Hindawi
International Journal of Digital Multimedia Broadcasting
Volume 2024, Article ID 6667738, 14 pages
https://doi.org/10.1155/2024/6667738

Research Article

Proposing Smart System for Detecting and Monitoring Vehicle Using Multiobject Multicamera Tracking

Phat Nguyen Huu,[1] Bang Nguyen Anh,[1] and Quang Tran Minh[2,3]

[1]*School of Electrical and Electronic Engineering, Hanoi University of Science and Technology (HUST), Hanoi, Vietnam*
[2]*Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam*
[3]*Vietnam National University Ho Chi Minh City (VNU-HCM), Ho Chi Minh City, Vietnam*

Correspondence should be addressed to Phat Nguyen Huu; phat.nguyenhuu@hust.edu.vn

Received 9 June 2023; Revised 6 December 2023; Accepted 22 March 2024; Published 29 April 2024

Academic Editor: Enas Musa Allozi

Nowadays, the strong development of the economy and society has driven the increase in traffic participation, making traffic management increasingly difficult. To effectively address this issue, AI applications are being applied to improve urban traffic management and operations. Therefore, we propose a smart system to detect and monitor vehicles across multiple surveillance cameras. Our system leverages data collected from traffic surveillance cameras and harnesses the power of deep learning technology to detect and track vehicles smoothly. To achieve this, we use the YOLO model for detection in conjunction with the DeepSORT algorithm for precise vehicle tracking on each camera. Furthermore, our system uses a ResNet backbone model for feature extraction of objects within each camera's frame. It utilizes cosine distance to identify similar objects in other cameras, facilitating multicamera tracking. To ensure optimal performance, our system is implemented using the NVIDIA DeepStream SDK, enabling it to achieve an impressive speed of 21 fps on each camera and an average of precision approximately 85% for three modules. The results of our study affirm the system's suitability and its potential for practical applications in the field of urban traffic management.

Keywords: DeepSORT; feature extraction; Re-ID; tracking vehicle; YOLOv5

1. Introduction

Multiobject multicamera tracking (MOMCT) is aimed at predicting the trajectories of all objects across multiple cameras. This means that when an object appears on multiple cameras, the system can recognize it as the same object and assign a unique ID to it throughout the entire system. The MOMCT system has numerous applications such as human surveillance, retail analysis, and intelligent traffic monitoring. In addition, the installation of multiple cameras on roads, intersections, toll stations, and traffic vehicles is becoming more popular with the development of information and communication technologies, so it has become necessary to automate the MOMCT system. In this article, we present a proposed system for vehicle tracking in a large city. This is a big challenge because the system does not have overlapping areas for each camera and is not diverse enough to handle illumination conditions and adverse weather. Furthermore, traffic in large cities like Hanoi or Ho Chi Minh City in Vietnam is dense, with vehicles moving chaotically as shown in Figure 1.

Current approaches commonly decompose the problem into distinct subtasks, including object detection, single-camera multiobject tracking, and cross-camera object association. Nevertheless, these methods still cannot be applied to real-world camera network systems due to slow processing speeds and hardware limitations. As illustrated in Figure 2, our proposed system shares similarities with the pipeline. However, we will leverage several algorithms and optimizations to reduce computational costs and enhance hardware acceleration. Firstly, we employ YOLO for real-time vehicle detection. Next, we utilize the DeepSORT algorithm for tracking vehicles within each individual camera. Subsequently, we use ResNet to extract appearance features of the

Figure 1: Illustrating image of urban traffic in Vietnam [1].

vehicles. Finally, we link objects and their trajectories across cameras using cosine similarity of feature embeddings.

Furthermore, we will present the data we utilized and the results of testing and evaluating the performance of the proposed MOMCT system in the context of real-world traffic scenarios. Our objective is to demonstrate that the proposed MOMCT system is a robust solution capable of practical deployment on a large scale in complex urban traffic environments. The key contributions of this paper are as follows:

- Constructing a dataset tailored to urban traffic in Vietnam
- Selecting deep learning models to detect objects and extract features of vehicles
- Deploying an algorithm to link objects using the extracted features and compare similarity based on cosine similarity
- Building and deploying the system written in C++ and optimized for speed using NVIDIA hardware to support multicamera CCTV setups

The rest of the paper includes four parts and is organized as follows. Section 1 introduces the related systems. In Section 2, we present related work. In Section 3, we present the proposal system. In Section 4, the proposed system is evaluated and the results are analyzed. In the final section, we give conclusions and future research directions.

2. Related Work

Vehicle tracking is an important task for various applications, such as traffic monitoring, urban management, security, and autonomous driving [2–9]. One of the single-camera methods for vehicle detection in urban traffic surveillance images is proposed by Zhang, Li, and Yang [2]. The method uses convolutional neural networks with feature concatenation to extract robust features from the images and classify them into vehicle or nonvehicle categories. The method achieved high precision and recall rates on the PKU-VD dataset. Another single-camera method for vehicle tracking is proposed by Nguyen, Thi, and Quynh [3]. The method uses YOLO to detect the lanes and obstacles on the road and a PID controller to adjust the steering angle and speed of the self-driving car. The method also uses a camera mounted on the car to capture the surrounding environment and send the data to a cloud server for processing. The method achieved high accuracy and stability on the UCF dataset.

One of the multicamera methods for vehicle tracking is proposed by Li et al. [4]. The method uses mask R-CNN for object detection and transfer learning for reidentification (Re-ID). The method also uses a graph-based optimization algorithm to associate the vehicle trajectories across different cameras. The method achieved the best performance on the AI City Challenge 2022 dataset. Another multicamera method for vehicle tracking is proposed by Nguyen et al. [5]. The method uses a deep neural network to extract features from the vehicle images and a k-nearest neighbor algorithm to match them across different cameras. The method also uses a shortest path algorithm to recommend the optimal route for the vehicles based on the traffic conditions. The method achieved good results on the HCMC dataset.

Some recent works on vehicle tracking are proposed in [6–8]. The former method uses YOLOv4 for object detection and an improved DeepSORT algorithm for object tracking. The method also uses a Kalman filter and a cosine metric to estimate the state and the appearance of the vehicles. The method achieved high precision and recall rates on the MOT16 dataset. The latter method uses YOLOv3 for object detection and classification and a convolutional neural network for license plate recognition. The method also uses a database to store and retrieve the vehicle information and a payment system to charge the toll fee. The method achieved high accuracy and efficiency on the Indian toll plaza dataset [8]. A different approach by Song et al. [9] proposed a transformer-based camera link model with spatial and temporal filtering to conduct cross-camera tracking. The model leveraged appropriate loss functions and distance measures to handle occlusion, illumination, and shadow challenges. The model showed the effectiveness on the NVIDIA Cityflow V2 dataset. However, the model required a large amount of training data and computational resources, which might limit its applicability in real-world scenarios.

Another work [10] proposed a deep learning–based framework for MOMCT that utilized mask R-CNN for object detection and transfer learning for Re-ID. This paper proposes a novel framework for MTMCT of vehicles that does not depend on any camera configuration or synchronization. The framework consists of two main components: metadata-aided Re-ID (MA-ReID) and trajectory-based camera link model (TCLM). MA-ReID is a method that combines metadata features (such as color, type, and license plate) and image features (extracted by a deep neural network) to reidentify vehicles across different cameras. TCLM is a method that uses the entry and exit information of vehicles to automatically construct a camera link model that captures the spatial and temporal relations among the cameras. By using TCLM, the framework can reduce the candidate search space for MA-ReID and improve the efficiency and accuracy of MTMCT. The framework achieved an IDF1 score of 76.77% on the CityFlow dataset, which outperforms

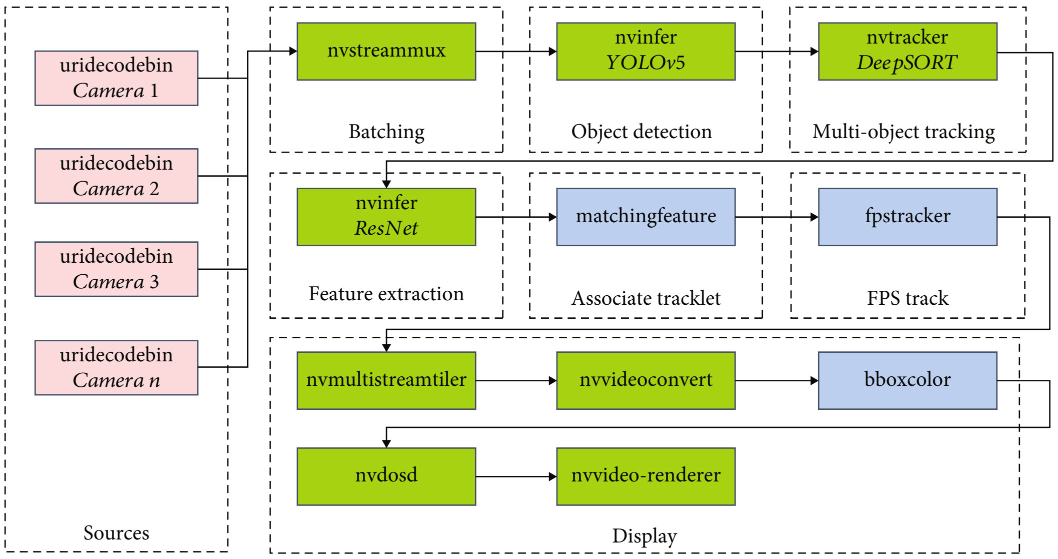

Figure 2: The block diagram of the proposed system.

the state-of-the-art MTMCT methods. However, it is worth noting that the use of TCLM requires knowledge of the entry/exit information of each object, making the method unsuitable for real-time applications.

However, vehicle tracking in urban environments poses many challenges, such as occlusion, illumination changes, camera motion, and scale variation [11–13]. Moreover, most existing methods for vehicle tracking are based on either single-camera or multicamera settings, which have their own limitations. Single-camera methods can only track vehicles within a limited field of view, while multicamera methods require complex camera calibration and synchronization. In addition, the development of algorithms for tracking multiple objects using cameras is being widely used in aquaculture [11, 12, 14]. In these documents, they use many cameras to monitor the bottom in shrimp farming to detect abnormalities and warn farmers. This is also a new research direction that we are implementing [13, 15, 16]. In the paper, we build an algorithm to predict water parameters using the long short-term memory (LSTM) model. We hope to develop these algorithms not only for smart transportation but also for other smart underwater monitoring systems such as smart shrimp ponds or smart homes. Another research direction is object identification based on 2D and 3D models. This is also a new approach that we are interested in [17–19].

To overcome these challenges, some recent works have proposed to use deep learning techniques for vehicle detection and tracking in urban environments. For example, the paper [20] by Pham et al. uses a four-step process: image preprocessing, vehicle detection, vehicle segmentation, and vehicle classification. The method uses a sliding window technique with a Haar-like feature extractor and an AdaBoost classifier for vehicle detection, a watershed algorithm for vehicle segmentation, and a support vector machine (SVM) classifier with geometric and color features for vehicle classification. The method achieved high accuracy in detecting and classifying vehicles in urban scenes in Vietnam. However, this method did not address the problem of vehicle tracking across multiple cameras, nor did it consider the temporal information of the vehicles. Moreover, this method used a relatively simple feature extractor and classifier, which might not be able to handle complex and diverse vehicles in urban environments.

In contrast to these works, our paper proposes a smart system for detecting and monitoring vehicles using MOMCT that does not rely on any camera calibration or synchronization. Our system leverages the power of deep learning technology and state-of-the-art algorithms to detect, track, and reidentify vehicles across multiple cameras in real time. Our system can handle dynamic camera changes, occlusion, illumination variation, scale variation, and other challenges in urban environments. Our system also outperforms the state-of-the-art methods on several benchmark datasets in terms of accuracy, efficiency, and robustness. We will describe our system in detail in the next section.

3. Proposal System

In this section, we mainly offer implementation solutions, designed to optimize the accuracy and speed of the system. An evaluation of the proposed method and the existing method is carried out from the results received. The purpose of the system is to track objects through various cameras. The system can detect the same vehicle and assign an ID (global ID) to it for an object appearing at the same time at many different cameras. Firstly, it is necessary to clearly define the objects that the system is targeting. It then provides suitable handling solutions for the objects.

The object of the system is the area of the traffic route with many surveillance cameras mounted at different corners. The integrated system will provide a multicamera monitoring solution for the route through which to know where the vehicles are going and their location.

Based on the actual requirements and context, the system has the following main functions:

- Detecting and classifying vehicles in traffic
- Vehicle tracking on each camera
- Multicamera vehicle tracking

3.1. System Overview. In this section, we propose the block diagram of the system as follows.

Figure 2 shows a brief description of the multicamera vehicle monitoring system, in detail:

- *Input*: The input to the system is a video stream streamed directly from the surveillance camera systems via the Real-Time Streaming Protocol (RTSP).
- *Vehicle detection*: The system will detect and classify the vehicles on the road using the YOLOv5 model from the input images.
- *Vehicle tracking*: Detected vehicles will use the DeepSORT algorithm to track each camera.
- *Feature extraction and Re-ID*: Each vehicle after having its ID on each camera (local ID) will go through the ResNet backbone to extract features. The Re-ID of each vehicle will then be performed based on the similarity cosine distance between the two feature vectors. If two vehicles have similar feature vectors, they will have the same ID on all cameras (global ID).
- *Output*: The system will display the location of all vehicles and localize them. The vehicles with the same global ID will have the same bounding box.

3.2. System Pipeline in DeepStream. In Figure 2, the system used a pipeline to run with the NVIDIA DeepStream engine. The blue elements are built by ourselves because these functions are not yet supported by NVIDIA DeepStream. The remaining green and pink are available elements of NVIDIA DeepStream. The pipeline of the system includes the following elements as follows:

- *uridecodebin*: used to decode video streams from different sources, such as video files, RTSP, or Hypertext Transfer Protocol (HTTP) input. When used in DeepStream, uridecodebin is used to take video data and convert them into DeepStream-compatible frames.
- *nvstreammux*: used to combine multiple videos into a single stream to process them simultaneously. It allows DeepStream applications to process multiple video streams at the same time, significantly reducing the hardware resources required for multistream processing
- *nvinfer YOLOv5*: used to perform processing on video and image data. This plugin provides the ability to integrate AI and deep learning machine models for object detection, classification, and behavior detection. In the object detection module, we use the YOLOv5 model. The output will be information about the position, size of bounding boxes, labels, and confidence.
- *nvtracker DeepSORT*: used to track objects in real-time video. This plugin offers a variety of different algorithms, including person-based and audience-based tracking.
- *nvinfer extract feature*: used to perform feature extraction model inference of each pretrained vehicle. The output will be the feature vector of the means.
- *matchingfeature*: used to match feature vectors, managing matched vehicles. The output will be the global ID of the vehicles. With this plugin that we have developed, we extract feature information from the metadata of NVIDIA DeepStream for detected objects. Subsequently, these extracted features of objects are compared to one another, and a global ID is assigned across the entire camera system if they match. Further details regarding the algorithm are presented in Figure 3.
- *nvmultistreamtiler*: used to split and display video streams on the screen or record them. This element provides the ability to display multiple video streams at the same time on the same screen, making it easy for users to observe and monitor them.
- *nvvideoconvert*: used to convert video formats between different types. This element provides the ability to convert popular video formats such as H.264, H.265, and MPEG-4 into other formats such as NV12, RGBA, or BGR.
- *bboxcolor*: used to generate random colors for bounding boxes and manage the colors until the object disappears from the frame. We built this element to be able to color the bounding boxes of matching cars the same color. With this plugin, we have designed it to support visualization. For vehicles with different IDs, the plugin generates a random color for the bounding box border. For vehicles with global IDs, the plugin generates a random color for the background of the bounding box. This enables supervisors or administrators to easily track trajectories and identify similar vehicles across the surveillance camera system. The visualization results of this plugin are presented in Figures 4 and 5.
- *fpstracker*: used to calculate the processing speed per thread. The unit of measurement is frames per second (FPS). With this plugin, supervisors can monitor the processing speed of each camera stream. It aids managers in gaining insight into the current state of the AI traffic surveillance system and any issues that may arise with the cameras.
- *nvdosd*: used to display overlay information on the video. This element provides the ability to add information such as name, address, time, and location information to the video to help users easily track and identify objects.

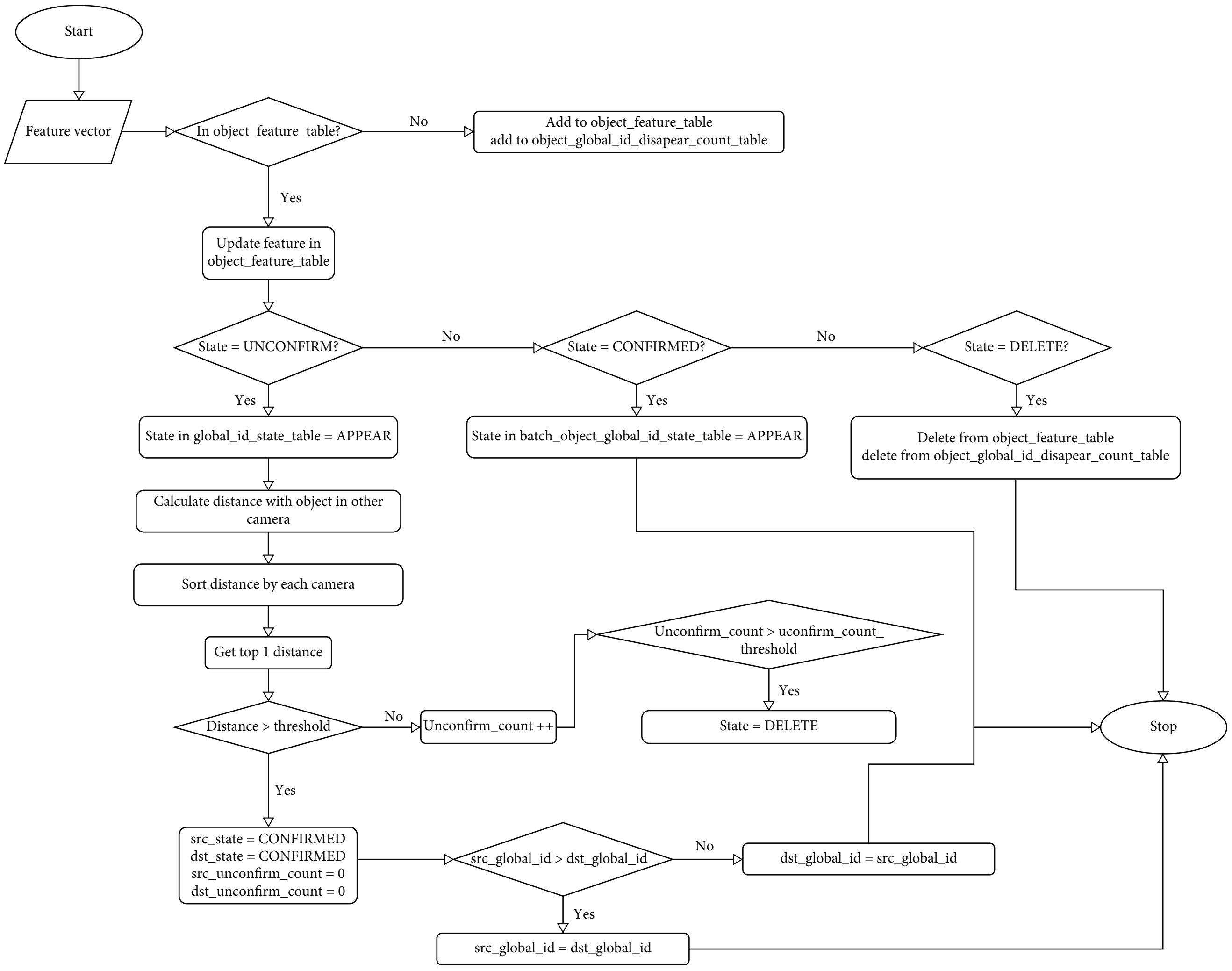
Start
Feature vector
In object_feature_table?
No
Add to object_feature_table
add to object_global_id_disapear_count_table
Yes
Update feature in object_feature_table
State = UNCONFIRM?
No
State = CONFIRMED?
No
State = DELETE?
Yes
State in global_id_state_table = APPEAR
Yes
State in batch_object_global_id_state_table = APPEAR
Yes
Delete from object_feature_table
delete from object_global_id_disapear_count_table
Calculate distance with object in other camera
Sort distance by each camera
Get top 1 distance
Unconfirm_count > uconfirm_count_ threshold
Yes
State = DELETE
Distance > threshold
No
Unconfirm_count ++
Stop
Yes
src_state = CONFIRMED
dst_state = CONFIRMED
src_unconfirm_count = 0
dst_unconfirm_count = 0
src_global_id > dst_global_id
No
dst_global_id = src_global_id
Yes
src_global_id = dst_global_id

Figure 3: Flowchart of matching and managing the track algorithm.

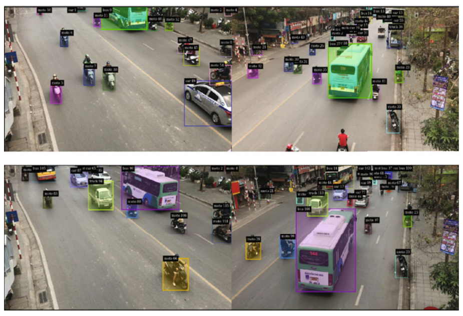

Figure 4: Result of the system with two video streams of urban traffic in Vietnam.

- *nvvideo-render*: used to display processed video streams to the screen. This element provides the ability to display video streams with high resolution and fast frame rate, making it easy for users to monitor and view processed video streams.

3.3. Media Detection Module Analysis. In this section, the objective is detected and classified for vehicles. We use the YOLOv5 model to detect vehicles. The output of this module will be the parameters of the location, type, and predicted probability of each vehicle.

We have built a dataset of urban means of transport that is suitable for the characteristics of Vietnam, where the traffic density is very high and does not follow a certain direction. The dataset is collected on urban roads with a top-down view. The device used in the data collection process is an iPhone 7 with a 12MP camera. The image of the dataset is extracted from the video recorded with full HD resolution as 1920×1080 pixels at a frame rate of 30 fps as shown in Figure 6.

We conduct data preprocessing according to the following steps after collecting data. We split the video into frames and then take one image for 10 fps. We then label the collected images with the LabelImg tool [21].

Figure 7 shows a high traffic density, which is typical of urban traffic in Vietnam. Vehicles stop at red lights and overlap each other.

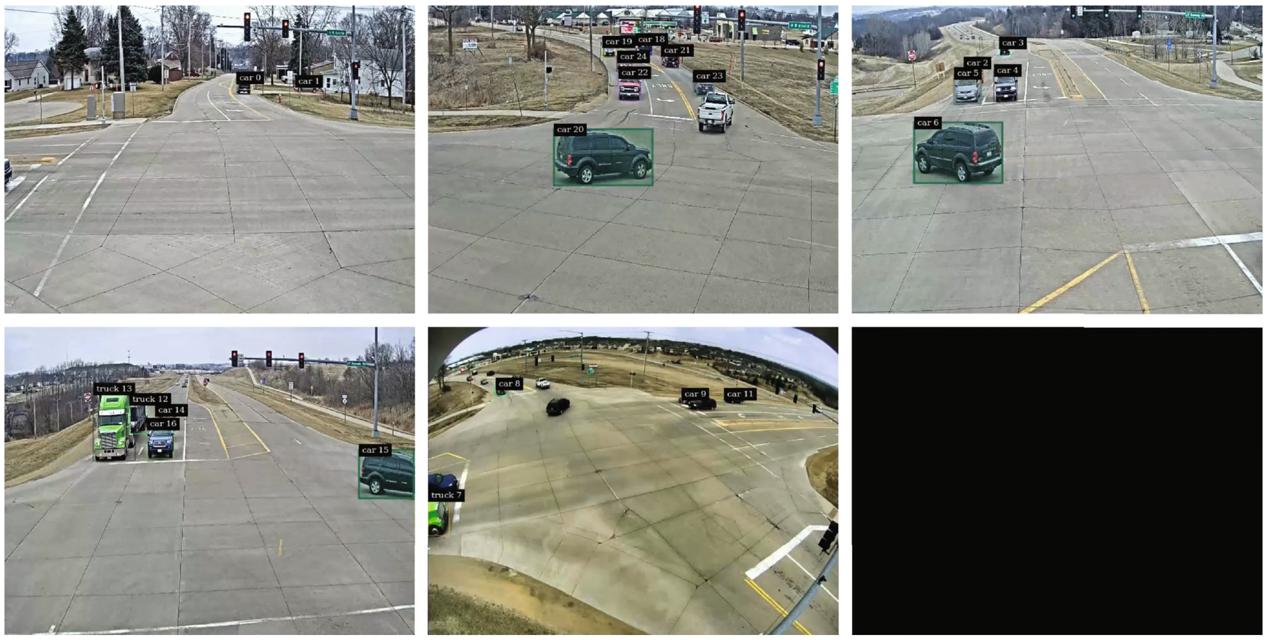

Figure 5: Result of the system with five video streams with the AI City Challenge dataset.

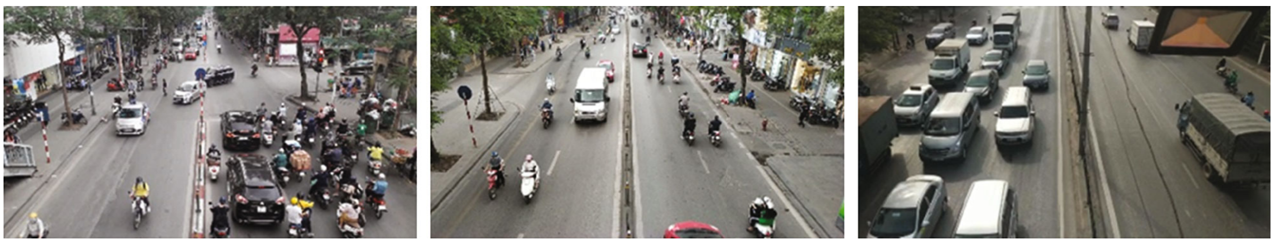
Figure 6: Illustrating several collected data samples.

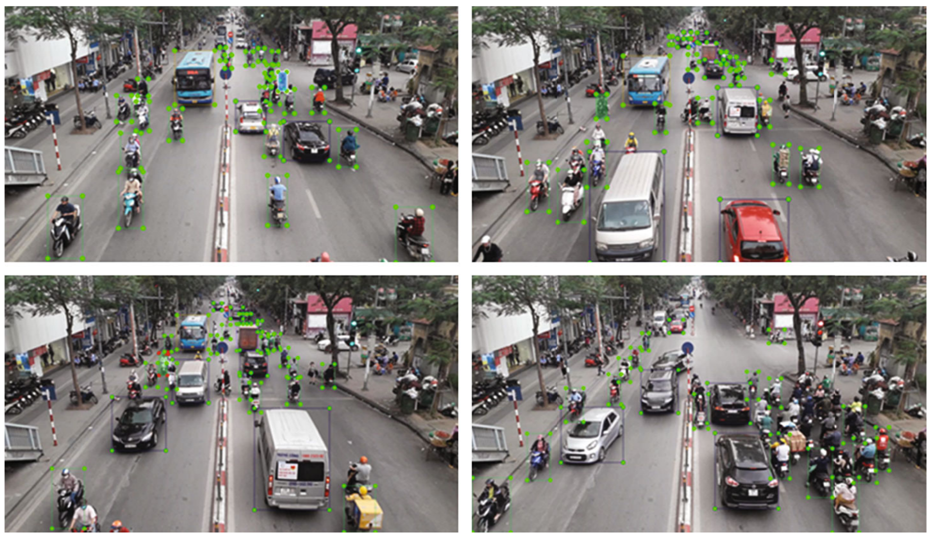
Figure 7: Data after being labeled.

When we have the labeled data, we proceed to divide the training data by the ratio (83:17). Eighty-three percent of images of the dataset are used for training, and the remaining 17% are used as a validation dataset. The number of images is shown in Table 1.

Details of the dataset and the several results applied to the object detection problem have also been presented in our paper [22].

During the evaluation of the trained vehicle detection model, we constructed a dedicated test dataset. This dataset comprises 3000 images captured from two distinct traffic routes in Hanoi, using the same equipment but on different roadways compared to the training data. The purpose of this test dataset is to provide an objective assessment of the accuracy of the trained model.

3.4. Solution Analysis and Design. We deploy the algorithm diagram for the object detection module after preparing the data as shown in Figure 8.

In Figure 8, the media detection algorithm flowchart has the following main steps:

- *Resizing image*: The size of the input is converted to a model-appropriate size. This has the effect of reducing the resolution of the input image making the calculation faster and increasing the performance of the model. In addition, resizing the image also helps to ensure that the objects are relatively the same size and not too large, thereby minimizing inaccurate prediction errors. However, if the image is resized too much, then small details can be lost leading to information loss and affecting the predictive ability of the model. Therefore, image resizing needs to be done carefully to ensure that its information is not lost too much and the accuracy of the model is preserved.
- *Processing*: At this step, the model computes and infers on the input. The input image will be predicted by the model of the vehicles and return information such as coordinates, size, vehicle type, and reliability.
- *Removing redundancy*: After having the prediction information of the model, we need to process to remove incorrect and duplicate predictions to get reliable objects. These objects will be returned as bounding boxes and corresponding labels.
- *Assigning metadata*: After obtaining the coordinates and dimensions of the bounding box, the type, and the reliability of the vehicles, proceed to assign the data of each vehicle to the metadata of the frame. This makes it easy to retrieve vehicle data for the following system modules.

3.5. Module Analysis by Vehicle. In this section, we use the DeepSORT algorithm to track traffic on each camera. The goal is to help the system track each vehicle on each camera. Its output is the track list of the means of transport.

Table 1: Data of training and validation for the model.

Dataset	Number of images
Train dataset	36,429
Valid dataset	7285
Total	43,715

After having information about each vehicle in each camera, we proceed to build a solution for the object tracking module.

In Figure 9, the main steps of the object tracking module on each camera are presented, including three steps as follows:

- *Getting information from metadata*: To be able to perform object tracking on each camera, the first thing is to get information about the object. This step will take the vehicle information such as coordinates and bounding box size, vehicle type, and confidence level to perform the next step of the module. This information is stored from the previous module of the system in metadata. We just need to access the frame metadata and retrieve each object metadata of the system.
- *DeepSORT*: In this step, after having enough information about the object, the DeepSORT algorithm will take steps to determine the ID object on each camera (local ID). More detail of DeepSORT has been presented in [23–27]. Each vehicle will have an ID on one camera after this step.
- *Assigning metadata*: After having the ID of each vehicle on each camera, this step will assign information about the ID object to the object_id field of each metadata object in the metadata frame.

3.6. Analysis of Feature Extraction Module and Re-ID

3.6.1. Data Collection and Preprocessing. In this module, we use the ResNet model to extract the features of the vehicles. Therefore, it is necessary to prepare a Re-ID dataset for vehicles. We decided to use the VeRi-776 dataset [28] to train the ResNet model for the vehicle Re-ID problem.

The VeRi-776 dataset was collected in urban traffic areas, with different vehicle rotation angles as shown in Figure 10. The dataset includes the following:

- More than 50,000 images of 776 vehicles were taken by 20 cameras covering an area of 1.0 km^2 over 24 h, which makes the dataset possible for the generalizability of vehicle Re-ID.
- The images are captured in a real-world unconstrained surveillance scene and labeled with different attributes, e.g., Bboxes, type, color, and brand.
- Each vehicle is captured by two to eight cameras with different angles, lights, and resolutions, bringing reality to the Re-ID vehicle problem.

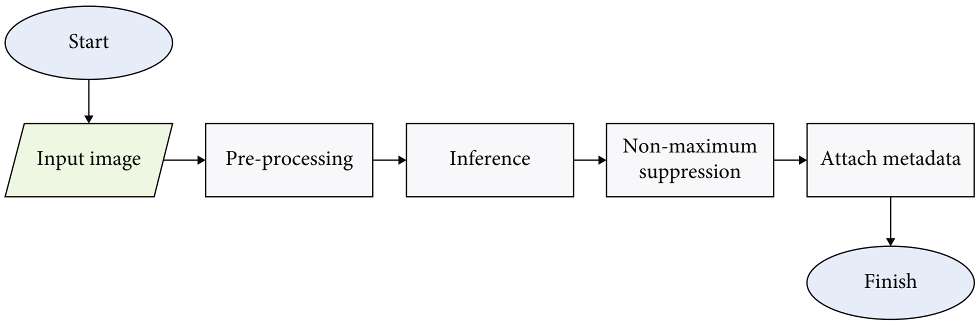

Figure 8: Flowchart of the vehicle detection algorithm.

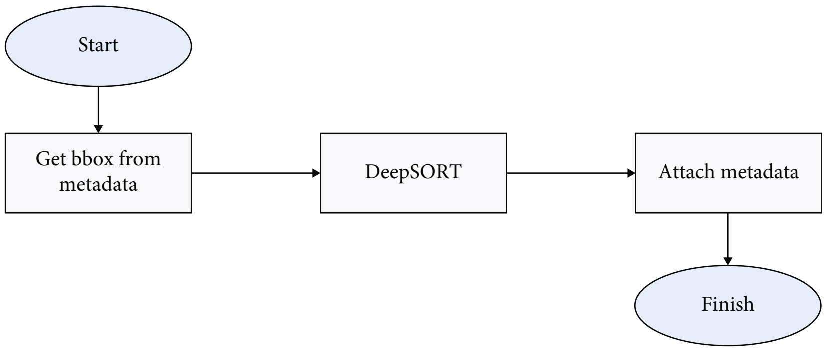

Figure 9: Flowchart of the object tracking algorithm on each camera.

Figure 10: Illustration of the images of the VeRi dataset [29].

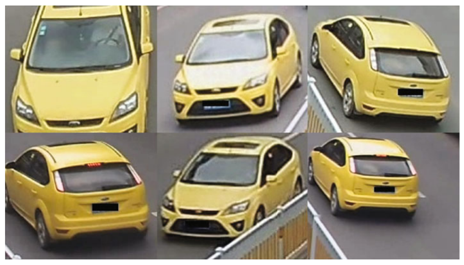

Figure 11: Images of a vehicle captured by six cameras in VeRi-776 [29].

Vehicle Re-ID is the problem of identifying and refinding a car in a dataset. Images are collected from many different cameras as shown in Figure 11. In the Re-ID problem, the data is divided into three datasets, namely, train, query, and gallery.

- *Train dataset*: It contains car images labeled with their unique identifiers. This volume includes images containing information about the vehicle taken from different cameras. The goal is to train the recognition model.
- *Query dataset*: The query dataset contains the car images used for finding the images in the gallery that has a similar car. This dataset is intended to be used to evaluate the model during training.
- *Gallery dataset*: It contains car images for the model to search. This dataset is intended to be used to evaluate the model during training.

3.6.2. Solution Analysis and Design. In Figure 12, the algorithm for the feature extraction module and Re-ID includes the following steps:

- *Getting information from metadata*: Similar to the previous module, to be able to perform the next steps, it is necessary to access metadata and retrieve the information of each object to be able to get enough information.
- *Feature extraction*: In this step, the means will be cut from the image after having information about the coordinates and sizes of the means in the image. It then will go through the ResNet backbone that has been trained previously for the Re-ID problem. Finally, the output will be a feature vector after calculating and inferring through ResNet.
- *Global track management*: After having obtained the feature vectors of each vehicle, this step will calculate

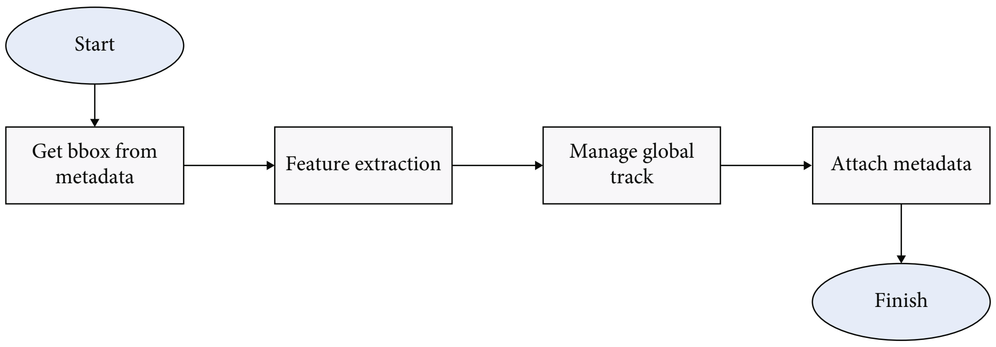

FIGURE 12: Flowchart of feature extraction and Re-ID modules.

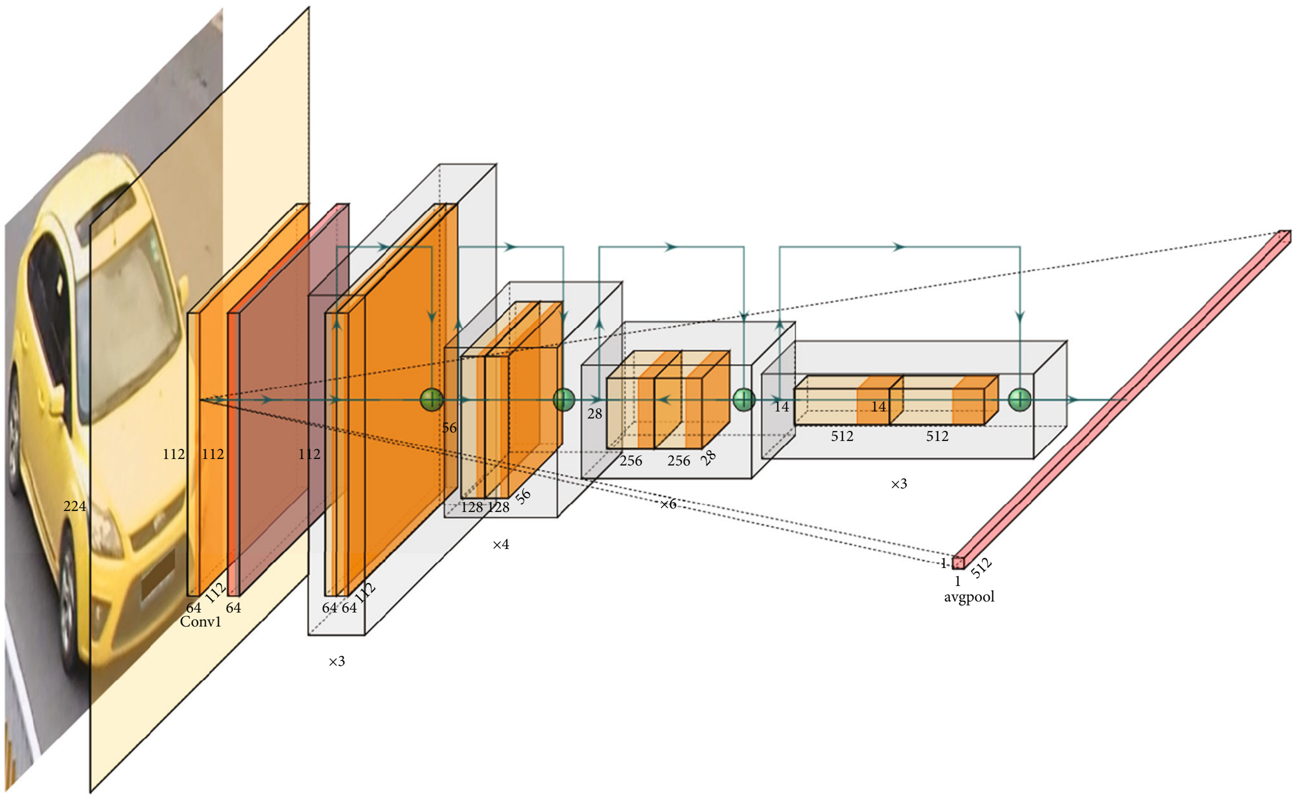

FIGURE 13: Using the ResNet34 model to extract vehicle features.

the similarity comparison between the feature vectors of the vehicles. It then will manage and store the information of the tracks such as the ID of the camera, object, and global into tables in metadata. This step output will be the global ID of each vehicle. This will be the ID object on all cameras.

- *Assigning metadata*: In this step, it is necessary to attach the global ID of each object to the metadata because the blocks behind can retrieve and plot it on the final output image of the system.

3.6.3. Designing Feature Extraction Module. One of the most used models in feature extraction and Re-ID problems is ResNet. ResNet can learn image features through the construction of residual classes, making it easier to train the model.

The feature extraction process is performed by the ResNet model trained for the Re-ID problem on the VeRi-776 dataset for vehicles as shown in Figure 13. After obtaining the features of the vehicle, we compare the similarity between the extracted feature vectors to search for vehicles from different cameras. In this feature extraction block, we decided to use the CNN backbone which is ResNet to extract the features of the vehicles into vectors of size 1×512. We then train the model on the VeRi-776 dataset. In the training process, we use the loss function centroid triplet loss (CTL) [30] to calculate the error

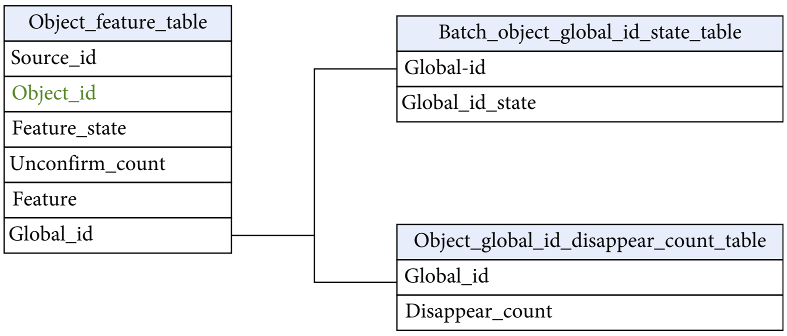

Figure 14: Tables for managing track information and features.

in the learning process. We train the Re-ID model with vehicles to solve the problem. When taking out the ResNet backbone in the Re-ID model for feature extraction, the image of the same vehicle is taken from many different angles and the feature vectors have the smallest similarity distance.

3.6.4. Designing Global Track Management Block. In this block, the extracted features will be processed and managed in the tables in the metadata of the system pipeline. The tables that store the characteristics and IDs of each vehicle are shown in Figure 14.

In Figure 14, three tables are used to manage the tracks and characteristics of the vehicles during the Re-ID for the vehicle. The tasks of the tables shown in Figure 14 are as follows:

- *object_feature_table*: It stores the vehicle's feature vectors and tracks during distance comparison. This table will contain information as **source_id** (ID of the camera containing the media image), **object_id** (local ID of the object on each camera), **feature_state** (status of media in the table: CONFIRMED: matched one vehicle in another camera, UNCONFIRM: newly added to the table, and DELETE: media needs to be removed from the table), **unconfirm_count** (frame number consecutive object in an UNCONFIRM state), **feature** (object feature vector), and **global_id** (global ID of the vehicle on all cameras).
- *batch_object_global_id_state_table*: It stores the appearance information of vehicles in a batch based on their global_id. At the end of each pipeline loop, this table will be dropped to store the information in the next new batch. This table will have two fields of information: **global_id** (global ID of the vehicle on all cameras) and **global_id_state** (the status of one vehicle appearing on all cameras based on global_id—APPEAR: one or more tracks with the same global_id appear in the batch and DISAPPEAR: in the whole batch, no track with the global_id appears).
- *object_global_id_disapear_count_table*: It plays the role of storing consecutive frame number information, and the track disappears from all cameras. This table will contain information such as **global_id** (global ID of the vehicle on the whole camera) and **disappear_count** (the number of consecutive frames global_id disappeared from all cameras in the system).

In Figure 3, a flowchart of the algorithm compares the feature vectors and manages the tracks in the tables. In this process, the similarity cosine distance is used to determine the distance between two feature vectors of two means from two different sources. The formula to calculate the similarity cosine distance of two vectors **A** and **B** is

$$\cos(\theta) = \frac{A \times B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

After getting the distance of one vehicle under consideration (src_object in src_camera) to vehicles in other cameras (dst_object in dst_camera), sort the distance by each dst_camera by value from high to low. Then, select the largest distance (Top 1) in each dst_camera and compare this distance with the distance threshold value (distance_threshold). If the distance threshold is greater than the threshold value, the two tracks match and will have the same global_id of the track with the smaller global_id. Otherwise, for two tracks that do not match, increase the unconfirm_count counter by one value. If the unconfirm_count value is greater than the unconfirm_count_threshold threshold value, delete the track from both tables.

4. Simulation and Result

4.1. Evaluation Parameters

4.1.1. Mean Average Precision (MAP). MAP is a measure of model quality in the object detection problem. It is used to measure the accuracy of the objects detected by the model and calculate the average of this accuracy over the entire dataset.

MAP is calculated based on the precision and recall values of the model on each feature class. Precision measures the ratio of the number of correctly detected objects (TP) to the total number of detected objects, while recall measures

Table 2: Model training hardware configuration.

CPU	GPU	RAM
Intel Xeon Silver 4216 @2.1 GHZ 16 cores 32 threads	NVIDIA Quadro RTX 5000 VRAM 16 GB	125 GB

Table 3: Laptop hardware configuration and embedded computer for performing the system.

Device	CPU	GPU	RAM
Lenovo Legion 5 15ARH05H	AMD Ryzen™ 7-4800H (2.90–4.20 GHz, 8 MB)	NVIDIA GeForce GTX 1660 Ti 6 GB GDDR6	16 GB DDR4
NVIDIA® Jetson AGX Xavier™	8-core NVIDIA Carmel Armv8.2 64-bit CPU 8 MB L2+4 MB L3	512-core Volta GPU with Tensor Cores	16 GB 256-bit LPDDR4x 137 GB/s

the ratio of the number of correctly detected objects to the total number of detected objects:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

To calculate MAP, we must first calculate the precision and recall values for each feature class and the area under the precision–recall curve for each layer as

$$\text{MAP} = \sum_n (R_n - R_{n-1}) \times P_n \quad (4)$$

where R_n and P_n are the recall and precision values at the point of order n on the *PR* curve, respectively. The first point on the *PR* curve is (0,1), and the last point is (1,0).

Finally, we calculate the average of overall feature classes to get the mapped value as

$$\text{MAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \quad (5)$$

A good object detail model will have a high MAP value. It is capable of detecting the right objects that do not cause too many errors.

4.1.2. Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA) mbox. MOTP and MOTA are two common parameters used to evaluate the performance of multiobject monitoring models.

MOTP is used to assess the accuracy of the location predictions of the objects monitored compared to their actual locations. The MOTP formula is calculated by calculating the average Euclidean distance between the location predictions of the monitored objects and their actual locations. The formula is shown as follows:

$$\text{MOTP} = \frac{1}{N}\sum_{i=1}^{N} T_i \times \sum_{j=1}^{T_i} d_{ij} \quad (6)$$

where N is the number of objects monitored during the evaluation process, T_i is the number of frames during the monitoring process of the object i, and d_{ij} is the Euclidean distance between the prediction and the actual location of the object i in the j frame.

Table 4: Model training results of detected vehicles.

Model	MAP@0.5	Precision	Recall
YOLOv5s	92.56%	94.21%	80.34%
YOLOv5n	91.84%	92.37%	76.64%

We see that the higher the MOTP value, the better the quality of the object monitoring system.

MOTA is used to assess the total number of wrong predictions of the system compared to the actual objects. It is calculated by subtracting the percentage of the wrong number of predictions (FP) and the number of subjects lost monitoring from the total number of actual objects. The formula is shown as follows:

$$\text{MOTA} = 1 - \frac{\sum_{i=1}^{N}(\text{FP}_i + \text{Missed}_i)}{\sum_{i=1}^{N}\text{GT}_i} \quad (7)$$

where N is the number of subjects monitored during the evaluation process, FP_i is the wrong number of objects i, Missed_i is the number of subjects i lost monitoring, and G T_i is the actual number of objects. The closer the MOTA value to 1, the better the quality of the object monitoring system.

4.1.3. Rank-1 mbox. Rank-1 is the index of the accuracy model, showing the number of times the most accurate image in the dataset is given compared to the total number of tested cases.

Rank-1 is calculated as follows:

$$\text{Rank-1} = \frac{\text{TruePredict}}{\text{TruePredict} + \text{FalsePredict}} \quad (8)$$

where TruePredict is the number of times that the only correct image prediction of the model is made and FalsePredict is the number of times that the wrong prediction is made.

For example, if a Re-ID model is tested on 1000 images and the model only gives the most accurate results

Table 5: Simulation results for vehicle detection and tracking on different hardware.

Hardware	Model	One stream (FPS)	Two streams (FPS)	Three streams (FPS)	Four streams (FPS)
Laptop	YOLOv5s	130.85	135.46	94.72	83.52
	YOLOv5n	143.51	128.34	113.28	90.23
Jetson Agx Xavier	YOLOv5s	183.25	163.31	145.27	128.43
	YOLOv5n	206.32	180.13	159.32	145.22
Laptop	YOLOv5s + DeepSORT	125.42	104.53	89.25	74.33
	YOLOv5n + DeepSORT	136.23	115.41	98.42	82.64
Jetson Agx Xavier	YOLOv5s + DeepSORT	152.34	123.74	104.20	91.43
	YOLOv5n + DeepSORT	186.20	151.84	129.53	116.03

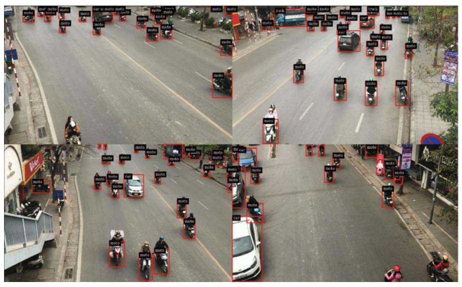

Figure 15: Results of running modules for detecting vehicles for four video streams.

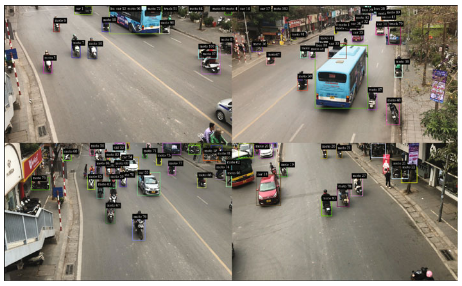

Figure 16: Results of running modules for tracking vehicles for four video streams.

in 800 cases, then rank-1 of this model will be rank-1 = 800/1000 = 0.8.

4.2. *Training Hardware Configuration.*

Table 2 is the configuration of the model training server used in the paper.

Table 3 is the configuration of two devices used to perform the system and evaluate its speed.

4.3. *Results of Simulation of Vehicle Detection*

4.3.1. *Training Result.*

We conduct the model training with different versions of YOLOv5 on hardware presented in Table 3 after collecting data and processing them. Two models YOLOv5s and YOLOv5n are trained in 300 epochs with batch size = 64 and image size = 640 that have the following training results as shown in Table 4.

Table 4 shows that the YOLOv5s model learns better and gives higher accuracy and parameters than YOLOv5n.

4.3.2. *Simulation Result.*

Performing simulation of both training models on a laptop and embedded hardware as configured in Table 3, we obtain the following results in Table 5 and Figure 15.

4.4. *Result of Vehicle Monitoring Modules.*

Simulation of both trained models combined with the DeepSORT algorithm on both test hardware obtained the following results as shown in Tables 3 and 5.

In Figure 16, each vehicle will have its local ID (ID of each vehicle on each camera). We have built the **bboxcolor** element in the system pipeline because it can randomly generate a different color for a local ID and will manage and delete it when the media is out of the camera frame.

Table 6: Results of training the Re-ID model on the VeRi-776 dataset.

ResNet34	Accuracy (%)
Rank-1	91.8
Rank-5	96.0
Rank-10	97.8
Rank-20	99.0
Rank-50	99.6

Table 7: Results of the speed of the system on embedded hardware.

Number of cameras	YOLOv5s (FPS)	YOLOv5n (FPS)
2	18.74	20.59
3	15.14	16.30
4	13.52	14.87
5	12.44	13.08

4.5. *Result of Characteristic Extract and Re-ID.*

Results of training the Re-ID model with the VeRi-776 dataset in 120 epochs on the hardware are presented in Table 6.

In Table 6, we can see that the trained Re-ID model has a percentage of the number of cases when the model gives the most accurate result (correct vehicle identification) for the searched vehicle as 91.8%.

4.6. Evaluating System Speed. Finally, we synthesize and simulate on embedded hardware devices after training and preparing the modules of the system. The results are obtained with two models as shown in Table 7.

In Table 7, it is shown that the system speed decreases significantly after integrating more feature extraction and Re-ID modules. This happens because the feature extraction block has to take as input an image of each vehicle and infers and computes the feature vector for each of them. When we perform on the entire system, there will be a lot of media that need to be handled through this block leading to a decrease in system speed.

In Figures 4 and 5, the vehicles with the rotating pulse bounding box are the vehicles that already have a local ID (the ID of each vehicle on each camera). However, we have not yet matched any of the vehicles in the other cameras. Vehicles colored inside the same bounding box are those detected by the system as matching and being one. At this point, the matched vehicles will have the same color filled inside the bounding box and will be managed until the vehicles disappear.

5. Conclusion

In this paper, we proposed a smart system for detecting and monitoring vehicles across multiple surveillance cameras using deep learning and hardware optimization. Our system consists of three main modules: vehicle detection, vehicle tracking, and vehicle Re-ID. Our system achieves high accuracy and efficiency in vehicle detection and tracking, as well as in Re-ID across different cameras. Our system also leverages the NVIDIA tools to ensure optimal use of hardware resources and real-time processing. We evaluated our system on a dataset of five video streams captured from different locations and times in Ho Chi Minh City. The results show that our system can achieve approximately 21 fps on embedded hardware and an accuracy of 93.56% for vehicle detection, 70.63% for vehicle tracking, and 91.8% for vehicle Re-ID.

However, our system still has some limitations that need to be addressed in future work. For example, our dataset is not diverse enough to handle various weather conditions, such as rain, storm, fog, or flood. Our algorithm for matching and managing tracks is also not robust enough to avoid wrong Re-ID of vehicles. Therefore, we plan to improve our system by

- combining more feature pools to store tracks to serve camera systems that do not overlap with each other
- optimizing the system to be able to run with more cameras
- adding some rules that force track management to be tighter
- integrating more models to solve problems such as license plate recognition, building the interface system for the application, and building more diverse datasets to be suitable for different weather contexts, etc.
- integrating rules to detect vehicles with traffic violation

We hope that our system can contribute to the field of urban traffic management and provide practical benefits for various applications.

Data Availability Statement

The system is developed based on our model that is presented in [22]. In this paper, we have developed three main points. Firstly, we use the dataset of means of transport collected by us in urban traffic routes in Vietnam. Secondly, we use a deep learning model to extract features of each vehicle and Re-ID on multiple cameras in the system. Finally, we run an entire system written in C++ on NVIDIA hardware for high speed and the ability to run multiple live video streams. All results are updated according to this data. In the paper, we use the VeRi dataset in [29]. We also confirm that all data are referred to in the paper.

Conflicts of Interest

The authors declare no conflicts of interest.

Funding

This research is funded by the Hanoi University of Science and Technology (HUST) under project number T2022-PC-013.

Acknowledgments

This research is funded by the Hanoi University of Science and Technology (HUST) under project number T2022-PC-013. The authors would like to thank HUST for the financial support.

References

[1] M. Ha, "The first city in Vietnam to develop electric transportation," 2022, https://thanhnien.vn/thanh-pho-dau-tien-o-viet-nam-phat-trien-giao-thong-dien-1851422504.htm.

[2] F. Zhang, C. Li, and F. Yang, "Vehicle detection in urban traffic surveillance images based on convolutional neural networks with feature concatenation," *Sensors*, vol. 19, no. 3, p. 594, 2019.

[3] H. Nguyen, "An efficient license plate detection approach using lightweight deep convolutional neural networks," *Advances in Multimedia*, vol. 2022, Article ID 8852142, 10 pages, 2022.

[4] F. Li, Z. Wang, D. Nie et al., "Multi-camera vehicle tracking system for ai city challenge 2022," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3264–3272, New Orleans, LA, USA, June 2022.

[5] P. N. Huu, P. T. T. Quynh, T. P. Ngoc, and Q. T. Minh, "Proposing a route recommendation algorithm for vehicles based on receiving video," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 4, p. 1487, 2022.

[6] I. Perera, S. Senavirathna, A. Jayarathne et al., "Vehicle tracking based on an improved DeepSORT algorithm and the YOLOv4 framework," pp. 305–309, 2021.

[7] M. A. Tahiri, H. Amakdouf, M. el mallahi, and H. Qjidaa, "Optimized quaternion radial Hahn moments application

to deep learning for the classification of diabetic retinopathy," *Multimedia Tools and Applications*, vol. 82, no. 30, pp. 46217–46240, 2023.

[8] S. K. Rajput, J. C. Patni, S. S. Alshamrani et al., "Automatic vehicle identification and classification model using the YOLOv3 algorithm for a toll management system," *Sustainability*, vol. 14, no. 15, p. 9163, 2022.

[9] H. Song, H. Liang, H. Li, Z. Dai, and X. Yun, "Vision-based vehicle detection and counting system using deep learning in highway scenes," *European Transport Research Review*, vol. 11, no. 1, p. 51, 2019.

[10] H.-M. Hsu, J. Cai, Y. Wang, J.-N. Hwang, and K.-J. Kim, "Multitarget multi-camera tracking of vehicles using metadata-aided Re-ID and trajectory-based camera link model," *IEEE Transactions on Image Processing*, vol. 30, pp. 5198–5210, 2021.

[11] Z. Hao, J. Qiu, H. Zhang, G. Ren, and C. Liu, "UMOTMA: underwater multiple object tracking with memory aggregation," *Frontiers in Marine Science*, vol. 9, article 1071618, 2022.

[12] H. Liu, X. Ma, Y. Yu, L. Wang, and L. Hao, "Application of deep learning-based object detection techniques in fish aquaculture: a review," *Journal of Marine Science and Engineering*, vol. 11, no. 4, p. 867, 2023.

[13] P. N. Huu and H. N. Duc, "Propose an automatic ammonia concentration of water measuring system combining image processing for aquaculture," *Journal Européen des Systèmes Automatisés*, vol. 54, no. 3, pp. 453–460, 2021.

[14] M. Saberioon, A. Gholizadeh, P. Cisar, A. Pautsina, and J. Urban, "Application of machine vision systems in aquaculture with emphasis on fish: state-of-the-art and key issues," *Reviews in Aquaculture*, vol. 9, no. 4, pp. 369–387, 2017.

[15] P. N. Huu, Q. T. Minh, and Q. T. Minh, "Designing water environment monitoring equipment for aquaculture in Vietnam," in *Artificial Intelligence in Data and Big Data Processing*, pp. 579–590, Springer International Publishing, Cham, 2022.

[16] P. N. Huu, Q. T. Minh, D. D. Dang et al., "Monitoring and forecasting water environment parameters for smart aquaculture using LSTM," in *2022 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 53–58, Ho Chi Minh City, Vietnam, December 2022.

[17] M. A. Tahiri, H. Karmouni, M. Sayyouri, and H. Qjidaa, "Correction to: 2D and 3D image localization, compression and reconstruction using new hybrid moments," *Multidimensional Systems and Signal Processing*, vol. 33, no. 3, p. 1069, 2022.

[18] M. A. Tahiri, H. Karmouni, A. Bencherqui et al., "New color image encryption using hybrid optimization algorithm and Krawtchouk fractional transformations," *The Visual Computer*, vol. 39, no. 12, pp. 6395–6420, 2023.

[19] M. A. Tahiri, F. Z. El Hlouli, A. Bencherqui et al., "White blood cell automatic classification using deep learning and optimized quaternion hybrid moments," *Biomedical Signal Processing and Control*, vol. 86, article 105128, 2023.

[20] L. H. Pham, T. T. Duong, H. M. Tran, and S. V.-U. Ha, "Vision-based approach for urban vehicle detection & classification," in *2013 Third World Congress on Information and Communication Technologies (WICT 2013)*, pp. 305–310, Hanoi, Vietnam, December 2013.

[21] heartexlabs, "heartexlabs/labelimg," 2017, https://github.com/heartexlabs/labelImg.

[22] P. N. Huu, B. N. Anh, V. T. N. Nam, T. M. Hoang, T. D. Nguyen, and Q. T. Minh, "Tracking and calculating speed of mixing vehicles using YOLOv4 and DeepSORT," in *2022 9th NAFOSTED Conference on Information and Computer Science (NICS)*, pp. 105–110, Ho Chi Minh City, Vietnam, October 2022.

[23] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, Phoenix, AZ, USA, September 2016.

[24] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME-Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[25] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[26] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, Beijing, China, September 2017.

[27] R. Pereira, G. Carvalho, L. Garrote, and U. J. Nunes, "Sort and Deep-SORT based multi-object tracking for mobile robotics: evaluation with new data association metrics," *Applied Sciences*, vol. 12, no. 3, p. 1319, 2022.

[28] Z. Zheng, T. Ruan, Y. Wei, Y. Yang, and T. Mei, "VehicleNet: learning robust visual representation for vehicle re-identification," *IEEE Transactions on Multimedia*, vol. 23, pp. 2683–2693, 2021.

[29] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "TPH-YOLOv5: improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 2778–2788, Montreal, BC, Canada, October 2021.

[30] M. Wieczorek, B. Rychalska, and J. Dbrowski, "On the unreasonable effectiveness of centroids in image retrieval," in *Neural Information Processing: 28th International Conference, ICONIP 2021, Proceedings, Part IV 28*, pp. 212–223, Springer, Sanur, Bali, Indonesia, 2021.