Journal of Computational Design and Engineering, 2024, **11**, 106–121

DOI: 10.1093/jcde/qwae016
Advance access publication date: 19 February 2024
Research Article

The identification of minor impact collisions in a long video for detecting property damages caused by fleeing vehicles using three-dimensional convolutional neural network

Inwoo Hwang[1] and Yong-Gu Lee[2,3,*]

[1]Vehicle Component Solutions, LG Electronics, 10, Magokjungang 10-ro, Gangseo-gu, Seoul 07996, Republic of Korea
[2]School of Mechanical Engineering, Gwangju Institute of Science and Technology (GIST), 123 Cheomdangwagi-ro, Buk-gu, Gwangju 61005, Republic of Korea
[3]Artificial Intelligence Graduate School, Gwangju Institute of Science and Technology (GIST), 123 Cheomdangwagi-ro, Buk-gu, Gwangju 61005, Republic of Korea
*Correspondence: lygu@gist.ac.kr

Abstract

A parked vehicle damaged by a hit-and-run can only be repaired at the expense of the owner, unless the fleeing vehicle is identified and the driver apprehended. Identifying the fleeing vehicle involves using a video investigation method that searches for perpetrators through CCTV footage of the crime scene. When the length of the recorded video is long, the investigation may require an extended amount of time from the investigator, resulting in an added burden on their daily work. Some commercial companies are using object recognition and tracking technology to detect hit-and-run incidents; however, detecting small movements of a vehicle during a minor collision still remains a challenge. Therefore, there is a need for a system that can detect small movement in a vehicle in a lengthy video. Automatic recognition and tracking require a sufficient amount of training dataset. However, such a dataset for hit-and-run incidents is not publicly available. One of the reasons behind this scarcity is that it may violate personal information protection acts. On the other hand, instead of using real accident videos, we could use actors to simulate such accident scenes. Although this may be feasible, creating such a dataset would require substantial costs. In this paper, we describe a new dataset for hit-and-run incidents. We collected 833 hit-and-run videos by recreating a parking lot using miniaturized cars. This dataset has been made publicly available through Kaggle. We used three-dimensional convolution neural network, which is frequently used in the field of action recognition, to detect small movements of vehicles during hit-and-run incidents. In addition, the proportion of the area that surrounds the target vehicle to the min-max box of the vehicle itself and the length of the input frame are varied to compare the accuracy. As a result, we were able to achieve better accuracy by using the lowest proportion and the shortest input frame.

Keywords: video action recognition, 3D-CNN, video surveillance

1. Introduction

Hit-and-run is a crime that is not specifically defined in road traffic laws. It occurs when a driver causes an accident or damages a parked vehicle and then flees the scene without providing assistance or exchanging information. Hit-and-run cases are difficult to prosecute and even when they are, the penalties are often light. As a result, many hit-and-run drivers are never caught.

To solve a hit-and-run case, investigators require information about the damaged area, the time of the accident, and the offender vehicle. The damaged area can be determined through photographs or video footage, while the time of the accident can be obtained from the vehicle's black box. In cases where the black box does not record the hit-and-run, investigators will have to rely on video surveillance footage from the surrounding area to search for the perpetrator. However, this method can be time-consuming as it necessitates watching hours of video footage. To reduce the time it takes to investigate a hit-and-run case, there are several methods that investigators can use. One of them is to designate a region of interest (ROI) and track any objects that enter it. Investigators can then record all of the footage from the ROI and review it to confirm the crash. This method is provided by intelligent CCTV solution companies and is primarily used for security purposes. However, it is not very effective at reducing investigation time, as it can generate a lot of unnecessary footage of vehicles that simply pass by the damaged vehicle.

To solve this problem, we need a function that can detect small shakings when a reference vehicle collides with another vehicle. This function will enable classifiers to accurately determine whether the vehicles are colliding. It is important to improve the function's accuracy and minimize false alarms that may occur when other vehicles simply approach the reference vehicle without colliding, when the reference vehicle is merely occluded, or when only a portion of it is visible. To avoid such false alarms, we should only pick when the reference vehicle and the other vehicle have truly collided. To do so, the classifier must accurately identify the moment of impact when the vehicles collide and a slight shake occurs.

Received: September 19, 2023. **Revised:** February 8, 2024. **Accepted:** February 12, 2024

2. Related Works

Our work relates to the general problem of spatial-temporal changes of objects where the input is a series of frames captured by a surveillance camera installed in a parking lot. Closely related subjects include anomaly detection, object tracking, and video action recognitions. We review relevant works in the following subsections.

2.1. Anomaly detection

The primary goal of CCTV video analysis is to detect abnormal situations such as crime scenes, natural disasters, and medical emergencies in recorded videos. However, this process requires significant manpower, as it involves analyzing long stretches of video footage. To address this issue, many studies have focused on automating CCTV monitoring to reduce the manpower required. To meet these social demands, the intelligent CCTV market has continued to expand in both public and private sectors, driven by changing social paradigms and increased safety demands. Rapid advancements in technology have led to the development of intelligent CCTV systems. These advancements include the integration of artificial intelligence and the Internet of Things. The intelligent CCTV systems use these technologies to enhance their functionality.

Anomaly detection is the process of distinguishing between normal and abnormal samples. It is applied in various fields, with efforts being made to adapt the approach to the unique characteristics of each domain. Traditional anomaly detection methods are trained using only normal samples, with any deviations from normal behavior classified as abnormal. For example, in video surveillance, anomaly detection might be used to detect bicycles, vehicles, or other unexpected objects in pedestrian areas or to identify traffic accidents. Video surveillance is a particularly important field for anomaly detection, as it enables the detection of strange behavior and other anomalies in surveillance videos.

If only the appearance is observed, the perpetrating vehicle that causes the hit-and-run generally shows a movement similar to the vehicle attempting to park. However, when the vehicle collides and slightly shakes, it differs from the typical vehicle movement. Therefore, to classify normal and abnormal cases, it is essential to extract relevant features accurately from the images. The current approach has been to utilize classical computer vision algorithms. Several methods have been proposed to recognize abnormal patterns by extracting hand-crafted features such as feature point detection and motion vector extraction. Common computer vision techniques, such as Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF) and Histogram of Oriented Gradients (HOG) have been utilized for this purpose (Bay *et al.*, 2006; Dalal & Triggs, 2005; Lowe, 2004). However, while these hand-crafted methods perform well in specific situations, they are less accurate in other environments.

In recent years, methods for automatically detecting abnormal situations have been proposed. These methods automatically extract and classify features within the deep learning networks such as convolution neural networks (CNNs), that gained enormous amount of popularity due to the rapid development of GPUs and artificial neural networks. This contrasts with the hand-crafted method, which manually extracts the features of the image and cannot be generalized. Autoencoders, which are often used in deep learning, have good performance in one-class classification (Ribeiro *et al.*, 2018). The autoencoder consists of an encoder and a decoder, which includes a CNN and a pooling layer. The autoencoder-based model can learn features of a normal region that are the main component of the data without specific labeling. The autoencoder model, which is learned only from normal data, reconstructs it with normal data even if abnormal values enter as input. Anomaly score is calculated using these properties and one-class classification is performed. Additionally, abnormalities are detected when sudden changes occur by measuring the amount of movement changes on the entire frames, using methods such as optical flow. Xu *et al.* (2015) used optical flow and autoencoder-based models containing motion information to learn normal data and used the trained model to detect abnormal data. Hasan *et al.* (2016) used HOG and Histogram of Optical Flow (HOF) as hand-crafted elements for motion characteristics and proposed an autoencoder based model. Biradar *et al.* (2019) extracted background images and performed anomaly detection by separating static and dynamic objects. Sultani *et al.* (2018) attempted to predict the time of the accident based on the anomaly threshold by using weak labeling and pretrained three-dimensional (3D)-CNN as backbone. Yao *et al.* (2019) predicted the expected trajectory of the vehicle using the bounding box information based on unsupervised learning to determine whether an accident occurred. Zhou *et al.* (2022) extracted spatial features using the object detection technique and extracted temporal features using HOF and a multilayer neural network to determine whether there was a traffic accident. Samani *et al.* (2022) detected abnormal situations in COVID-19 situations using a 2D detector. Kim *et al.* (2022) proposed a real-time monitoring warning system that can be applied in e-scooter sharing services.

2.2. Object detection and object tracking

Object recognition technology was initially developed for detecting people. If pedestrians can be detected accurately on camera, this can be used for front-end risk notification in the field of autonomous driving and security monitoring. Object recognition technology has previously used hand-crafted methods to detect objects. Viola and Jones (2001) proposed the Haar feature-based cascade classifier, which is a classic and the most popular algorithm. Dalal and Triggs (2005) proposed a HOG-based methodology to identify objects by extracting local gradient distribution characteristics of images. Zhu *et al.* (2006) improved the speed by applying the cascaded technique to HOG. They applied HOG by creating blocks of various sizes and positions. Felzenszwalb *et al.* (2010) proposed a part-based model method that can detect not only by the overall appearance of an object but also by using the part information of the object. This method can detect even when the shape of the object changes.

Recently, with the increase in computing power, the development of deep learning and the discovery of CNNs, deep learning has also begun to apply in the field of object recognition and is currently showing outstanding results. Initially, the localization process, region proposal, and classification were conducted separately. In the region proposal phase, inefficient methods such as sliding windows were initially used; however, computer vision technologies such as selective search were utilized, or localization problems were solved by selecting through region proposal networks, etc. (Girshick *et al.*, 2014; Ren *et al.*, 2017). When the candidates of the object are determined by the corresponding method, the object is classified through the classification model. The two-stage method has high accuracy; however, the processing speed is slower than the one-stage method to be introduced later. The two-stage method involves performing the region proposal step

and classification together (Dai *et al.*, 2016). The method of simultaneously performing region proposal and classification finds only objects of a predetermined location and size. These locations and sizes are selected to be used in most situations, such as anchor boxes. This method is referred to as the one-stage method (Liu *et al.*, 2016; Redmon *et al.*, 2016).

In the field of object tracking, a method has been proposed that tracks only a single object (Held *et al.*, 2016) such as utilizing CNNs for the entire image (Wang *et al.*, 2015) or Kalman filters using values from object recognition results have been proposed (Wojke *et al.*, 2017). Attention mechanisms have also been utilized (Lee *et al.*, 2023). It is believed that all these technologies have exceeded the level of human perception.

Intelligent CCTV companies use object recognition and tracking technology for crime prevention. The system allows users to designate areas of interest. When new objects enter these areas, an alarm is issued and the time is recorded. When this technology is applied to a hit-and-run, an ROI is designated around the affected vehicle. Subsequently, when other vehicles approach the area of interest, that incident with the trajectory of those vehicles is recorded. However, this methodology has the disadvantage of being inefficient in reducing the time used for the investigation. This is because it provides a lot of unnecessary information, such as the trajectories of vehicles that simply pass through the area of interest without colliding with the affected vehicle.

2.3. Action recognition

Action recognition in video covers various areas such as video summarization, surveillance systems, video retrieval, and video prediction. Video summarization aims to reduce the amount of data in a video. Surveillance systems are used for security purposes. Video retrieval helps in finding a specific content within a video. Video prediction forecasts future events in videos. Action recognition is a longstanding and important computer vision task. Researchers have studied action recognition for many years, as demonstrated by the works of Bao and Intille (2004) and Zhang and Tao (2012).

Action recognition is a process that aims to understand the movement of an object over time and identify it in an image. To achieve this, motion information over time is often used. Optical flow is a common method for obtaining motion information. The motion information is added or processed based on the RGB information of an image. This helps correct for camera shakes, which can affect the accuracy of the action recognition process (Poleg *et al.*, 2016).

There have been significant changes in action recognition before and after the advent of deep learning. Before deep learning, spatial and temporal information was obtained using various hand-crafted elements. Those include histograms of optical flows (Wang *et al.*, 2013), SURFs with improved SIFT performance and HOG (Laptev & Lindeberg, 2006). These were used to enhance the performance of action recognition.

After the advent of deep learning, attempts were made to solve the action recognition problem using CNN. The features learned using 2D-CNN and the temporal characteristics can be captured through various fusion methods with multiframe input (Karpathy *et al.*, 2014). Furthermore, temporal information can be modeled using 2D-CNN, Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) (Donahue *et al.*, 2015). Police signals have also been recognized using 2D-CNN and LSTM (Baek & Lee, 2022).

3D-CNNs have been developed to utilize temporal information. The structure of the 3D-CNN network uses methods validated on 2D-CNN (Ji *et al.*, 2013; Tran *et al.*, 2015). Attempts have also been made to use pre-trained 2D-CNN weights with relatively little data (Carreira & Zisserman, 2017). Optical flow and RGB were entered as two main streams to separate spatial and temporal information (Simonyan & Zisserman, 2014). Alternatively, to minimize hand-crafted elements, the same video was entered as two streams with different frame rates (Feichtenhofer *et al.*, 2019). Currently, many experiments are being conducted to modify the backbone and two-stream structures showing high performances. Action recognition can be classified into large categories such as sports and cooking, as well as individual movements in sports or detailed behaviors such as walking, running, and gestures. It is expected that the characteristics of such action recognition can express the movement of the vehicle in detail. Therefore, in this paper, the action recognition methodology is used to distinguish between small shaking of the vehicle from other normal cases.

The datasets used for action recognition research include Kinetics, Sports-1M, HMDB-51, and AVA (Gu *et al.*, 2018; Karpathy *et al.*, 2014; Kay *et al.*, 2017; Kuehne *et al.*, 2011). In these datasets, each short video clip represents one class that includes several multiple actions. However, in the case of the hit-and-run dataset, most frames show normal vehicles that are parking and moving in the scenes. And only a small fraction of the video includes the actual collision. It is necessary to pick or isolate the frames starting from the first frame of the shaking until the frame where the shaking stops. In other words only the relevant frames should be used as an input to the training model as being classed as a hit-and-run case. We apply the 3D-CNN to the isolated frames for further processing. Figure 1 shows the overall structure of the proposed system.

This paper contributes in three areas. First, it is shown that hit-and-run can be solved by applying 3D-CNN. The hit-and-run is regarded as an action recognition problem that distinguishes between the cases where the vehicle is shaken and where it is not. Second, a hit-and-run dataset was collected using a radio-controlled (RC) car to reduce the cost. The RC car was used to simulate the movement and collision of real vehicles. This method not only solved safety problems that could result when real accidents are performed but it also reduced costs by more than 100 times as compared with the real-sized performances. The filming environment was configured to fit the size of the RC car by reflecting the actual parking lot. Third, we propose a data preprocessing method that reflects the characteristics of general action recognition datasets and other hit-and-run datasets. In particular, we propose a method of introducing r values to input data only for the parts needed for the problem-solving. The r value represents the degree of the surrounding environment of the damaged vehicle. In other words, it defines the margin of areas surrounding the damaged vehicle. Finally, we propose a method of analyzing scenes that the network can misdetect and an enhancement in the method by preprocessing the dataset so that the scenes can be more robustly distinguished.

3. Dataset

In this section, we discuss how we constructed the environment where the collisions were staged, design of the scenarios for the training, and data preprocessing. In the data preprocessing, varying lengths of video clips that were collected on the designed scenarios are quantized in a fixed-frame-lengths so that they can be easily consumed and processed in the model training and testing.

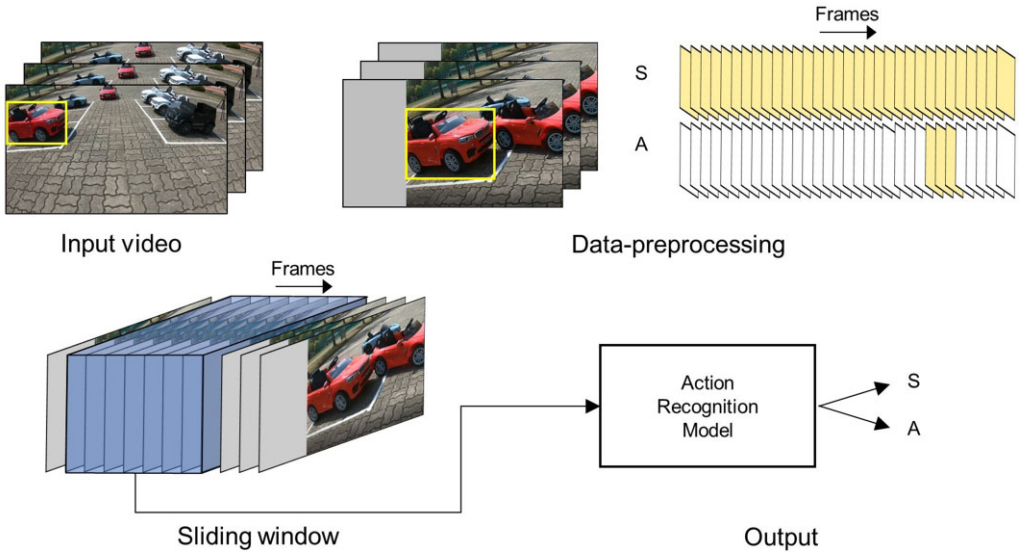

Figure 1: Overall structure of the proposed system.

Figure 2: RC cars for dataset collection.

3.1. Environment

Obtaining hit-and-run clips from CCTV is challenging due to concerns regarding potential infringement of personal information. Therefore, it is necessary to collect the datasets directly to ensure sufficient data. Direct collecting involves hiring actors to perform actions according to a designed scenario. However, following problems must be handled properly when datasets are collected directly. First, guaranteeing the diversity of datasets requires a large number of vehicles, which can be expensive to build and maintain. Second, vehicle collisions incur repairment costs and results in lapse of time needed for the vehicle to be repaired. Third, there is a risk of human injuries and a safety manual must be established to prevent such risks.

To solve this problem, we used commercially available remote-controlled miniature toy cars, as shown in Fig. 2. By using toy cars instead of actual vehicles, costs can be reduced by more than 100 times. The RC car's appearance is similar to that of an actual vehicle; however, the upper part of the car on which the infant rides is different. To evaluate the similarity between the RC cars and the actual vehicle, we compared the detection and tracking results of the RC cars by using the pre-trained weights of the YOLOv5 and DeepSORT models trained for real vehicles. These models were trained using the BDD-100K dataset, which is a large dataset of road images. Sample results are shown in Fig. 3. We can see that the RC car was correctly detected and tracked, using the weights learned from real vehicles. The results of tracking the movements of actual vehicles and RC cars were also similar. One concern was the mismatch of the aspect ratio of the RC car to that of the actual vehicle. However, data augmentation techniques are often used to expand the data to various aspect ratios (T. He *et al.*, 2019). It is widely accepted that by training with differently scaled aspect ratios, detection performance can be improved. Our experiments have shown that within the range where the spatial characteristics and features of the actual vehicle images are maintained, the CNN network is indifferent to different aspect ratios. Hence, we used the dataset collected from the RC car for training our model to identify the minor impact collisions.

The dataset was collected using a GoPro HERO6 BLACK camera, with shooting settings detailed in Table 1. The aim was to capture the appearance of an underground parking lot, where hit-and-run accidents typically occur. To achieve this, we referenced CCTV footage of underground parking lots, as well as videos from web crawling and alleyways of residential areas. We also consulted with on-site police officers and carefully considered the camera's location, angle, shooting height, and the positioning of vehicles to construct the shooting environment. The shooting height was selected as 80 cm, which is about one-third of the height of a typical parking lot CCTV camera installation. This was chosen to match the height of the RC car, which is one-third the height of an actual vehicle. The shooting angle was set at 30 degrees, at which point the upper side of the camera was not visible. When photographed with these settings, the resulting image resembled the actual parking lot shown on the left side of Fig. 4. The comparison

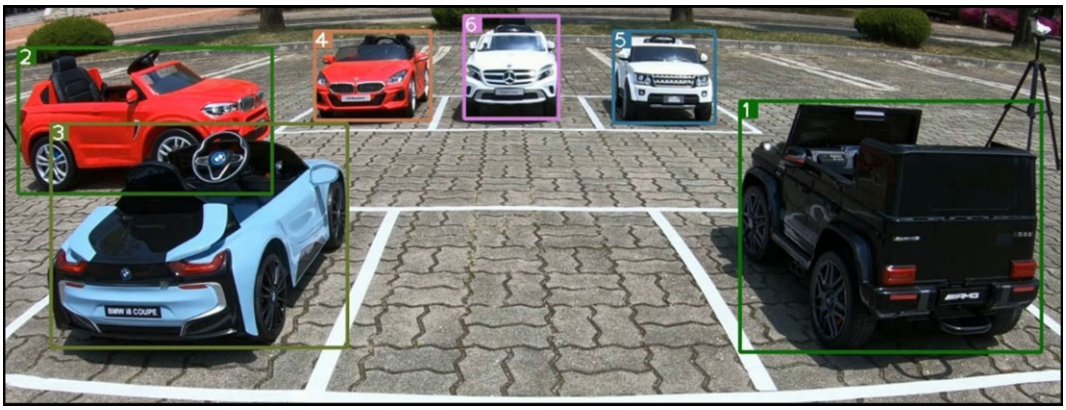

Figure 3: RC cars with object recognition and tracking.

Table 1: Camera settings.

Environment	Specification
Camera model	GoPro HERO6 BLACK
Frame per sec	30
Aspect ratio	16:9
Field of view	118.2
Resolution	1920×1080
Height from ground	0.8 m
Distance from car	1.2 m
Shooting angle	30 degrees

of the real parking lot with our staged environment is shown in Fig. 4. The distance and size measurements of the shooting stage, including the location of the RC car and the camera, are illustrated in Fig. 5.

Our collision detection system is designed to be highly robust, with the primary determining factor being any noticeable movement of the vehicle that has been impacted. This detection is dependent on the relative motion of the car with respect to the camera's line of sight. We found that the primary failure scenario occurs when the direction of the car's movement is parallel to the camera's line of sight. To analyze the cause of this failure, we have conducted a detailed analysis to determine the limiting distance of movements that can be detected by a CCTV camera. Our calculations involve determining the field of view (FOV) of the camera, calculating the number of pixels per unit distance in both horizontal and vertical resolutions, and then defining a threshold

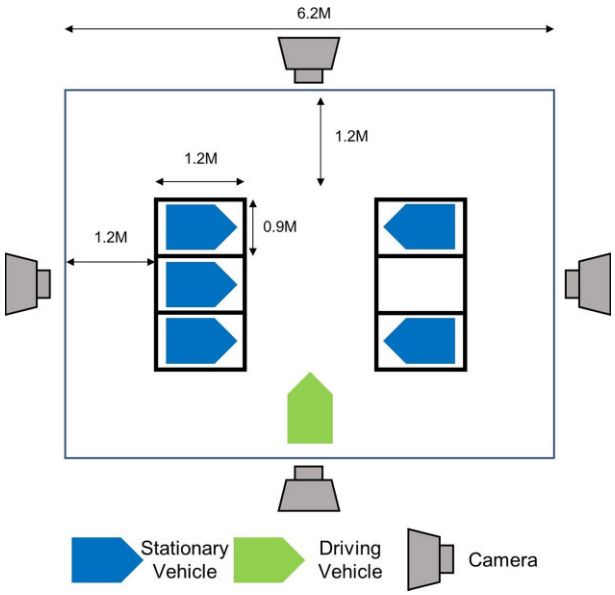

Figure 5: Recoding layouts and camera settings.

for noticeable pixel differences. For a GoPro HERO6 Black camera specifications, we calculated that the camera can discern movements of the impacted vehicle as long as it moves more than approximately 5.4 mm horizontally and 7.0 mm vertically in the or-

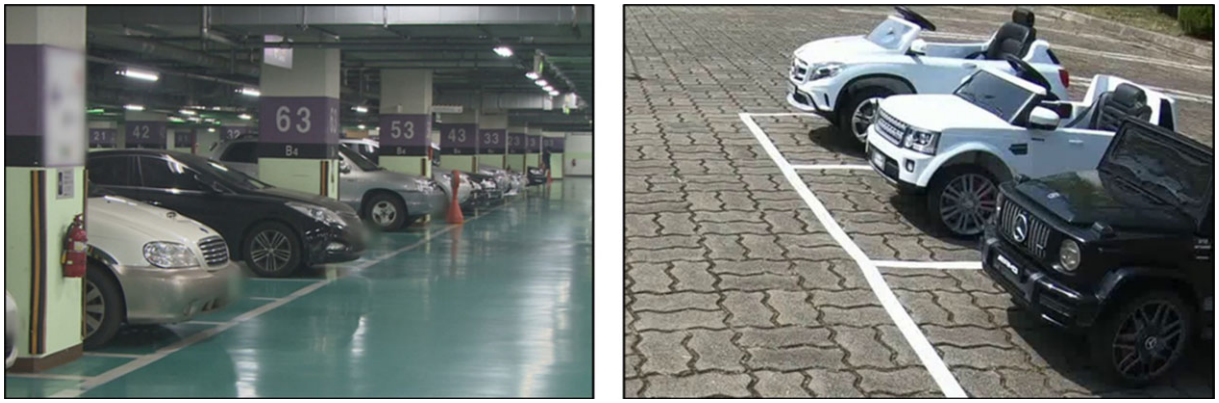

Figure 4: Actual parking lots (left panel) and collected datasets (right panel).

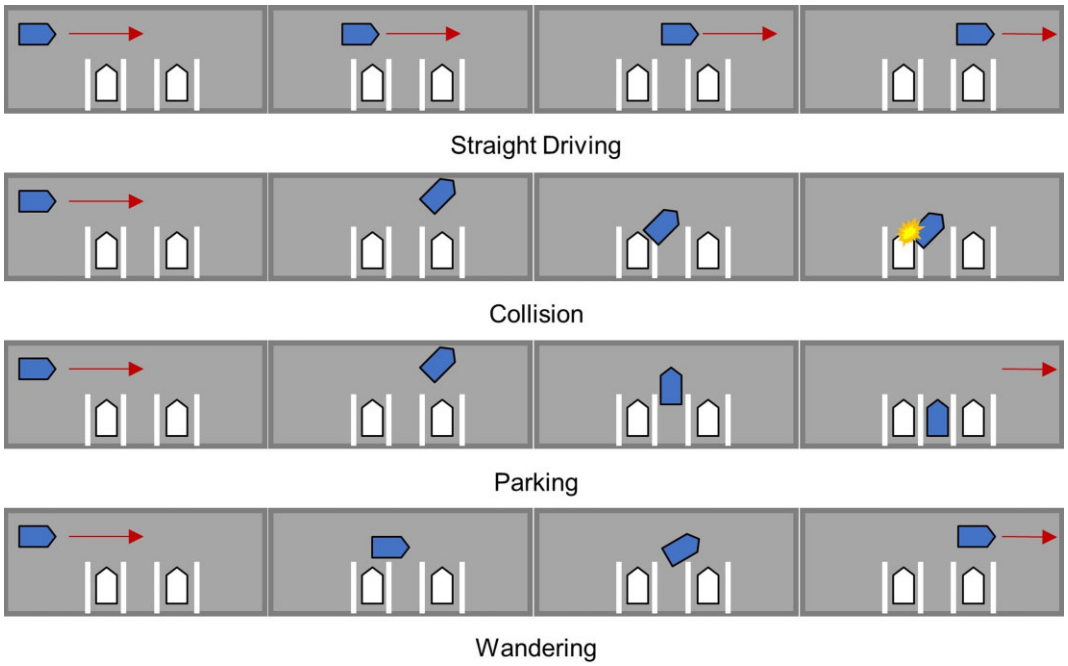

Figure 6: Dataset scenario.

thogonal direction to the line of sight. Detailed calculations for the horizontal and vertical resolutions δ_h and δ_v are as follows:

$$\delta_{h=}\left(2\times L\times\tan^{\theta_h}/_2\right)/_{n_h=5.4\text{mm/pixel}}$$
$$\delta_{v=}\left(2\times L\times\tan^{\theta_v}/_2\right)/_{n_v=7.0\text{mm/pixel}}$$

where, L is the distance to target (10 m); θ_h the horizontal FOV (91.6 degrees); n_h the number of pixels of the Charge-Coupled Device (CCD) in the horizontal direction (3840); θ_v the vertical FOV (74.35 degrees); and n_v the number of pixels of the CCD in the vertical direction (2160).

3.2. Dataset scenario

To diversify the dataset, we photographed six RC cars by alternating from four different directions, simulating a parking lot. As underground parking lots typically have no rain and provides consistent illumination, we conducted the data collection during rain-free daylight hours. Each video captured the vehicle's actions based on specific scenarios, each lasting 10 to 15 s. The recording started from the moment the vehicle entered the parking lot and continued until the action was completed.

Figure 6 illustrates four scenarios: parking, straight driving, collision, and wandering. The collision scenario is classified as class A, while the remaining scenarios are classified as non-collision class S. Parking refers to cases where the vehicle successfully parks and stops next to the reference vehicle. Straight driving refers to cases where the entering vehicle drives straight passing the reference vehicle without any special maneuvers. Collision describes cases where a collision occurs while attempting to park. Wandering refers to cases where unexpected events occur, such as temporarily stopping or approaching a reference vehicle while attempting to park in a straight driving scenario. The collected datasets include various situations. The collision scenario was thoroughly designed to recreate vehicle collisions from various directions. The wandering scenario, which does not involve a collision, simulates situations where it may appear that a collision may occur any soon but does not actually happen, such as closely approaching a reference vehicle while attempting to park. The parking scenario not only positions the vehicle in the parking space but also recreates the movement trajectory, simulating the actions of an actual driver during parking. In most scenarios, the reference vehicle was occluded by another vehicle. We collected a total of 833 videos, with the collection details presented in Table 2. The hit-and-run dataset is available for free download at the provided Kaggle URL (https://www.kaggle.com/datasets/inwoohwang2/parking-vehicle-hit-an-run-dataset).

Table 2: Collected dataset details.

Directions	Class	Number of videos
North	A	48
	S	142
South	A	59
	S	190
Left	A	61
	S	170
Right	A	41
	S	122
Total		833

Data labeling was carried out by classifying each scenario according to the image name. For class A, bounding box labeling was employed to identify the vehicle involved in the collision. We labeled the frames in the range from the start of the vehicle's collision up to and not including the frame when the shaking caused by the collision stops. This method allowed us to include frames for the duration of two vehicles actually making contacts and at the same time demonstrating minor movements. For comparisons, we also extended by including some of the frames prior to the start of the collisions during model testing. For class S, bounding box labeling was performed on the two vehicles adjacent to the parking space. These two vehicles are likely to be obscured by the driving vehicle, so labeling them assists in tracking the driving vehicle's movement.

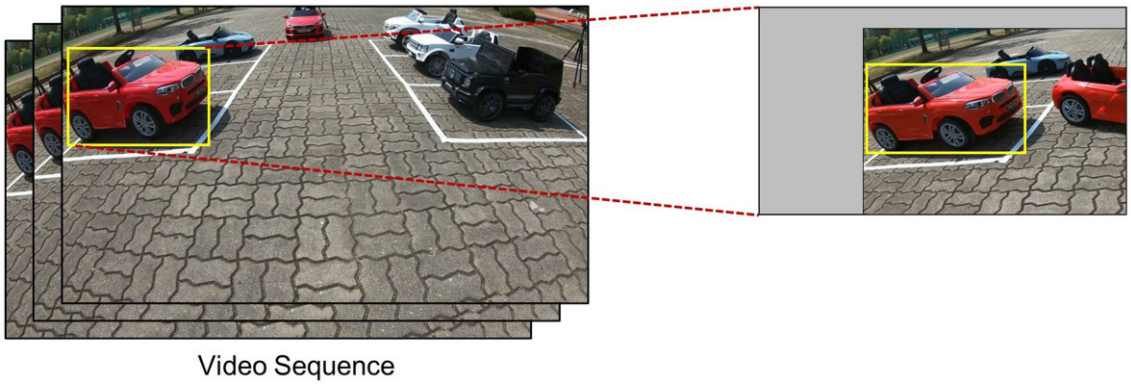

Figure 7: Proposed data preprocessing method.

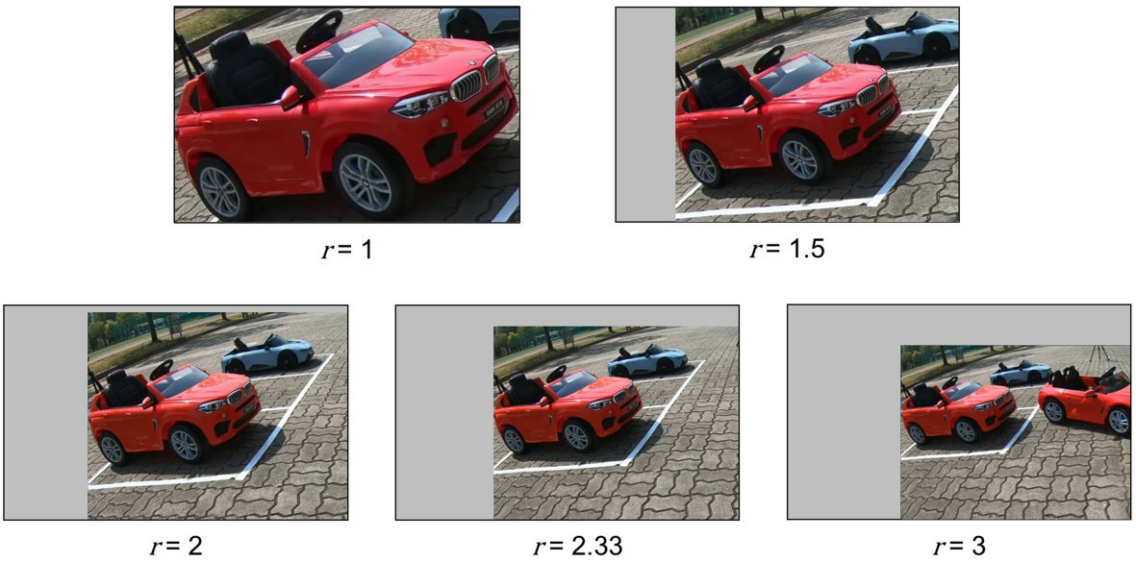

Figure 8: Peripheral margin range of reference vehicle according to the *r* value.

3.3. Data preprocessing

To achieve optimal results in deep learning, the availability of meaningful training data and the choice of the model used play significant roles. Additionally, the preprocessing method is also important. For efficient learning, it is essential to utilize a reliable data preprocessing method. This process includes classifying the data to align with the learning objective and removing any unnecessary data that might hinder the learning process.

As previously mentioned, we employed a 3D-CNN network-based action recognition algorithm. This algorithm was used to detect hit-and-run incidents and distinguish between collision and non-collision scenarios. Two factors were considered when applying the collected dataset to the network. The first factor was the length of the input video clip. For class A, we performed experiments by dividing the length of each video clip into 9, 15, and 30 frames. In these divisions, the last frame corresponds to the moment when the collision concludes. The reference frame length for each class A is as follows. In 9-frame clips, the footage contains only the reference vehicle already crashing and shaking. In 30-frame clips, the vehicle approaches and collides with the reference vehicle, causing shaking. In 15-frame clips, the properties were intermediate between 9-frame and 30-frame clips.

The second factor is the location of the damaged vehicle and the degree to which the surrounding environment is incorporated. The extent of the surrounding environment is controlled by the variable *r*. In the case of a hit-and-run incident, this method is straightforward to implement because the location of the damaged vehicle is already known. Instead of utilizing a full image that contains unnecessary information as input, this method adjusts the extent to which the affected vehicle and its surroundings are observed. The variable w_c denotes the width of the cropped image, while w_v represents the width of the reference vehicle. The reference vehicle is precisely positioned in the center of the cropped image. If centering is not possible, padding is added as illustrated in Fig. 7. The *r* value, representing the degree of the surrounding environment of the damaged vehicle, is shown in Fig. 8. By changing the *r* value the training clips are augmented and expanded. The 833 videos become 1457 video data.

$$r = \frac{w_c}{w_v}$$

As the value of *r* decreases, the system concentrates more on the affected vehicle. Consequently, it can be more focused on the problem of recognizing a collision and pay less on the vehicle's trajectory. On the other hand, as the value of *r* increases, the model

Figure 9: Images with ambiguous collision status.

Table 3: The video clips used for training.

Video clip frame	Class	Number of video clips	Total
9 Frame	A	209	32 975
	S	32 766	
15 Frame	A	209	19 547
	S	19 338	
30 Frame	A	209	9487
	S	9278	

relies more on the vehicle's trajectory. However, this may lead the model to incorporate unrelated surroundings, potentially resulting in a misinterpretation of the collision of the affected vehicle.

The images below are all non-collision images. In the image depicted in Fig. 9, it is challenging to tell whether the top three pictures are displaying a state of collision. Additionally, the appearance of a slowly moving vehicle and a vehicle slightly displaced due to a collision are quite similar. To reduce the false alarm rate, it is necessary to differentiate between these two situations. Therefore, we divided the non-collision videos into frames of a specific length and added them to the training as labeled to be non-collision. The video augmented by the r value was also divided into frames of the same length. Table 3 summarized the number of video clips used for training.

We used a fixed frame length for training data. However, during the testing process, video data of different lengths can be used. We utilized the sliding window method during testing. The entire video is segmented into frames of the same length as used in the training process with a stride of one. This causes much overlappings between testing video clips.

4. Model

In this section, we compare between two choices of network models for our purposes. First is the 2D-CNN-RNN model which combines 2D-CNN with RNN for temporal considerations. Second is the 3D-CNN model that considers time as a third dimension in space therefore unifies the space and time. We also discuss about the used data augmentation techniques and the better understanding of the proposed model through the use of the class activation map (CAM) for highlighting where in the image the network is concentrating more to identify the accidents.

4.1. 2D-CNN-RNN model

In the field of computer vision, effectively handling continuous time series of images entails more than merely leveraging a CNN function. The system must also possess the ability to process sequential images. While CNNs are suitable for processing short-term information, they are not ideally suited for learning temporal information across the entire dataset. Conversely, an RNN is well-suited for modeling sequence data; however, it does not excel at learning spatial information. Donahue *et al.* (2015) proposed a combined CNN-RNN network that synergizes the strengths of both neural network structures for video recognition. This network employs a CNN as an encoder and an RNN as a decoder. Figure 10 illustrates the fundamental concept of the CNN-RNN structure.

4.2. 3D-CNN model

The hit-and-run dataset presents distinctive characteristics that resemble general action recognition datasets. For example, in a scenario involving a baseball player pitching, there are several distinct body movements, each contributing to the overall action of pitching. Similarly, the hit-and-run dataset comprises a scene where a vehicle enters, attempts to park, and eventually collides. This process is crucial for reproducing the given situation. However, it is essential to understand that not all these movements should be categorized as a vehicle collision. In any given video, a collision should be interpreted as occurring only when the vehicle makes direct contact and demonstrates discernible shaking.

In the proposed method, the primary factor for distinguishing whether a vehicle has collided or not is the shaking of the reference vehicle. The trajectory information of the vehicle, based on the r value, may also influence the collision determination. Ob-

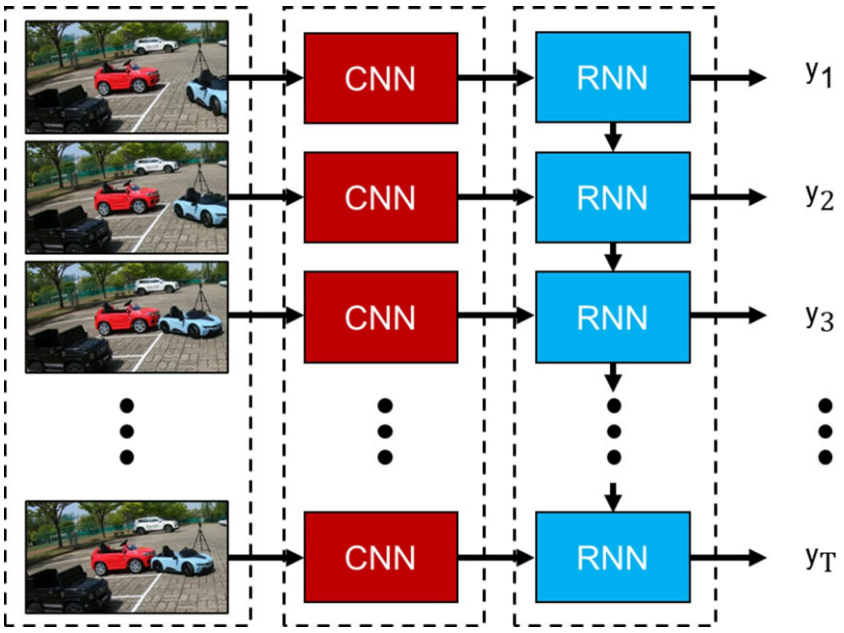

Figure 10: 2D-CNN-RNN model for action recognition.

serving vehicle shaking requires both spatial and temporal information. Therefore, we chose a 3D-CNN, which extends the concept of 2D-CNN in the temporal direction, to effectively utilize both types of information.

As a video is an extension of an image along the time axis, an image serves as the base. The 3D-CNN structure extends the time-axis concept of the 2D-CNN structure and incorporates various concepts from 2D-CNN. It also adopts structures such as VGGNet, ResNet, and GoogLeNet which have shown strong performance in 2D-CNN.

The backbone network used in this study is based on Two-Stream Inflated 3D ConvNets (I3D), which leverages the benefits of well-performing 2D-CNNs like ResNet and Inception (He *et al.*, 2016; Szegedy *et al.*, 2015) that represent efficient operations in 2D-CNN. The I3D network structure, as depicted in Fig. 11, expands the dimensions of the 2D-CNN structure. This enables the use of pre-trained weights, thereby enhancing both performance and capabilities. The structure of the inception module, as shown in Fig. 12 (Carreria & Zisserman, 2017), is utilized. The network accepts a continuous RGB image as input. A $1 \times 1 \times 1$ convolution is employed at the network's head to reduce the feature maps. The softmax is then used to classify whether a collision has occurred or not.

Contrary to the conventional use of I3D, we decided not to utilize optical flow. This decision stemmed from its apparent limitations in detecting shakes in vehicles caused by collisions, particularly when those vehicles are moving slowly. Furthermore, the computational load required to generate optical flow images greatly exceeds that of RGB images. However, the slight improvement in performance does not adequately justify the extra computational expense.

4.3. Data augmentation

Data augmentation is a well-known method for improving the performance of deep learning models especially when the training dataset is not large enough to guarantee generalization. Data augmentation can vary for each image in image data. However, in the case of video data, all frames of the video should use the same data augmentation method. Augmented data should closely resemble the original data. In this study, a data augmentation method was selected that preserves the characteristics of vehicle movement. Shaking is aimed to be simulated as close to the reality as possible. For instance, we excluded the vertical flip from our methods. This decision was made because CCTV cameras are rarely overturned completely, causing the footage to be upside down. However, a horizontal flip can diversify the data while maintaining reasonable image features. We applied color jitter to create various illumination effects and rotation was used to produce different photographic compositions. We also applied random crop to augment the dataset for the purpose of adding collisions in restricted and occluded views.

4.4. Class activation map

When CNN is used for classification problems, a fully connected layer is typically included at the end. However, flattening this last layer can result in the loss of localization information, which can complicate the identification of which part of the network determines the decision-making process. The challenge is addressed by employing the CAM, which leverages global average pooling instead of flattening. This method applies the feature map from the backbone to the fully connected layer and generates a new weight. By multiplying the weight with the feature map, we can create a heat map for a specific class. This heat map can help identify the foundation of the network's prediction for a particular class (Zhou *et al.*, 2016). Figure 13 illustrates the structure of CAM.

In the context of a 3D-CNN, we calculate the heat map similarly to a 2D-CNN, using the final layer as the output. The heat map highlights the local areas of the image that receive more attention by increasing intensity, indicating their relative importance to the model's decision-making process. Figure 14 provides an illustration of the CAM at the moment of collision, the point of contact on the vehicle is highlighted in red, indicating that the network considers both the vehicle's shaking and the contact point. CAM allows for the visualization of relevant parts of the input data, thereby enabling domain experts to understand the model's be-

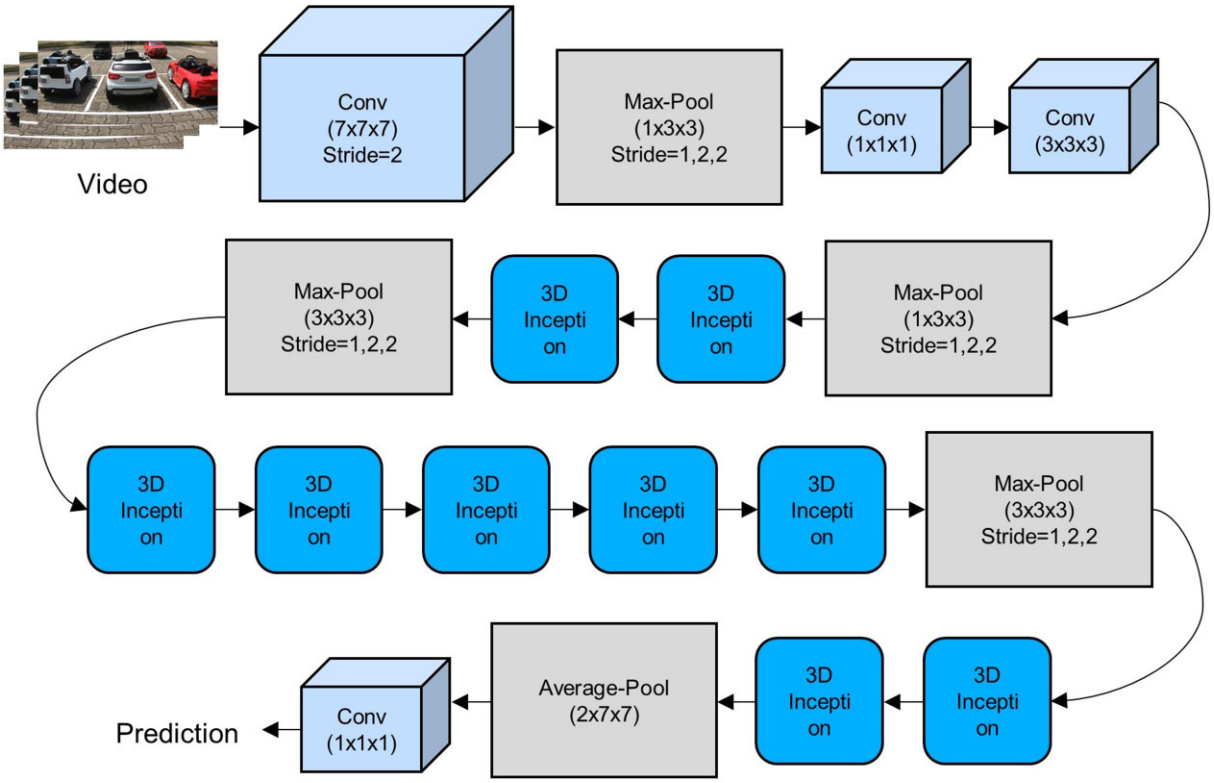

Figure 11: I3D model for action recognition.

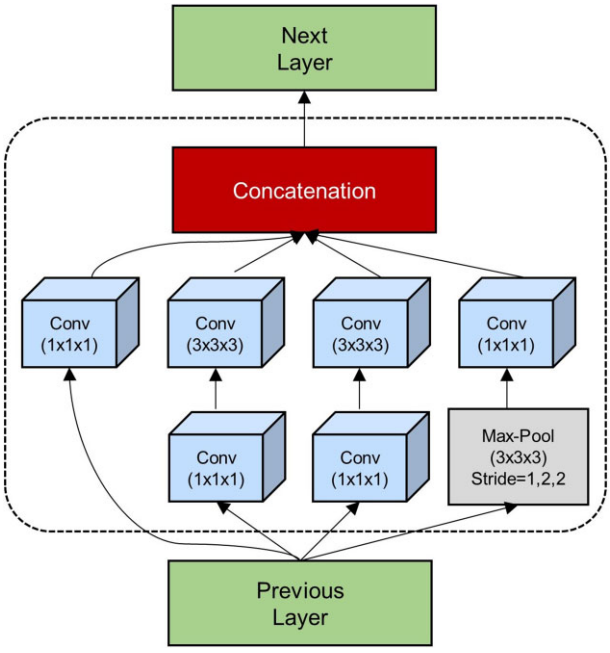

Figure 12: Inception module.

havior better. Additionally, the results of this visualization can be used to enhance the model's accuracy and interpretability.

5. Experiments

The comparison of the outputs from the two models that were used to identify the collisions is discussed in this section. The detailed specification of the experimental environment is shown in Table 4.

The choice of different models on the results was evaluated by comparing the CNN-RNN and the 3D-CNN. The CNN-RNN algorithm was developed using PyTorch. We utilized ResNet-152 pre-trained on ImageNet. The CNN encoder transforms all 2D images into 1D vectors. Meanwhile, the decoder uses an LSTM to receive a sequential input vector from the CNN encoder and output another sequence. For the final prediction, a fully connected layer was employed. We set the batch size to 256 and the number of epochs to 15. The learning rate was selected as 0.00 001 and we used the Adam optimizer.

For training, 1164 out of a total of 1457 videos were used, and the remaining 293 videos were used for testing. Among the 293 videos used for testing, class A contained 41 videos and class S consisted of 252 videos. The accuracy was calculated as follows:

$$\text{Accuracy} : \frac{TP + TN}{TP + TN + FP + FN}.$$

The accuracy is a binary classification problem where Positive denotes the video was identified as an accident (class A) and non-accident (class S), otherwise. The results of the CNN-RNN structure are presented in Table 5 below.

A bias towards the class S was tended to be exhibited by the model in the CNN-RNN architecture. This is due to the fact that number of trained samples for class S outnumbered that of class A by 50 to 1. There was no significant difference in accuracy based on the r value size or the frame length.

For the 3D-CNN architecture, we did not use pre-trained weights. The hyperparameters were set as follows. The batch size was set to 15. The number of epochs was set to 100. A learning rate of 0.00 001 was set, and Adam was used as the optimizer. Early stopping was applied to prevent overfitting. The results of the 3D-CNN architecture are presented in Table 6.

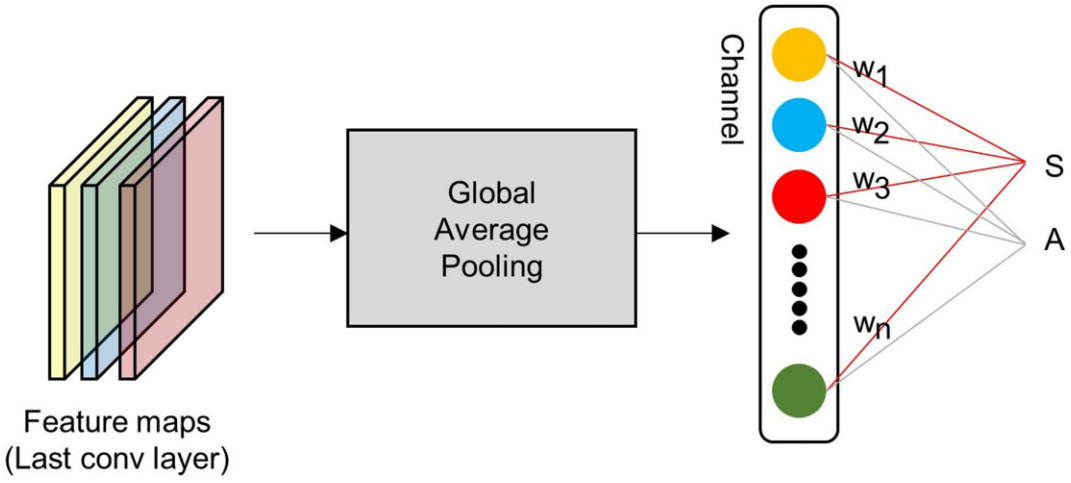

Figure 13: CAM in I3D model.

Figure 14: Result of CAM.

Table 4: Computing environment.

Parts	Specification
GPU	Nvidia quadro RTX 8000
CPU	Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz
RAM	500 GB LRDIMM DDR4
NVIDIA CUDA Cores	8 × 4608cores
OS	Ubuntu 18.04 Linux OS
Language	Python
Framework	PyTorch

Table 5: Action recognition accuracy comparison (2D-RNN).

Frame length (*r*)	Recall	False alarm	Accuracy
9 ($r = 1$)	39.02%	3.17%	88.74%
9 ($r = 1.5$)	41.46%	0.40%	91.47%
9 ($r = 2$)	34.14%	1.19%	89.76%
9 ($r = 2.33$)	34.14%	3.57%	87.71%
9 ($r = 3$)	19.51%	5.56%	83.96%
15 ($r = 1$)	24.39%	5.16%	84.98%
15 ($r = 1.5$)	39.02%	0.79%	90.78%
15 ($r = 2$)	24.39%	3.57%	86.35%
15 ($r = 2.33$)	21.95%	3.17%	86.34%
15 ($r = 3$)	29.27%	5.95%	84.98%
30 ($r = 1$)	36.59%	14.28%	78.84%
30 ($r = 1.5$)	36.59%	6.75%	85.32%
30 ($r = 2$)	26.83%	4.37%	86.01%
30 ($r = 2.33$)	24.39%	4.37%	85.66%
30 ($r = 3$)	17.07%	3.17%	85.66%

Table 6: Action recognition accuracy comparison (3D-CNN).

Frame length (*r*)	Recall	False alarm	Accuracy
9 ($r = 1$)	80.49%	9.52%	89.08%
9 ($r = 1.5$)	68.29%	1.98%	93.86%
9 ($r = 2$)	80.49%	3.97%	93.86%
9 ($r = 2.33$)	63.41%	2.38%	92.83%
9 ($r = 3$)	43.90%	5.56%	87.37%
15 ($r = 1$)	85.37%	4.37%	94.20%
15 ($r = 1.5$)	56.10%	7.14%	87.71%
15 ($r = 2$)	75.61%	7.54%	90.10%
15 ($r = 2.33$)	58.54%	5.16%	89.76%
15 ($r = 3$)	63.41%	3.17%	92.15%
30 ($r = 1$)	92.68%	2.38%	96.93%
30 ($r = 1.5$)	85.37%	6.35%	92.49%
30 ($r = 2$)	80.48%	3.17%	94.54%
30 ($r = 2.33$)	53.66%	1.98%	91.81%
30 ($r = 3$)	63.41%	3.57%	91.81%

The 3D-CNN did not exhibit as much biasing as the CNN-RNN and showed significantly better performance. The accuracy was found to be better for smaller values of r. Additionally, longer frame lengths resulted in better performance. The best performance was achieved with $r = 1$ and a frame length of 30 frames.

In overall, 3D-CNNs outperformed CNN-RNN structures for several reasons. The CNN-RNN structure requires determining optimal hyperparameters for each network separately. This is because the CNN model and the RNN model exhibit different characteristics. The CNN-RNN model is strong in preserving sequential time information but suffers from gradient vanishing problem. However, in the context of classifying vehicle collisions and non-collisions, the inclusion of sequential time information is not essential. The critical factor in this task is the shaking scene of the vehicle.

To evaluate how our model performs on real data, we demonstrate our performance on 20 actual hit-and-run videos. Figure 15 shows the snapshots of these actual data. The sources of the videos are given in the Appendix 1. Although our primary focus is on low-speed collisions, we acknowledge the importance of considering high-speed scenarios. Items 5 and 18 in our dataset represent high-speed collisions, offering insights into the model's performance under such conditions. These cases, along with relevant footage (where available), are included in Appendix 1 for

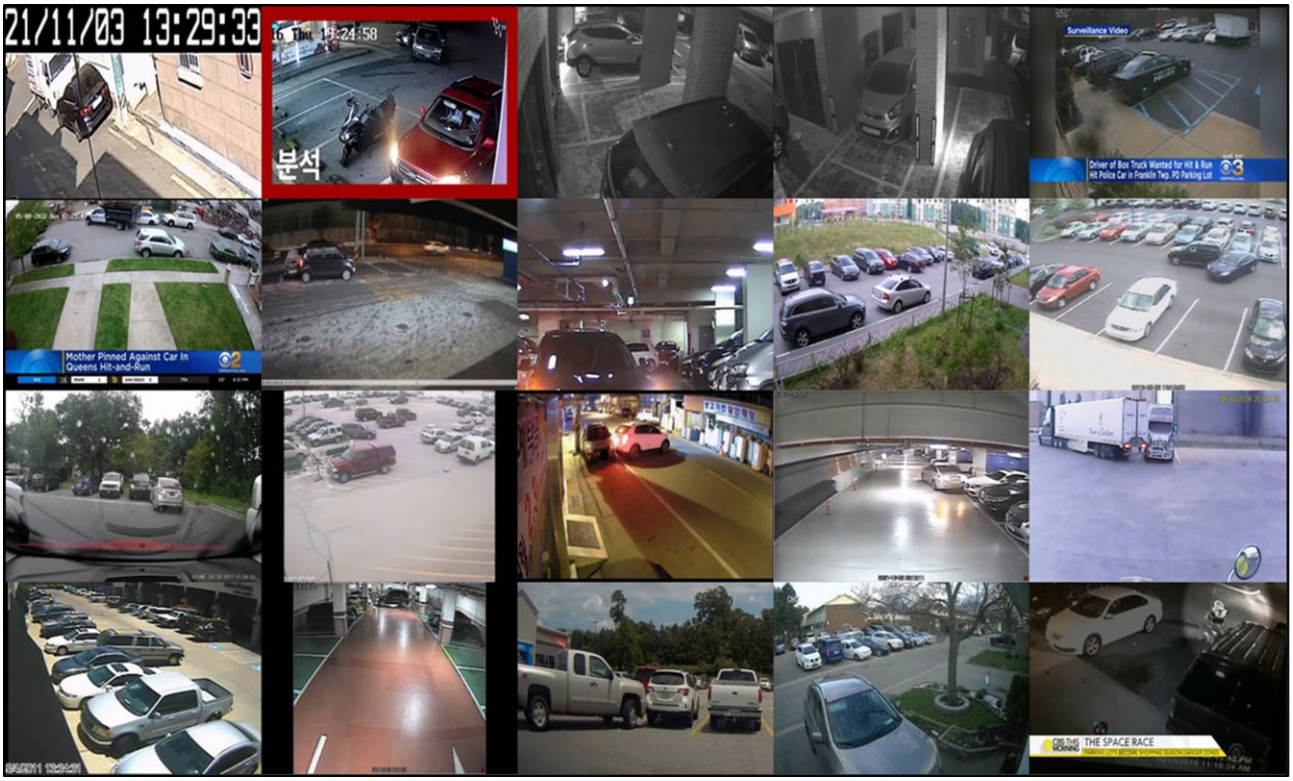

Figure 15: Web crawling data.

further reference. We used the optimal hyperparameters of $r = 1$ and a frame length of 30 frames based on our experimental results. This approach achieved a recall of 75% (15 out of 20 videos). To supplement the insufficient web crawling data, we augmented the accident videos. One approach involved rearranging the obstacles in the scenes to occlude the reference vehicle from various angles. Additionally, we introduced variations in brightness to simulate day and night conditions, thereby expressing different lighting conditions. There is a disparity in vibration characteristics between real vehicles and our mock-up models. The difference in tire damping between an RC car and a real car does lead to variations in vibration duration. However, for detecting the minor movement of the cars in collision, the amplitude of movement is critical. For this matter, the difference between the RC car and a real car is marginal and within acceptable limits for our purposes.

Figure 16 shows the augmented data, which consisted of 12 h of footage and 12 collision scenes. The augmented data demonstrated similar performance to the web crawling data, achieving a recall of 83.33% in detecting 10 out of 12 collisions. The results of the web crawling and augmented web crawling data can be found in Table 7.

6. Discussion

Various effects to the performance of the model, including the marginal value *r*, are discussed in this section.

6.1. Performance based on frame length

Better overall performance than the CNN-RNN was exhibited by the 3D-CNN structure and here we analyzed its results. We found that the recall decreases when the length of the input frame is short. This means that there are many cases where a collision occurs with the reference vehicle; however, it is not recognized as a collision. This issue arises because the input information is too brief. Consequently, the network cannot distinguish between a situation where another vehicle obstructs the reference vehicle as it passes by and a scene where the vehicle shakes due to a collision. Scenes where a vehicle passes by without a collision, occluding the reference vehicle, occur more frequently than scenes where a vehicle collides with and shakes the reference vehicle. Collision situations are more prone to be misinterpreted as non-collision events by the model due to this imbalance. When the frame length increases, the model can take in more information. It allows the model to discern the distinctive pixel changes between situations where other vehicles occlude the reference vehicle while passing by and situations where collisions occur.

6.2. Performance based on marginal value *r*

The *r* value primarily influences the scope of visual data considered in detecting collisions. When the *r* value is at its minimum, our detection model focuses exclusively on the pixel values within a tightly defined box encompassing the impacted vehicle. This narrow focus is designed to analyze the immediate visual changes precisely at the point of impact. However, as the *r* value increases, our algorithm expands its scope to include pixel values in the margins surrounding the impacted vehicle. This broader view allows for a more comprehensive analysis of the collision context, incorporating the behavior of nearby vehicles and environmental factors.

The purpose of adjusting the *r* value, therefore, is to meticulously control the extent to which surrounding elements are factored into the collision determination process. By varying the *r* value, we can assess how the inclusion of adjacent areas influences the detection accuracy. This nuanced approach is critical for enhancing the overall accuracy and reliability of our collision detection system.

Figure 16: Results of augmented web crawling data.

Table 7: Performance of web crawling video (3D-CNN).

Video	Recall
Web crawling	75%
Web crawling + augmentation	83.33%

Figure 17: Image incorrectly detected as class S.

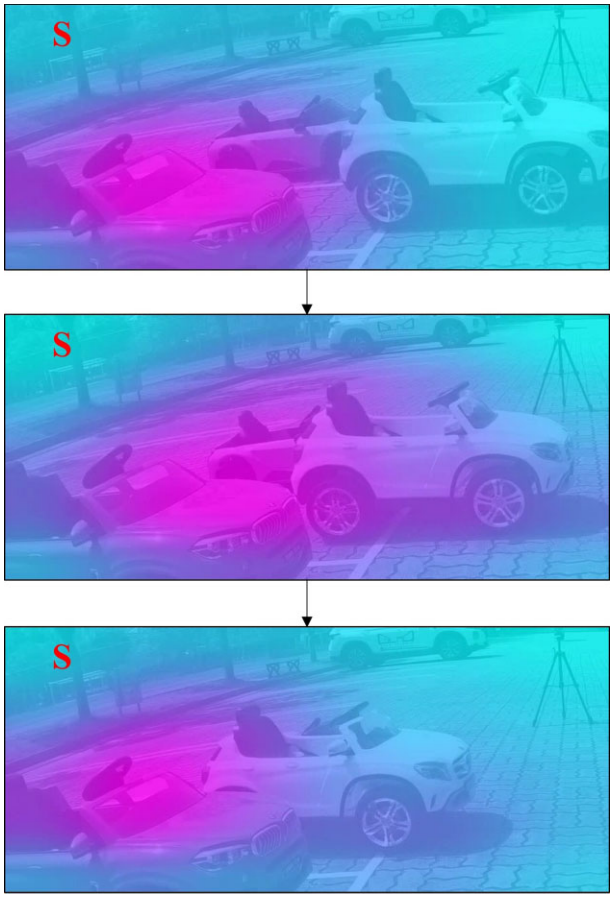

Figure 18: Image with reference vehicle occluded.

Based on the findings presented in Table 6 and Fig. 17, it can be inferred that when the input includes more areas surrounding the vehicle, the model is more prone to fail to detect a collision, even when one does occur. With an increase in the value of r, the heatmap shifts focus towards the space between the vehicles. This suggests that the model identifies collisions based on the vehicle trajectories rather than their shaking movements.

As shown in Fig. 18, when the value of r is small, the vehicle passing the reference vehicle may cover most of the reference vehicle. However, the model does not recognize this as a collision. This is because it is able to distinguish between the general movement of the vehicle and the movement caused by the collision.

7. Conclusions

A 3D-CNN-based network for detecting small shaking of a vehicle in hit-and-run is proposed in this paper. An RC car, due to its lower stiffness and lesser mass compared with a real vehicle, would at first thought to exhibit a larger relative motion during an impact. However, this is counterbalanced by the fact that the impacting object in our experiments was also proportionally smaller and lighter, similar to the RC car. Consequently, the observed motion of the RC car during impact turned out to be significantly smaller than what would be expected in a real vehicle collision. This discrepancy in the physical dynamics of the RC car impacts has a

notable implication for our study. Our training dataset, which captured these smaller scale impulsive movements, inadvertently led to our model being more sensitive to impact detection. This increased sensitivity is due to the model being trained on data where even minor impacts result in visible motion, a scenario less likely in real vehicle collisions due to their greater mass and stiffness. This approach not only enhances the generalizability of our model but also ensures that it remains sensitive and accurate when applied to real-world scenarios involving actual vehicles.

Our dataset includes RC cars, which showed reasonably good object detection accuracy using YOLOv5 as compared with real vehicles. We utilized a modified I3D models that have performed well in the field of action recognition. We proceeded with a 1-stream network because optical flow information has a negative effect on accuracy. We compared accuracy using two conditions. Those are frame length in the 3D-CNN model and the input margin extent of the vehicle's surroundings. Additionally, we applied CAM to visualize how the 3D-CNN model works. Our results demonstrated that the model performed better with a frame length of 30 frames and with the smallest value of r. A high false alarm rate was observed for larger values of r. We collected our own dataset without utilizing publicly available open datasets. Our dataset only includes parking lots. We believe more data are needed to improve the model's performance. As for a future work, more data collection is necessary to carefully learn diverse collision behaviors in hit-and-run accidents. New locations to include are alleys or densely populated residential areas, and it is necessary to collect datasets that reflect the real situation, taking into account the angle of the cameras of the CCTV installed. We believe such effort will enable the creation of a model that can be robustly applied in more hit-and-run scenarios.

Acknowledgments

This work was conducted by Center for Applied Research in Artificial Intelligence (CARAI) grant funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD) (UD230017TD), Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE; P0020535, The Competency Development Program for Industry Specialist), and GIST Research Project grant funded by the GIST in 2023.

Conflict of interest statement

None declared.

References

Baek, T., & Lee, Y.-G. (2022). Traffic control hand signal recognition using convolution and recurrent neural networks. *Journal of Computational Design and Engineering*, **9**, 296–309. https://doi.org/10.1093/jcde/qwab080.

Bao, L., & Intille, S. S. (2004). Activity recognition from user-annotated acceleration data. *Lecture Notes in Computer Science*, 1–17. https://doi.org/10.1007/978-3-540-24646-6_1.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. *Computer Vision – ECCV 2006*, 404–417. https://doi.org/10.1007/11744023_32.

Biradar, K. M., Gupta, A., Mandal, M., & Vipparthi, S. K. (2019). Challenges in time-stamp aware anomaly detection in traffic videos. *arXiv preprint arXiv:1906.04574*. https://doi.org/10.48550/arXiv.1906.04574.

Carreira, J., & Zisserman, A. (2017). Quo Vadis, action recognition? A new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2017.502.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. *arXiv preprint arXiv:1605.06409*. https://doi.org/10.48550/arXiv.1605.06409.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. https://doi.org/10.1109/cvpr.2005.177.

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Darrell, T., & Saenko, K. (2015). Long-term recurrent convolutional networks for visual recognition and description. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2015.7298878.

Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). SLOWFAST networks for video recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv.2019.00630.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**, 1627–1645. https://doi.org/10.1109/TPAMI.2009.167.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2014.81.

Gu, C., Sun, C., Ross, D. A., Vondrick, C., Pantofaru, C., Li, Y., Vijayanarasimhan, S., Toderici, G., Ricco, S., Sukthankar, R., Schmid, C., & Malik, J. (2018). Ava: A video dataset of Spatio-temporally localized atomic visual actions. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2018.00633.

Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., & Davis, L. S. (2016). Learning temporal regularity in video sequences. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.86.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.90.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 558–567). IEEE. https://doi.org/10.1109/cvpr.2019.00065.

Held, D., Thrun, S., & Savarese, S. (2016). Learning to track at 100 FPS with deep regression networks. *Computer Vision – ECCV 2016* (pp. 749–765). https://doi.org/10.1007/978-3-319-46448-0_45.

Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D Convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**, 221–231. https://doi.org/10.1109/TPAMI.2012.59.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2014.223.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., & Zisserman, A. (2017). The kinetics human action video

dataset. *arXiv preprint arXiv:1705.06950*. https://doi.org/10.48550/arXiv.1705.06950

Kim, E., Ryu, H., Oh, H., & Kang, N. (2022). Safety monitoring system of personal mobility driving using deep learning. *Journal of Computational Design and Engineering*, **9**, 1397–1409. https://doi.org/10.1093/jcde/qwac061.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., & Serre, T. (2011). HMDB: A large video database for human motion recognition. *2011 International Conference on Computer Vision* (pp. 2556–2563). https://doi.org/10.1109/iccv.2011.6126543.

Laptev, I., & Lindeberg, T. (2006). Local descriptors for spatio-temporal recognition. In MacLean W. J. (Ed.), *Spatial coherence for visual motion analysis. SCVMA 2004. Lecture notes in computer science* (Vol. **3667**, pp. 91–103). Springer. https://doi.org/10.1007/11676959_8.

Lee, S., Woo, T., & Lee, S. H. (2023). Multi-attention-based soft partition network for vehicle re-identification. *Journal of Computational Design and Engineering*, **10**, 488–502. https://doi.org/10.1093/jcde/qwad014.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Computer Vision – ECCV 2016* (pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**, 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

Poleg, Y., Ephrat, A., Peleg, S., & Arora, C. (2016). Compact CNN for indexing egocentric videos. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. https://doi.org/10.1109/wacv.2016.7477708.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.91.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**, 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031.

Ribeiro, M., Lazzaretti, A. E., & Lopes, H. S. (2018). A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*, **105**, 13–22. https://doi.org/10.1016/j.patrec.2017.07.016.

Samani, H., Yang, C.-Y., Li, C., Chung, C.-L., & Li, S. (2022). Anomaly detection with vision-based deep learning for epidemic prevention and control. *Journal of Computational Design and Engineering*, **9**, 187–200. https://doi.org/10.1093/jcde/qwab075.

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*. https://doi.org/10.48550/arXiv.1406.2199.

Sultani, W., Chen, C., & Shah, M. (2018). Real-world anomaly detection in surveillance videos. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2018.00678.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2015.7298594.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3D convolutional networks. *2015 IEEE International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv.2015.510.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (pp. I–I). IEEE. https://doi.org/10.1109/cvpr.2001.990517.

Wang, H., Kläser, A., Schmid, C., & Liu, C.-L. (2013). Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, **103**, 60–79. https://doi.org/10.1007/s11263-012-0594-8.

Wang, L., Liu, T., Wang, G., Chan, K. L., & Yang, Q. (2015). Video tracking using learned hierarchical features. *IEEE Transactions on Image Processing*, **24**, 1424–1435. https://doi.org/10.1109/TIP.2015.2403231.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. *2017 IEEE International Conference on Image Processing (ICIP)*. https://doi.org/10.1109/icip.2017.8296962.

Xu, D., Ricci, E., Yan, Y., Song, J., & Sebe, N. (2015). Learning deep representations of appearance and motion for anomalous event detection. *Procedings of the British Machine Vision Conference 2015*. https://doi.org/10.48550/arXiv.1510.01553.

Yao, Y., Xu, M., Wang, Y., Crandall, D. J., & Atkins, E. M. (2019). Unsupervised traffic accident detection in first-person videos. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 273–280). IEEE. https://doi.org/10.1109/iros40897.2019.8967556.

Zhang, Z., & Tao, D. (2012). Slow feature analysis for Human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**, 436–450. https://doi.org/10.1109/TPAMI.2011.157.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2921–2929). IEEE. https://doi.org/10.1109/cvpr.2016.319.

Zhou, Z., Dong, X., Li, Z., Yu, K., Ding, C., & Yang, Y. (2022). Spatio-temporal feature encoding for traffic accident detection in VANET environment. *IEEE Transactions on Intelligent Transportation Systems*, **23**, 19772–19781. https://doi.org/10.1109/TITS.2022.3147826.

Zhu, Q., Yeh, M.-C., Cheng, K.-T., & Avidan, S. (2006). Fast human detection using a cascade of histograms of oriented gradients. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (pp. 1491–1498). IEEE. https://doi.org/10.1109/cvpr.2006.119.

Appendix 1. Web Crawling Data Details

Video	FPS	Resolution	Video length	Accident time	URL
1	30	640 × 480	26	17	https://www.youtube.com/watch?v=l9KMgfV5BeU
2	29.47	1280 × 720	2:55	46	https://www.youtube.com/watch?v=AH5jkmpq_Jc
3	29.97	1280 × 720	1:04	17	https://www.youtube.com/watch?v=4_43LiZyfnQ
4	29.97	720 × 480	51	24	https://www.youtube.com/watch?v=X4qMykwdxNo
5	29.78	1280 × 720	43	33	https://www.youtube.com/watch?v=nIuFYkC7y2Q
6	24.86	1280 × 720	1:54	47	https://www.youtube.com/watch?v=VwRT2tcsFmk
7	30	1280 × 720	3:00	2:27	https://www.youtube.com/watch?v=Z-W96lVRhzI
8	15	1920 × 1080	6:02	20	-
9	29.67	1280 × 720	49	10	https://www.youtube.com/watch?v=OH3hD5C2HYs
10	30	1280 × 720	14	10	https://www.youtube.com/watch?v=HhbkdO9VPn8
11	10	528 × 190	6	4	https://www.clien.net/service/board/cm_car/16653661?combine=true&q=%EB%AC%BC%ED%94%BC%EB%8F%84%EC%A3%BC+cctv&p=2&sort=recency&boardCd=&isBoard=false
12	30	1280 × 720	15	6	https://www.youtube.com/watch?v=0D94uatULm0
13	30	1280 × 720	22	11	https://www.youtube.com/watch?v=0zfObwJW5qE
14	30	854 × 480	43	25	https://www.youtube.com/watch?v=X8DeXsAwnbU
15	30	1280 × 720	2:00	46	https://www.youtube.com/watch?v=7cbeyX8OAZU
16	30	1920 × 1080	1:13	37	https://www.youtube.com/watch?v=aIU4BpGLWSc
17	15	1920 × 1080	6:28	2:26	-
18	30	1280 × 720	1:32	1:12	https://www.youtube.com/watch?v=v9r3cMHdJ0s
19	30	1920 × 1080	1:31	20	https://www.youtube.com/watch?v=9Ezgb3eJnaI
20	30	480 × 360	25	7	-

Received: September 19, 2023. **Revised:** February 8, 2024. **Accepted:** February 12, 2024