937–4949. [CrossRef]
26. Amato, G.; Carrara, F.; Falchi, F.; Gennaro, C.; Meghini, C.; Vairo, C. Deep learning for decentralized parking lot occupancy detection. *Expert Syst. Appl.* **2017**, *72*, 327–334. [CrossRef]
27. Chen, Z.; Guo, H.; Yang, J.; Jiao, H.; Feng, Z.; Chen, L.; Gao, T. Fast vehicle detection algorithm in traffic scene based on improved SSD. *Measurement* **2022**, *201*, 111655. [CrossRef]
28. Li, Y.; Chen, Y.; Yuan, S.; Liu, J.; Zhao, X.; Yang, Y.; Liu, Y. Vehicle detection from road image sequences for intelligent traffic scheduling. *Comput. Electr. Eng.* **2021**, *95*, 107406. [CrossRef]

29. Jung, H.; Choi, M.K.; Jung, J.; Lee, J.H.; Kwon, S.; Jung, W.Y. ResNet-Based Vehicle Classification and Localization in Traffic Surveillance Systems. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 934–940. [CrossRef]
30. Luo, W.; Xing, J.; Milan, A.; Zhang, X.; Liu, W.; Kim, T.K. Multiple object tracking: A literature review. *Artif. Intell.* **2021**, *293*, 103448. [CrossRef]
31. Sun, Z.; Chen, J.; Chao, L.; Ruan, W.; Mukherjee, M. A Survey of Multiple Pedestrian Tracking Based on Tracking-by-Detection Framework. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 1819–1833. [CrossRef]
32. Hou, X.; Wang, Y.; Chau, L.P. Vehicle Tracking Using Deep SORT with Low Confidence Track Filtering. In Proceedings of the 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Taipei, Taiwan, 18–21 September 2019; pp. 1–6. [CrossRef]
33. Buric, M.; Ivasic-Kos, M.; Pobar, M. Player Tracking in Sports Videos. In Proceedings of the 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Sydney, Australia, 11–23 December 2019; pp. 334–340. [CrossRef]
34. Parico, A.I.B.; Ahamed, T. Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT. *Sensors* **2021**, *21*, 4803. [CrossRef]
35. Liu, T.; Liu, Y. Deformable Model-Based Vehicle Tracking and Recognition Using 3-D Constrained Multiple-Kernels and Kalman Filter. *IEEE Access* **2021**, *9*, 90346–90357. [CrossRef]
36. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple Online and Realtime Tracking. *arXiv* **2016**, arXiv:1602.00763. Available online: http://xxx.lanl.gov/abs/1602.00763 (accessed on 15 April 2023).
37. Zhang, Y.; Sun, P.; Jiang, Y.; Yu, D.; Weng, F.; Yuan, Z.; Luo, P.; Liu, W.; Wang, X. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. 2022. Available online: http://xxx.lanl.gov/abs/2110.06864 (accessed on 10 May 2023).
38. Cao, J.; Pang, J.; Weng, X.; Khirodkar, R.; Kitani, K. Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking, 2023. Available online: http://xxx.lanl.gov/abs/2203.14360 (accessed on 10 May 2023).
39. Jocher, G.; Chaurasia, A.; Qiu, J. YOLO by Ultralytics. 2023. Available online: https://github.com/ultralytics/ultralytics (accessed on 23 May 2023).
40. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. *CoRR* **2016**, abs/1612.08242, Available online: http://xxx.lanl.gov/abs/1612.08242 (accessed on 10 May 2023).
41. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *CoRR* **2018**, abs/1804.02767, Available online: http://xxx.lanl.gov/abs/1804.02767 (accessed on 12 May 2023).