An Integrative Approach to Pedestrian Detection Based on Sparse Optical Flow and Convolutional Neural Network Model

Hosik Choi

The Graduate School

School of Electrical and Electronic Engineering

Yonsei University

December 2022

An Integrative Approach to Pedestrian Detection Based on Sparse Optical Flow and Convolutional Neural Network Model

A Masters Thesis
Submitted to the Department of
Electrical and Electronic Engineering
and the Graduate School of Yonsei University
in partial fulfillment of the
requirements for the degree of
Master of engineering

Hosik Choi

December 2022

This certifies that the master's thesis
of Hosik Choi is approved.

Thesis Supervisor: DaeEun Kim

Euntai Kim

Dong-Hyun Kim

The Graduate School

School of Electrical and Electronic Engineering

Yonsei University

December 2022

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Hosik Choi*)

Table of Contents

List of Figures

List of Tables

Abstract

Visual information plays a very important role in interacting with the real world. In the field of computer science, image processing algorithms inspired by human visual information systems have been widely studied. Traditional research topics include optical flow algorithms, which extract a motion vector between images using brightness constancy assumptions, and Histogram of Gradient (HOG) algorithms that detect objects by learning the boundary values of pixels. State-of-the-art, deep learning techniques in artificial intelligence research mimic neural structures in the human brain, and are an area of active development. However, small computers such as Micro Controller Unit (MCU) and Internet of Things (IOT) devices in industry create computational time restrictions to utilizing the state-of-the-art techniques.

In this dissertation, a light-weight object detection algorithm is designed to perform efficiently, even in situations where computer performance is limited. It reduces the computational load and increase the object detection performance by combining the existing traditional techniques and the latest techniques. Rather than using the entire state-of-the-art deep learning model, optical flow vectors are calculated to estimate moving objects in the beginning of the detection process. Then, in the last phase of the algorithm, machine learning and neural network models including deep learning are applied to extract image features for object classification.

The proposed method begins by introducing proposed feature point reset function, moving window and target estimator function based on the Lucas-Kanade method for sparse optical flow estimation. These functions improve performance when tracking moving objects and reduce computational time greatly. Next, it classifies the moving objects in the area extracted from the proposed optical-flow-based algorithm. When applying learning-based classification models, training data is a very important factor for the performance of the learning models. Training data relationships and confirmed coverage effects are determined among datasets by performing experiments to evaluate test datasets. In the end, a HOG+SVM machine learning model, You-Only-Look-Once version 5 (YOLOv5), and simple CNN-based Small Person Detection (SPD) deep learning models are used to classify the moving objects with the optical-flow-based algorithm. A learning-based classification model is applied to the object-existing Region of Interest (ROI) which is extracted from the optical-flow-based model. The optimal parameters are found by adjusting the crop size and resize ratio parameters according to the size of the ROI. Proposed CNN-based deep learning model SPD is operated on the

proposed separated ROI method which is pre-process step. ROI is separated according to threshold of aspect ratio from the experiment analysis. Object detection performance such as FPS, recall, and precision are compared for the machine learning technique the HOG+SVM, HOG Multi-scale, OF+HOG+SVM, YOLOv5n, OF+YOLOv5n models, and the proposed deep learning technique OF+SPD model. The comparison results shows better performance in computational speed and object detection performance for proposed detection algorithm.

This work presents a pedestrian detection algorithm having high computational speed through sparse optical flow and a lightweight CNN-based neural network model. First, it employs an object localization function with a moving window detector, which mitigates the flaws of the sparse optical flow. Next, an SPD neural network model consisting of fewer CNN layers performs object classification. A performance comparison evaluation experiment of the proposed method was conducted upon a CPU device, which showed good detection performance and faster computation speed compared to existing object detection algorithms. This indicates that the proposed pedestrian detection algorithm could be applied using only a CPU device without the need for an expensive GPU device. The proposed algorithm can serve as an economic and efficient algorithm for real-time pedestrian detection in industrial areas where low-performance computing devices need to be used.

Key words : moving object tracking, optical flow, moving window, target estimator, object detection, cross test, influence graph, deep learning, object classification, pedestrian detection, machine learning, neural network, lightweight algorithm

Chapter 1

Introduction

In recent years, there has been increased interest in developing and applying object detection systems using optical flow and deep learning (Cheng and Han, 2016; Liu et al., 2020; Prasad, 2012; Zhao et al., 2019; Zou et al., 2019). Moving object tracking and detection methods are applied to wide research fields including surveillance systems, traffic monitoring systems and recognition of workers and pedestrians (Aslani and Mahdavi-Nasab, 2013). Object detection is the study of how to implement and automate the tasks that human visual systems can do in the field of computer vision. In this research field, image or video data is collected, processed, and analyzed, and various topics in each process are being studied.

For example, there is software in vehicles that analyzes whether there are other vehicles or people around the vehicle by receiving and processing images taken around the vehicle. From the analysis, it is possible to obtain location and motion information of surrounding objects for preventing collisions. The latest object detection models adopt a deep neural network structure, and various application cases arising nowadays with the development of new technology. Despite the development of object detection models and their wide application cases, various issues still exist. In this vein, it is discussed why this paper proposes a computationally efficient object detection algorithm.

1.1 Why efficient pedestrian detection?

It is important to track and detect moving objects in many applications, such as surveillance cameras and unmanned vehicle cameras, to track workers in factory environ-

ments to ensure personnel safety and to recognize pedestrians. Conventional algorithms to track and detect moving objects include frame difference algorithms, background subtraction algorithms, optical-flow-based algorithms, and learning-based algorithms. Deep learning-based algorithms are computationally complex, require sufficient training samples, and are not suitable for real-time processing on a board without graphical processing units (GPUs). In addition, conventional moving-object detection and tracking algorithms cannot effectively detect and track targets obscured by obstacles within the images or when an image is distorted by camera vibrations.

In the industrial area, there are cheap, low-computing power CPU devices, and various uses of devices in vibration-prone environments, with distortion-prone camera frames. One particularly desires a robust, time and financial cost-effective algorithm in industrial fields where various variables exist. Moving objects are targeted for surveillance systems of traffic flow, detecting pedestrians, and workers in industrial fields. Therefore, it requires the algorithms for moving object tracking and classifying whether a tracked object is a person or not, with improved computational speed and detection performance required to the industry. When using a classification model, it is important to learn features such as the distortion of fish-eye camera images used for capturing wide areas in industry.

1.2 Motivation and objective

To address this problem, this paper proposes a moving object tracking and detection algorithm that can be applied to various applications. The proposed algorithm can efficiently eliminate the noise from camera vibration in the image frames and perform continuous tracking for moving objects obscured by obstacles. Specifically, the conventional sparse optical flow algorithm (Lucas-Kanade, LK) is enhanced to detect and track multiple moving objects at a low computational cost. Moreover, the corner extraction algorithm (Shi–Tomasi) is used to track feature points to detect and track moving targets. A moving window detector and memorized estimator are used to enhance the detection performance, ensure the robustness of the algorithm to noise, and improve the worker safety. In particular, the moving window detector uses the window memory at each feature point as the window size and detects and tracks the moving target by evaluating whether the feature point is noise or a moving object. The location history of the detected points is memorized, and a halted or invisible target is identified from

the location history of the feature points. Subsequently, the estimator decides whether the target state is maintained.

It needs to classify moving objects in the ROI extracted from the proposed optical flow method. Before applying learning-based classification models, the effects of the different types of training dataset is studied. Recently, there have been many complex and diverse structures and high-performance deep learning models. These models require a large amount of learning data and learning time. As mentioned above, instead of studying the structure of the model, techniques such as the amount of efficient learning data, learning epochs, and transfer learning are being studied as elements outside the model (Zhu et al., 2012). From this point of view, distorted image datasets are used to examine characteristics with relationship among the datasets. Using the proposed strategy, the performance of model could be increased by changing only the learning data composition.

After studying on the effects of the training data, a zoom and classification function are developed, even in the region of features which are hardly trained, including faraway and small objects. It do not only reduce computational cost, but also develop performance or additional functionality. Hard cases in the data selection experiments can be solved with proposed optical flow method and machine learning classifier such as the HOG+SVM or YOLOv5 models. ROI extracted from the optical flow object tracking method will be zoomed in various ratios like 0.7 or 1.7, according to the tracked ROI size. This adaptive resizing method will be helpful to keep object features in the frame when using object classification models. Various experiments are conducted to examine the optimal crop and resize parameters, and which classification models are effective. The objective is to find a model with an efficiently reduced computational cost and improved detection performance by finding the optimal parameters.

1.3 Organization of dissertation

The organization of this dissertation is as follows. Chapter 2 describes the background for this research, including various object detection applications of optical flow, machine learning, and deep learning methods. Chapter 3 introduces an algorithm to detect and track the moving object using sparse optical flow and other developed functions. Chapter 4 provides an efficient strategy for selecting training datasets, in which the object detection performance is maintained. Chapter 5 explains the overall process of pro-

posed optical flow and machine learning method, which detects and classifies objects with lower computational cost and better performance compared with other traditional methods. Finally, Chapter 6 presents a summary of the whole dissertation, conclusions from the results, and discussions of potential future work improving proposed method.

Chapter 2

Background

In recent times, object detection has been extensively researched (Zou et al., 2019). The difficulties and challenges in object detection problems must be considered, including issues such as improving the speed of the detection model so that it can be used in real-time applications, various rotation angles and sizes of objects, dense and occluded object classification, and object position estimation. In other computer vision problems, there are also issues regarding images under varying camera views, illuminations, distorted camera lenses, and noise-filled environments.

Object detection suffers from restrictions depending on the detection target, operating environment, and purpose. The detection model could be operated in a surveillance system that monitors traffic flow, pedestrians, and workers in industrial sites. In this case, an optical flow method for extracting pixel motion information between consecutive frames may be advantageously applied. While considering the background, various types of optical flow methods and advanced techniques will be examined, and limitations and improvements will be analyzed. First, I consider tracking the moving object and classifying it through a learning-based model. The learning-based object detection model is particularly influenced by the features of the learned image. Accordingly, it is necessary to examine the types and characteristics of conventional training datasets as well as studies that improve the performance of the model by applying data.

To solve object detection problems that cannot be solved through an efficient learning dataset strategy, optical flow and a learning-based detection model can be combined. For this, various machine learning and deep learning models will be examined, including each of their characteristics, limitations, and improvements. Based on these

examinations, an improved object detection model can be designed by combining an optical flow method and a learning-based classification model, enabling to reduce the computational cost of the deep learning model and mitigate the window detector size problem of machine learning. Finally, such an approach can be robust even in distorted datasets, reduce the amount of computation consumed by object localization and classification, and detect moving objects regardless of size.

2.1 Moving object tracking

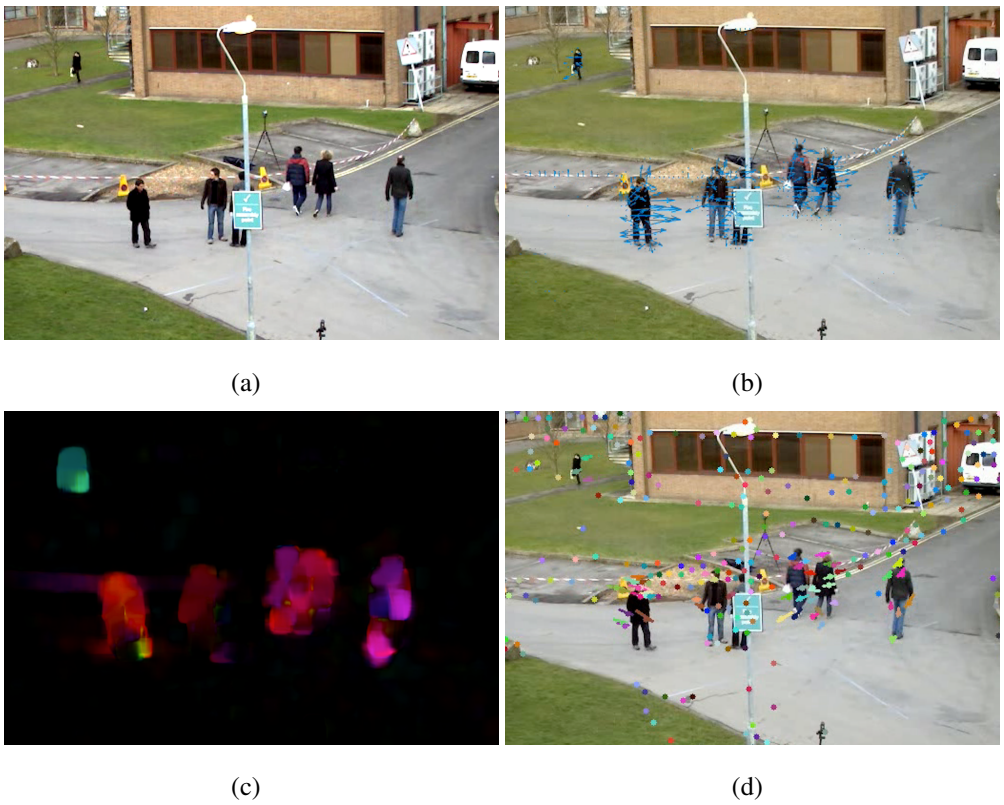
In recent years, methods to track and detect moving objects have been widely studied. Before the development of learning-based algorithms, methods based on the optical flow, frame difference, and background subtraction were typically used. Moving-object tracking algorithms such as dense optical flow, frame difference, and background subtraction are used to determine the movement of objects and were successful in accurately detecting objects. However, these algorithms involve substantial computational time cost. Existing moving-object tracking algorithms suffer from limitations with regard to computational time and can only work in limited environments.

2.1.1 Optical flow-based model

Optical flow is the pattern of motion in the image, represented by the direction and distance distribution of pixels between the previous frame and the next frame (Horn and Schunck, 1981). It assumes that the pixel intensity of an object moving between consecutive frames remains unchanged and that neighbouring pixels have similar motion. It is a crucial consideration in moving-object tracking because optical flow allows to determine how much the object moves in the image and in which direction.

There are two ways to calculate optical flow: sparse optical flow, which calculates only a few pixels, and dense optical flow, which calculates the entire image's pixels. The Gunnar–Farneback algorithm is a representative moving-object tracking algorithm among dense optical flow methods (Farnebäck, 2003). It calculates the optical flow using all pixels of the image, so there is no need for a separate feature point to track. However, it has the disadvantage that the speed is low because the optical flow vector is calculated using all pixels.

A representative example of the sparse optical flow method is Lucas–Kanade algorithm



(a) (b)

(c) (d)

Figure 2.1: Samples of representative optical flow methods (modified from (Horn and Schunck, 1981; Farnebäck, 2003; Bouguet et al., 2001)). (a): Input image. (b): Horn-Schunck method. (c): Farneback. (d): Lucas-Kanade.

(Bouguet et al., 2001). In this algorithm, the motion vector is computed using a small 3×3 window patch, under the assumption that neighbouring pixels move similarly. Unlike the dense optical flow method, the Lucas–Kanade algorithm calculates an optical flow vector by utilizing only feature points in an image. The optical flow vector is calculated by utilizing feature points such as corner points, which has the advantage that the processing speed is very high compared to using all pixels. It focuses on speeding up the processing of moving-object tracking algorithms so that they can operate seamlessly even on low-performance equipment. Therefore, I intend to develop and propose an object tracking algorithm based on such a sparse optical flow method.

Various optical flow techniques for moving-object detection have been proposed. Moving-object tracking was realized using optical flow and motion vector estimation (Agarwal et al., 2016; Kale et al., 2015; Aslani and Mahdavi-Nasab, 2013), and the approach was noted to exhibit a strong object tracking ability for the same scene in various views. To perform real-time object detection and tracking, feature extraction was conducted

using the pyramid LK optical flow, as a sparse optical flow technique (Wang and Yang, 2018). To enhance the tracking accuracy, the corners were detected for tracking, the sub-pixel corners were determined, the video in each frame of the image layered in the image pyramid was examined to calculate the optical flow at the top corner, and the next pyramid was considered the starting point of the pyramid. This process was repeated until the bottom pyramid image. Notably, object recognition can be supplemented in a moving camera situation with technological advancements.

Certain researchers attempted to perform object recognition using the optical flow based on a camera attached to a moving vehicle (Kim and Kwon, 2016). Several movements were captured within the scene, and the ego motion was separated from the background. However, when the scene moved instead of a fixed camera, many false positives occurred. The authors attempted to relax the stationary cameras restriction by using traditional moving-object detection methods and introducing additional steps before and after the detection. For cameras to be attached to heavy equipment, a fisheye camera with a wide-angle range can be used. Certain researchers developed an approach to track pedestrians and cars in fisheye images (Bertozzi et al., 2015), using low-cost sensors and four fisheye cameras with a wide range. Unwarping technique is used to pre-process distorted images, followed by object classification and tracking. A novel equipment design and sensing system (Safety 360) was developed to provide equipment operators with a surround-view (Teizer, 2015a). However, there is a problem that the background subtraction is needed, to create an algorithm that works better by combining optical flow with cameras in various situations. The following section introduces and limits to the background subtraction algorithms used in combination with optical flow.

2.1.2 Background subtraction problem

There is a fatal flaw in the object tracking model based on optical flow. In the previous survey, it is difficult to extract optical flow vectors for a desired target in images acquired from cameras that are ego-motioned, such as cameras attached to moving vehicles. To overcome this issue, there are various background subtraction methods (BSM) that distinguish moving backgrounds from foregrounds (Piccardi, 2004; Brutzer et al., 2011; Garcia-Garcia et al., 2020; Sobral and Vacavant, 2014). Figure 2.2 shows sample images of the results after applying background substitution methods (KaewTraKulPong and Bowden, 2002). Approaches based on frame difference, opti-

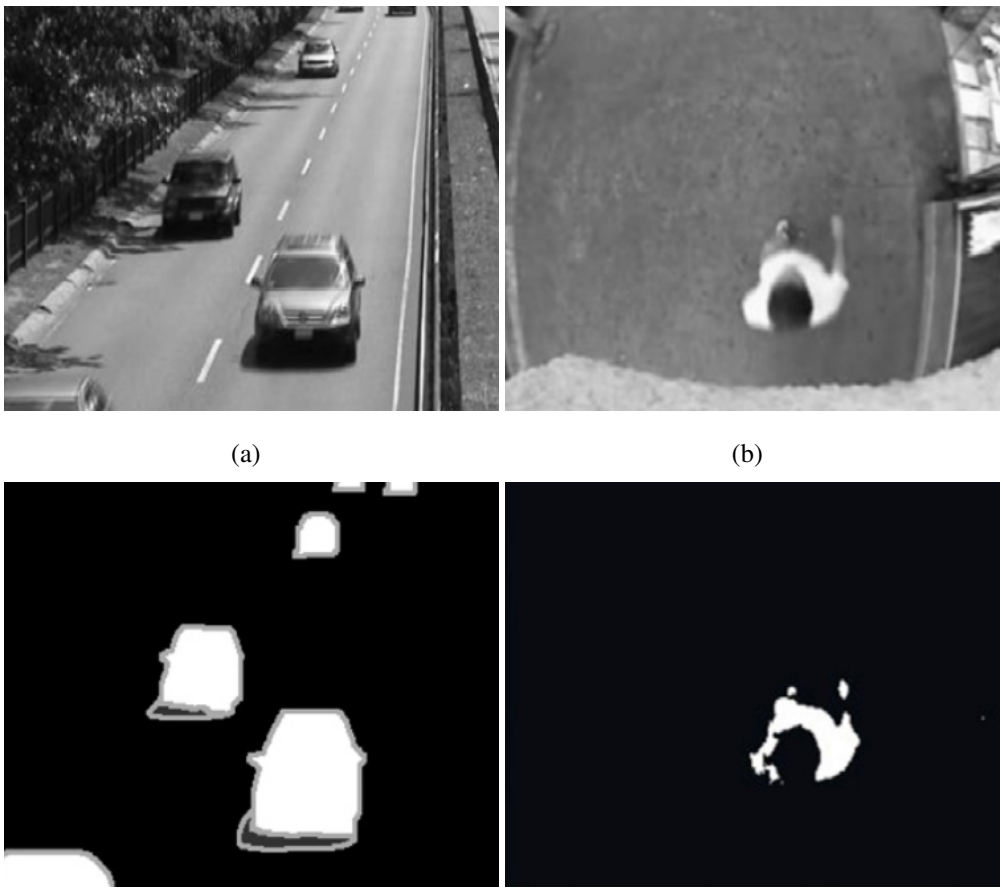


Figure 2.2: Sample images of applying background substitution method (modified from (Garcia-Garcia et al., 2020)).

cal flow, machine learning, and deep learning techniques are being studied in BSM. These techniques further use filters such as Gaussian averaging, temporary median filter, mixture of gaussians, kernel density estimation (KDE), sequential KD application, co-occurrence of image variations, and linear and statistical computations. The background subtraction technique should be appropriately selected based on how to limit the target object to be extracted from the background motion depending on various average processing and filter processing techniques and the computational speed of the computer.

Object detection performance for a complex background can be enhanced using the optical flow. Object detection may not be effective when the frame difference technique is integrated with the optical flow technique. Certain researchers developed algorithms to perform background modeling tasks, using edge detection to solve problems (Chen

et al., 2016). Another study attempted to solve the background subtraction problem in a moving camera environment based on optical flow (Diamantas and Alexis, 2018). The corresponding approach obtains motion information of the camera from a sensor inside the camera. After examining the correlation between camera motion information and optical flow parameters, it is possible to select optical flow vector information for the desired target object from the moving background from the multi-linear regression model. In this approach, parameters are optimised for the root environment where city buses roam.

Another study aimed at object segmentation based on the dense optical flow (Kushwaha et al., 2020; Chan, 2013). Segmentation problems based on optical flow involve the task of distinguishing them from moving objects more precisely in a moving background. Accordingly, this approach labelled each pixel to distinguish between foreground and background, and set the threshold to the magnitude of the motion flow vector to define moving pixels. The proposed technique calculates the homography matrix, stabilizes the camera motion, and performs statistical background extraction modeling based on the single Gaussian background modeling approach. By utilizing the dense optical flow vector, precise background subtraction can be performed by comparing the change in motion vector magnitude for all pixels between previous frames. However, as such a dense optical flow involves operations on all pixels of the image, it is computation-intensive.

2.1.3 Application of moving object tracking

Moving-object detection techniques can also be applied for safety evaluations. Certain researchers used computer vision technologies to measure the vibration of buildings (Chou and Chang, 2021). This approach could help evaluate the condition of the building, and minute movements of the building were detected using several sensors. The efficiency of the existing motion extraction methods was compared, using commercialized cameras and the LK optical flow instruments as experimental equipment. In general, when heavy equipment is operated and large vibrations and physical forces are applied to the ground at construction sites, shaking occurs throughout the structure. After pre-processing the image, this shaking can be monitored through monocular vision and detection of the obstacle around which the shaking occurs (Son et al., 2017).

Object tracking algorithms have also been applied for ensuring personnel safety during the operation of heavy equipment such as unmanned excavators (Rasul et al., 2021;

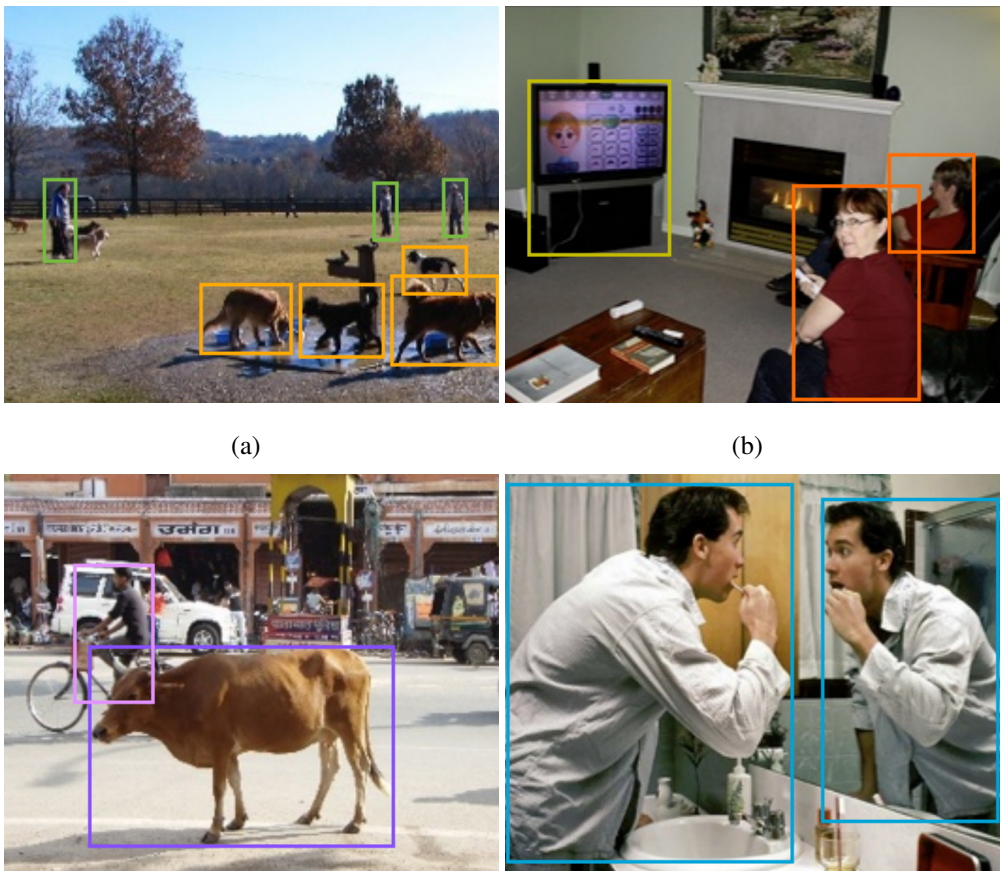
Leger et al., 1999; Son et al., 2019). Motion detection and object tracking were performed using Velodyne VLP-16 light detection and ranging sensors. Moreover, motion predictions could be performed by analyzing the physical movements and estimating the activity areas. This paper presents the proposed technique based on the optical flow (Chae and Yoshida, 2010; Teizer et al., 2010; Teizer, 2015b; Chi and Caldas, 2012; Ishimoto and Tsubouchi, 2013). Moreover, a stereo vision sensor-based monitoring system using more than one image can help distinguish various objects and represent them as three-dimensional information to ensure accurate monitoring (Chi and Caldas, 2012; Ishimoto and Tsubouchi, 2013). Specifically, this technique can provide the three-dimensional geometry, high-resolution image correction, and color and textural information to enhance the monitoring accuracy. However, for various lighting conditions, low resolution and high-performance camera systems may be required. Another approach can recognize and track objects by analyzing the behavior of workers at the construction site (Gong and Caldas, 2011). The moving objects are detected based on the optical flow, the joint probability around the detected objects is calculated using the naïve Bayesian model, and the workers' actions are categorized to track and recognize the objects.

Moreover, an object detection study was performed to clarify the influence of images distorted in environments such as those involving movement of the background, camera shaking, and rotation (Fu et al., 2016). Moving-object detection could be performed using images recorded at large distances, such as top views (Zhu et al., 2020). Images with camera equipment movement or noises were pre-processed through background compensation. From this, it can be devised a way to make the object detection model more robust in images acquired from distorted or high-position cameras. In the following data for object detection section, various data format and data processing studies are investigated to examine the effect on object detection according to the characteristics of the image as mentioned above.

2.2 Data for object detection

2.2.1 Data source

The Microsoft Common Objects in Context (MS COCO) dataset, samples of which are shown in Figure 2.3, is a public dataset provided by Microsoft for computer vision purposes such as object detection, segmentation, and keypoint detection (Lin et al.,



(a) (b)

(c) (d)

Figure 2.3: Sample images contained in MS COCO dataset with label boxes (modified from (Lin et al., 2014)).

2014). Several recent papers on object detection use the COCO 2017 dataset for model performance evaluation. Object detection libraries based on neural networks, including TensorFlow and PyTorch object detection API, provide pre-trained models with COCO 2017 datasets. The COCO 2017 dataset has a training dataset consisting of 118,000 images, a validation dataset consisting of 5,000 images, and a test dataset consisting of 41,000 images from 80 classes. The MS COCO dataset has multiple objects in one image, more complex data than other datasets, and stricter standards in mAP. In addition, there is an MS COCO Captions dataset in which caption information containing contextual descriptions is added for each image from the MS COCO dataset (Chen et al., 2015). In deep learning-based object detection models, these MS COCO datasets can be used as base datasets for learning basic object forms in the process of examining dataset features.

The PASCAL Visual Object Classes Challenge (PASCAL VOC) dataset is one of the

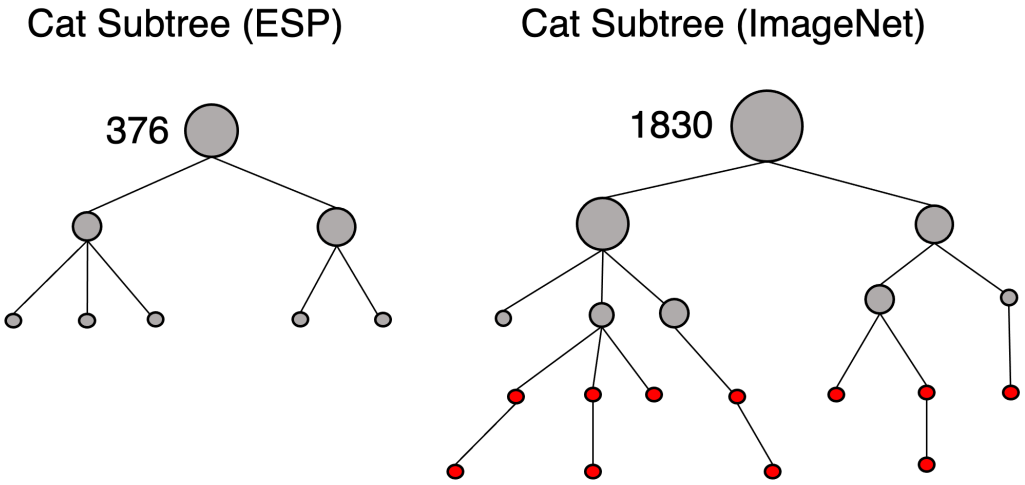


Figure 2.4: Comparison of the cat and cattle subtrees between ESP and ImageNet (modified from (Deng et al., 2009)).

representative datasets used in object detection competitions for computer vision fields such as object detection, segmentation, action classification, and large scale recognition (Everingham et al., 2010). The PASCAL VOC 2012 dataset comprises training and validation data, which are 11,530 images including 27,450 object annotations and 6,929 segmentations. PASCAL VOC, MS COCO, and Google Open Images datasets are representative datasets that are widely used for model performance evaluation purposes, and many detection and segmentation deep learning packages are pre-trained and distributed based on those datasets. PASCAL VOC uses XML format as an annotation file format that contains object information in an image and has 20 object classes. The MSCO dataset uses the JSON format as an annotation file format and has 80 object classes. The Google Open Images dataset uses CSV format as an annotation file format and has 600 object classes. The annotation file contains the date when the data was created for a single image, data license, id, segmentation, bounding box of objects contained in the image, pixel area, size, category, and other data.

The ImageNet dataset is a large amount of open-source public image database (Deng et al., 2009). Node size is the portion of images contained within. The red-colored circle denotes ImageNet's own categories. The dataset consists of 1000 classes and contains more than one million data. Approximately 1.2 million are used for training

and 50,000 for validation. The amount is so large that the learning dataset capacity is approximately 138 GB and the verification dataset capacity is approximately 6 GB. In particular, deep learning researchers interested in classification problems tend to utilize ImageNet datasets consisting of approximately 1000 photos per class and with vast class types. The ESP is a dataset created by people called ESP gamers, who actively participate in image labeling through games (Von Ahn and Dabbish, 2004). In Figure 2.4, when ImageNet is compared to ESP datasets, ImageNet datasets, which boast a vast class type, are more stratified than ESP datasets and have more data volumes every quarter of the subtree.

Fisheye cameras can cover the blind spots with a wide angle view, but produce strongly distorted images (Miyamoto, 1964). There have been studies to calibrate fisheye images for rectification (Chao et al., 2020; Xue et al., 2019; Yin et al., 2018). Adaptation or rectification process is often required to obtain real physical properties of image objects. Yet a deep learning model has been tested for the fisheye camera environment (Blott et al., 2018), and without rectifying the image directly, mapping or augmentation methods for the distorted images have been applied. A parking slot detection algorithm has been tested using fisheye camera images (Xu and Hu, 2020). They collected the dataset under various environments of parking slot with sunny, cloudy and rainy weathers. However, acquiring many types of environmental data could be inefficient and it needs extra time and cost (Sun et al., 2017). Thus, an appropriate selection of data or a sufficient amount of data is needed for high accuracy, depending on the property of target subjects (Cho et al., 2015). While the Fish-Eye dataset reflects image distortion due to camera lens characteristics, there are a large-scale hierarchical multi-view RGB-D object dataset photographed for the same places and objects at various points (Lai et al., 2011). Since image datasets obtained according to camera measurement characteristics, photographing methods, and environment have various characteristics, using these datasets will be easier to learn image features that are not well learned with deep learning models.

2.2.2 Data processing

At present, state-of-the-art deep learning models have been applied in various areas. Various deep learning-based moving-object detection algorithms consisting of deep convolution networks such as YOLOv5s and R-CNNs have been developed. These methods are highly sensitive to changes in the background brightness, which can in-

crease the probability of erroneous detection. Thus, the dataset used in model training has considerable impact on the performance of the model. In order to obtain the desired effect in a specific area, it is important to analyse the characteristics of the dataset to be used for model learning.

Deep learning models have been built using many complex and diverse structures. These models require a large amount of learning data and learning time. As mentioned above, instead of studying the structure of the model, aspects such as the number of efficient learning data, learning epoch, and transfer learning are being studied as elements outside the model that influence performance. Systems that relies on deep neural network models are the most efficient and accurate. However, there are still many blind spots, and there is a limit to object recognition capability. Accordingly, as image processing algorithms based on deep learning have been developed, acquiring training data of high quality and large size becomes a key differentiator. Driver assistance systems using vision sensors such as monocular cameras and stereo cameras, which are the focus of considerable research, are instrumental in securing the driver's safety as well as the surrounding environment.

There are several deep learning models that are widely used in applications to detect labeled objects. As a state-of-the-art deep learning model used for object detection, YOLOv5 provides high detection performance (Redmon et al., 2016; Huang et al., 2018; Shafiee et al., 2017). MobileNet with SSD reduces computing cost considerably compared to large-sized neural models (Liu et al., 2016; Chiu et al., 2020). These training models consist of many layers for extracting feature maps and require a huge training dataset. However, collecting for these training datasets be time-consuming and expensive. Furthermore, efficiently selecting training datasets in machine learning is a challenge (Zhu et al., 2016; Moore and Lewis, 2010; Mehryary et al., 2016).

In recent times, state-of-the-art deep learning models have been applied in various areas. The dataset provided for the model training substantially affects the learning performance. In order to obtain the desired effect in a specific area, the characteristics of the dataset to be used for model learning must be analyzed. In one study, data-augmentation-based data imputation was used to create a strong monitoring model for situations where a specific sensor failed in a system that monitors the structural state of a building (Hou et al., 2022). They augmented the dataset containing characteristics of sensor failure using models such as Generative Adversarial Networks (GAN) and Long Short-Term Memory (LSTM). Using this method, the deep learning model

becomes more robust to noisy sensor data by learning the characteristics of the data acquired from a faulty sensor. A deep learning model was used to examine the big data in the medical field through a visualization algorithm (Qiu and Lu, 2021). Another deep learning model used various types of sensor data to identify the inlet pipe blockage level in a centrifugal pump (Kumar et al., 2021).

An example of chained technology is a Deformable Part-based Model (DPM), a model that recognizes objects based on the decomposition of a kinematically-inspired object and a meticulously designed representation represented by a graphic model. It uses discriminant learning of graphical models and can build models of high precision performance for object classes. Manually engineered representations, along with models trained on shallow discriminants, have been one of the best-performing paradigms for tasks related to object classification. However, in the past, Deep Neural Networks (DNNs) have been used as powerful machine learning models. DNNs have notable differences from the existing approaches to classification. They have the ability to train more complex models than shallow ones. This expressivity and robust training algorithms allow them to learn powerful feature extraction without manual-design features. In addition, DNNs have been validated in terms of function and effectiveness in the ImageNET classification task across thousands of classes. They not only classify objects, but also track the location accurately, and can be used in the task of object detection. Even with limited computing resources, DNNS are an appropriate technique because they involve classifying classes with diverse sizes and finding objects in the same image.

Furthermore, deep learning models have been widely used in applications to detect labeled objects. As a state-of-the-art deep learning model used as an object detection model, You-Only-Look-Once version 5 (YOLOv5) provides high detection performance (Redmon et al., 2016; Huang et al., 2018; Shafiee et al., 2017), and MobileNet with Single Shot Multi-Box Detector (SSD) substantially reduces computing cost, when compared with a large size of neural models (Liu et al., 2016; Chiu et al., 2020). These training models consist of multiple layers for extracting feature maps and require a large training dataset. However, collecting data for these training datasets can be time-consuming and expensive. There have been studies focused on efficiently selecting training datasets in machine learning (Zhu et al., 2016; Moore and Lewis, 2010; Mehryary et al., 2016). Therefore, it aims to present an efficient learning strategy to reduce the amount of data by appropriately screening training data to be learned in a

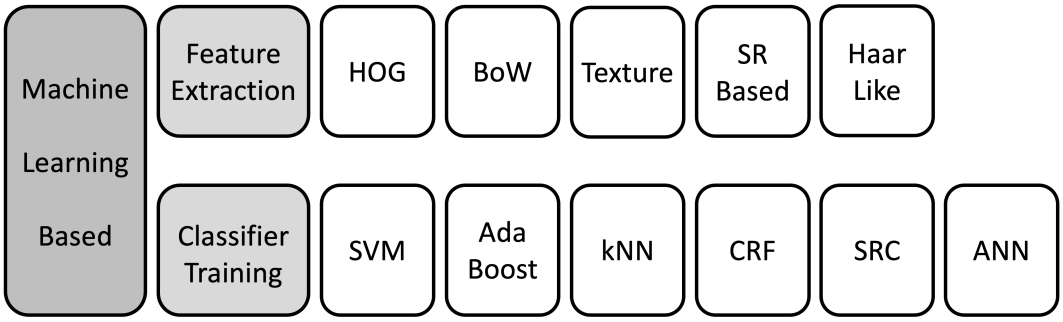


Figure 2.5: Taxonomy of methods for machine learning based feature extraction and classifier training methods (modified from (Cheng and Han, 2016)).

complex deep learning model structure. In the deep learning model in various environments, I propose a data classification created using the characteristics of the sensor and data utilization model that compensate it.

2.3 Models for object detection

2.3.1 Machine learning-based model

An optical flow-based object tracking model can be used to detect areas where moving objects exist. However, determining whether the object is human can be more effective at preventing false alarms in detecting target objects as compared to using optical flow by itself. Furthermore, such an object detection model can extract useful information for the target objects and be used to ensure human safety and pedestrian traffic at industrial sites.

Before studying such cases, various machine learning-based object detection methods are reviewed (Cheng and Han, 2016). Figure 2.5 illustrates the crucial processes in the performance of object detection: feature extraction and classifier training. Among feature extraction methods, there are five types of feature extractor: Histogram of oriented gradients (HOG), bag-of-words (BoW), texture, sparse representation based (SR), and Haar-like features. In classifier training methods, there are six types of classifier training methods: support vector machine (SVM), AdaBoost, k-nearest-neighbour (kNN),

conditional random field (CRF), sparse representation-based classification (SRC), and artificial neural network (ANN).

Before the studying cases, I review various machine learning-based object detection methods in survey In Figure , there are mainly crucial steps that play important roles in the performance of object detection: feature extraction and classifier training. In feature extraction methods, there are typical five types of feature extractor including Histogram of oriented gradients (HOG), bag-of-words (BoW), texture, sparse representation based (SR), and Haar-like features. In classifier training methods, there are typical six types of classifier training including support vector machine (SVM), AdaBoost, k-nearest-neighbor (kNN), conditional random field (CRF), sparse representation-based classification (SRC), and artificial neural network (ANN).

Feature extraction involves mapping from image pixels to high-dimensional data space. Primarily, object detection performance is affected by the feature space of the input image, so feature extraction is very important. HOG features represent objects using the distribution of gradient intensities and orientations in spatial regions, and they better capture the edges of local shape information of objects (Dalal and Triggs, 2005). The BoW feature model has the advantages of simplicity, efficiency, and invariance under camera view changes and background noises (Fei-Fei and Perona, 2005). The texture feature aims to calculate local density variation and patterns on the surface of an object, which is important when identifying textural objects such as airports, buildings, urban areas, and vehicles (Eikvil et al., 2009). SR-based features are used in hyperspectral image denoizing and classification, and their core idea is to sparsely encode high-dimensional original signals by structural primitives in a low-dimensional manifold (Chen et al., 2011). Haar-like feature is mainly used for face detection and is now widely used for object detection (Viola and Jones, 2001). It is composed of a set of rectangular regions that are positive- or negative-weighted. The features are calculated with a sum of pixel values in rectangular regions according to their corresponding weights. Compared to HOG and Haar-like features, other methods incur higher computational costs because of usual usages in high-resolution images, and they require more precise calculations. Therefore, a representative feature extractor can be applied to HOG or Haar-like features for accurately detecting objects.

The kNN classifier is a conventional method for classifying data into n classes, and it is effectively used in industrial fields (Cover and Hart, 1967). However, it is unsuitable when dealing with complex object features such as distortion, rotation, and varying

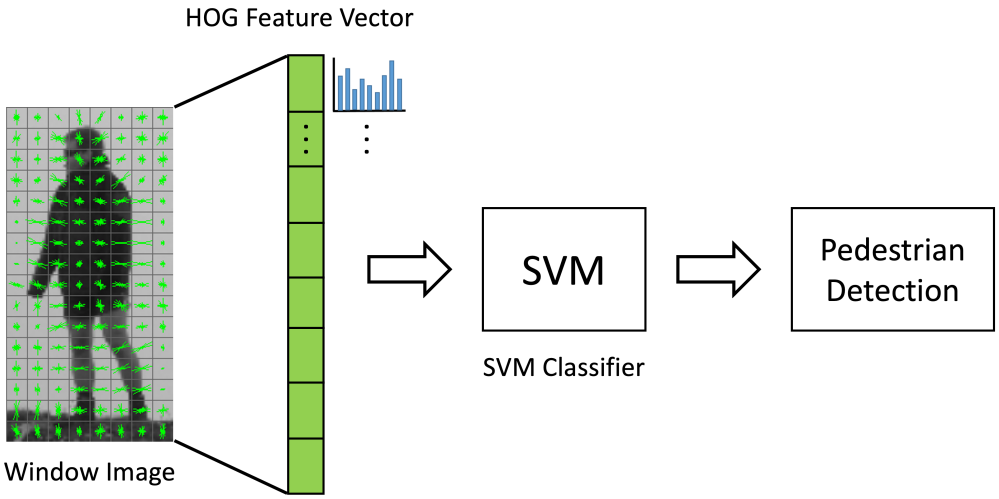


Figure 2.6: Schema diagram of pedestrian detection using HOG feature and SVM classifier (modified from (Dalal and Triggs, 2005)).

sizes. The AdaBoost algorithm is widely used as a machine learning-based algorithm that combines weak classifiers to make a strong classifier by adjusting the weights of training data (Freund et al., 1996). It is usually applied with a Haar-like feature-based object detection model to classify Haar-like features. Finally, SVM is one of most popular and effective machine learning algorithms for solving classification problems such as pedestrian classification (Vapnik, 1999). It has been widely used in object detection models such as object recognition, road extraction, change detection, multi-class object detection, and pedestrian detection (Inglada, 2007; Das et al., 2011; De Morsier et al., 2013; Bai et al., 2014). The SVM classifier is usually applied with HOG features in object detection models. In Figure 2.6, such a HOG+SVM model can achieve simple and powerful detection performance, but SVM has limitations with regard to variance of objects size, rotation, and distortion. Consequently, this model results in high computational cost to get a high detection performance when used by itself, because the SVM window detector has to slide in detailed movement.

In one approach, an optical flow is detected in the setting ROI, and an object classification is attempted when the optical flow vector value exceeds a threshold (Do, 2020). After the detection line is set in the preset ROI and the pixel value within the detection lines exceeds the threshold, the optical flow is calculated around the area. If the calculated optical flow value exceeds the threshold, the HOG feature is calculated with a determined size near the area and input into the SVM classifier to determine whether

the object is a person. The limitation of this approach is that the area of interest where the optical flow is to be calculated is fixed in advance, so it is not possible to prepare for the case where the object appears elsewhere. The proposed method computes optical flow based on feature points of moving objects in the entire image. Furthermore, the presented approach only reduces the area in which HOG-based classifiers are calculated, and the performance of the classifiers themselves is not improved. The HOG feature that is input to the classifier can be improved by adaptively resizing the ROI to the input dimension of the learned SVM classifier from the HOG+SVM.

Another approach performs pedestrian detection by learning optical flow vector information and HOG feature information together with the SVM classifier (Ramzan et al., 2016). This technique computes a dense but simple optical flow motion vector and learns it using a HOG feature. However, as the optical flow motion vector is used, the proposed technique may be adversely affected by the change of light. In certain situations, the performance is improved compared to the model that detects objects using only the existing HOG feature, but the computation requirement increases because the optical flow motion vector is calculated more densely. In order to minimise optical flow calculation, sparse optical flow method is adopted in the proposed lightweight algorithm. Another approach uses frame difference techniques to find pixels that change on the screen, set the area where pixels are clustered as ROI, and use a HOG detector (Hariyono et al., 2014). This study contributes to reducing the amount of computation cost, but only reduces the area in which the HOG detector is to be computed, and there is no contribution affecting the detection performance of the HOG detector.

In Figure 2.7, there is a study to detect large-sized objects in an image with HOG features and fixed-dimensional SVM classifiers while reducing the input image resolution (Li et al., 2019). The SVM classifier learns a fixed-dimensional feature vector and classifies objects. The disadvantage of the SVM classifiers is that the size of the object that is close to the image cannot be contained in a fixed HOG feature window, making it difficult to detect using the SVM classifier. To overcome this challenge, the technique adjusts large-sized objects to the size of the window detector, i.e., the size of the SVM input vector dimension, in an image pyramid manner. While these methods are effective in detecting large objects, they are not suited to detecting small objects. If a small object needs to be found by enlarging the image as in this method, the window detector will have more areas to process and incur a huge amount of computation. Instead, only a small portion of the area must be enlarged to fit the window detector size, so that the

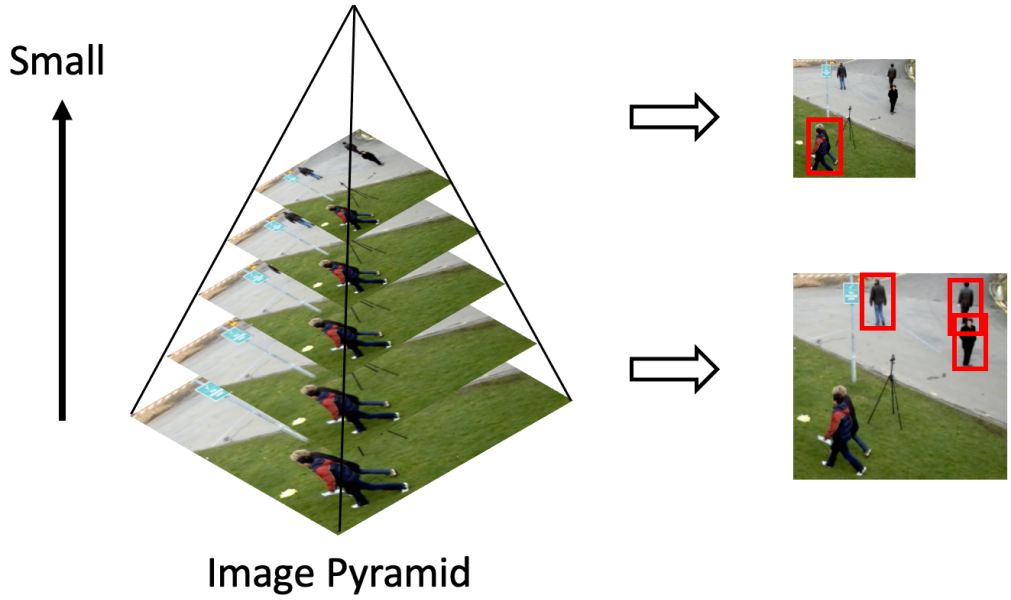


Figure 2.7: Schema diagram of HOG multiscale method (modified from (Li et al., 2019)).

small object is detected better and the detection computation is reduced.

There are various classifier models based on machine learning approaches. The classifier is used to distinguish target objects from diverse classes and to capture more hierarchical, semantic, and informative representations for the objects. Classifiers such as SVM (Wang et al., 2015), AdaBoost (Freund and Schapire, 1997), and DPM (Felzenszwalb et al., 2009) are usually used. Among these classifiers, DPM is a model that flexibly combines deformation cost and target parts to handle severe deformation. It is deficient in low-level functionality designed with the help of graphic models and kinetically inspired parts disassembly. on the results of using these local feature extractors and easily learnable architectures are presented in the PASCAL Visual Object Classes (VOC) object detection competition, and real-time embedded systems have been designed with a low burden on hardware. The reason is that when searching to classify objects through the sliding window method, the generation of detection target candidates is duplicated, inefficient, and inaccurate. It also has the disadvantage of having to manually determine their labeling.

Machine learning techniques have also been used to detect and recognize objects. The motion of earth movement equipment was detected based on the vision at ground level (Roberts and Golparvar-Fard, 2019). Images were acquired from excavators or

dump trucks, objects were tracked using the convolutional neural network (CNN) deep learning model, and routes were extracted using the hidden Markov model (HMM). Notably, the HMM leverages trajectories to train a Gaussian mixture model, and the probability density function of each activity can be determined using support vector machine (SVM) classifiers. For real-time vehicle detection and tracking for gas station surveillance, an approach based on the Adaboost classifier and optical flow tracking was proposed (Xiang et al., 2016). Specifically, the Adaboost algorithm was used to train the classifier with Haar-like features extracted from positive and negative samples of the gas station vehicles. Optical flow tracking method was performed to extract the corner points of the vehicle areas and match the positions of these corners in the consecutive frames in real-time.

There are studies that propose a model that can recognize robust object extraction and tracking in any situation based on memory. Although these methods often have high recognition rates and can work in real time, they all contain objects of interest without extensive fake information and background, so there is a problem in that they rely on manually segmented data or labeled databases. In today's era that requires complex image data and more accurate and detailed object recognition, I am not only interested in classifying images but also accurately estimating the class and location of objects included in the image, which is the object recognition and detection technology. A major advance in object detection can be rapid, thanks to improvements in object representation and machine learning models.

Some studies propose a model that can recognize robust object extraction and tracking in any situation based on memory. Many advances have been made using technologies such as SIFT (Lowe, 1999), SURF (Bay et al., 2008), and ViolaJones (Viola and Jones, 2004). A method of handling feature extraction and selection in the visual images was proposed for detecting moving objects in video sequences (Kursun and Favorov, 2010), which only focuses on the objects of interest while removing fake information data. Although these methods often have high recognition rates and can work in real time, they all contain objects of interest without extensive fake information and background, so there is a concern that they rely on manually segmented data or labeled databases. In today's era that requires complex image data and more accurate and detailed object recognition, it is not only important to classify images but also to accurately estimate the class and location of objects included in the image, i.e., simultaneous object recognition and detection technology.

Thanks to improvements in object representation and machine learning models, there have been rapid advancements in object detection. However, some of these have the disadvantage of incurring considerable computational cost in training networks (Felzenszwalb et al., 2009; Szegedy et al., 2013). Advances in this field will have an immense impact on object detection technology, which can be considered as a learning system, as well as in developing neural network algorithms using successful neural networks and related learning systems. However, owing to large changes in viewpoint, pose, occlusion, and lighting conditions, it is difficult to accurately recognize an object, and it is necessary to locate the object. The object detection task aims to detect objects' locations in a given image as well as categorize them correctly. The pipeline of an existing object detection model can be mainly divided into three stages: information area selection, feature extraction, and classification.

Information area selection is related to the fact that various objects can appear anywhere in the image and have diverse proportions and sizes, so scanning the entire image using a multi-scale sliding window is the first thing to do. This exhaustive strategy can find every possible position of an object, but it has certain drawbacks. It is computationally expensive owing to the large number of candidate windows and too many duplicate windows. It is also difficult to expect good results if fixed types of sliding window are applied. The second step is feature extraction, which means that in order to recognise other objects, it is necessary to extract visual features with semantic and robust features. Typical examples are SIFT (Lowe, 2004), HOG (Dalal and Triggs, 2005), and Haar-like (Lienhart and Maydt, 2002) features. This is a technology created by simulating the behavior of recognizing objects in the human brain. However, it is impossible to fully describe all kinds of objects given the variety of shapes, lighting conditions, and backgrounds, and considerable manual intervention is required. From this perspective, deep learning-based object detection model is needed for being more robust in a variety of camera environments.

2.3.2 Deep neural network-based model

There are several concerns in the domain of data acquisition, such as computational cost, environmental property, and target objects. To overcome this, neural network algorithms such as DNN are being developed. They can learn more complex functions than shallow ones and do not need manual learning through strong discipline. Since the proposal of Recursive-Convolutional Neural Network (R-CNN), several improved

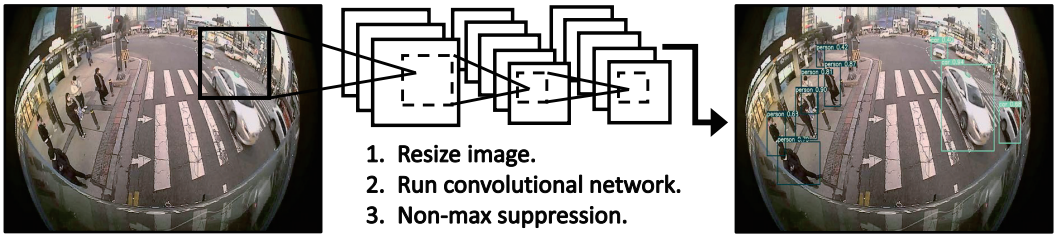


Figure 2.8: Simple diagram of YOLO model structure (from (Redmon et al., 2016)).

models have been suggested, including fast R-CNN that jointly optimizes classification and bounding box regression tasks (Girshick, 2015), faster R-CNN that takes an additional subnetwork to generate region proposals (Ren et al., 2015), and YOLO that accomplishes object detection via a fixed-grid regression (Redmon et al., 2016). Each of them improves detection performance compared to the conventional R-CNN and makes real-time object detection more achievable. In the Approaches section, the datasets used in the experiment, which are COCO, Woodscape, and a custom dataset obtained from the hydrogen bus with multiple fisheye cameras installed, are described. In the Experimental Results section, the contents of comparative analysis of the results according to the proposed learning method are explained.

Object detection, a sub-concept of computer vision technology, is an automation technique that identifies objects of interest in an image by distinguishing them from the background. This technique connects the categories of objects representing objects by setting a bounding box to detect the correct objects, and deep learning technology is mainly applied. Figure 2.8 shows YOLOv5, an object detection algorithm based on deep learning that is used in proposed method. YOLOv5 can ensure that the real-time update processing speed can keep up with the speed of change in the surrounding environment, which is one of the biggest challenges of autonomous driving (Qiu et al., 2017). It has high detection accuracy and high-speed detection. In general, YOLOv5 is largely composed of a backbone and a head. The backbone uses CSP-Darknet as the part that extracts the feature map from the image. This is similar to the configuration of YOLOv4. The head searches for the position of the object based on the extracted feature map, and algorithm sets the anchor box first and then uses it to create the final bounding box for detecting the object. YOLOv5 has four types of backbones, i.e., s, m, l, and x based on the model size. The present study used the s model, called YOLOv5s, which consists of a one-step detection algorithm and a two-step detection algorithm.

One study on deep learning models to detect small objects (Liu et al., 2021; Tong et al., 2020) claimed that with the recent development of deep learning models, the detection performance of objects of medium or large size has improved significantly. However, detection of objects as small as 20×20 size remains a challenge. In the study, the underlying CNN layer does not contain enough information about small object recognition present in the image. Furthermore, training datasets, which lack the distinction between background and foreground, degrade the model's small object recognition performance. Models including SSD are said to recognize objects of various sizes in an image, combining several feature maps (Liu et al., 2016; Fu et al., 2017). However, combining multiple layers of feature maps improves recognition performance for small objects, but a tradeoff occurs, which decreases the computational speed.

Besides the above, another study found that if only a small object part is entered into the model, not the entire image, satisfactory performance is achieved even with a small CNN model (Li et al., 2018). Instead of finding small objects in the entire image, only a small 96×96 pixel area of the desired position within the image is entered into the model. The study attempts to classify human facial expressions in the middle of the image and claims that a model consisting of two CNN and max pooling layers has satisfactory recognition performance. It makes this challengeable small object detection problem as including many features of small objects. In this study, the problem of inputting small areas into the model is solved by assuming that small objects exist in areas extracted with optical flow method.

Certain researchers developed a deep-learning-based framework for tracking UAVs. In this approach, moving objects (UAVs) were accurately detected at a high speed by modifying and improving CNN models based on YOLOv3-tiny in a real-time measured video stream (Yavariabdi et al., 2021). The algorithm was characterized by multiple detection steps and tracking steps between frames. In the multiple detection phase, the FastUAV-NET architecture used five insertion units and a pyramid network. In the multi-tracking step, the detected boundary box was tracked using the scale-adaptive Kernelized Correlation Filter (sKCF). Thus, algorithms to detect UAVs could be applied to every sixth frame, and efficient and accurate tracking could be performed in intermediate frames through sKCF (Montero et al., 2015). This approach could effectively address the challenges associated with the high speed of UAVs, changes in the UAV scale and aspect ratio, variations in the illumination condition and camera viewpoint changes, and reflected light and shadows.

The system that relies on deep neural network model is the most efficient and accurate technology. However, there are still many blind spots, and there is a limit to recognizing objects. Accordingly, as the image processing algorithm based on deep learning developed, it became the key that acquires training data of high quality and large size. Driver assistance systems using vision sensors such as monocular cameras and stereo cameras are of immense help in securing the driver's safety as well as the surrounding environment, and many associated studies are being currently conducted.

In particular, heavy equipment vehicles and buses face many blind spots compared to general automobiles, and as the risk of a collision sharply increases when the vehicle is reversing, rear cameras mounted on vehicles are gradually becoming more popular. For this reason, many studies have focused on detecting common objects and pedestrians behind vehicles (Kim et al., 2016; Tadjine et al., 2013; Ma et al., 2009; Yankun et al., 2011). The main objectives of their research are to reduce the risk of collision and to detect and inform the driver of moving objects behind the vehicle in motion (Braillon et al., 2006). To avoid missing objects in these blind spots, a wide range of visual equipment and sensor equipment is required, as well as suitable algorithms to effectively train using limited data resources.

One study on object detection and semantic segmentation uses information from various sensors as features in an unmanned vehicle environment (Feng et al., 2020). The research uses information such as RGB image, LiDAR, and 3D points obtained from Radar sensors. As pedestrian safety is important in a vehicular environment, a method of fusing various deep learning models to obtain accurate pixel location information where objects exist is suggested. The study adopts a two-stage fast R-CNN, one-stage YOLO, and a multi-modal model that combines SSD models and various sensor information. Another study uses a deep learning model that segments moving objects (Chen et al., 2021). Using 3D LiDAR data, binary classification is performed for each image pixel from the model of structure. It inputs the depth differences between frames and pixel positions into CNN layers for encoding and decoding of convolutional outputs. Using the inter-frame difference information, it is possible to generate a useful feature with a small input value only for moving objects. However, this method requires high-performance computing devices and expensive sensors, so it is not suitable for the purpose of the problem considered in this study.

2.3.3 Application of learning-based models

There are studies for pedestrian detection algorithm. A recent study applied various features to make high-level semantic feature to detect pedestrian (Liu et al., 2019). They suggest a new perspective for pedestrian detection, compared with sliding window classifier or anchor-based prediction in tradition approaches. That paper proposes a detector that scans for feature points on the image such as corners, blobs, and other feature extractors, and they optimize the suitable convolutional network for these features. Another researcher uses filtered channel features to improve pedestrian detection performance (Zhang et al., 2015a). One study classified deep infrared pedestrian based on deep neural networks using the automatic image matting method (Liang et al., 2019). This method provides efficient solution to improve classification performance of infrared pedestrian using deep neural network, while maintaining network structure.

There is a study that learns and calculates optical flow based on a deep learning model and combines it with a classification model such as the YOLO model (Zhang et al., 2019b). The researchers attempted to exploit optical flow information from video frame and apply object detection technology to the videos. The application of moving object detection models to video data is a challenge of conditions such as motion blurring, video defocusing, and partial covering. Therefore, the algorithm was developed to accurately detect and track moving objects by deciding the position of the bounding box. It uses the feature map of optical flow value within two adjacent frames obtained through FlowNET 2.0. The target moving objects in the frame are detected from YOLOv3 detection model. The method of calculating the optical flow using learning model is that the interference time is less than the dense optical flow and the performance is similarly maintained. However, it is unsuitable for the purpose of studying lightweight algorithms, as it requires several times the computational cost rather than traditional sparse optical flows such as Lucas-Kanade method.

There have been various studies on object detection for improving model efficiency and detecting small objects. A recent study increased efficiency of incremental transfer learning using scalability of knowledge distillation for fast object detection (Yuwono et al., 2022). To detect small objects in high-resolution satellite images, deep feature extraction is adapted using dilated convolution and contextual information for improving on model efficiency and detection of small objects (Wu et al., 2021). Another study on detecting small objects applies a clustering algorithm based on YOLOv3 to improve model instability (Wang et al., 2021). There is a study to detect coal-gangue, which pro-

poses fine-grained object detection based on spatial and channel attention mechanisms (Lv et al., 2021).

Recently, many researchers developed approaches to detect moving objects by using optical flow and deep learning based model. With the widespread application of unmanned aerial vehicles (UAVs), moving objects have been attempted to be detected and tracked using cameras within the UAV (Zhang et al., 2019a). In this approach, the moving objects were detected by subtracting a background changing in a complex manner from an image captured by a moving camera. The algorithm extracted motion areas based on optical flow and removed the background to perform clustering around moving objects. Noise was eliminated by removing a false foreground based on time and space consistencies. A frame skip strategy was used to accelerate the algorithm.

In addition, moving objects for UAVs were detected by obtaining images in real time. A dense optical flow technique was used; however, the background was assumed to be fixed. By obtaining top-view images in the aviation domain, the map for moving objects can be extracted using background removal and mean shift segmentation techniques. Notably, dense optical flow techniques are time intensive, and nearly 3.5s are required per frame, which limits the application of such techniques to heavy equipment such as microcontroller units (MCUs) (Zhang et al., 2015b). To address this limitation, I use a sparse optical flow technique.

For UAVs, certain researchers proposed a robust onboard visual algorithm based on the reliable global-local object model for 2D and 3D object tracking to achieve a reasonable computational time (Fu et al., 2016). This approach is based on global matching and local tracking. The algorithm initially identifies feature correspondences. An improved binary descriptor is developed for global feature matching, and an iterative LK optical flow algorithm is used for local feature tracking. Furthermore, an efficient local geometric filter is used to manage the outlier feature correspondences based on a new forward–backward pairwise dissimilarity measure, thereby ensuring pairwise geometric consistency. In another study, objects on the ground were identified using the YOLOv3 deep learning model for UAVs (Meng et al., 2020). The images recorded by aircraft were transmitted to computers using in-flight communication systems, and neural network models were implemented through the computers. It can be thought of combining using optical flow techniques for local feature tracking and classifying objects using deep learning models. Inspired by these application studies, it can get the idea of the sparse optical flow used for object tracking and the object detection

(a) (b) (c)

Figure 2.9: Process of tile selecting method. (a): Tiling for different CNN input size of 544x544 (2 tiles). (b): 352x352 (6 tiles). (c): 256x256 (12 tiles). (from (Plastiras et al., 2018)).

algorithm for classifying objects based on that area.

Figure 2.9 shows a selective tile processing technique to create an efficient CNN-based object detection model (Plastiras et al., 2018). In this study, the size of the tile is determined according to the input size parameter of the CNN model, and the input image is divided into several tile regions. The proposed method selects a tile with objects in each divided tile area and then inputs only the selected tile area into the CNN model to detect the object. As objects may appear in excluded tiles, the proposed technique introduced a function called tile reset. This is similar to the feature reset function of the optical flow method proposed in this paper. In conclusion, this work increases the computational processing speed by reducing the size of the input layer of the CNN in a way that deals with the resolution of the input image. These approaches can be seen as similar to the algorithms that combine the optical flow-based ROI and classification model proposed in Chapter 5. However, they merely remove unnecessary areas of images taken from high positions from drone equipment of high-performance CPUs. In contrast, the approach proposed in this paper removes unnecessary regions through an optical flow-based tracking model and classifies only objects in the regions of interest. Furthermore, it can overcome the fixed SVM dimension problem of machine learning techniques through resizing methods to improve detection performance.

2.4 Summary of Chapter 2

In this chapter, the basic background of data and algorithm for object detection models are introduced. In moving-object tracking section, optical flow techniques for calculating and tracking pixel movements and various object tracking models using them are introduced. Dense optical flow and background subtraction are not suitable for the lightweight detection model proposed in this paper because of its high computational cost. Accordingly, it aims to track moving objects from fixed cameras, such as surveillance systems.

In the moving-object tracking application, the detection performance was affected by image characteristics in the case of distorted images such as wide-angle cameras or images taken from high-position cameras. In the data processing section, the effect of datasets on model performance, abnormal data value classification and learning strategy, and amount of learning data was analyzed. The features and relationships of dataset are analyzed and used to establish the coverage capabilities and combination learning strategies of dataset features.

In the object detection section, the characteristics of learning-based classification models from the perspective of various camera environments and noise tolerance, and the disadvantages of high computational cost are examined. The learning-based model is noticeably influenced by the feature of the learning data, and it is difficult for small objects at a distance to be learned. The classification model can be used within the ROI where the location of the object is estimated through moving-object tracking. By resizing the ROI to make the outer features of the moving object more distinct, it is possible to better detect the subset of the dataset containing small objects that are difficult to learn. Finally, the proposed algorithm employs deep learning models and CNN classifiers to better detect the features of moving objects in the ROI extracted by the optical flow method.

Chapter 3

Moving object tracking based on sparse optical flow

In this chapter, a new approach for moving-object detection and tracking is proposed. One of its major attributes is to improve the accuracy performance and reduce the computation time while responding to moving objects or moving pedestrians. The proposed method is based on the sparse optical flow approach, i.e., a coarse-grained optical flow, but it includes the corner feature reset with a moving window. A sequence of images effectively finds the flow of moving objects, and thus the moving window of image frames easily captures moving targets without wasting much time.

The moving window detector improves the noise filtering and the detection rate by looking at a history of optical flows. In a hazardous environment, such as the construction sector, there may be the risk of encountering many obstacles including walls and trees, and various optical flow patterns are often observed, when pedestrians should be detected. The memory-based target estimator plays the role of monitoring the targets or pedestrians without missing targets when they move around or stagger at some positions. Even if a moving target is initially recognized, the target may move continuously with occasional pauses. Using this estimator, the last position of a moving target is estimated, which improves the performance of detecting moving objects in a row.

This detection algorithm was adapted in the embedded board system, Raspberry Pi 4B, for real-world applications. The experimental results demonstrate that the suggested approach is effective for preserving the detection performance even with an embedded device having low computing power. According to the experimental results,

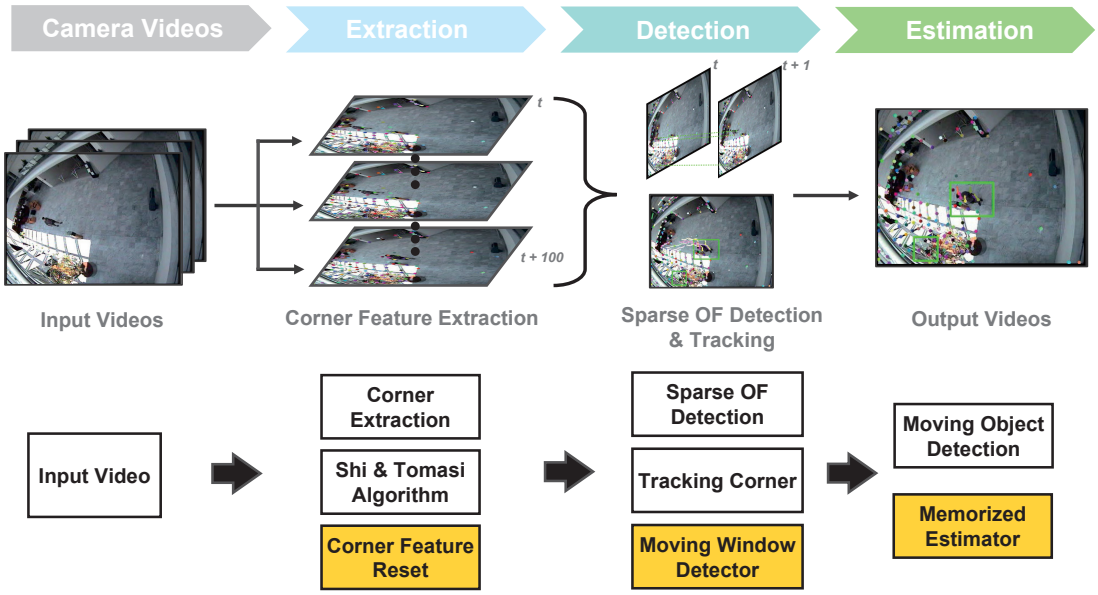


Figure 3.1: Overview of the proposed algorithm; the green boxes show detection of moving objects.

the proposed method shows similar or higher accuracy performance compared to the conventional algorithms for moving-object detection using optical flows or vision processing algorithms, e.g., Lucas–Kanade method and Farneback's method, in addition to HOG and Haar-like methods. It also takes lower computing time than dense optical flows and other vision processing algorithms. The approach works well even for blurred images or noisy image frames. This content is also available in a published journal (Choi et al., 2022b).

3.1 Methods

I propose a method based on optical flow algorithm to track moving objects. In order to operate smoothly in small embedded equipment, it applies sparse optical flow among optical flow algorithms. The sparse optical flow uses corner point as a feature in which the optical flow is calculated. There is a function that generates this corner point, and I would like to introduce a corner feature reset function that optimizes the corner feature function. By assigning window memory for each corner feature, I add moving window detector and memory estimator functions to enhance detection performance and robustness to noise.

Figure 3.1 shows the process flow of the proposed real-time video-based algorithm.

The proposed algorithm is based on sparse optical flow, which receives each frame of the video as the input and calculates the motion degree of the objects. Specifically, the algorithm calculates the motion information from all pixels of each frame, not as a dense optical flow (Farneback) (Farnebäck, 2003) but Lucas-Kanade(LK) as a sparse optical flow (Bouguet et al., 2001). In addition, I extract the corner feature during LK optical flow calculation and estimate the moving object using only the moving information of this feature. More accurate and efficient motion information can be obtained by adding the regenerative function of the corner. I input videos with resolutions of 383×288 and 768×576. Every N frames, the corner feature points are generated with Shi-Tomasi algorithm (Shi et al., 1994). The sparse optical flow algorithm calculates the moving information of feature points. The moving window detector collects the feature points for optical flow. If the tracking moving object disappears behind a wall, obstacles, or outside of a frame area, the memorized estimator checks the region around the target.

The LK method uses feature points to track the optical flow, which may render object detection over large distances challenging. In contrast to dense optical flow techniques, which evaluate all the pixels on the frame and neighboring pixels, LK optical flow calculates feature points such as corners to facilitate tracking. Notably, if the moving objects are at a large distance, they are difficult to distinguish from noise. In detecting the motion information for each frame, a noise filtering function is introduced to alleviate the camera vibrations and light spread phenomena. To detect and track the moving objects obtained through feature extraction and LK optical flow, I introduce a memorized estimator to estimate the position of moving objects by memorizing information regarding the last missing position. This approach is expected to be effective in situations in which the object stops moving or is obscured by certain obstacles in the video input.

Notably, the proposed algorithm is based on the pyramid LK method but incorporates corner reset, noise filtering, and moving-object estimation (yellow box in Figure 3.1). Moreover, I aim to implement the proposed algorithm in embedded systems that can be used in places such as construction sites to ensure personnel safety.

3.1.1 Corner feature reset

For sparse optical flows, corner features are detected, and a set of features is used for optical flow calculation to decrease the computational time. The corner features are

Figure 3.2: Example of corner feature reset method.

typically generated for a given image snapshot by using the Shi–Tomasi algorithm. Although moving objects can be tracked using optical flows, they may be hidden near obstacles or walls and appear again, or a new object may be observed in the image. Thus, corner feature reset must be performed to identify all the moving objects. Corner features are regenerated in regular period to prevent this problem. However, the process is not implemented for every frame because corner feature algorithms are time intensive.

Figure 3.2 shows an example of corner features based on the corner feature reset. The red box shows that the corner feature reset method regenerates the corner features in a period of time. The green box shows the detected moving object using proposed algorithm. The pedestrian features are retained for continuous tracking. The red box shows the region that the moving pedestrian traverses. This region may not be relevant for tracking anymore, but another moving object may be present in the area. Corner feature reset is aimed at identifying the corner features in the region for the optical flow of another moving object.

3.1.2 Moving window detector

The original pyramid LK method is vulnerable to noise such as that pertaining to light smudging in the input image and camera vibration. Even if no moving object is present in the actual optical flow, false positives may be induced owing to even small noises. To overcome these problems, it is used a filtering function and enhance the detection performance in Figure 3.3. The moving window method memorizes the feature points for optical flow within each of n frames. The green boxes show detection of a moving

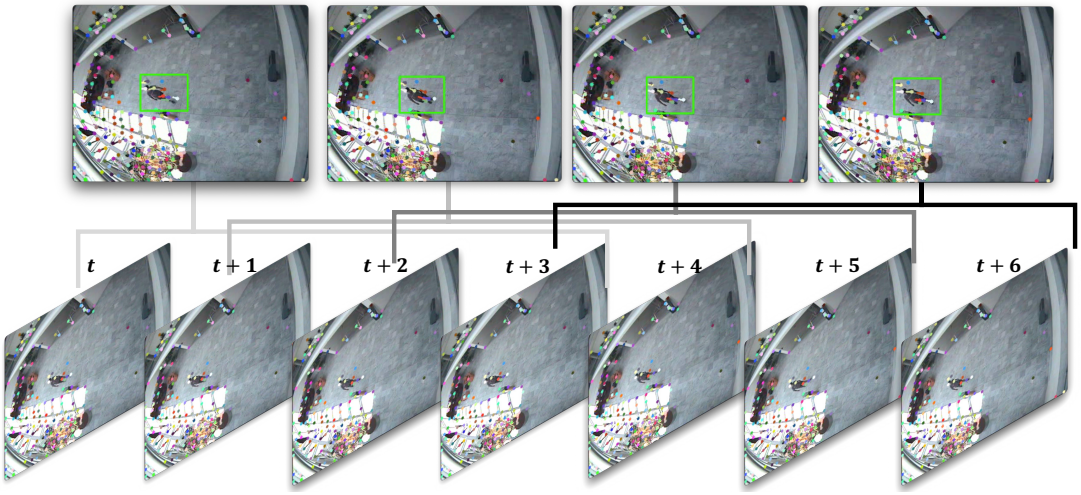


Figure 3.3: Example of the moving window detector method.

object.

$$MW_{i,t} = \{P_{i,t-n}, P_{i,t-n+1}, \cdots, P_{i,t-1}\} \tag{3.1}$$

The moving window function aims to memorize the feature points for optical flow within each of n frames. The presence of noise is evaluated by determining if the movement of the feature during an interval is less than or equal to the distance threshold. The window memory container $MW_{i,t}$ in Equation (3.1) continually saves the location of feature point i at time t over M points, $P_{i,t}$ $(i \in M)$. If $t < n$, the memory size is less than the n memory capacity. In contrast, when $t > n$, window memory overflow occurs, the oldest memory $P_{i,t-n}$ is removed from the window memory container, and the recent memory $P_{i,t}$ is pushed to the end of the container. When corner reset is implemented and the point is tracked, the window memory container of the point is maintained, not reset.

$$\Delta P_{i,t} = \sum_{t'=t-n}^{t-1} \Delta P_{i,t'} = \sum_{t'=t-n}^{t-1} (P_{i,t'+1} - P_{i,t'}). \tag{3.2}$$

To enhance the detection performance, especially when capturing a distant moving object, the moving window method changes the distances of the first and last locations in the window memory, to determine whether the feature is a moving object. Instead of the sum of moving distances within a period, the threshold measurement is based on the varied distances of the points because the vibration related noise moves continuously

Figure 3.4: Example of the memorized estimator method.

in a certain period. In contrast to the summed value, the change in location in the period does not cumulate the moving distances, and only the changes in the initial and final point locations are determined. Thus, the change in point locations is defined in a constant time interval, $\Delta P_{i,t}$, and use it instead of the sum of changes in the point locations *P(x,y)* from time $t-n$ to $t-1$, as shown in Equation (3.2).

$$L_{i,t} = dist(P_{i,t-n}, P_{i,t}), \quad D_{i,t} = \begin{cases} 1, & \text{if } L_{i,t} \geq \alpha \\ 0, & \text{if } L_{i,t} < \alpha \end{cases} \tag{3.3}$$

In Equation (3.3), $L_{i,t}$ is the movement distance, measured using the Euclidean distance method, in units of pixels. Moreover, α is the distance threshold to determine whether the point is a noise or moving object. $D_{i,t}$ is a flag that operates the moving window detector function and identifies whether point $P_{i,t}$ is a moving target or vibration noise, based on $L_{i,t}$. For example, if $D_{i,t} = 1$, the green detection box is generated around point $P_{i,t}$. A high distance threshold can block the sensor noise and obtain more definite movements, demonstrating lower recall but higher precision performance. The performance metrics must be adjusted based on the detection environments.

3.1.3 Memorized estimator

I propose a function to estimate the location of the tracking object, even if it is temporarily stopped or hidden behind obstacles, by estimating the feature points pertaining to the target in Figure 3.4. The green boxes show detection of moving objects. The green detected box is maintained on the target over a given time span.

$$E_{i,t} = \begin{cases} \tau, & \text{if } D_{i,t} = 1 \\ E_{i,t-1} - 1, & \text{if } D_{i,t} = 0 \text{ and } E_{i,t} > 0. \end{cases} \tag{3.4}$$

In Equation (3.4), τ is the estimation time. $E_{i,t}$ represents the estimator for the feature point i at time t and is reset to τ when feature point i has a detection state of $D_{i,t} = 1$, determined using Equation (3.3). If the moving window detector $D_{i,t}$ is set as zero and the detector $E_{i,t}$ is more than 0, the memorized detector $E_{i,t}$ is reduced to $E_{i,t} - 1$ in each time step.

The non-zero estimator ($E_{i,t} > 0$) attempts to detect hidden targets. For example, if certain feature points lose the tracking target or the target is hidden because it is beyond the camera frame or behind walls or obstacles, the memorized estimator continues to track the feature points of the target. The green detected box is maintained on the feature point while estimation time $E_{i,t}$. In addition to the effect of the moving window detector, a higher τ increases the recall and decreases the precision performance.

3.2 Experimental environment

The proposed method is influenced by feature point characteristics such as the corners for calculating the optical flow and tracking the next point. Experiments are performed to identify an effective feature generating algorithm by comparing several algorithms. First, it is specified the control parameters for all algorithms: corner reset interval, maximum number of corners and corner distance, semi-metric parameters, number of missing boxes (non-existent feature points in the label box), and number of real-generated features. To select the optimal parameters for each algorithm, I investigate the semi-metric comparison results, apply the selected parameters to the algorithms and evaluate the performance. The follow subsections describe the semi-metric parameters of each algorithm and performance evaluation.

3.2.1 Dataset

I conduct an experiment by applying the proposed algorithm to two large datasets and perform a comparative analysis with other existing algorithms, as shown in Figure 3.5. Figure 3.5a–c show the first dataset: Videos 1, 2, and 3 correspond to walk data for a person walking in a straight line in a hallway, flow data for back and forth movement,

(a) (b) (c) (d)

Figure 3.5: Examples of datasets (CAVIAR and PETS2009). (a): Video 1(Walk). (b): Video 2(Browse). (c): Video 3(Browse Whilewaiting). (d) Video 4(Pedestrian).

and waiting data for a person pausing in the middle and then continuing to walk back and forth, respectively. The red boxes show the ground truth contained in the datasets. The dataset has a resolution of 384 $\times$ 288, and the number of video frames are 790, 1042, and 610. The proposed algorithm is compared with dense and sparse optical flow algorithms. In the second dataset, video 4 shows the surroundings, containing more individuals than those in the first set. The resolution is 768 $\times$ 576, and the number of frames is 794. The proposed algorithm is compared with pedestrian detection algorithms (dense and sparse optical flow algorithms and Hog and Haar-like methods).

The first dataset focuses on the recognition and evaluation performance of moving objects instead of pedestrian shapes, and the second dataset is aimed at comparatively analyzing the proposed algorithm with algorithms that can estimate shapes and moving objects. The two datasets are significantly different: The first dataset contains slowly walking people, whereas the second dataset contains rapidly walking people. In the first dataset, the moving targets are more difficult to detect because of the presence of fewer people and people who are walking slowly. Because the two datasets have different resolutions, I consider the resolutions of 384 $\times$ 288 and 768 $\times$ 576 as small scale and large scale, respectively, to ensure a fair comparison. In the comparison of the optical flow methods, video 4 is resized to the small scale (video 4_S, resolution 384 $\times$ 288). When comparing machine learning methods, videos 1–3 are resized to the large scale (videos 1–3_L, resolution 768 $\times$ 576).

3.2.2 Missing box of corner parameters

In the proposed method, to detect moving objects, I calculate the optical flows of each feature point, determine the next point movement location, and detect moving objects by inspecting the moving window memories of the points. Therefore, the feature points

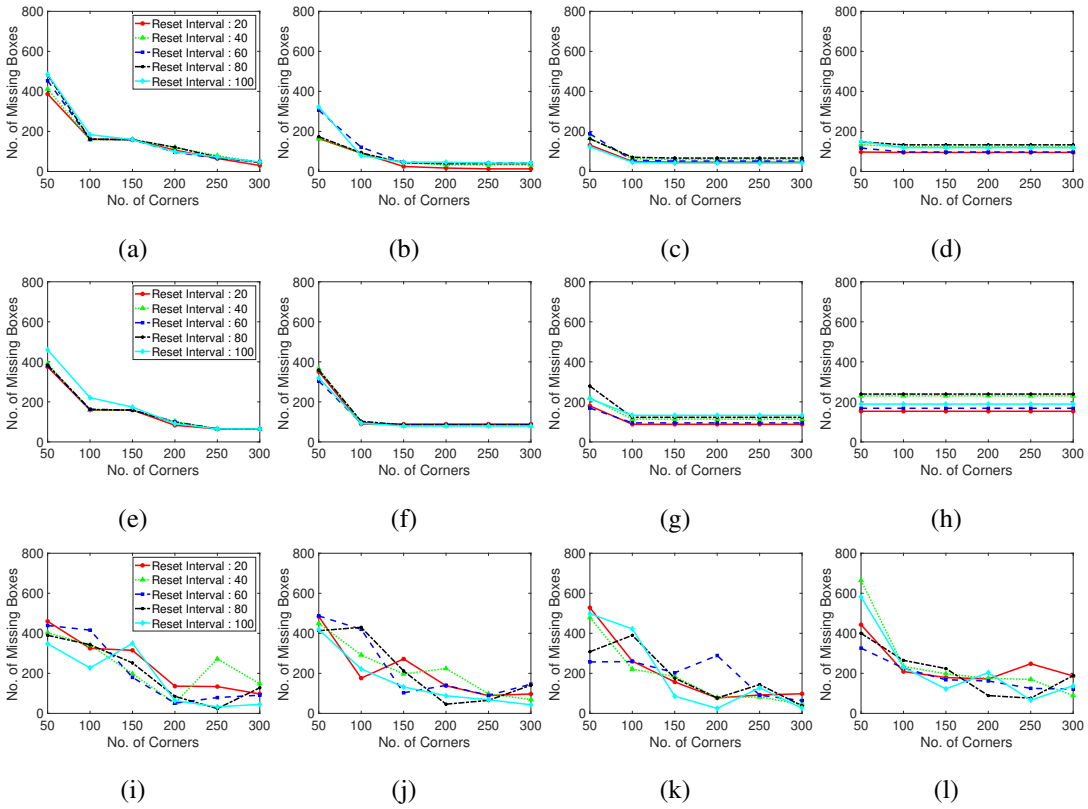


Figure 3.6: Number of missing boxes with various corner extraction methods. (a–d): Shi-Tomasi method (a pixel distance of 5, 10, 15, and 20 between corners). (e–h): Harris. (i–l): Random grid.

for calculating the optical flow and next movement are essential and important components of the proposed method. The feature points are typically spotted around existing moving objects and those that recently appeared in the frame. The feature points tracked on the moving objects remain on the target objects.

To observe this situation and enhance the performance of the corner generation algorithm, I define a parameter, that is, the number of missing boxes, which counts the label boxes of non-existent feature points to determine the optimized values. In Figure 3.6, a higher number of missing boxes means that the feature point generation algorithm does not generate feature points to calculate optical flows. The corner detection methods control the number of corner features and the pixel distance between corners. Test dataset is video 1, 'Browse WhileWaiting1.mpg' from CAVIAR dataset. A lower number of missing boxes corresponds to a higher detection rate. When both number of generating corners and distance between corners low, corner features are generated densely, and it causes a lot of missing boxes.

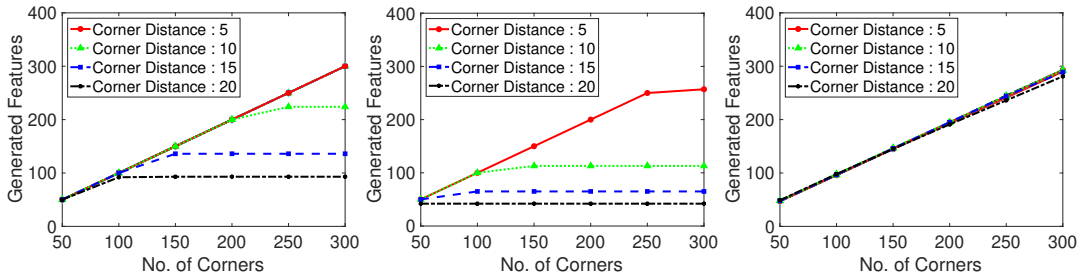


Figure 3.7: Generated features with various corner extraction methods. (a): Shi-Tomasi. (b): Harris. (c): Random grid.

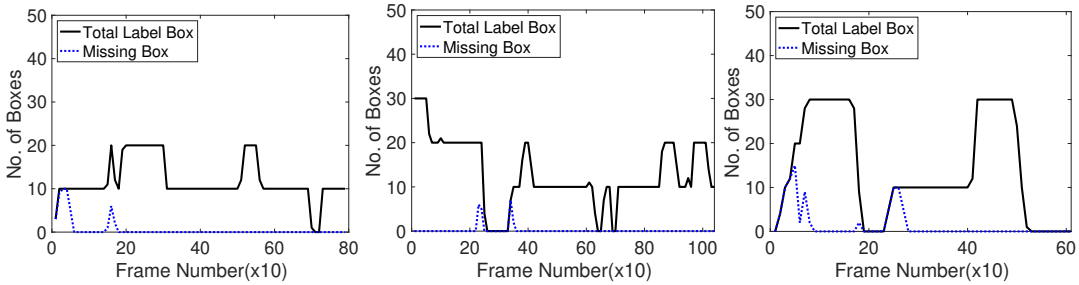


Figure 3.8: Number of missing boxes in every 10 frames with Shi-Tomasi method. (a): Video 1. (b): Video 2. (c): Video 3.

I specify the real generated corner numbers for the above mentioned experiment environments to examine the influence of the corner features on the number of corners, as shown in Figure 3.7. Test dataset is video 1. Shi-Tomasi corner extraction tends to make more feature points than Harris corner extraction. If the number of corners is large, the frames per second (FPS) for the processing is high. In contrast, if the number of corners is small, the accuracy of feature tracking to determine the optical flow are high. Certain feature point generating algorithms have dependent parameters such as the corner distance versus corner number. In the case of corner-generating algorithms, setting the corner distance limits the maximum number of generation points, depending on the inter-corner distance, maximum number of corners, and presence of corners in the frame. If the corner distance is excessively high, the number of corners is low at a given frame size. I examine the number of real generated corners or feature points for each feature generating method (Shi–Tomasi and Harris corner extraction algorithms and random grid point methods).

Considering the corner feature extraction results based on the number of missing boxes, the parameters of the corner-generating algorithms are determined to optimize the performance. To specify the best corner generation framework, I inspect the number of missing boxes in 10 frames of three typical public datasets. Figure 3.8 shows that the missing boxes are exposed at first, as indicated by the black solid line when the moving objects appear or disappear, and disposed off. Test datasets are videos 1, 2, 3. This observation indicates that the feature points to track the moving objects are effective and appropriate.

3.2.3 Evaluation of corner parameters

To evaluate the influence of the corner generation on the performance, the recall and precision for the corner generation algorithms are measured. The Shi–Tomasi and Harris corner extraction algorithms and randomly located point method are compared in terms of the LK optical flow tracking feature points. The variance in the corner generation parameters is the same as that in the previous subsection. In this section, the results of only one test dataset (Browse_WhileWaiting1.mpg) is presented owing to the limited space. The recall, precision, and number of missing boxes, and number of generated features are presented in the following figures.

The following metrics are typically used in object detection experiments: True positive (TP) means successful detection of the ground truth labels, false positive (FP) means detection failure, and false negative (FN) indicates the number of non-detected labels. The precision, recall, and F-score are determined as $TP/(TP+FP)$, $TP/(TP+FN)$, and $2\times(Precision\times Recall)/(Precision+Recall)$, respectively. In this case, β is 1. Precision indicates the detecting accuracy rate, recall represents the proportion of detected true labels, and the F-score is a generalized measurement considering both the recall and precision. These typical evaluation measurements are used in the following analyses.

Figure 3.9 shows the recall results for various corner parameters of the Shi–Tomasi corner extraction algorithm (first row), Test dataset is video 1. Harris corner extraction algorithm (second row), and randomly generated features (third row). With the increase in the maximum number of corners, the recall performance steadily increases and becomes convergent and stable. The Shi–Tomasi corner extraction algorithm outperforms the other algorithms in terms of the recall. As shown in Figure 3.10, the Shi–Tomasi and Harris corner extraction algorithms exhibit similar precision performances and

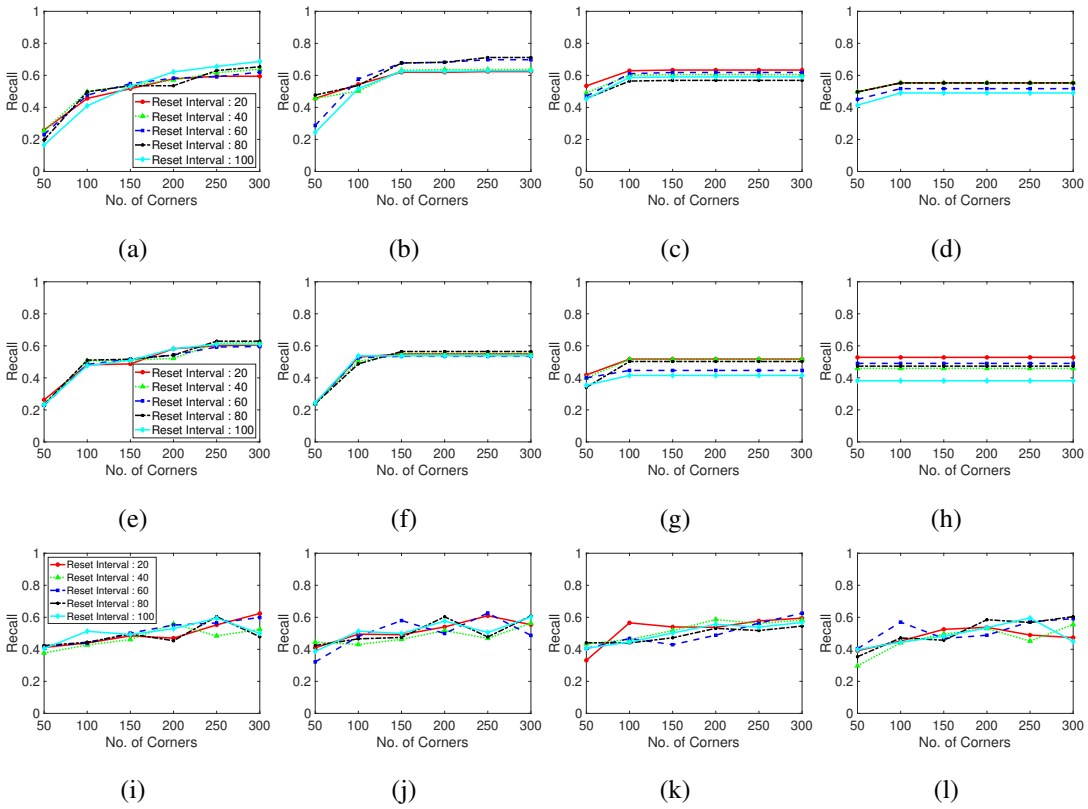


Figure 3.9: Recall performance with various corner extraction methods. (a–d): Shi-Tomasi method (a pixel distance of 5, 10, 15, and 20 between corners), (e–h): Harris. (i–l): Random grid.

the values converge, although the Shi–Tomasi algorithm is slightly superior. Thus, the most effective feature generating method is the Shi–Tomasi corner extraction algorithm with the maximum number of generated corners being 150–200. Corner reset interval parameters exhibit similar results over 60 frames. The corner distance parameters results are similar in four columns. I select the corner distance parameter as 10 pixels (second column), which corresponds to a stable and high performance in terms of the recall and precision. In the following analyses, I choose the best parameter values for the considered methods.

Moreover, I evaluate the grid located point results. The randomly located point method exhibits an unstable performance, likely because of the stochastically generated point locations. Thus, well-distributed tracking point must be used when implementing a limited number of generation corner number. Distance between corner refers the parameter using in the corner extraction algorithm, which determines the distance between extracted corners. In the case shown in Figure 3.11, setting a maximum corner

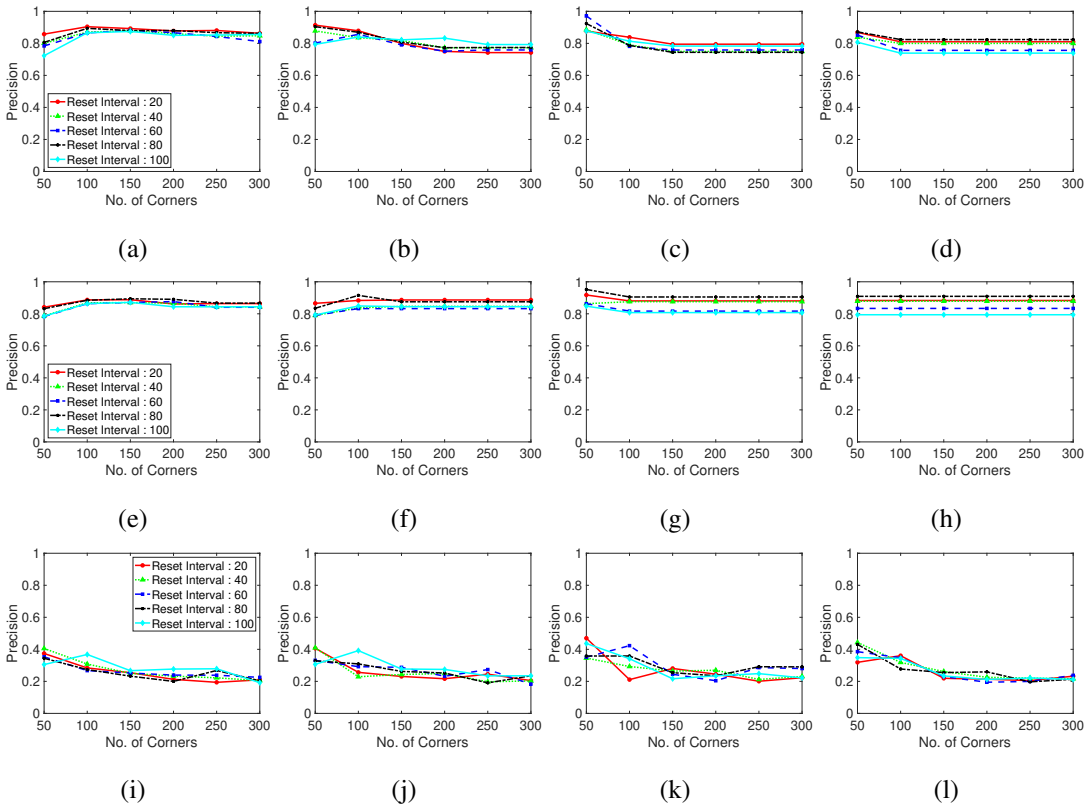


Figure 3.10: Precision performance with various corner extraction methods. (a–d): Shi-Tomasi method (a pixel distance of 5, 10, 15, and 20 between corners), (e–h): Harris corner. (i–l): Random grid.

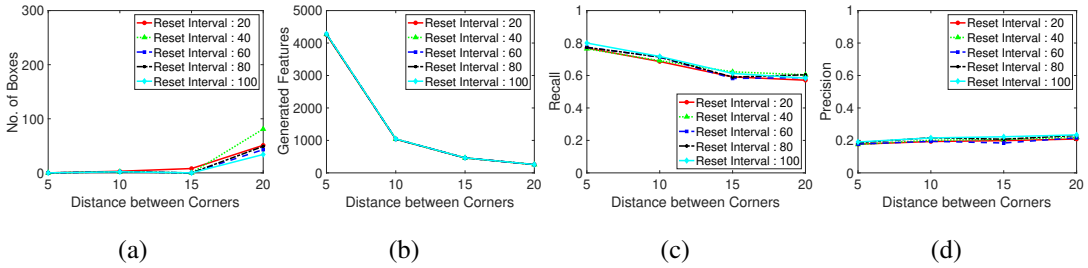


Figure 3.11: Performance with a regularly-spaced grid of sampling points. (a): Missing boxes. (b): Generated features. (c): Recall. (d): Precision.

number is meaningless because the points are generated according to the grid of the constant corner distance. Therefore, I investigate the corner distance and corner reset interval parameters in terms of the number of missing boxes, number of real generated features, recall, and precision. The grid generated point method exhibits a large number of missing boxes, but the recall and precision are low. Moreover, the second plot in Figure 3.11 shows that the generation of excessively many tracking points decreases

the computational speed.

3.3 Experimental results

The proposed method differs from other moving-object detection algorithms owing to the implementation of the moving window and estimator method on the LK optical flow algorithm, which enhances the object detection performance and helps overcome the limitations of sparse optical flow techniques. The estimator operates synergistically with the moving window method by preventing the failure of determining the sparse optical flow. Specifically, when the moving window tracking for the change in the location of the feature points fails, the estimator is activated. The estimator remembers the last location of the tracking object and predicts the presence of the disappeared object on the spot. Therefore, I conduct experiments to examine the detection performance with changes in the parameters of the window and estimator: window sizes, distance thresholds, and estimation times.

I test four datasets: Browse_WhileWaiting1.mpg, Browse1.mpg, and Walk1.mpg from the CAVIAR dataset for comparing moving-object detection algorithms using typical sparse and dense optical flow; and PETS09-S2L1.webm from the ETS2009 dataset for comparing pedestrian detection algorithms with Hog and Haar-like SVM detection models (Dalal and Triggs, 2005; Viola and Jones, 2001). The experiments for recall and precision are independent of the accuracy because this parameter is influenced by the algorithm parameter settings. Notably, FPS is influenced by the electric power stability of the device. Raspberry Pi 4 is used, and thus, 20 experiments are conducted, and the average and standard deviation of the 20 values are considered. It examines the effects of the moving window and estimator and compare the performance of other object detection algorithms.

3.3.1 Results with changes in window size

The window size is a key parameter of the moving window function in the proposed method. The function contains the locations of all tracking points (x,y) in the moving window memory from time $t - \textit{window size}$ to t, and thus, each window memory has a specific size. When the window memory is full, the last location memory is eliminated, and the recent location memory is pushed to the end of the list. Therefore, a constant window memory size is maintained. The window tracks the change in the location

of each point and decides whether it is a moving object or noise by considering the moving distance threshold. The influence of the change in the window size is reflected in terms of the true number of detection boxes, recall, and precision. The total number of alarms is the total number of predictions obtained using the proposed detection method, and this value is compared with the true number of detection boxes.

In the case of a small window size, the history of the tracking point is limited, and the change in the points' locations is observed. The probability of the point being identified as a noise instead of a moving object is higher. In contrast, for a large window size, the history of the tracking point is adequate, and the point location can be tracked to examine if it is moving object. However, the detection of true target boxes may be missed because considerable time is required for the evaluation, and the window may fail to track the point when the moving object disappears. For example, if the window size is 40 and the object is moving in 50 frames, the detection box has less than 10 frames. Therefore, the window size must be properly selected. I conduct experiments with different window sizes and examine their influence on the number of detected boxes and accuracy.

As shown in Figure 3.12, the number of detected moving object boxes increases with the window size, and the difference in the ratio of the total predicted detections and true detections increases. This gap signifies that as the window size increases, the proposed method tends to erroneously indicate that moving objects exist. In other words, extremely high window sizes deteriorate the detection performance because more time is required to decide whether the target is in a moving state. However, at larger window sizes, the number of true detected moving objects stabilizes but the number of prediction alarms increases. This aspect indicates that the above mentioned phenomenon likely has another explanation. When the algorithm examines a longer history of the tracking point and location changes (when the window size is larger), it is more likely to identify the target point as moving object even when it is stationary state. If the tracking point movement distance exceeds the moving window distance threshold, the moving window detection algorithm judges the object to be moving when the change in the location exceeds the threshold. Therefore, the object is likely to be predicted as a moving object even when the target tracking point stops. Sparse optical flow cannot easily track fast-moving objects and thus an object may be considered to be moving even when the tracking point does not lie on the moving object.

With the change in the window size, there occurs a crossing-over point at which the

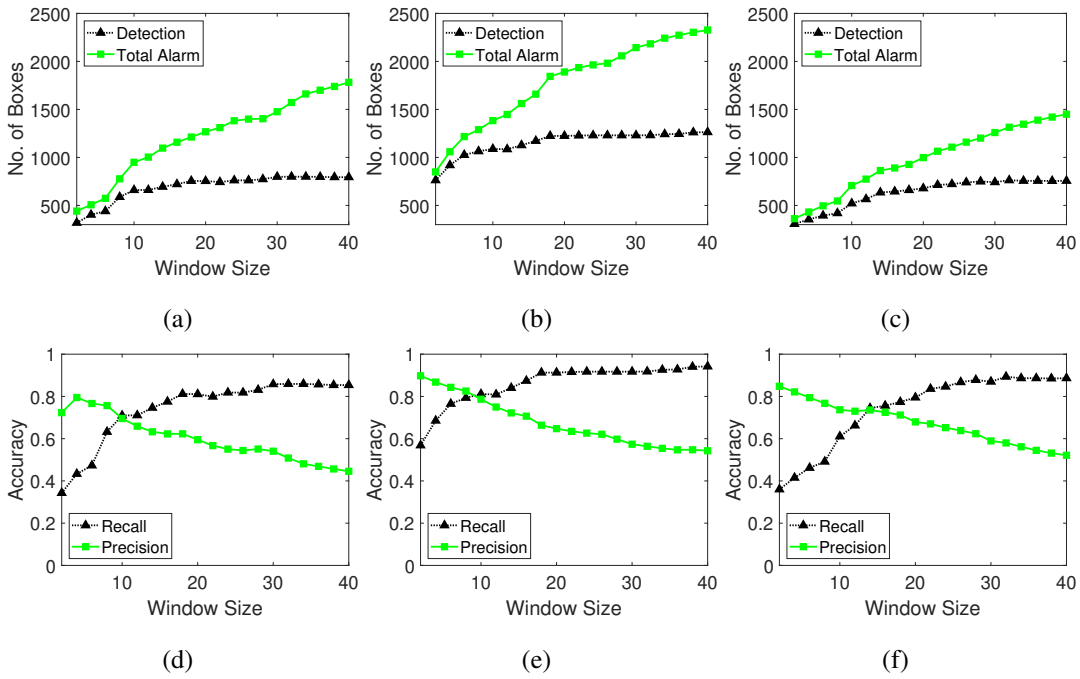


Figure 3.12: Number of boxes and accuracy of the moving window method; test datasets are video 1 for (a,d), video 2 for (b,e), and video 3 for (c,f). (a–c): Total predicted detections (total alarm) and true detections. (d–f): Recall and precision performance.

recall and precision curves intersect. This point is likely an optimal value to ensure a stable performance between the recall and precision as well as the F-score (F1). The total true detection number steadily increases and adversely influences the recall performance. Thus, longer tracking of the change in the point location leads to the detection of more true moving objects because the moving window detector obtains the interpretation based on a longer history at each point. The precision decreases as the recall increases because the optic flow tracking points that remain at and depart from moving object are considered to be in the moving state by the moving window memory. I validate this analysis by investigating the output detection labelled video. The red labelled box is the ground truth box, the green labelled box is the detection result of the moving-object detector algorithm, and the blue labelled box pertains to false detections. The findings indicate that the optimal window size is 10 frames.

3.3.2 Results with changes in distance thresholds

Figure 3.13 shows the number of false detections to validate the noise filtering effect based on the distance threshold in the moving window method. A high distance threshold prevents the detection of the noises from vibrating cameras in locations such

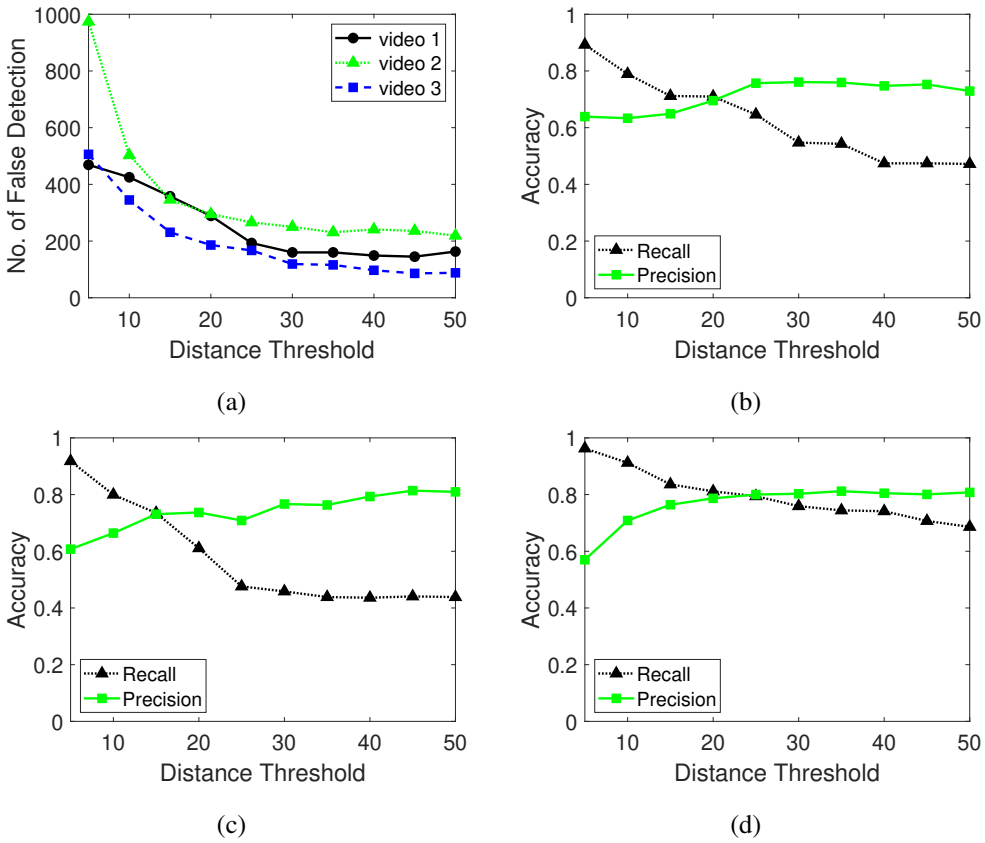


Figure 3.13: False detections and accuracy of the moving window method. (a): False detections for videos 1, 2, 3. (b): Recall and precision for video 1. (c): Video 2. (d): Video 3.

as construction fields. In contrast, a low distance threshold enables the detection of minute noisy vibrations and small moving objects. A lower distance threshold corresponds to a higher recall and lower precision. The black line shows that high threshold distance filters the noises and tiny movements. The green line shows that noise filtering in the moving window distance threshold method helps achieve more precise results.

3.3.3 Results with changes in estimation time

The memorized estimator remembers the last position at which the calculation was stopped for a certain period (frames) and continues prediction even when the moving object stops or disappears from the video. To evaluate and select optimal parameters for this function, I determine the number of detections according to the estimation time and existence of the estimation function. The performance indicators of recall and precision are determined.

Figure 3.14 shows the results of the number of detections, detection elapsed time, recall, and precision. When the proposed detector uses the estimator, the non-estimated detection number refers to the number of successful detections of the moving object. When the proposed detector uses only the moving window and distance threshold method, the estimated detection number means the detection counts when the detector estimates the target. The detection time is the elapsed time until the proposed detector identifies a moving object when working on an estimated or non-estimated tracking target. As shown in the first row of Figure 3.14, a larger tracking estimating time corresponds to a decreased probability of detecting moving objects than that pertaining to non-estimated tracking. When the estimation time is longer, the estimator spends more time on the moving targets and less time on the non-estimated targets because the detector for the moving objects implements the estimation more frequently. Moreover, the elapsed time until a moving object is detected is less than that for the non-estimated detection tracking. This finding shows that the estimator works faster when detecting moving objects if they temporarily stop, by carefully observing the movement.

The memorized estimator influences the detection accuracy. When the object is not sensitively detected, the detector misses the moving target and the estimator supplements the insufficient information of the moving target by memorizing the target's last location information such as the moving window memory. As shown in the last row in Figure 3.14, the recall is stable as the estimation time increases higher; however, the precision decreases because the estimator remains activated when the moving target stops or moves behind obstacles such as walls, trees, or roofs, when observed through a top-view camera.

The proposed algorithm has three key parameters. Based on the experimental results, I determine the optimal parameters for the corner features (generation method, maximum corner number, and corner distance), memorized moving window (window size and distance threshold), and memorized estimator (estimation time).

In Figure 3.15, LK method corresponds to the sparse optical flow, which is the basis of the proposed algorithm, MV corresponds to the proposed moving window detector without the memorized estimator, and MW + Est pertains to framework with the memorized moving window and memorized estimator. The recall and precision of the proposed algorithms (MV and MW + Est) are comparable to the existing algorithm. Thus, the proposed model exhibits a high accuracy and reproducibility for actual moving objects. Figure 3.15 shows the influence of the proposed methods (moving window

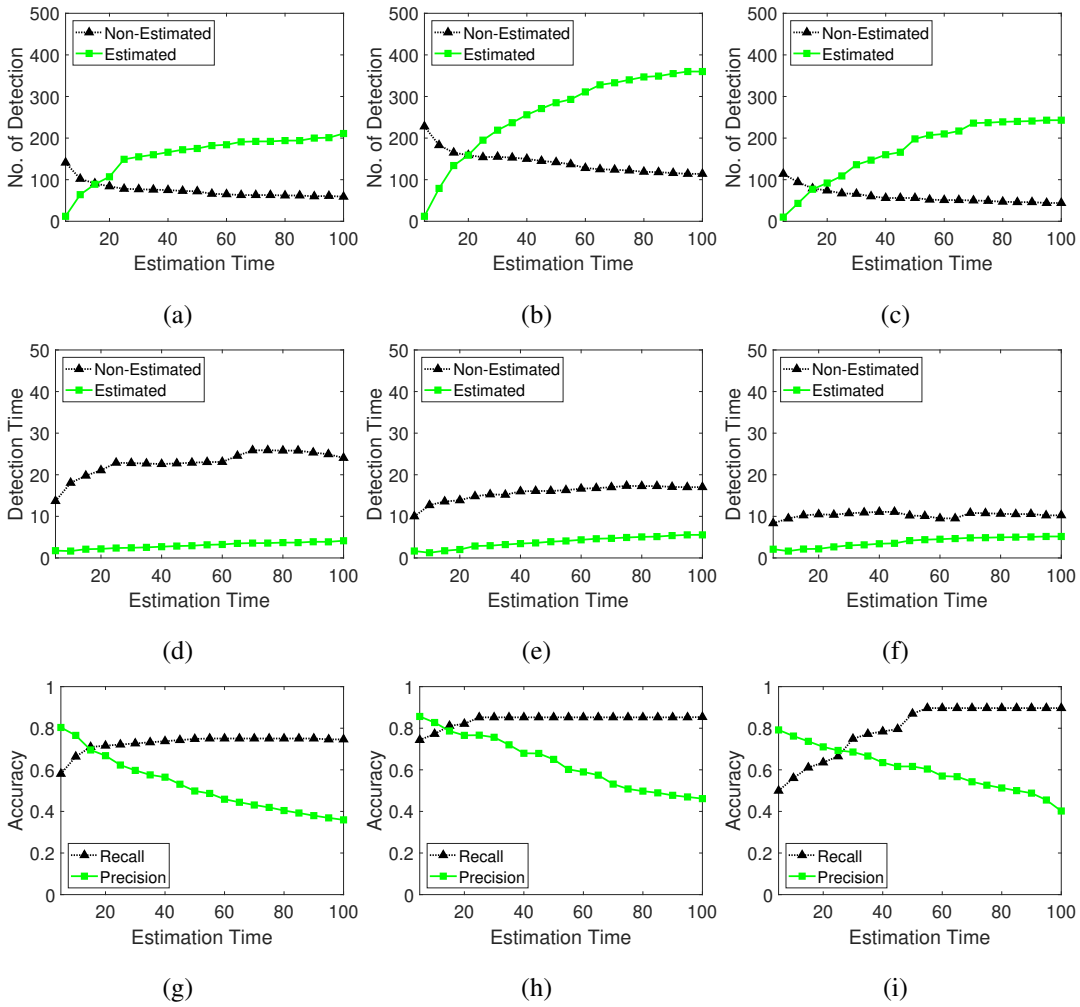


Figure 3.14: Estimation time of the memorized estimator method. Test datasets are video 1 for (a,d,g), video 2 for (b,e,h), and video 3 for (c,f,i). (a–c): Number of detections. (d–f): Tracking time. (g–i): Recall and precision.

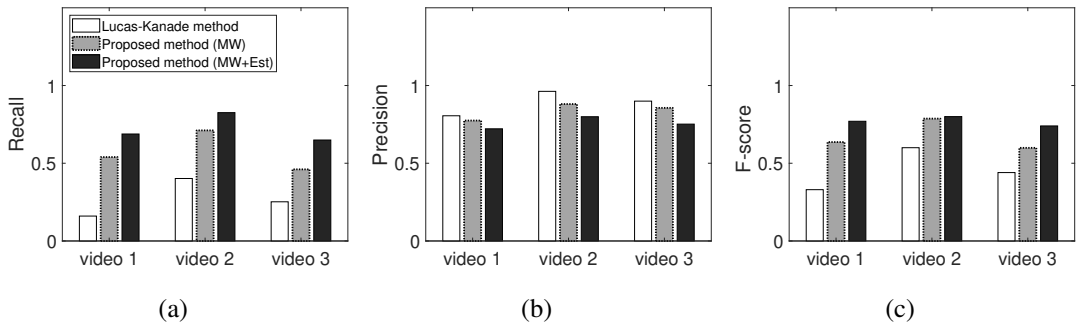


Figure 3.15: Performance of proposed methods. (a): Recall. (b): Precision. (c): F-score.

detector and memorized estimator) on the recall and precision for three input videos. The white bar corresponds to a low recall and high precision accuracy, which is not

suitable for detection and may increase the possibilities of accidents pertaining to missing moving objects such as workers in the industrial field. The proposed moving window detector and memorized estimator can help enhance the safety and detection performance. Both methods exhibit higher recall and precision. Moreover, the estimator enhances the true detection rate, and the estimator influences the false detection rate for disappeared or stationary tracking objects. As shown in Figure 3.15c, the F-score slightly increases. In the industrial field, the safety of workers from dangers is more important than the false alarm rate. In particular, mis-detections of workers operating near heavy equipment may lead to fatalities, whereas false alarms may simply be considered cautionary.

3.4 Comparisons

3.4.1 Comparison with other conventional methods

I halve or double the resolutions of the input videos according to the object detection algorithm. Machine learning detection algorithms such as Hog and Haar-like algorithms typically learn objects with resolutions of 64×128. The resolution of videos 1, 2, and 3 is 384×288 and that of video 4 is 768×576. The resolution of videos 1, 2, and 3 is converted to 768×576 and compared with video 4 in terms of machine learning object detection algorithms; the videos are named videos 1_{L}, 2_{L} and 3_{L}. Moreover, the resolution of video 4 is converted to 384×288 for optical flow object detection algorithms, and it is named video video 4_{S}.

The LK optical flow method using in the comparison experiment is a typical moving object detection algorithm based on sparse optical flow, and a simple noise filtering function of the moving distance threshold is applied between the locations of the previous and present pixel. The Farneback method is also a typical moving object detection algorithm based on dense optical flow and simple noise filtering function of the optical flow magnitude threshold for pixels. The Hog and Haar-like object detection algorithms are famous machine learning classification methods based on the learning weights of pedestrian. All the algorithm codes are sourced from the official OpenCV community site .

The method consists of the moving window system and the target estimator. Although the computational speed of the proposed algorithm is similar to that of typical sparse

Table 3.1: Comparison of proposed method with object detection methods using optical flow; video 4_S indicates a small scale of images from video 4.

Methods	Input	TP	FP	FN	Precision	Recall	F-score	FPS
Proposed Method	video 1	653	123	277	0.84	0.70	0.77	48.02 ($\pm$0.70)
	video 2	1038	207	303	0.83	0.77	0.80	45.31 ($\pm$0.50)
	video 3	595	156	258	0.79	0.70	0.74	46.36 ($\pm$0.93)
	video 4_S	4028	609	622	0.88	0.86	0.87	30.96 ($\pm$1.71)
Lucas–Kanade Method (Bouguet et al., 2001)	video 1	185	20	745	0.90	0.20	0.33	51.70 ($\pm$1.56)
	video 2	583	24	758	0.96	0.43	0.60	50.75 ($\pm$1.28)
	video 3	249	29	604	0.90	0.29	0.44	50.51 ($\pm$1.13)
	video 4_S	3736	476	914	0.89	0.80	0.84	36.62 ($\pm$1.49)
Farneback Method (Farnebäck, 2003)	video 1	654	159	276	0.80	0.70	0.75	4.27 ($\pm$0.06)
	video 2	1041	264	300	0.80	0.78	0.79	4.29 ($\pm$0.05)
	video 3	563	112	290	0.83	0.66	0.74	4.23 ($\pm$0.05)
	video 4_S	3442	417	1208	0.89	0.74	0.81	4.23 ($\pm$0.06)

optical flow in Table 3.1, as indicated by the FPS in the embedded system (Raspberry Pi 4), it achieves a higher recall and precision. Even at a slightly lower FPS, the proposed method outperforms the existing algorithm in terms of the recall, precision, and F-core. The dense method exhibits a low FPS but recall, precision, and F-score are similar to the proposed method, which performs extensive calculations for the optical flow vector and magnitude of all pixels. This finding indicates that the proposed method can optimize the costs of the applied functions (moving window detector and memorized estimator) and achieve a higher performance than the existing moving object detection algorithms based on the optical flow. In experiments on video 4_S, a lower FPS than other videos is achieved, and there are more moving objects in terms of the TP. The results for video 4 corresponds to a slightly increased recall but comparable precision. As mentioned, videos 1–3 have slow walking people, whereas video 4 has many people who walk rapidly. The proposed method exhibits a reasonable prediction performance in the presence of vibration noise. In the case of videos 1–3, the proposed method can effectively distinguish the slowly moving target and vibration noises. In contrast, the LK method evaluates slowly moving targets as vibration noises and thus cannot detect moving targets. In video 4, many pedestrians move rapidly, and thus, the LK method can effectively detect targets. In other words, the proposed method can robustly distinguish vibration noises and moving targets.

Table 3.2 indicates that the recall, precision, and FPS for the proposed algorithm are

Table 3.2: Comparison of the proposed method with other pedestrian detection methods; video 1_L, 2_L, 3_L indicate a large scale of images from videos 1, 2, 3.

Methods	Input	TP	FP	FN	Precision	Recall	F-score	FPS
Proposed Method	video 1_L	708	118	222	0.86	0.76	0.81	16.58 (±0.40)
	video 2_L	1076	208	265	0.84	0.80	0.82	16.43 (±0.28)
	video 3_L	645	119	208	0.84	0.76	0.80	16.61 (±0.29)
	video 4	4117	440	533	0.90	0.89	0.89	12.20 (±0.09)
Pedestrian Detector (HOG) (Dalal and Triggs, 2005)	video 1_L	5	2	925	0.71	0.01	0.01	1.29 (±0.01)
	video 2_L	40	37	1301	0.52	0.03	0.06	1.29 (±0.01)
	video 3_L	2	2	851	0.50	0.001	0.001	1.28 (±0.02)
	video 4	2886	63	1764	0.98	0.62	0.76	1.29 (±0.01)
Pedestrian Detector (Haar-like) (Viola and Jones, 2001)	video 1_L	6	1	924	0.86	0.01	0.01	2.04 (±0.02)
	video 2_L	22	0	1319	1.00	0.02	0.03	2.00 (±0.02)
	video 3_L	1	10	852	0.09	0.001	0.001	1.96 (±0.03)
	video 4	2912	341	1738	0.90	0.63	0.74	1.83 (±0.02)

higher than the typical pedestrian classifiers, Hog and Haar-like methods. The Hog and Haar-like detection algorithms incur high calculation costs because they calculate the masks of the pixels and classify whether the pixels are objects from prebuilt learning weights. Notably, machine learning models cannot effectively detect objects that are distorted or rotated from those in the learning model. In the case of videos 1–3_L, the proposed method detects many moving objects, but the Hog and Haar-like methods miss the objects owing to distortion. In video 4, which is typically used for machine learning detection algorithm, the proposed method exhibits a comparable precision and higher recall than the compared algorithms. The findings indicate that different sizes and distortions of pedestrian objects affects the detection accuracy of Hog and Haar-like methods. In other words, the proposed method can outperform the machine learning algorithms in moving object detection.

3.4.2 Comparison in noisy videos

Table 3.3 summarizes results for noisy video frames (blurred, poisson, gaussian, and salt-pepper noise in four videos), obtained using the proposed method and other methods. The considered noises are representative types of image noise. In the case of blurred noise, pixels in the blurred frame are filtered and averaged with five neighbouring pixels. Poisson noise is a type of electronic noise generated with the averaged distribution of extended scaling to the input pixel values. Gaussian noise is generated with gaussian distribution of zero mean and 0.01 variance. Salt-pepper noise is gener-

Table 3.3: Test results with blurred, poisson, gaussian and salt-pepper noises.

Methods	**Input**	**Blurred**				**Poisson**			
		Precision	**Recall**	**F-score**	**FPS**	**Precision**	**Recall**	**F-score**	**FPS**
Proposed Method	video 1	0.68	0.65	0.67	48.54	0.80	0.59	0.68	47.81
	video 2	0.83	0.75	0.79	45.34	0.80	0.72	0.76	45.37
	video 3	0.78	0.57	0.66	45.76	0.72	0.64	0.68	45.10
	video 4_S	0.86	0.88	0.87	31.43	0.89	0.85	0.87	31.27
Lucas–Kanade Method (Bouguet et al., 2001)	video 1	0.55	0.14	0.23	51.12	0.74	0.17	0.27	51.09
	video 2	0.82	0.33	0.47	51.34	0.75	0.36	0.49	50.51
	video 3	0.64	0.26	0.37	50.87	0.59	0.28	0.38	49.96
	video 4_S	0.92	0.58	0.71	36.49	0.90	0.52	0.66	35.88
Farneback Method (Farnebäck, 2003)	video 1	0.86	0.65	0.74	4.27	0.63	0.64	0.64	4.26
	video 2	0.86	0.73	0.79	4.30	0.86	0.70	0.77	4.29
	video 3	0.96	0.58	0.73	4.22	0.89	0.52	0.65	4.21
	video 4_S	0.90	0.74	0.81	4.21	0.86	0.70	0.77	4.21

Methods	**Input**	**Gaussian**				**Salt & Pepper**			
		Precision	**Recall**	**F-score**	**FPS**	**Precision**	**Recall**	**F-score**	**FPS**
Proposed Method	video 1	0.59	0.52	0.55	47.29	0.63	0.43	0.51	46.18
	video 2	0.68	0.69	0.68	44.65	0.64	0.54	0.58	43.46
	video 3	0.65	0.54	0.59	44.76	0.52	0.53	0.52	44.20
	video 4_S	0.88	0.77	0.82	30.21	0.86	0.69	0.77	29.43
Lucas–Kanade Method (Bouguet et al., 2001)	video 1	0.40	0.13	0.20	50.13	0.29	0.07	0.11	48.56
	video 2	0.72	0.25	0.37	49.87	0.33	0.14	0.20	46.93
	video 3	0.57	0.22	0.32	49.01	0.30	0.12	0.17	47.04
	video 4_S	0.91	0.47	0.62	35.19	0.80	0.37	0.50	33.85
Farneback Method (Farnebäck, 2003)	video 1	0.70	0.32	0.44	4.18	0.51	0.31	0.38	3.63
	video 2	0.88	0.49	0.63	4.19	0.78	0.48	0.60	3.74
	video 3	0.83	0.35	0.49	4.17	0.56	0.35	0.43	3.60
	video 4_S	0.76	0.63	0.69	4.04	0.73	0.63	0.68	3.37

ated with a noise density of 0.05, which affects 5% of the pixels. Blurred and poisson datasets correspond to weak noise, and gaussian and salt-pepper datasets correspond to strong noise. Table 3.3 is presented in two parts: The upper table corresponds to weak noise, and bottom table corresponds to strong noise. The datasets include original noises from camera vibrations and image quality, and I add more intense noises to the datasets to verify the robustness of the methods against noise.

The Hog and Haar-like methods exhibit inferior detection performance and low robustness in various environments because of the fixed pretrained filter weight, distortions, and various sizes of the moving targets in the frame. I compare the LK and Farneback methods (as sparse and dense optical flow methods, respectively). The parameters of

the compared methods is optimized to ensure a fair comparison. For the LK method, the corner quality parameter ranges from 0.01 to 0.1 to avoid a large number of corners being generated on the noise. For the Farneback method, I set the mean N parameter from 3 to 30 to normalize the baseline of decisions between noises and moving targets, which smoothens the generated optical flows. The precision, recall, F-score, and FPS results are compared.

The results presented in Table 3.3 are considered to evaluate the robustness of the proposed method. The precision, recall and F-score of the proposed method for videos 1, 2, 3, and 4_S is slightly deteriorated. The performance of the LK method is significantly deteriorated on the noisy datasets, especially in the case of salt-pepper noise. The LK method cannot effectively decide whether the feature points are noise or moving target owing to a large number of features generated on the noise. For the Farneback method, the recall is significantly decreased in the case of gaussian and salt-pepper noise videos. Because the Farneback optical flow is calculated on all pixels including noises, it can eliminate the noise effects, but loses the sensitivity of moving target detection. in the case of blurred noise, the Farneback method calculates the optical flow that is lower than that for normal datasets, resulting in slightly lower recall and higher precision. In contrast, the proposed and LK methods lose precision owing to the ambiguous generated corners on the blurred spot. The result of video 4_S are similar for all methods in Table 3.1; however, in the cases shown in Table 3.3, the performance of the LK and Farneback methods are considerably different. This video has fast moving and many pedestrians without noises. However, the addition of blurred, gaussian, and salt-pepper noise renders video 4_S challenging, more moving objects are detected as noise and vice versa.

The proposed method memorizes the location history of each feature point in the moving window and tracks the target in the window. The proposed method can effectively distinguish the additional noise and moving targets because each feature point has its own window memory. The proposed method outperforms the LK method and achieves a higher FPS than the Farneback method. Therefore, Table 3.3 demonstrates the robustness of the proposed method in noisy environments.

3.5 Summary of Chapter 3

Moving object detection and tracking are technologies applied to wide research fields including traffic monitoring and recognition of workers in industrial fields. However, the conventional moving object detection methods have faced many problems such as much computing time, image noises, and disappearance of targets due to obstacles. In this chapter, I introduce a new moving object detection and tracking algorithm based on the sparse optical flow for reducing computing time, removing noises and estimating the target efficiently. The developed algorithm maintains a variety of corner features with refreshed corner features, and the moving window detector is proposed to determine the feature points for tracking, based on the location history of the points. The performance of detecting moving objects is greatly improved through the moving window detector and the continuous target estimation. The memory-based estimator provides the capability to recall the location of corner features for a period of time, and it has an effect of tracking targets obscured by obstacles. The suggested approach was applied to real environments including various illumination (indoor and outdoor) conditions, a number of moving objects and obstacles, and the performance was evaluated on an embedded board (Raspberry pi4). The experimental results show that the proposed method maintains a high FPS (frame per seconds) and improves the accuracy performance, compared with the conventional optical flow methods and vision approaches such as Haar-like and Hog methods.

Chapter 4

Data selection for deep learning model

Several vision-based object detection models have been applied on image databases. Deep learning models including YOLOv5 can be applied to object detection in fisheye images. However, towing to the geometric distortions and perspective changes, adaptation is required to recognize objects and pedestrians in these fisheye images. Furthermore, the environmental conditions, such as indoor or outdoor and day or night, also influence the object detection. In general, a single deep neural network model learned from an environmental dataset is not well fitted to another image dataset. Fisheye lens cameras have a wide viewing angle, which can be used for monitoring pedestrians or automobiles with a vehicle. Images from six fisheye cameras mounted on a hydrogen-powered bus as well as from another type of fisheye camera on an excavator vehicle were collected under diverse environmental conditions and camera positions. The vehicles are supposed to monitor a moving path of pedestrians in the surrounding area, with the potential for developing autonomous driving in the future.

The custom image datasets are classified into six category subjects depending on the environmental conditions, whether the images are captured indoor or outdoor, day or night, or the views are at low or high angles. The capability of deep neural networks learned for a subject being run compatibly for another subject is investigated. Each subject dataset has training data and test data. The YOLOv5 architecture was applied to each image dataset for training and a cross test among six subjects was conducted. From the results, an influence graph is built between a set of subject datasets by comparing how much the performance of image classification is improved when a deep neural network trained for one subject is tested on another subject. In the experiment,

the relationship graph of the custom subjects is determined by training 5%, 20%, and 100% of the entire dataset for each subject and testing the trained network on the validation datasets for the other subjects. Interestingly, similar influence graphs for inter-subject relationship were obtained even with varying percentages of the training set. If a positive effect between a pair of subjects is found in the influence graph, the dataset can be included in the training data.

In this chapter, a cross-test approach is proposed between a pair of subjects to confirm if the subjects have different coverage for object detection. It can easily mark the shadow zone for a pair of the custom subjects in which objects or pedestrians are not detected in the dataset for a given subject but are satisfactorily detected for another subject used for training. From the cross-test approach and the influence graph, it can be determined whether a deep neural network for a subject can be compatibly used for another subject, and also whether it can choose or collect relevant subjects of image data to improve the accuracy performance of object detection for a given subject. This content has been submitted to a journal (Choi et al., 2022a).

4.1 Methods

This study aims to establish an efficient strategy for the selection of training data for object detection to reduce computing time and regulate the amount of input data needed for training. First, I check the effect of each dataset through training and validation according to the ratio of each dataset, the independent effect on the experimental dataset that can be obtained from the hydrogen bus, and the open dataset defined as a common base set. Thereafter, it is applied to six fisheye images obtained in real time to YOLOv5s using multi-fisheye cameras with multiple channels and construct a system that informs the driver of the results. In addition, it builds the influence graph among the six custom training datasets using test performances and prove coverage area on different datasets.

Figure 5.1 shows overview of the proposed strategy. A large amount of data is generally required when various environments are tested or when one wish to learn image data of distorted images or test images under varying illumination conditions. In the object detection problem using deep learning, the higher the number of data to be learned, the more helpful it is to improve detection performance. It is acquired to training sets collected in six diverse environments and prepared two representative public training

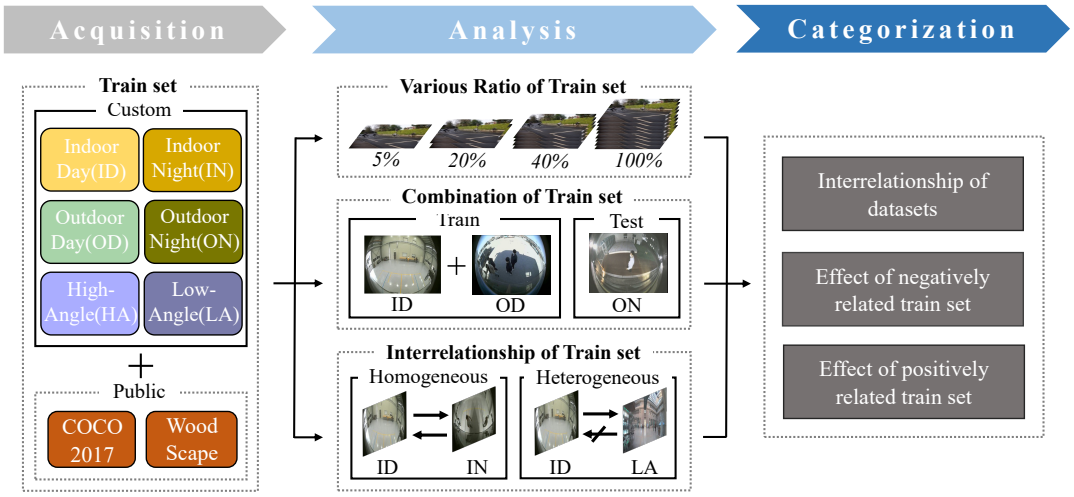


Figure 4.1: Overview of the proposed method.

sets for use in object detection. It is used that various ratios of custom training sets to determine the most efficient ratio for a training dataset. Using two combinations of training sets, I tested another training set and verified the impact of the additional training set. It is studied the relationship between the custom training set and homogeneous and heterogeneous training sets. By analyzing the custom training set, it determines its characteristics, correlation, and image filtering. Therefore, the core goal of the proposed method is to properly cross-learn the common base set and custom data required in the current problem scenario so as to increase the recognition rate in any problem scenario. The following subsection introduces how to utilize the proposed method in this chapter, the public data of COCO and Woodscape, and custom datasets acquired from distorted camera images to improve object detection performance.

4.1.1 Various portion of dataset

Neural network models are dependent on the sort of training data provided. All training sets include the public datasets of COCO and Woodscape as a base training set. The common base set has a large number of images, compared to the custom datasets. The features of the datasets can be analyzed by evaluating the test performances. Each of six custom datasets has its own training set and test set. It allows a set of options for training a custom dataset. Training is conducted on custom datasets comprising 5%, 10%, 20%, 40% 60%, 80%, and 100% of the entire training set. For every model-training experiment, the common base set consisting of COCO and Woodscape is always included in combination with the above choice. It tries to determine the percentage of

data that would be sufficient to model a given dataset. From the result, efficient data selection for various scenarios can be achieved.

4.1.2 Combination of dataset

It tries to verify the improvement of detection rates on the test datasets, by combining two training sets. For example, learning two diverse types of datasets can result in improved performance for another type of test set. Even if all of the learning data acquired from these results are not learned, learning unnecessary data can be prevented by classifying data with similar tendencies and learning among the corresponding data groups. In this study, public data is basically always included in learning data, and various experiments are conducted while additionally configuring custom data as learning data. When combining data, two or more custom data types are added.

When there are two or more data types, it must be verified that specific types of data maintain their learning influence. When combining two types of custom data, the first data is called custom 1 and the second data is called custom 2. To verify the independence of the two custom data, it compares the performance change between learning only custom 1 data and learning the combination of custom 1 and custom 2 data. As public data is always included in learning, public data and custom 1 data are included when learning only custom 1 data; it refers to this as the base training set, which is the basis for two combinations of custom data. The public data consists of COCO and Woodscape datasets, and I abbreviate those public datasets as C. The public dataset plays a role of the base set and is included at any time for learning related to the common base set. It is often chosen either the training set as the base set alone or the base set plus one of six custom subjects (ID, IN, OD, ON, LA, HA). Here, the training set is denoted by B.

If a common base set B is defined for neural network training, one subject of the dataset can be added to the common base set in order to create a new training set. It is then observed the effect of the model oriented towards one subject on another target subject by comparing the performance with the common base set and with the common base set with the addition of the subject. In the experiments, the common base set is set up using COCO and Woodscape, or one subject plus those two datasets. In fact, the common base set can be extended to multiple subjects plus the base set C. To observe the effect of a subject on a target subject, the network learns one subject X together with the common base set and it is tested on a target subject Y. Based on the result, I can

find the relationship between subject X and subject Y. It can find the influence graph for pairs of subjects where the edge provides information of how much the performance can be improved from that of the common base set.

$$\begin{cases} B : Y \\ B + X : Y \end{cases} \tag{4.1}$$

In Equation 4.1, B represents one of seven base datasets, which are COCO+Woodscape (denoted as C) datasets, ID, IN, OD, ON, HA, and LA. X represents training data which is different from B. Y represents the test dataset after learning B or B+X dataset. It can be denoted to the set $B = \{ID, IN, OD, ON, HA, LA\}$ union C, $X = \{ID, IN, OD, ON, HA, LA\}$, and $Y = \{ID, IN, OD, ON, HA, LA\}$. It will train dataset X with various datasets B and test on dataset Y. This method verifies whether a particular dataset X influences a target dataset Y regardless of all the base sets.

$$\alpha = f(b + x, y) - f(b, y) \tag{4.2}$$

In Equation 4.2, it defines the influence value as α. In the function $f(b+x,y)$, $b \in B$ is one of the base training data, $x \in X$ is added to the training data, and $y \in Y$ is a test dataset. The parameter $b+x$ in $f(b+x,y)$ indicates the training dataset composed of b and x datasets. The next parameter y is a testing dataset. The output value of $f(b+x,y)$ is the test performance on the data set y by the neural model trained with the data $b+x$.

$$\begin{cases} X \underset{\hat{\alpha}}{\longrightarrow} Y, & \text{if } \hat{\alpha} \geq \tau \\ X \underset{\hat{\alpha}}{\dashrightarrow} Y, & \text{if } \hat{\alpha} \leq -\tau \end{cases} \tag{4.3}$$

In Equation 4.3, the normal black-colored arrow indicates a positive relation, and dashed arrow indicates a negative relation. For example, it would like to examine whether the influence of learning dataset OD is maintained when OD is learned with other training datasets. Accordingly, I set X as OD and train OD with other six base training sets (C, C+ID, C+IN, C+ON, C+HA, C+LA) one by one. Using the trained models, it is evaluated for the model on one test set (ID, IN, ON, OD, HA, LA).

For validating the trend on learning effects of X, I pick the representative index value $\hat{\alpha}$ among the set of α. First, it is eliminated meaningless values and outliers in the set

of α. Second, I choose the median value of the set as $\hat{\alpha}$. If $\hat{\alpha}$ is over the threshold τ, it depicts the value $\hat{\alpha}$ on the arrow. Finally, the arrows indicate the trends on the learning effect of dataset X. When the $\hat{\alpha}$ is very high, it shows a thick arrow to indicate a strong trend or a thin arrow to indicate a normal trend.

4.1.3 Relationship of dataset

Through the combination of training using a variety of ratio of custom datasets and using custom datasets, it can be seen that each dataset has its own characteristics. Through the preceding methods, I analyze the data characteristics in the training set I have. For example, data acquired indoors features little change in the background because the camera is fixed, while the background in data acquired outdoors frequently changes because the camera is moving on the bus. Data acquired in an environment where there are many changes in the background may have characteristics that make it more difficult to learn the target object than in an environment where the background does not change. Even in the case of day and night, in data acquired at night, it is difficult to distinguish between object and background, so the object detection model will not properly learn the features of the object. Depending on whether the angle of the camera acquiring the data is high or low, the object used for the learning model may suffer from distortion. The environmental characteristics obtained from these data can be derived by analyzing the results of model training in various ratios and combinations of training datasets.

It uses characteristics of datasets to examine the influence and relationship of custom datasets on each other. First, in the experimental results of combination of base sets, combinations with another custom dataset are represented as an influence graph. This can help to infer the characteristics of the base training set and its relationship with other data by learning and testing it with five other custom datasets fixed. For example, if a specific dataset B is determined and the influence on X and other test sets after learning each of the five X except B is generally positive, it can be said that B acts as a positive catalyst with other datasets. In contrast, if it is generally negative, it can be said to serve as a sub-catalyst. If the influence is a mixture of positive and negative, the relationship can be considered irrelevant.

4.1.4 Coverage effects of dataset

From Section 4.1.3, it is a selected pair of related datasets in the influence graph, then learn each dataset and evaluate it on the same dataset. I want to check the negative and positive influences from the actual detection result photos according to the relationship that the learned dataset has on the evaluation dataset. Here, I define the shadow zone between a pair of the custom subjects where objects or pedestrians are not detected in the dataset for a target subject but are well detected for the subject used for training. To draw the shadow zone on an image canvas for a target subject, a window size of 5×5 pixels in a frame are considered as tiles and the mis-detection rate is calculated on each tile.

If the mis-detection rate of a tile is over a given threshold of 0.3, the tile belongs to the shadow zone, and it is painted with translucent and blue-colored polygons of density 0 to 1.0. The shadow zone refers to false negative (FN). The mis-detection rate is calculated by 1-recall at each tile. If the detection rate of a tile is over a given threshold of 0.7, the tile belongs to the green zone, and it is painted with translucent and green-colored polygons of density 0 to 1.0. The green colored zone is the inverse of the shadow zone, which is the same as true positive (TP) at each tile. It can be also considered to false-alarmed zone, which is same as false positive (FP).

When $X \dashrightarrow Y$, which is a negative relationship, the characteristics of dataset X, which do not match the characteristics of dataset Y, will be learned, and this shows the areas where detection fails in the result of C+X:Y. Such an area becomes a shadow zone that the learning dataset X has for the evaluation dataset Y. When $X \longrightarrow Y$, which is a positive relationship, the characteristics of dataset X that fit well with the characteristics of dataset Y will be learned, and this shows the areas where detection succeeds in the result of C+X:Y. By training and testing with negative datasets, I can find it difficult to learn the shadow zone in the target dataset. With positive datasets, it can identify the features of positive datasets that can complement the shadow zone, which is difficult to learn. Thus, learning datasets with specific relationships one by one makes it possible to identify the shadow zone and the dataset characteristics that complement this area.

I observe the effects of adding a positive dataset to a learning dataset. From the results of validating two combination-dataset learning, it examines the effectiveness of learning multiple negative and positive datasets and datasets with positive relationships together. When the negative-related dataset and the positive dataset are learned together

and this combination set is applied to the evaluation dataset, the area that was not detected by the negative dataset can be reinforced by the effect of the positive dataset. In particular, when two datasets with positive relationships are learned together and applied to the evaluation dataset, the detection coverage areas of each positive dataset are combined and improve detecting performance.

4.2 Experimental environment

4.2.1 Camera measurement system

The fisheye camera is an ultra-wide-angle lens camera that produces strong visual distortion intended to create wide panoramic or hemispherical images (Miyamoto, 1964). Fisheye lenses capture the wide-angled camera view. Instead of capturing images in the way of straight or rectified linear perspective, fisheye lenses use own mapping method, which captures images in a convex non-rectilinear perspective. In the typical 35-mm film format of lens, the normal focal length of the fisheye lens is 8 mm to 10 mm for circular images and 15 mm to 16 mm for full frame images. This type of lens also has other applications, such as reproducing images taken through fisheye lenses or generated through computer graphics onto a hemispherical screen. The fisheye lens is also popular in scientific photography and geometry, and is widely used as a peeping hole door viewer to provide users with a wide view.

In this experiment, I use a 35-mm circular fisheye camera having DX-format, producing cropped circle images as shown in Figure 4.2. The circular fish lenses consider a 180° hemisphere and projects the image within a circle frame. Some circular fisheye lenses were used in an orthogonal projection model. The fisheye camera has a vertical viewing angle of 180° and a horizontal and diagonal viewing angle of 180°. Most circular fisheye lenses have a smaller central area of the frame than other conventional lenses, so the edges of the frame tend to be completely dark. The fisheye cameras are mounted onto six sides of the bus as shown in Figure 4.2. These cameras capture images of the front, back, front-left, front-right, rear-left, and rear-right as shown in Figure 4.2. The camera is supposed to detect pedestrians or bikers moving around the bus with a wide range of frame. The wide view allows the camera to capture many objects in the region of interest.

Figure 4.3 shows the foreground of the excavator, the appearance and location of the

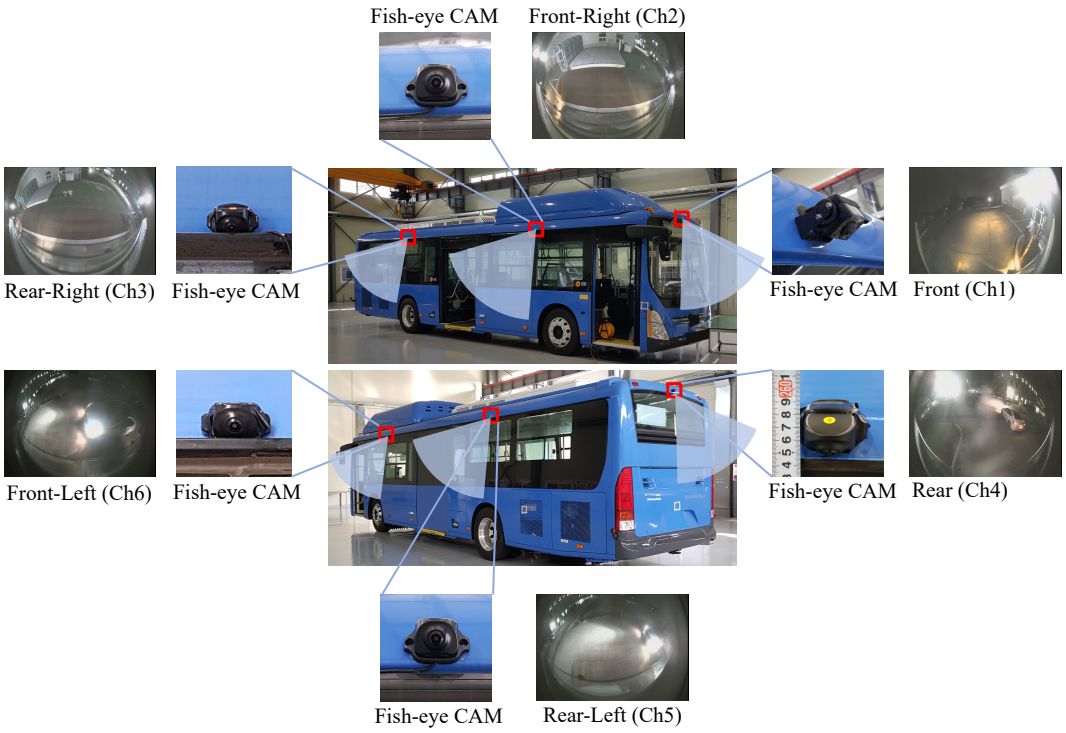


Figure 4.2: Fisheye cameras settings mounted on a hydrogen-powered bus; six fisheye cameras were used to collect the surrounding images.

fisheye camera attached to the excavator, and the data acquired from this camera. This camera also captures views of 180° vertical, horizontal, and diagonal angles. The camera is attached to a very high and low height, and the angle is also very high and low. The top-located camera takes a downward direction at a high angle, showing high distortion. The bottom-located camera captures forward at a low angle, showing low distortion.

The collection of the custom image datasets are classified into six subjects depending on whether the images are captured indoor or outdoor and in daylight or at night as well as from a high angle or low angle of the view position: Indoor Day (ID), Indoor Night (IN), Outdoor Day (OD), Outdoor Night (ON), High-Angle (HA), and Low-Angle (LA). The ID, IN, OD, and ON subjects are obtained from a hydrogen-powered bus and the HA and LA subjects from an excavator (actually the excavator cameras are repositioned in the indoor environment). The datasets ID, IN, OD, and ON are acquired on a moving bus under various environments. The datasets HA and LA are acquired in a fixed excavator environment. The same types of fisheye cameras are mounted on a bus but a different type of fisheye camera with a change of image distortions is used for the excavator.

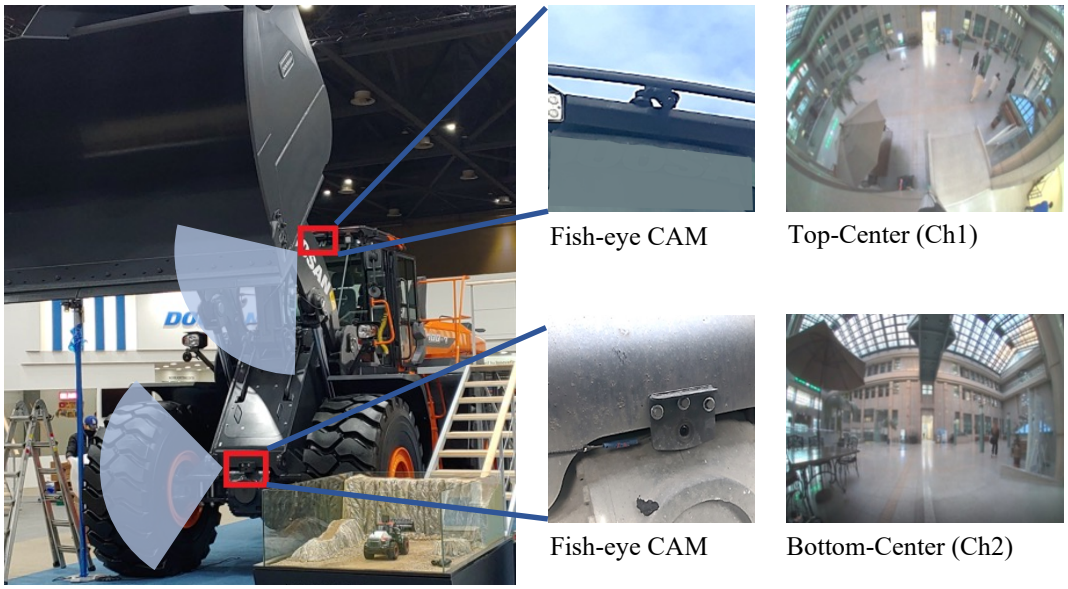
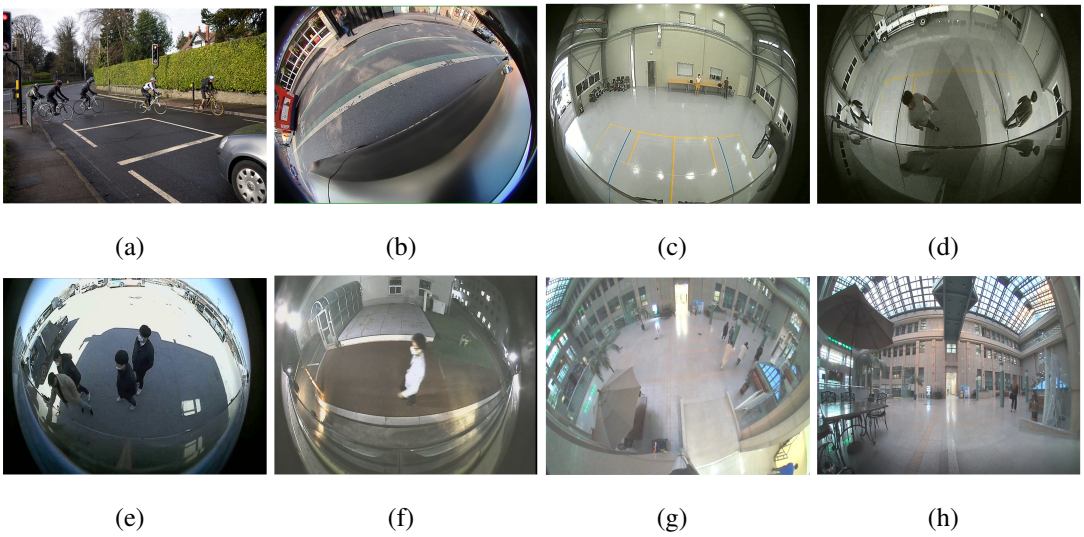


Figure 4.3: Fisheye cameras settings mounted on an excavator.

4.2.2 Dataset

In the experiment, I use eight types of datasets shown in Figure 4.4; (a)–(b) two common base sets (open-access public data) and (c)–(h) six custom image datasets (a collection of image datasets acquired manually). It is defined for object classes as Person, Bicycle, and Car (PBC) for the deep learning model; these classes are designed for the safety of pedestrians on the road. The COCO dataset has 71,103 images having persons, bicycles, and cars, the Woodscape dataset has 11,249 fisheye images. The custom datasets are collected with fisheye cameras having a wide-angle view. The images were taken in indoor and outdoor environments during the day and the night. Another dataset of images was acquired at two different positions of a fisheye camera, called High-Angle view, which is captured at 4-m height, and Low-Angle view, which is captured at 1.5-m. The datasets (c)–(h) have 845, 391, 2515, 2565, 1000, and 1000 images, respectively. Each dataset shows its own environmental characteristics of various environments with illumination, camera altitude, and distortion.

In table 4.1, the characteristics of the custom dataset are classified by dividing them into several categories. The characteristics of the data are classified by considering the environment in which the data was obtained, the time zone, the state in which the camera was attached, and the angle of the camera while looking at the dataset image. The ID dataset was acquired indoors during the day while the camera was fixed on a stationary bus, and the IN dataset was acquired during the night. OD datasets



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.4: Examples of public and custom datasets. (a): COCO 2017. (b): Woodscape. (c): ID. (d): IN. (e): OD. (f): ON. (g): HA. (h): LA.

Table 4.1: Properties of six custom datasets.

Dataset	Images	Environment	Illumination	Camera Distortion	Camera Ego-motion	Camera Angle	Camera Location
ID	845	Indoor	Bright	High	Fixed	High	Front
IN	391	Indoor	Dark	High	Fixed	High	Right
OD	2515	Outdoor	Bright	Middle	Moving	High	Front
ON	2565	Outdoor	Dark	Middle	Moving	High	Right
HA	1000	Indoor	Bright	High	Fixed	Very High	Top
LA	1000	Indoor	Bright	Low	Fixed	Low	Bottom

were acquired outdoors during the day with cameras fixed to buses traveling around the city, while ON datasets were acquired during the night. The HA and LA datasets were obtained from indoor and daytime fixed surveillance cameras, characterized by a wider range of filming than the ID and IN datasets. In this section, I will systematically establish the characteristics of the data through a more detailed analysis method and examine the relationship between the characteristics of the data.

4.2.3 Deep learning model

You-Only-Look-Once (YOLO) is a typical single-step object detection algorithm based on deep learning. The advantage of this deep learning model is to see the full image and process the object location and classification in a single stage. The YOLOv5

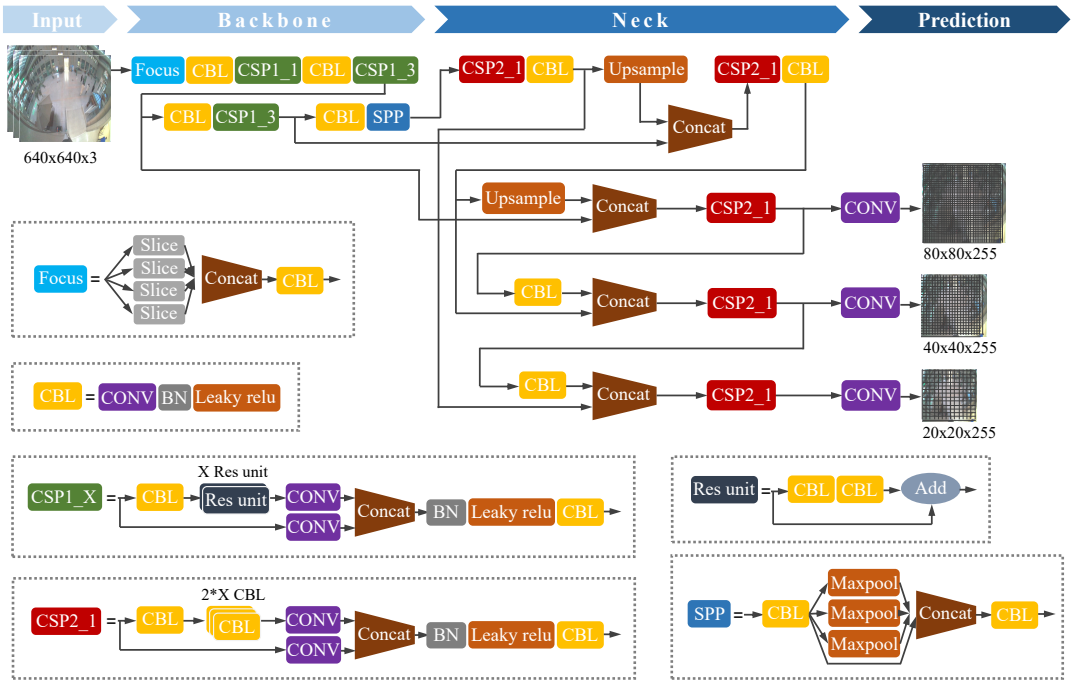
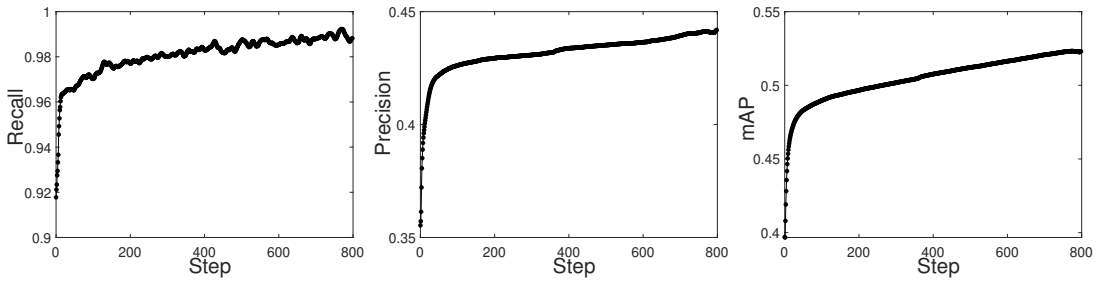


Figure 4.5: Architecture diagram of YOLOv5 model network (modified from (Gao et al., 2022)).

model, i.e., the latest version, achieves high performance and real-time object detection. I use YOLOv5 in the PyTorch framework to reduce computing costs. There are various versions of the YOLOv5 model depending on the number of layers and parameters. Performance is typically measured through Mean Average Precision (mAP) and Intersection over Union (IoU). Network training was performed using an AMD Ryzen 9 3900X CPU and two RTX 2080TI SLI GPUs.

Figure 4.5 shows the architecture diagram of the YOLOv5 model. The model consists of three types: a backbone that extracts features, a neck that improves the detection performance by fusing the extracted features, and a head (prediction) that converts features into bounding box parameters. The backbone extracts feature maps of various sizes from input images through multiple layers of convolution and pooling layers. Convolution with batch normalization and leaky ReLU (CBL), cross stage partial (CSP), and spatial pyramid polling (SPP) techniques are used in this layer. CBL is a block consisting of a convolution layer, batch normalization, and leaky ReLU active functions, and is a block that is used by default to extract features. The CSP is a method of performing convolution operations on only a portion of the feature map and integrating it with the rest. By passing only a few feature maps through the convolution



(a) (b) (c)

Figure 4.6: Recall, precision and mAP performance on the public training set with 800 iterations of epoch.

layer, the computation volume can be reduced, and performance can be improved by efficiently performing gradient flow in the back-propagation process. SSP improves performance by pooling feature maps into filters of various sizes and merging them again.

The neck part fuses various sizes of feature maps. It improves performance by mixing low-level and high-level features using path aggregation network (PAN). The head part converts the feature extracted from the neck into the final output of the network using the convolution layer. It converts the parameters of the bounding box (x,y,w,h), the probability that the object exists, and the probability that the class exists.

4.3 Experimental results

4.3.1 Results with various portion of single dataset

To determine the training period for an iteration of sample learning, I monitored the transition of performance as the epoch increased. For the results, the YOLOv5s trained the common base set of COCO and Woodscape. The performances were mostly saturated at approximately 50–100 epochs as shown in Figure 4.6. Thus, the epoch for training time was set to 100 iterations for the experiments described below.

Figure 4.7 shows the mAP for each custom dataset depending on the various portions of the training dataset. From these results, approximately 5% of the entire training set can be sufficient to train the characteristics or trend of a pattern of the training dataset without OD dataset, which needs more than 5% of the dataset for similar performances of other datasets. If stable performance is needed, it can select 10% of the dataset

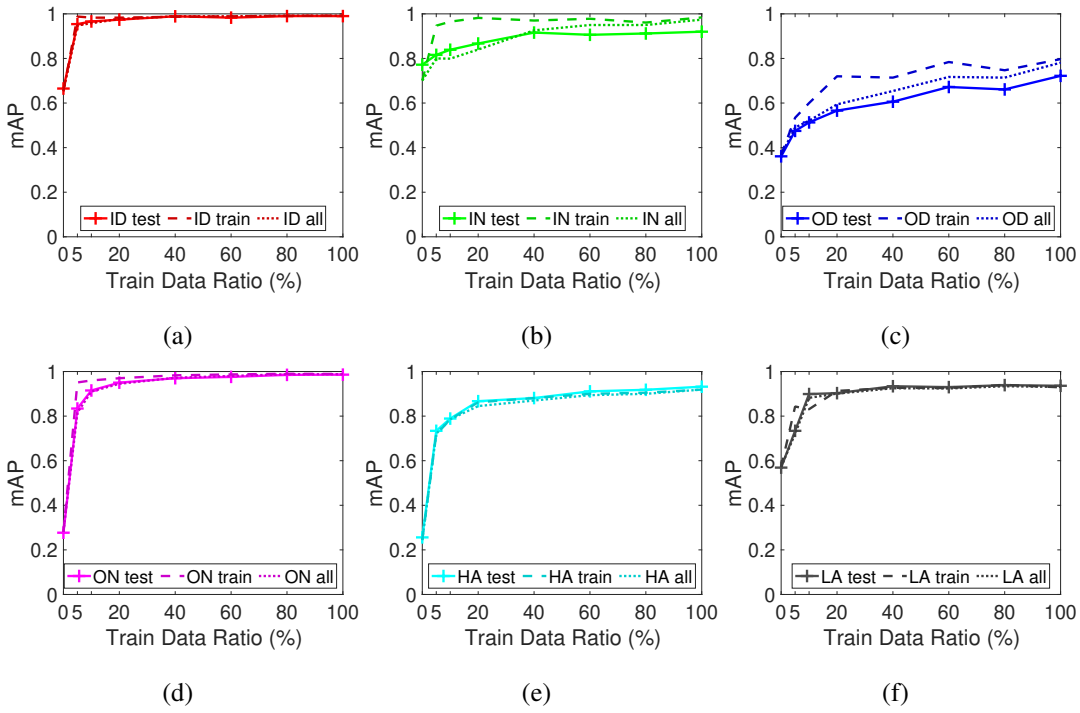


Figure 4.7: Changes in portions of the full training set. (a): C+ID:ID. (b): C+IN:IN. (c): C+OD:OD. (d): C+ON:ON. (e): C+HA:HA. (f): C+LA:LA.

instead of 5%. A relatively small portion of the training dataset can capture the trend of the dataset. These datasets are considered to have homogeneous properties to keep the data patterns consistent. The feature map of images within a custom dataset may be similar and the training model can easily detect the fundamental features with a small portion.

The ON and HA datasets show large improvement, meaning that the characteristics of both datasets have been easily trained. The environment of the ON dataset is dim illuminations; the objects are indistinguishable from a dark background. However, a small portion of the dataset is sufficient to support training. The proposed approach works well for this type of image dataset, where images cannot be easily used from common base set. The LA dataset contains a distorted image of a fisheye lens, but it has a view similar to the camera view commonly found in common base set (Woodscape dataset). Figure 4.7 shows the relationships between custom datasets. After training each selection of the portion of the training set for the custom dataset, the effect was obtained from the average performance against the test sets of the other custom datasets. Percentages indicate changes in performance compared to after training with only common base set. It indicates that the characteristics of the training dataset are

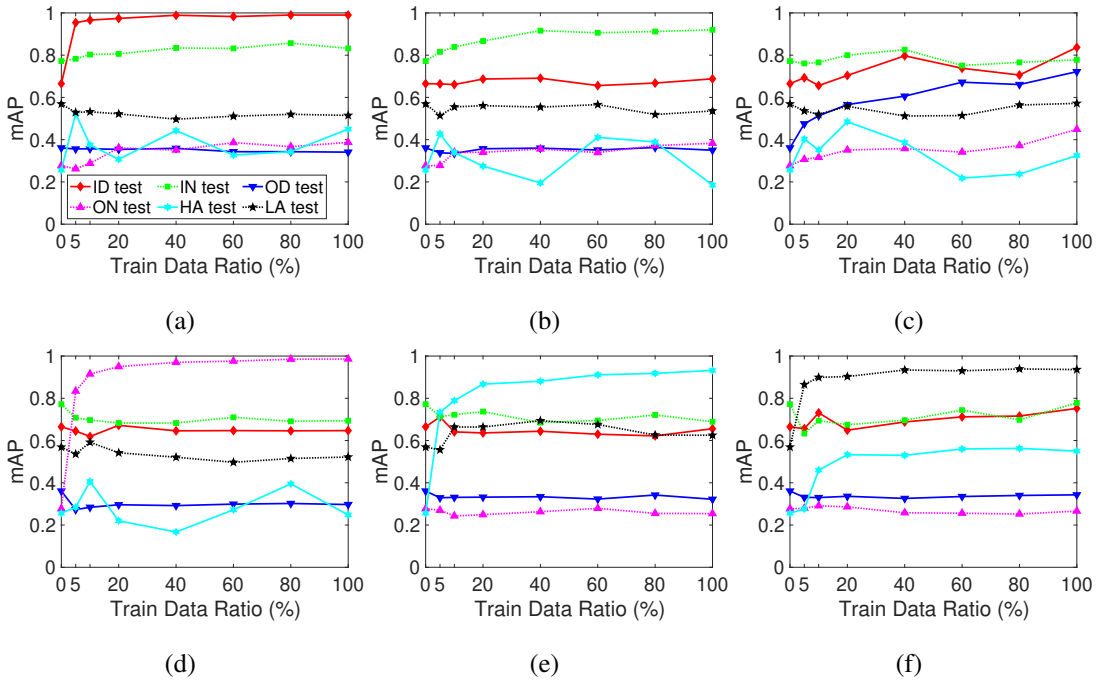


Figure 4.8: Performance according to the various percentages of the training set. (a): Test of training ID. (b): IN. (c): OD. (d): ON. (e): HA. (f): LA.

relevant.

In contrast, the OD dataset seems to contain data that takes time to be trained in YOLOV5s. It is probable that there are a bunch of types of heterogeneous datasets in the OD dataset. The image was taken from an outside driving bus camera. The dataset includes images distorted by fisheye cameras and dynamic background images in motion frames, as well as crowds and distorted images of pedestrians hidden by adjacent objects. Training dynamic environment datasets provides a hint that it is more difficult than clean and reliable image datasets. In the indoor dataset environment (see Figure 4.7(a)-(b)), objects are clearly distinguished because a certain level of lighting is observed within a particular dataset. This type of image dataset seems to be widespread in the common base set.

Figure 4.8 shows performance mAP according to eight percentage sets of the training set, as tested on the six test sets. Six colored lines in the graph denotes tested datasets. The test sets are indoor-day, indoor-night, outdoor-day, outdoor-night, high-angle, and low-angle, and they are shown in red, green, blue, magenta, cyan, and black colored lines, respectively. It is trained each according to the ratio of the number of data singly using six custom datasets, and applied them to the test set of the common base set

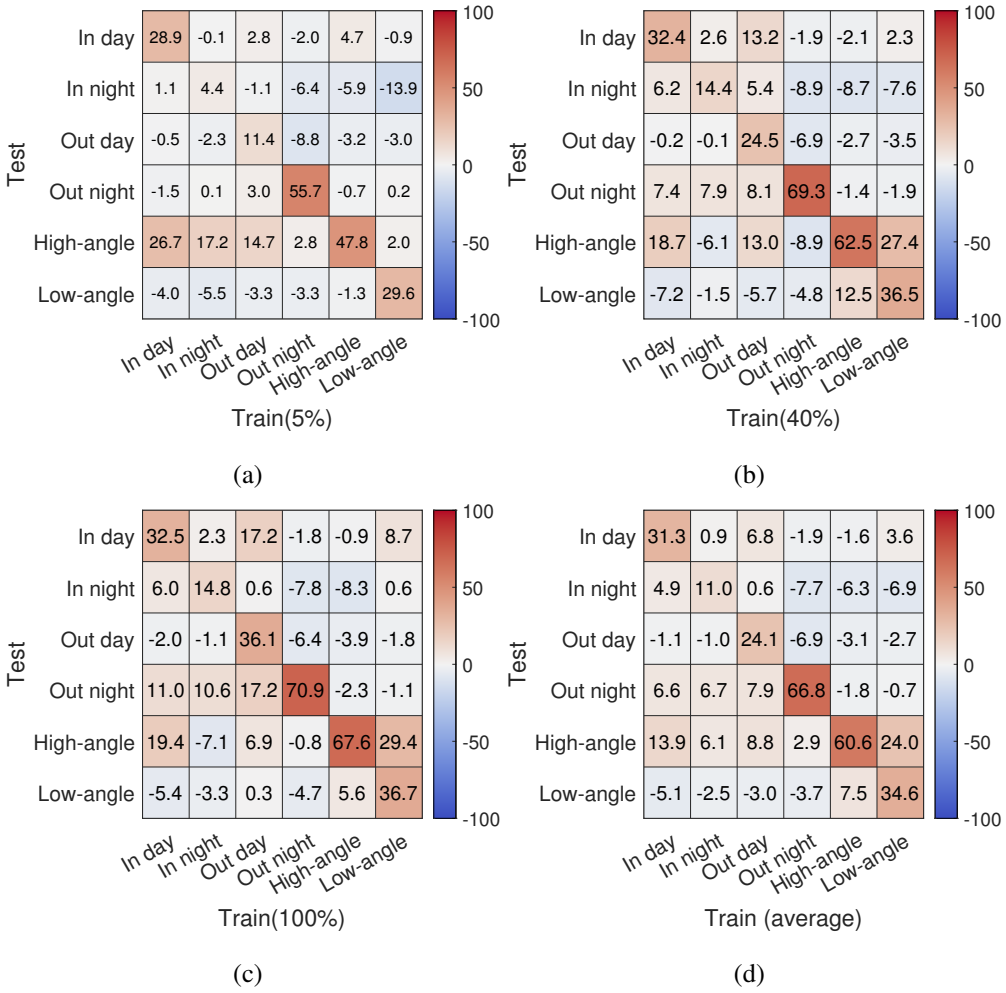


Figure 4.9: Difference from only training common base set. (a): 5% of each training set. (b): 40%. (c): 100%. (d): average of 5 to 100%.

and six datasets to derive the results in Figure 4.8. This is according to the ratio of the training sets, and 0% is the result when only a common base set is trained. In addition, it shows the result of training each single dataset, and the test performance of the single trained dataset is significantly improved. In particular, the results of the indoor day in Figure 4.8(a) and the outdoor night in (d) are clearly improved. The results of high angle and low angle in Figure 4.8(e),(f) are somewhat irrelevant.

Figure 4.9 shows the difference from the common base set training performance result by fixing the ratio (e.g., 5%, 40%, 100%) of six custom training sets and applying them to each test set. Figure 4.9(d) also shows the average test performance of all ratios of the training set. This allows me to estimate the correlation between the single datasets and the learning rate of the single datasets. For example, high-angle dataset and low-

angle dataset have a close relationship. In contrast, the outdoor night dataset and indoor night dataset have a relationship that hinders training performance.

4.3.2 Results of combination of base dataset

Table 4.2 summarizes the results obtained by applying the combination of two training data to each single test data with training data ratio 100%. The meaning of Y is six test sets and the meaning of X is six training sets added to the base training set, denoted as B+X:Y. Each numerical value at colored cell indicates a change of test performance from learning only base training set B to learning added-on base training set B+X model. While the results of the previous single dataset were derived based on the test results of the common base set, the results through these two combinations were combined pairwise from six custom datasets, and the single training performance of one of the two datasets were all It was applied to a single dataset. For example, after training by combining the indoor day dataset and the indoor night dataset, check the difference based on the 100% learning result of the indoor day. The combination for the test consists of indoor day+indoor night, indoor day+outdoor day, indoor night+outdoor day, indoor night+outdoor night, high angle+low angle, and indoor day+low angle.

In Table 4.3, trends of training effects on each datasets are similarly maintained in both the combination results at training ratios of 100% and 20%. Through this, it is possible to identify the characteristics of single data based on the common base set and to grasp the correlation between the common base set and the data more closely based on one type of single data. For example, when training by combining indoor day and indoor night, the high-angle test performance is good in two datasets based on the common base set (Table 4.2), but it is actually significantly lower based on the indoor day. Conversely, based on indoor night dataset, it is better at a higher value than lowering by indoor day dataset.

In Table 4.2, it is learned in combination with the remaining six base training sets that hold the training set X and do not contain X, and each learning model on six test sets is evaluated. For each learning model, the evaluation results are shown in the line of the graph. This can be expressed as B(var)+X(fixed):Y(fixed), i.e., as a relational expression. The line graph analysis helps to examine which base training set differs from a specific X through the evaluation of a learning model including various base training sets. The yellow line is the performance change between C+X and C learned models without other custom datasets, which represents the pure influence of X. The

Table 4.2: Test performance after 100% of learning each custom dataset according to the seven base training sets.

X	Y					
	ID	IN	OD	ON	HA	LA
ID	0.325	0.06	-0.02	0.11	0.194	-0.054
	0	0	0	0	0	0
	0.302	0.017	0.016	0.014	0.143	-0.009
	0.154	0.052	0.006	-0.01	0.03	-0.002
	0.344	0.148	0.035	0	0.186	0.126
	0.334	0.161	0.035	0.095	-0.005	0.063
	0.237	0.064	0.005	0.092	0.004	0.008
IN	0.023	0.148	-0.011	0.106	-0.071	-0.033
	0	0.105	0.025	0.01	-0.122	0.012
	0	0	0	0	0	0
	-0.126	0.153	-0.03	-0.003	-0.075	0.016
	0.007	0.231	0.025	-0.004	-0.045	-0.099
	-0.009	0.238	0.023	0.133	-0.021	0.062
	-0.098	0.144	-0.003	0.097	-0.021	-0.002
OD	0.172	0.006	0.361	0.172	0.069	0.003
	0.001	-0.002	0.387	0.052	-0.095	0.055
	0.023	0.011	0.342	0.063	0.065	0.052
	0	0	0	0	0	0
	0.192	0.002	0.422	0.001	-0.105	-0.06
	0.027	0.141	0.343	0.153	-0.024	0.03
	-0.071	0.04	0.324	0.126	-0.167	0
ON	-0.018	-0.078	-0.064	0.709	-0.008	-0.047
	0.001	0.01	-0.009	0.599	-0.016	0.133
	-0.034	0.005	-0.028	0.599	0.018	-0.113
	0.002	-0.082	-0.003	0.538	-0.182	-0.11
	0	0	0	0	0	0
	0.002	0.008	-0.017	0.729	-0.01	0.069
	-0.063	-0.083	-0.052	0.713	-0.076	0.001
HA	-0.009	-0.083	-0.039	-0.023	0.676	0.056
	0	0.018	0.016	-0.038	0.477	0.173
	-0.041	0.007	-0.005	0.004	0.726	0.151
	-0.154	0.052	-0.057	-0.042	0.583	0.083
	0.011	0.003	0.008	-0.003	0.674	0.172
	0	0	0	0	0	0
	-0.093	-0.063	-0.015	-0.007	0.375	0.007
LA	0.087	0.006	-0.018	-0.011	0.294	0.367
	-0.001	0.01	0.007	-0.029	0.104	0.429
	-0.034	0.002	-0.01	-0.02	0.344	0.398
	-0.156	0.04	-0.055	-0.057	0.058	0.364
	0.042	0.001	-0.006	-0.007	0.226	0.415
	0.003	0.026	0.006	0.005	-0.007	0.318
	0	0	0	0	0	0

Table 4.3: Test performance after 20% learning each custom dataset, according to the seven base training sets.

X	Y					
	ID	IN	OD	ON	HA	LA
ID	0.309	0.034	-0.007	0.084	0.051	-0.047
	0	0	0	0	0	0
	0.291	0.028	0.012	0.024	0.215	0.013
	0.274	0.035	0.033	0.037	-0.037	-0.009
	0.302	0.12	0.029	0.005	-0.033	0.101
	0.343	0.062	0.003	0.066	-0.008	-0.042
	0.326	0.107	0.021	0.057	0.001	0.004
IN	0.022	0.095	-0.004	0.063	0.019	-0.008
	0.004	0.089	0.015	0.003	0.183	0.052
	0	0	0	0	0	0
	0.032	0.044	0.03	0.021	-0.094	0.119
	-0.031	0.189	0.028	-0.002	0.025	0.065
	0.038	0.123	0.014	0.074	-0.025	-0.028
	0.061	0.224	0.017	0.013	0.058	0.016
OD	0.039	0.028	0.205	0.074	0.229	-0.011
	0.004	0.029	0.245	0.027	0.141	0.027
	0.049	-0.023	0.239	0.032	0.116	0.116
	0	0	0	0	0	0
	0.087	0.016	0.292	0.01	0.072	0.028
	0.174	0.075	0.279	0.098	-0.032	0.036
	0.024	0.153	0.228	0.085	-0.046	-0.006
ON	0.007	-0.089	-0.065	0.673	-0.036	-0.027
	0	-0.003	-0.029	0.594	-0.12	0.121
	-0.046	0.005	-0.033	0.608	-0.03	0.046
	0.055	-0.101	0.022	0.609	-0.193	0.012
	0	0	0	0	0	0
	0.014	-0.056	-0.044	0.708	-0.009	-0.079
	-0.011	0.013	-0.022	0.669	0.005	0.008
HA	-0.029	-0.035	-0.029	-0.028	0.611	0.095
	0.005	-0.007	-0.019	-0.046	0.552	0.1
	-0.013	-0.007	-0.011	-0.017	0.567	0.075
	0.106	0.012	0.045	-0.004	0.35	0.142
	-0.022	-0.002	-0.008	0.007	0.638	0.043
	0	0	0	0	0	0
	0.015	-0.018	0.007	-0.019	0.329	0.014
LA	-0.016	-0.097	-0.025	0.009	0.277	0.334
	0.001	-0.024	0.003	-0.018	0.227	0.385
	0.023	0.032	-0.004	-0.041	0.316	0.358
	-0.031	0.028	-0.002	0.02	0.002	0.339
	-0.034	0.005	0.018	0.005	0.318	0.369
	0.028	-0.08	0.011	0.018	-0.005	0.253
	0	0	0	0	0	0

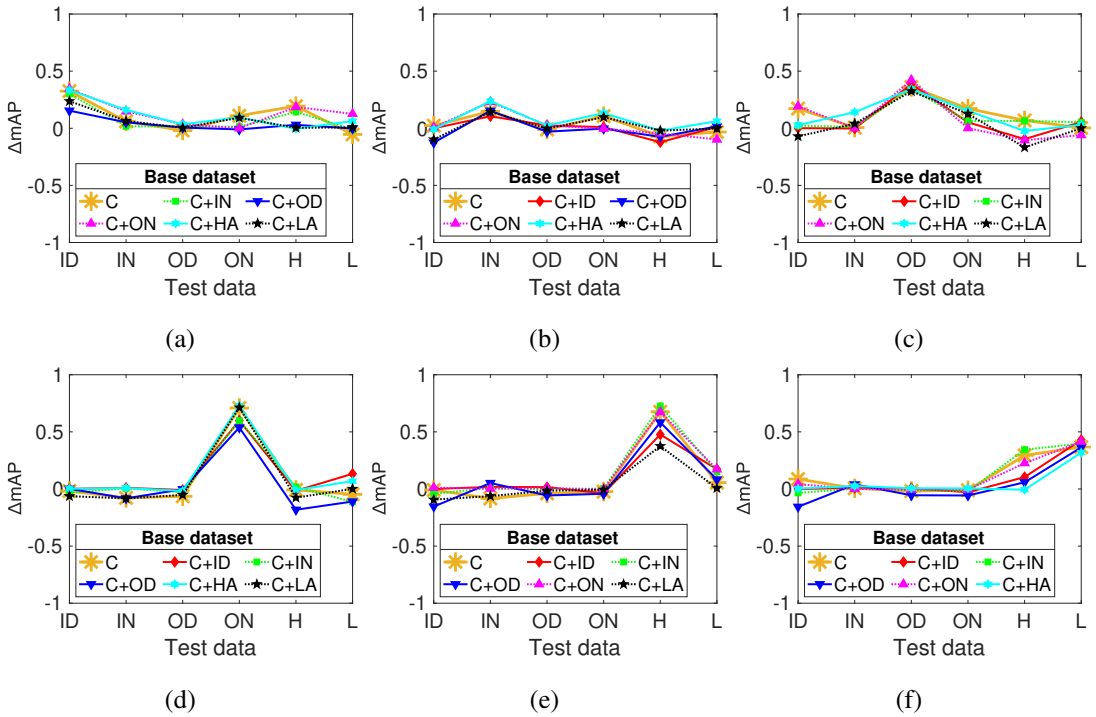


Figure 4.10: Monitoring of various base training set on the six added training set and six test set. (a): Added training set is ID. (b): IN. (c): OD. (d): ON. (e): HA. (f): LA.

yellow line serves as a reference line for examining the relationship between other base training sets and X. The results for the same X and test set are learned and evaluated on the same type of data, so the same influence is maintained on Table 4.2 in general. Therefore, when combining two custom datasets, it can be seen that the effect of the added learning data appears with the effect of the already existing learning data every time one data is added.

In Figure 4.10(a), each plot excepts base training set result same as added training set. Blue and black lines representing the influence of C+OD and C+LA base training sets generally appear lower than yellow baseline, suggesting that OD and LA are heterogeneously related to ID. Unlike ID, the backgrounds of images in OD change rapidly, while in LA, the camera angle faces the front. Likewise, in Figure 4.10(c), the red and black lines representing the influence of C+ID and C+LA base training set appear below the yellow baseline. This can be seen as a heterogeneous relationship where objects are distorted according to the characteristics of data acquired outdoors and indoors as well as the difference in angles of installed cameras.

In Figure 4.10(d), except for C+OD base training set, the remaining base training sets appear to have a positive effect on the ON training set. The data characteristics of the

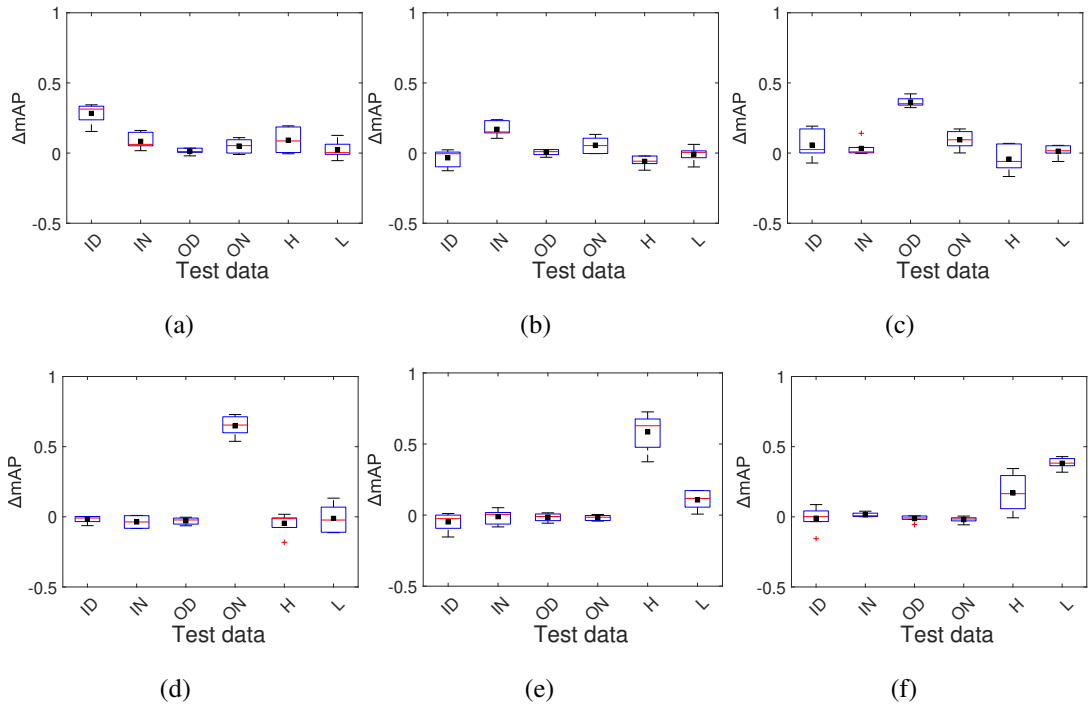


Figure 4.11: Performance distribution for the test set according to the added training set, regardless of the base training set. (a): Added dataset is ID. (b): IN. (c): OD. (d): ON. (e): HA. (f): LA.

outdoor environment in images captured at late night in ON require learning data to supplement the object because the object is not clear. In contrast, OD has data characteristics with varying backgrounds, so OD cannot clearly learn the feature of the object in ON. Therefore, it can be seen as heterogeneous with ON and other custom datasets. From Figure 4.10(d), it can be seen that the properties with respect to the image background influence the learning data. Test results in Figure 4.10(e), (f), and (h) show many deviations between base training sets. Learning HA with C+LA as a base degraded HA performance, while learning LA with C+HA as a base degraded HA performance. The deviation of the base training sets indicates that B and X are not independent, and that the learning data effect of X may be affected by B.

In Figure 4.11, I analyze box plots that visualize summary statistics such as intermediate, maximum, minimum, and outliers to verify the consistency of base training sets' influence on a particular X. When multiple Bs are learned with a particular training set X, the large width of the box means that a particular X does not consistently maintain the learning effect of any B. X, which exhibits large deviations, is influenced by specific B in the results shown in Figure 4.10, indicating that the data characteristics of B

and X are correlated with each other. If an abnormal value appears, it can be seen that the corresponding B and X are in a heterogeneous relationship with respect to test set Y where the abnormal value is shown. The high box appearance in the box plot, which is $X = Y$, indicates that all Xs maintain consistency in their learning effects regardless of B.

4.3.3 Results of relationship on custom dataset

In the previous combination on base training set section, I observed the effectiveness of training by adding custom data as X to another custom data as B. The result of evaluating the training by combining custom data (X) with base training set (B) was obtained in the test set as Y, which is summarized as B+X:Y. When examining the learning effect of X, the differential value obtained by subtracting the result of B:Y from the evaluation result of B+X:Y was viewed as the influence of training set X. That is, the base training set appears as a reference line for examining the influence of the added X. The number of possible training set combinations is equal to the number of combinations of seven datasets (B) and six datasets (X). It is observed the influence of B and the characteristics of the dataset as fixed B and changed X, and I looked at the influence and characteristics of X as fixed X and changed B. In addition, B and X were fixed and various datasets were evaluated as test set Y was changed, and from this, the effect of combination training set and the characteristics of test set were examined.

It is shown the results of the two-combination evaluation obtained in the previous section as a bar graph in Figure 4.12. The six columns accord with six test sets and the six rows accord with six training sets added to the seven base training sets, which are colored bars in each small plot. Performance of the colored bar indicates a change of test performance from learning only base training set B to learning added-on base training set B+X model. The horizontal line represents the added training set X, while the vertical line represents test set Y. The seven bars of various colors in each plot are base training sets, each representing C, C+ID, C+IN, C+OD, C+ON, C+HA, and C+LA results. The two-combination evaluation result table obtained in the previous section is visualized as a bar graph so as to examine the overall influence of added training set X on test set Y on any base training set. As the diagonal component of the matrix is the same as the training set and the test set, overfitting occurs, and the performance evaluation result is highly effective for any base training set. What is unusual is that symmetric components other than diagonal components do not tend to be the

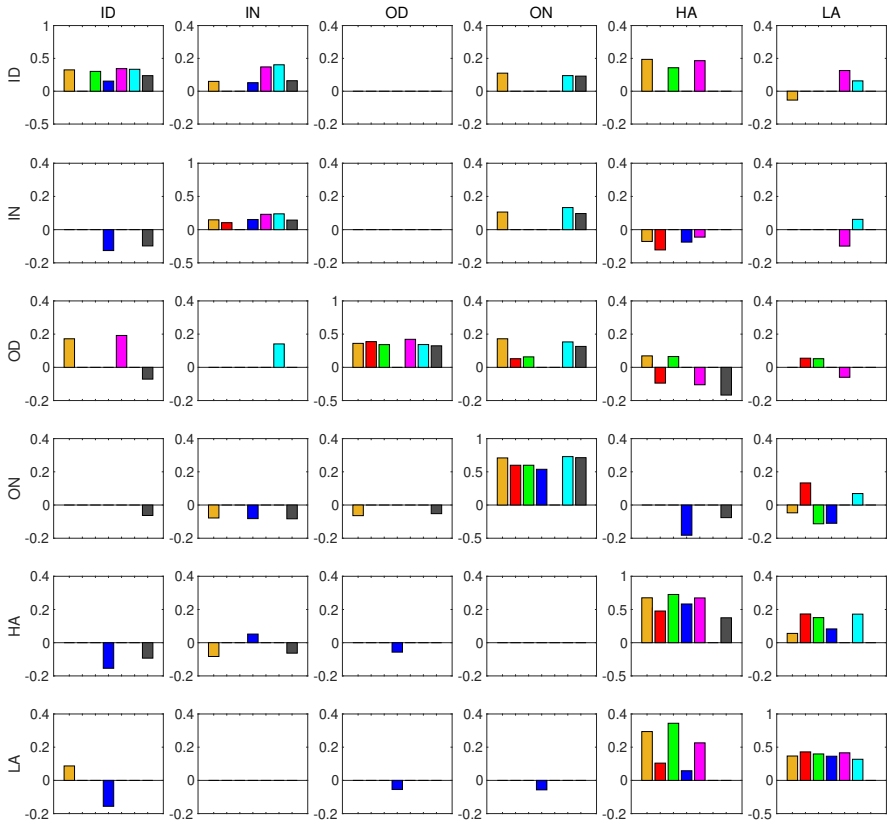


Figure 4.12: Comparison of relationships within the training and test datasets, according to the seven base training sets.

same. This shows that when training data and test data are opposite to each other, their influence may be different. In the future, the relationship between homogeneous and heterogeneous datasets will be examined in more detail through analysis.

In the diagonal component of the matrix, it can be seen that the learning result is not better than that of other datasets. This means that the IN dataset is not well-learned and requires more learning or is already sufficiently learned, so there is no need to learn more. Looking at the results of the single dataset analysis so far, IN showed high-enough learning results using only COCO and Woodscape, which are public datasets, and when learning by adding IN data, the additional learning effect is lower than that of other datasets owing to the dark influence of illumination. In contrast, as the ON dataset was acquired in a darker environment with more weak lighting than the IN dataset outdoors, the learning result was very low when using only the public dataset, and it

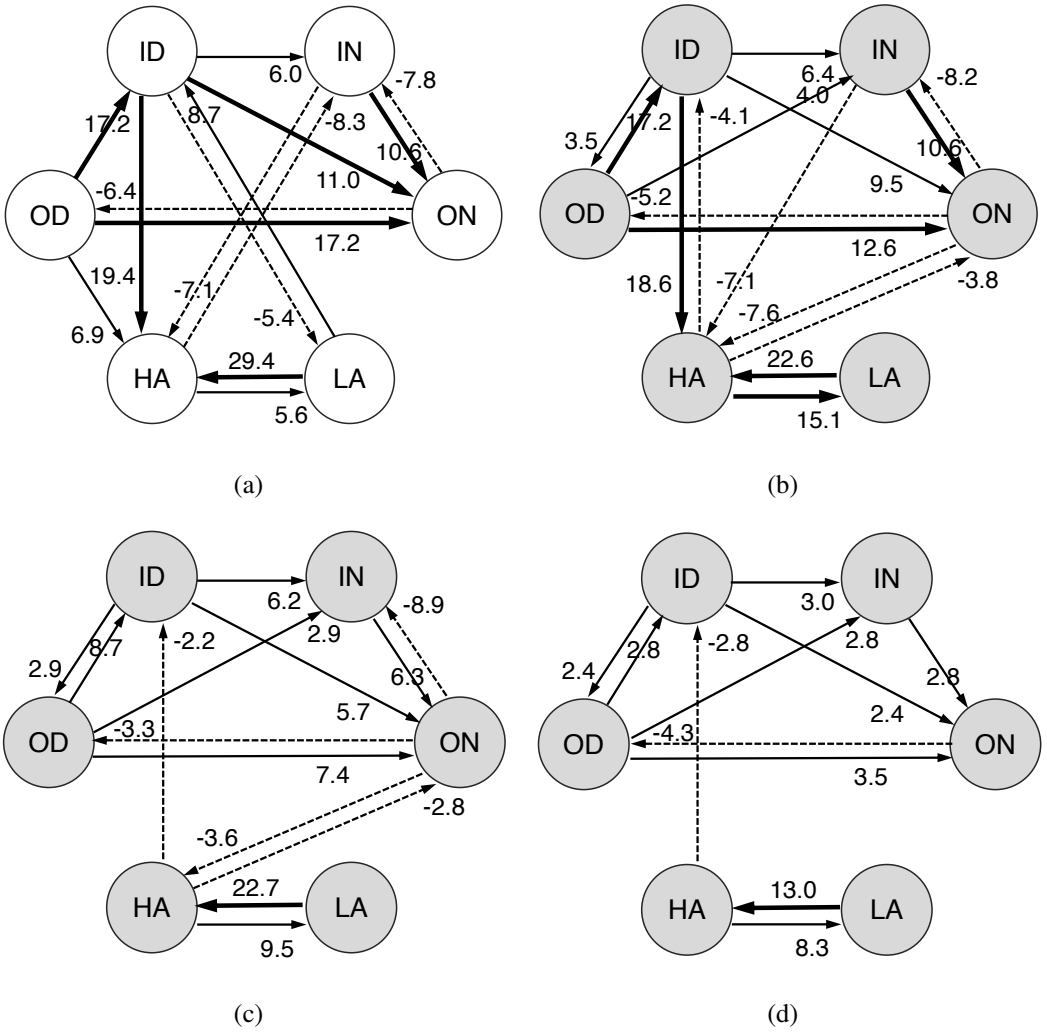


Figure 4.13: Partial ordered influence graph from training two combination of datasets. (a): Only common base set. (b): Considering all base datasets (training dataset ratio to 100%). (c): 20%. (d): 5%.

was confirmed that even a small ON dataset substantially improved the learning result. Although HA and LA datasets are acquired in similar environments, the size of objects in the data tends to be generally small because of the wide range of photographs. This clearly demonstrates the learning effect of the HA and LA datasets, and it can be inferred that the learning effect is better than that of LA because the camera filming the HA dataset was at a very high angle at a high position.

When looking at symmetric components except diagonal components, HA and LA datasets show positive correlations and homogeneous relationships compared to other datasets, whether they are influenced by similar indoor locations. If X is the ID and Y

is the HA, the ID has a positive effect on the HA, whereas if X is the HA and Y is the ID, the effect of the HA on the ID is irrelevant. This allows me to infer a partial ordered relationship because the ID dataset contains more objects at close distances than HA, and thus has a better effect of learning near as well as far objects. Similarly, when X is OD and Y is ID, OD positively influences the learning of the ID, whereas when X is OD and Y is ID, OD datasets obtained outdoors from a more complex background than the ID have a more dominant effect.

In Figure 4.13, it is learned with B fixed with six X and graph the evaluation results for X and five other Y. Nodes in the graph indicate the training set, and each arrow represents median value within the change of performance based on the other base datasets. I assume that there is no relationship when the change of performance is shown to be approximately zero. This is to examine the influence relationship of X according to B by schematizing the line corresponding to each B in Figure 4.10. When X is added to B and trained, the test results for any Y are rather degraded, which is indicated by a negative arrow. Looking at the overall influence relationship of the base training set, all results have a mixture of positive and negative tendencies. Learning with IN and HA dataset along with another dataset X generally improves the learning performance of X, OD and LA dataset reduce X, and ID and ON dataset can be seen as either retaining X or independent of X. This overall relationship distribution of base training sets provides clues as to which dataset to prioritize when constructing training sets.

From a different perspective, it can look at the tendency of the relationship arrows from one node to another for all B. In Figure 4.13, OD in (a), (b), and (c) has a positive relationship with ON, regardless of any base datasets, and IN in (a) and (b) has a negative relationship with HA. HA and LA in (a), (b), (c), and (d) have positive relationships with each other, such that the same two-way relationship indicates that the two datasets are homogeneous types. In the relationship between the ID and IN of (a), (b), (c), and (d), the learning model with the ID is positive for IN.

HA and LA differ in camera angle, which indicates that the objects contained in the dataset are small because they are far away. Here, I can see that the ID dataset with illumination being bright, environment being interior, and camera angle being lower high than HA improves the near-field object detection performance of HA. The positive effects of HA and LA on each other can be considered as being acquired in a similar background environment apart from the items of the currently classified data

Table 4.4: Specified performance of B+X:Y.

Test	**Train**	**P**	**R**	**mAP**	**Test**	**Train**	**P**	**R**	**mAP**
HA	C	0.410	0.281	0.256	ON	C	0.260	0.391	0.277
	C+ID	0.700	0.417	0.450		C+ID	0.421	0.462	0.387
	C+OD	0.585	0.310	0.325		C+IN	0.377	0.469	0.383
	C+LA	0.615	0.547	0.550		C+OD	0.597	0.447	0.449
	C+ID+OD	0.599	0.344	0.355		C+ID+IN	0.382	0.507	0.397
	C+ID+LA	0.757	0.534	0.554		C+ID+OD	0.514	0.489	0.439
	C+OD+LA	0.514	0.408	0.383		C+IN+OD	0.440	0.520	0.446
	C+ID+OD+LA	0.549	0.434	0.409		C+ID+IN+OD	0.446	0.486	0.434

characteristics. In the case of LA datasets, the relationship is negative for other test sets, and when other datasets evaluate LA, the relationship is irrelevant, indicating that the characteristic of camera angle is a heterogeneous relationship between LA and other datasets.

Table 4.4 summarizes the results of evaluation on target subjects ON and HA with a combination of subjects for training; they show strong links in the influence graph shown in Figure 4.13(a). There are selected test datasets from strictly positive bold arrows in the influence graph. Training datasets are COCO and Woodscape common base set denoted to C, positively related single, two, and three-combination datasets. P is precision performance, R is recall, and mAP is the mean average precision. I cannot say that more subject collections for learning are helpful to improve the performance of object detection on a target subject. The relation between a pair of subjects is to check if a deep learning model for one subject can be compatibly run on another subject. Even when one subject influences a target subject positively and another subject is also a positive influence on the target, it may not guarantee the performance improvement with a combination of the two subjects for learning, as the two subjects are not compatible with each other in the influence graph. It implies that learning various environmental conditions together may not be effective.

It is expected that learning a large number of subjects on a target subject may improve the performance of object detection. By identifying the coverage area of a deep learn-

Table 4.5: Characteristics of the fisheye cameras and train-test results among subjects; using the influence graph, positive or negative effects on a target test subject are shown.

Test	Train	Camera Distortion	Camera Ego-motion	Camera Angle	Camera Location	Relationship	Performance(%)		
							Train 100%	Train 20%	Train 5%
ID	HA	High	Fixed	Very High	Top	Negative	-4.1	-2.2	-2.8
	OD	Middle	Moving	High	Front	Positive	17.2	8.7	2.8
IN	ON	Middle	Moving	High	Right	Negative	-8.2	-8.9	3.0
	ID	High	Fixed	High	Front	Positive	6.4	6.2	3.0
	OD	Middle	Moving	High	Front	Positive	4.0	2.9	2.8
OD	ON	Middle	Moving	High	Right	Negative	-5.2	-3.3	-4.3
	ID	High	Fixed	High	Front	Positive	3.5	2.9	2.4
ON	HA	High	Fixed	Very High	Top	Negative	-3.8	-2.8	-2.0
	ID	High	Fixed	High	Front	Positive	9.5	5.7	2.4
	IN	High	Fixed	High	Right	Positive	10.6	6.3	2.8
	OD	Middle	Moving	High	Front	Positive	12.6	7.4	3.5
HA	IN	High	Fixed	High	Right	Negative	-7.1	1.9	-1.1
	ON	Middle	Moving	High	Right	Negative	-7.6	-3.6	2.8
	ID	High	Fixed	High	Front	Positive	18.6	5.1	1.2
	LA	Low	Fixed	Low	Bottom	Positive	22.6	22.7	13.0
LA	HA	High	Fixed	Very High	Top	Positive	15.1	9.5	8.3

ing model and its characteristics, I will analyze the influence effect of how these results come about. I inspect the positive and negative related subjects as per characteristics and performance evaluation of the subjects. I select the candidate subjects in a pair of positive and negative relationships. Calculating mis-detected rate of model learning each inter-related subject, I will mark the shadow zone. It will be helpful to explain the effects of various environmental conditions on the subject.

In Table 4.5, I classify the characteristics classified for each dataset according to the relationship between the datasets discussed in the partial ordered graph. This is to perform efficient data selection when combining data by comparing and analyzing data characteristic items and relationship. Finally, it shows that learning performance changed in the related training and testing datasets according to the training ratio of 100%, 20%, and 5%.

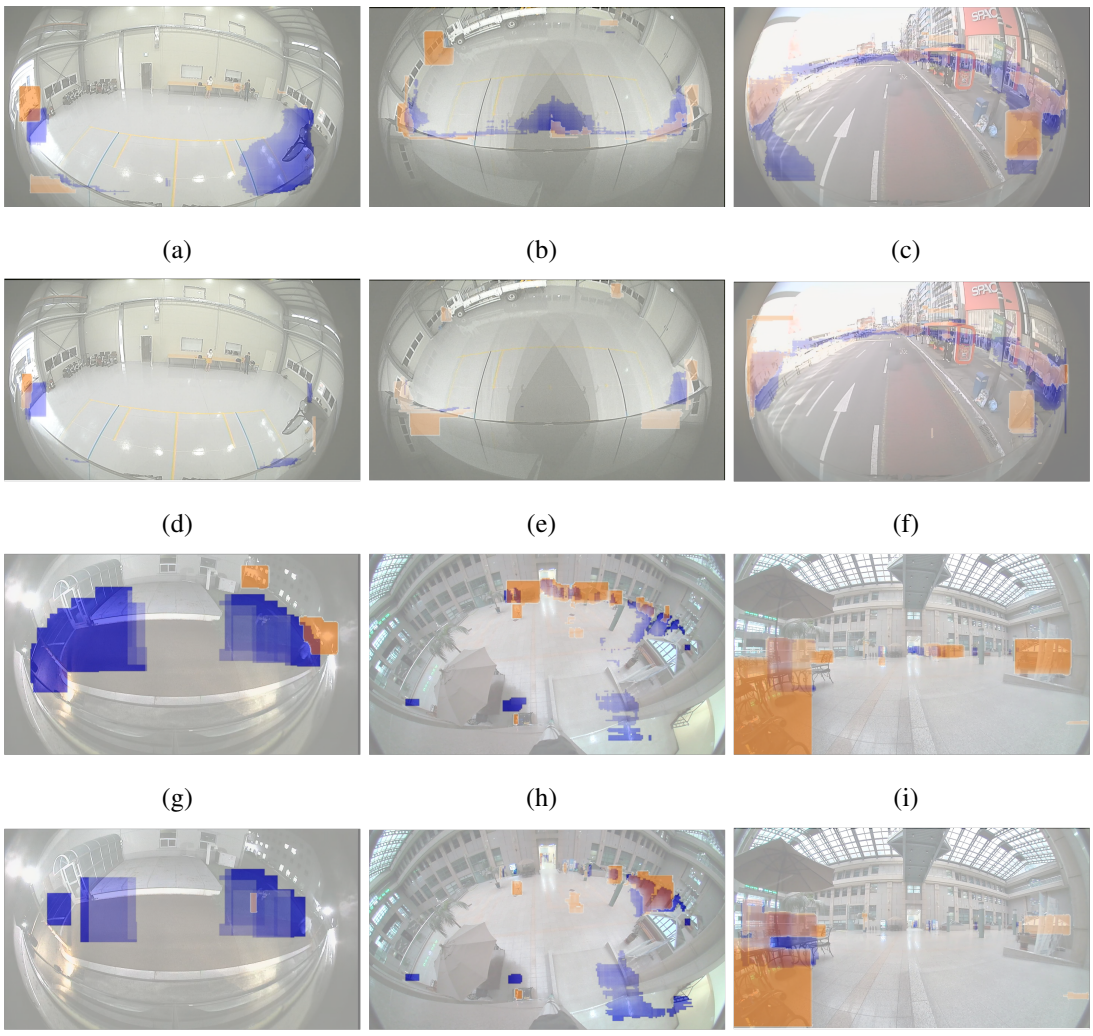
In the case of test set ID, several datasets have a negative effect. The IN dataset is

different from the ID dataset in that illumination is dark, and ON has the characteristic that the environment is outdoors and the camera is attached to the moving bus. It can be seen that these characteristics negatively affect the bright ID dataset. This means that the same occurs when the test set is OD and the training set is ON, and that the characteristic where illumination is bright has an advantage over the characteristic where illumination is dark. When the test set is ON, the IN dataset also has a positive influence even though Illumination is dark, but when the test set is IN, the ON dataset shows a negative relationship, indicating that the influence may differ depending on whether the data are acquired indoors or outdoors. Even when the training set is OD, the characteristic of outdoor environments is that positive or negative relationships alternately appear depending on the test set. Therefore, it can be seen that the illumination characteristic absolutely has a relation where bright is superior to dark, and the environmental environment has a correlation with the environment of the test set.

When it comes to HA, there are some dynamic changed results in performance of training 20% and 5%. It can be explained that subject HA is hard to learn under various environmental conditions such as very high camera angle, and similar indoor place, compared with training IN, ON, and ID. In this case, there should be more training subjects IN, ON, and ID, when the target subject is HA. When training subject is LA, it shows consistent influence on target subject HA. This is because HA and LA have similar indoor environment condition. Thus, I can state that similar environmental condition makes it easier for the model to learn than datasets showcasing diverse conditions.

4.3.4 Results of coverage effects in custom dataset

The results presented in Table 4.5 and Figure 4.13 show the partial ordered relationship between datasets. Translucent and blue-colored polygons indicate the shadow zone, which is hard to detect over a 0.3 mis-detection rate. Translucent and orange-colored polygons indicate the false-alarmed zone, which is hard to detect over a 0.3 false-alarmed rate. In Figure 4.14, I identify images of learning and evaluation results based on these results. Learning and evaluation are conducted by grouping each of the six datasets into a pair of negative and positive relationships. The learning dataset and the evaluation dataset are paired with different types, and the datasets have heterogeneous data characteristics with negative and positive relationships as presented in Table 4.5.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 4.14: Pairs of tested custom datasets on training sets of negative (above in pair) and positive (below) relationships. (a): C+HA:ID. (b): C+ON:IN. (c): C+ON:OD. (d): C+OD:ID. (e): C+ID:IN. (f): C+ID:OD. (g): C+HA:ON. (h): C+ON:HA. (i): C+ON:LA. (j): C+OD:ON. (k): C+LA:HA. (l): C+HA:LA.

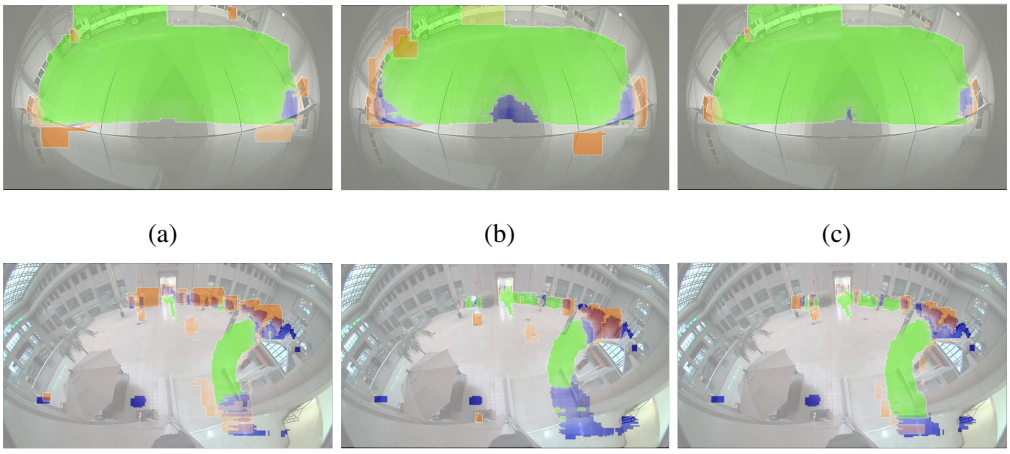
For example, if the test set is the subject ID, the training sets, which have a negative relationship with ID, is subject HA, and a positive relationship is OD as presented in Table 4.5.

The translucent blue area shown in Figure 4.14(a) is characterized by the light of the day and the edge. This blue area refers to an area in which object detection is not well performed owing to the data characteristics. When this is trained by the subject OD, a positive dataset in the partial ordered relationship, the area is improved as shown in Figure 4.14(d). In Figure 4.14(b), a shaded area covered by lighting is seen during the

night time, which is confirmed to be improved when learning with a distinct ID dataset of light and shade. Figure 4.14(c) is when the test set is OD, and owing to the nature of the OD dataset, it is difficult to specify areas where object detection is difficult, perhaps because of the characteristics of frequent background changes during the day. In Figure 4.14(g), It can be seen that the test set is ON and the dark light and distorted edges are not well learned. This can be compressed by learning an OD dataset with bright light and shade. In Figure 4.14(h) and (i), the blue area covers a long distance, and the object is very small, so detection is difficult. This compacts vulnerable areas by learning from each other to LA and from LA to HA. In this way, it is possible to comprehensively learn areas that are not well learned according to data characteristics with complementary characteristics. Also, this shadow zone implies the meaning of inverse recall performance.

The translucent orange area shown in Figure 4.14 shows results of false-alarmed tiles at density of zero to one. Thus, orange-colored zone implies the meaning of inverse precision performance. As the result of Figure 4.14, a positive dataset in the partial ordered relationship makes better results in precision performance than a negative one. It implies that positive related subject learns better object patterns to test subject. Sometimes, there is false-alarmed zone where is actually no labeled in Figure 4.14(b). It is result of learning ID subject and it detects better on the area close to camera. So it detects objects reflected on the mirror of the bus. In Figure 4.14(k),(l), the result of learning LA or HA subject detects easier objects faraway from camera. This result supports the learning effect of the partial ordered relationship and data characteristics.

In Figure 4.15, datasets with positive relationships have been learned one by one and the results of learning two datasets in combination have been presented. Green-colored region indicates green zone and orange-colored region indicates false-alarmed zone. From Table 4.5, I select datasets with positive relationships for the same test dataset. The first and second columns of Figure 4.15 represent the frames of the part that is vulnerable owing to the dataset characteristics. They are the results of learning the single dataset of the positive relationship. The third column shows the enhanced performance in learning combination of these two datasets. The translucent blue area show mis-detected shadow zone. The translucent orange area show detected false-alarmed zone. The translucent green area show detected zone. The shadow zone is the same as false negative (FN) over the mis-detection rate 0.3, at each tile. The mis-detection rate is calculated by 1-recall, at each tile. Orange-colored zone is false-alarmed zone, which



(a) (b) (c)

(d) (e) (f)

Figure 4.15: Result of single and combination learning sets. (a): Result of C+ID:IN. (b): C+OD:IN. (c): C+ID+OD:IN. (d): C+ID:HA. (e): C+LA:HA. (f): C+ID+LA:HA.

is same as false positive (FP) over the false-alarmed rate 0.3 at each tile. Green-colored zone is the inverse of the shadow zone, which is the same as true positive (TP) over the detection rate 0.7 at each tile.

Figure 4.15(b) is the result of learning a dataset OD on a target subject IN, and it is not possible to detect a pedestrian close to the camera in the middle. Figure 4.15(c) are the result of learning datasets ID and OD together. From the result, dataset ID of Figure 4.15(a) improve the shadow zone in dataset OD of (b). Dataset ID in Figure 4.15(a) capture objects close to the camera in the middle better than dataset OD, and dataset OD captures objects around the outside better than ID. By additionally learning OD in the dataset ID, Figure 4.15(c) show the results of reinforcing this area. Figure 4.15(d)-(f) are the results of testing on a high-angle dataset, which contains objects from nearby to faraway distances. It can be inferred that the dataset ID, which can better capture objects at close distances, has a positive relationship with a portion of dataset HA. Dataset LA is a dataset containing long-distance objects as seen in dataset HA, and from this point, it has a positive relationship with dataset HA. Figure 4.15(d) is a result of learning dataset LA, and it is shown that an object that is considerably far away is captured. Figure 4.15(e) is a result of learning a dataset ID, and an object nearby camera is captured. In Figure 4.15(f), dataset LA and ID were learned together, and the performance was improved by reflecting the characteristics of both datasets. As a result, I could verify the enhanced effect of combining two positive datasets using the partial ordered relationship.

4.4 Summary of Chapter 4

Fisheye cameras having wide viewing angles have been used to monitor pedestrians or automobiles from vehicles. Given geometric distortions and perspective changes, adaptation is required to recognize objects. The images are collected from fisheye cameras mounted upon a hydrogen-powered bus and an excavator. The image datasets were classified into six category subjects depending on environmental factors. Then, I developed a cross-test approach to find the relationship graph among a collection of those subjects, which builds up a deep learning model for one subject and evaluates the model on a target subject. It can thus be estimated to the coverage of deep learning models on a target subject for object detection. From the results, it can choose or collect relevant subjects of image data to improve the accuracy performance of object detection for a given subject.

Chapter 5

Pedestrian detection based on convolutional neural network model

In the previous chapter, strategies to cover the shadow zone by combining interrelated datasets were described. The area is shown according to the distorted lens or the viewpoint of the camera. When learning from datasets including many objects located far away from the camera, the deep learning model becomes better at detecting them but still fails to detect them in detail. It is assumed that the poor detection performance is due to small objects suffering from a large loss of input features in the deep learning model. To mitigate this and better detect the object pattern, it only targets the pixel area where the desired object is estimated to exist as the input feature for the model. As the object of interest is a moving object, the area of interest can be extracted through the optical-flow-based object tracking method proposed in Chapter 3. Combining the optical-flow-based method and the deep learning method allows the pixels of the region of interest to be input features of the classification model and reduces computational costs for object localization. To this end, a trained classifier model with a small number of layers and a simple function is needed.

Before applying the neural network model, one can also consider the machine learning-based classification model. The machine learning-based model for object detection is described in the background of Chapter 2. Before studying a light deep learning model, it is possible to think of a machine learning model with a lower computational cost than a deep learning model, but which has a similar function to a deep learning model. There are SVM models that classify objects using a feature vector of preset input dimension

and an effective HOG feature for object detection. Therefore, a method utilizing the optical-flow-based method is proposed so as to reduce the computational cost and improve the performance of the learning-based classification model by focusing on the image features of the region of interest. But, SVM models are vulnerable to distorted object size.

Finally, object detection is performed in a deep learning or neural network model, by entering the ROI containing moving objects extracted from sparse optical flow. With the neural network models, experiments are conducted upon the latest model including YOLOv5, to check whether the YOLOv5 model is effective for ROI extracted from Optical Flow (OF) or not. If the ROI includes multiple objects, the aspect ratio of the ROI is checked and the ROI divided equally according to the ratio. This replaces the localization function of individual objects as much as possible with sparse optical flow, and it will be able to noticeably reduce the computational load. A simple Convolutional Neural Network (CNN) model that is only responsible for classification functions is designed, and then the detection performance and computation speed improvement on the separated ROI are determined. Models that fit the purpose and situation are presented by evaluating performance with existing models that differ from the proposed model. This content will be published in a journal (Choi and Kim, 2022).

5.1 Methods

There are various surveillance applications to detect pedestrians with CCTV video stream or automotive camera. Through the sparse optical flow method (Choi et al., 2022b), it is possible to estimate the location of moving objects with a small amount of computation. To classify whether a set of moving points is a pedestrian, the classification model is applied. Following this concept, I wish to check the validity and limitations of the machine learning models. Machine learning models often use a method of sliding a fixed size of window in the whole canvas image, and apply the detection operation to the window frame. In this paper, the region selection approach is applied with localization of moving objects, and then the region is scaled into the input of machine learning models including CNNs and decision classifier. For region selection, it can take a pre-processing method of cropping the area extracted from the optical flow and resizing it to improve detection performance. it could take a region partitioning method that divides the moving object region into sub-regions so that multiple objects

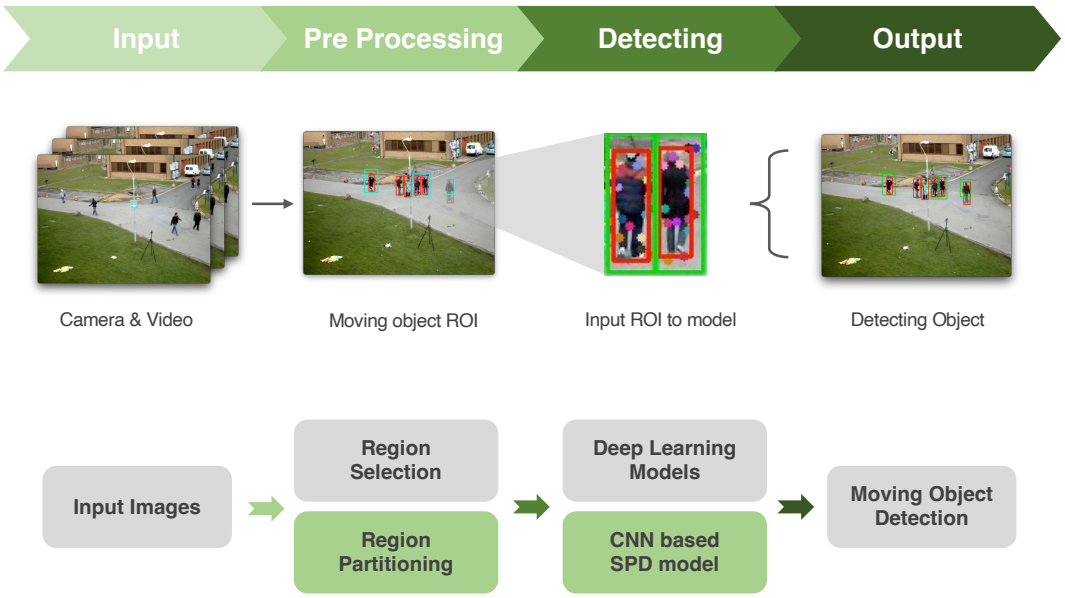


Figure 5.1: Overview of the proposed method; red-colored markers for the ground truth label, cyan-colored markers for region of moving objects and green-colored markers for successful detection.

are identified. Each sub-region becomes an input to the CNN model to classify whether the object is a pedestrian.

Figure 5.1 shows an overview of the approach, consisting of moving object tracker with optical flow, cropping and resizing operator, and CNN model. Sparse optical flow method with memorized estimator (Choi et al., 2022b) is effective to track moving objects in the video stream. A collection of moving objects can select the ROIs. Then the regions are cropped and resized into a fixed size of window frames, which include the area where any moving object exists, and exclude the static background. The regular size of window images becomes an input of the machine learning model. When an object is not detected by the object classification model due to a small size of objects, the resized object image has a better chance of being classified correctly.

In this paper, I compare a well-known classical machine learning model, HOG+SVM, a combinational model of HOG features and SVM classifier, deep learning models such as YOLOv5, YOLOv5s, YOLOv5n, and the suggested approach, SPD CNN model for pedestrian detection. The SPD model is a CNN-based learning model with a small number of layers and focuses on classification features for a fixed size of image window. To this end, I propose a two-phase approach, the region selection with sparse

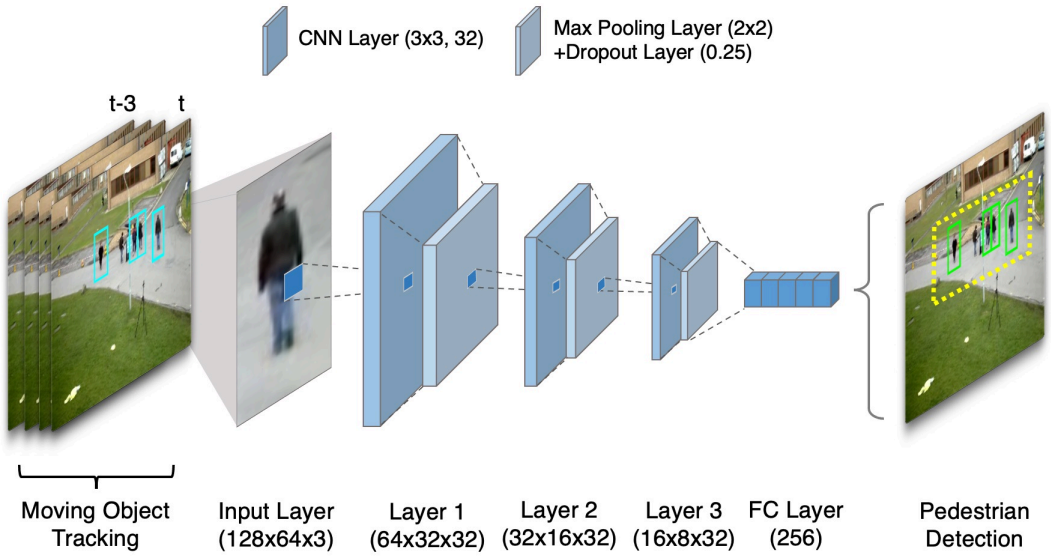


Figure 5.2: Structure diagram of the suggested SPD network model; yellow-colored dashed box indicates successful detection

optical flow and the neural network model.

5.1.1 Small-sized pedestrian detection (SPD) network

The deep learning model for object detection extracts visual features from the input image and performs object localization and object classification. Reducing the amount of computation performed during object localization could mean a lightweight deep learning model can be configured because only a small amount of object classification function remains. I intend to design a model consisting of simple CNN layers and FC layers for object classification functions alone. The input dimension of this neural model is given 128x64 pixels, which assumes a standardized size of a normal pedestrian object. The model is named SPD network model for the purpose of detecting pedestrians appearing in the video for surveillance.

The proposed SPD network model is shown in Figure 5.2. The figure represents one sample of SPD model which consists of one input layer of 128x64 and three CNN layers with 3x3 filters. It solves the problem of parameter amplification caused by the accumulation of CNN layers by adding a max-pooling layer with a 2x2 mask that extracts representative features to each CNN layer. The number of layers varies from one to six, and the number of filters is 32 or 64 for each layer. The number of nodes of

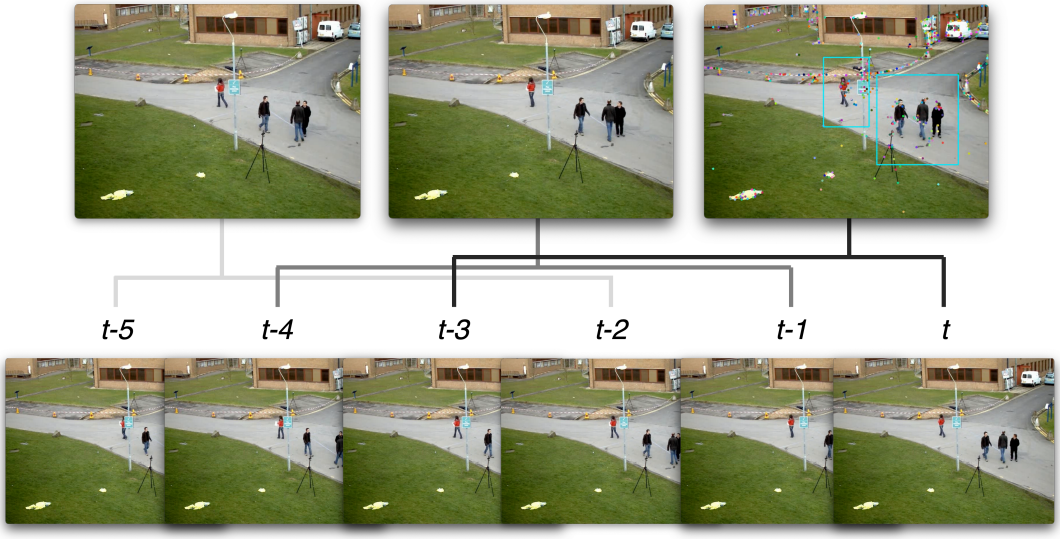


Figure 5.3: Example of moving time window tracker; cyan-colored boxes indicate moving objects tracked

the FC layer for classification is set to 256, and the dropout layer is applied after the FC layer with a parameter of 0.5.

Without the max pooling layer, the number of parameters increases exponentially, making it impossible to learn models or perform detection operations. Dropout layer is added after the max pooling layer to prevent overfitting and improve the generalization performance of the model. In addition to the learned data, objects that have not been seen in the evaluation environment appear, so overfitting degrades performance in the test environment. An FC layer is added to the last layer of the model to perform object classification through image features extracted from the CNN layer.

5.1.2 Region selection for moving objects

Optical flow has been applied to track moving objects in the video stream in various existing approaches. To reduce the computing time, the sparse optical flow algorithm (Choi et al., 2022b) has been suggested to match pixels surrounding the corner feature points and estimate the image flow. The Lucas–Kanade algorithm (Bouguet et al., 2001) is typically used in the sparse optical flow approach. In the paper, I follow the sparse optical flow method with moving window and target estimator (Choi et al., 2022b). Feature points are extracted using the Shi–Tomasi method (Shi et al., 1994)

with the feature reset function to renew feature points for each frame.

A key feature is the moving time window detector (as shown in Figure 5.3), which stores and observes the location of the feature point where the optical flow occurs, and determines and tracks whether it belongs to a moving object. That is, the feature point is recorded in the moving window array, and then the detector checks whether the position of the feature point is changed by a certain distance within a certain time window. If the feature point moves from the starting point to a distance beyond its threshold, it is regarded as a moving object. If the matched feature point in a sequence of images is hovering within a certain threshold distance from the starting point, the feature point is classified as noise. This process reduces the probability of misjudging a moving object.

The moving object tracking algorithm generates a tracking box at feature points around the moving object. It regards the tracking box area generated from feature points on a moving object as an ROI. I crop the region and apply the object classification model only to this region area, not the whole image. Tracking moving objects requires a small amount of computation, so it is useful as a pre-processing stage of pedestrian detection.

Region selection has two types, depending on a loose or tight bounding of the region. Sparse optical flow monitors movement of feature points in the video stream. A region around the feature points of interest can be selected with wide margin so that it can cover more objects or observe the whole bodies of pedestrians; only parts of the human body can be selected as feature points for optical flow, which may degrade the detection performance. In contrast, a tight bound of the region is obtained with a narrow margin over a set of feature points. The HOG feature for pedestrian detection has a vector representation, and it needs the whole size of HOG feature vectors for better detection performance. Thus, a loose bounding box is appropriate to cover the pedestrian appearances. There may be multiple objects in the region, and sliding a regular type of window frame may be useful to detect the target object.

A deep learning model, for example, YOLOv5 model consists of both object localization and classification function. It automatically processes where objects are and which are classified, and thus a wide canvas image can be helpful to detect pedestrians, although a larger size of input image needs more computing time for neural networks. A loose bound of region selection would be helpful to increase the detection performance for the model.

Suggested model, the SPD network is specialized in detecting a target object with a fixed size of window frame, assuming a pedestrian is fitted to the window. The network itself can detect pedestrians even with parts of human body if the training dataset includes such images. Thus, a tight bound of the region is sufficient to recognize an object without searching for multiple objects. However, even a tight bound of region may include multiple pedestrians moving together. Region partitioning process may be required to make subdivision of region to detect each person. Each sub-region is given to the SPD network to check if it includes a pedestrian. The HOG+SVM method can also use the region partitioning method in a similar manner.

5.1.3 Pedestrian detection models

The region selection method crops the ROI from the moving time window tracker based on sparse optical flow. Then, the region is given to the object classification model. Using the whole image as an input of the learning model brings a high computational burden.

In this paper, I test the histogram of gradient (HOG) features and the support vector machine (SVM) classification as a typical machine learning model. For the HOG+SVM classification model, the whole canvas image is scanned with a fixed size of window frame. The model tests if each window frame has the HOG feature for a target object. The SVM determines the classification. The above region selection with optical flow can be applied to the HOG-based approach. A region for moving objects in an image is cropped with wide margin, and then the window sliding can be applied similarly within the region. A larger size of window frame has more detailed information, but it needs more computing time to process. There is a trade-off between the accuracy performance and the computing time. Thus, an appropriate size of window frame is required for the application purpose.

For deep learning models, it can also consider the two-phase approach, region selection with optical flow and object classification. The OF+YOLOv5 method applies the state-of-the-art deep learning model, YOLOv5, to the ROI extracted from optical flow. There are small-sized YOLOv5 models, YOLOv5s and YOLOv5n for small and nano sizes, respectively, that I test in the paper. They are categorized according to the number of layers and filters of model structure. The YOLOv5s and YOLOv5n models consist of 29 layers, but have 7.2M and 1.9M parameters, respectively. The difference between the models is the number of filters in the layers, with YOLOv5s having twice as many

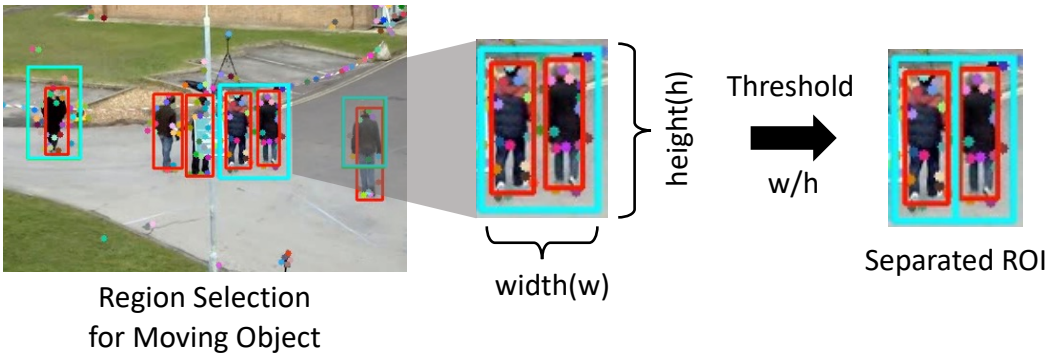


Figure 5.4: Example of region partitioning

filters as YOLOv5n. Later, it will be compared for the performances of deep learning models with / without region selection. The detection performance and computing time vary depending on size of input image as well as what learning models are used.

5.1.4 Region partitioning

Sparse optical flow has an advantage of requiring a small amount of computation, but the motion vector extraction is vulnerable to the background image and large distance movement. Also, the region selected from optical flows may not completely reflect the original shape of a moving object. If multiple objects move together nearby, various optical flows may occur, resulting in an ROI including multiple objects at once.

From this, only a part of the human body may be included in the ROI, or an ROI may have many pedestrians. Distortion may occur in the process of converting this ROI into the input size of the model. I wish to verify the classification performance and robustness for distorted images or part images. The CNN models can be expected to better capture features with convolution filters by learning distortion images or part images of objects.

To divide the ROI obtained with the optical flow into sub-regions, I propose a subdivision method according to the aspect ratio of the ROI as shown in Figure 5.4. Subdivision process involves diving up an ROI into two or more parts, putting them as separate, standalone sub-regions. Each sub-region is resized and tested with the pedestrian detection model to see if it can be classified as a pedestrian.

A region for moving objects is checked if it can be divided in terms of the aspect

ratio. The aspect ration is the ratio of width to height of a region. Figure 5.4 shows an example of region partitioning. The threshold is determined by examining the average of aspect ratios of the ROI according to the number of correctly classified objects. The algorithm roughly estimates the number of pedestrians with the aspect ratio over a region selected from the optical flow, that is, how many sub-regions are available to assign each for a pedestrian. Each partition of the ROI becomes an input to the detection network to perform classification, and the number of pedestrians increases if each partition has a successful pedestrian detection. The true positive (TP) score will be increased for a bounding box region.

Without looking at the ground truth label, I do not know how many objects exist within the ROI. Therefore, when performing performance evaluation, the TP evaluated for recall performance is increased by one for each sub-region ROI with a detection success. In the experiment, I will check how well the classification model detects individual objects. If no object is detected even after applying the region-partitioning method, the original region before partitioning should be checked for detection.

5.2 Experimental environment

5.2.1 Dataset

To learn CNN model, the customized dataset is collected. Individual or multiple person objects are cropped from the COCO2017 dataset which includes various forms of objects, even distorted images (Lin et al., 2014; Chen et al., 2015). A total number of person images are 252,309 and the number of non-person images are 529,336. To validate the trained CNN model, I use 10,621 person-object images and 23,645 non-person images. Non-person objects include various object classes and partial backgrounds. Then the trained CNN model is applied to detect pedestrian for the PETS2009 video dataset. The video resolution is 768 $\times$ 576 pixels, and the number of frames is 794. This dataset includes many pedestrians walking with fast or slow speed on the road in an arbitrary direction. Similarly, machine learning models with HOG features use the same image set for training and testing.

5.2.2 HOG-based approach and deep learning models

The pedestrian detection using the HOG features of the input image and the SVM classification is a commonly used algorithm. Gradient features of image and their local histogram form HOG features. The orientation directions are measured in a sample patch of images, and their distribution can be a feature to characterize an object or pedestrian. The SVM classifier learns this HOG feature with the training set. The HOG+SVM model, a combinational model with the HOG feature and the SVM decider will be tested as a machine learning model for comparison. The model tests only a fixed size of window frame for HOG features and so the test window should be slid repeatedly over the whole canvas image to find its matched object. Generally, the method is vulnerable to the size and distortion of target objects to be detected. To solve this problem, there is an approach with multi-scale HOG feature using varying scales in an image pyramid (Li et al., 2019).

A well-known deep learning neural network model, You Only Look Once (YOLO), is often used for object detection. There are variations of the model, YOLO, YOLOv2, YOLOv3, YOLOv4 and YOLOv5, and each has its own characteristics for application. Prior to the YOLO model, R-CNNs split the image into several sections and the analyzed the image using a CNN model. Therefore, even if object detection was performed on one image, it was like analyzing several images. However, YOLO has a powerful property of viewing images only once as an integration of image features, which allows detecting objects in real time. Its contextual understanding of the object class is higher than that of other models, showing low false-positive rate. Also, it shows an effective learning of generalized object class, allowing to see where objects are and what objects are in a bounding box. To prevent overfitting, the number of learning epochs stops at the number of iterations at which the validation loss is not significantly reduced. However, the YOLO model has a relatively low accuracy for small objects. As such, further improvements are needed to increase the accuracy as well as reduce the computing time.

5.3 Experimental results

The approach suggested in this paper for pedestrian detection consists of two stages: one is to select regions over moving objects found with optical flow, and the other is to apply the classification model to a given size of window frame. I suggest an

Table 5.1: Comparison of YOLOv5 models with various input sizes; YOLOv5s and YOLOv5n indicate small-sized and nano-sized models of YOLOv5, respectively

Model	YOLOv5s							
Input size	64x64	96x96	128x128	160x160	192x192	256x256	320x320	640x640
Recall	0.036	0.258	0.529	0.601	0.652	0.794	0.837	0.912
Precision	1.000	0.987	0.987	0.987	0.983	0.967	0.948	0.941
F-score	0.070	0.409	0.689	0.747	0.784	0.872	0.889	0.926
FPS	27.916	24.179	22.640	19.429	18.278	14.716	11.899	4.970
Model	YOLOv5n							
Input size	64x64	96x96	128x128	160x160	192x192	256x256	320x320	640x640
Recall	0.002	0.068	0.262	0.394	0.509	0.691	0.797	0.883
Precision	1.000	0.995	0.999	0.990	0.986	0.980	0.975	0.943
F-score	0.004	0.127	0.415	0.564	0.671	0.810	0.877	0.912
FPS	31.048	28.969	27.680	24.099	21.987	18.305	15.383	6.450

SPD model consisting of small-sized CNNs, and it is compared with the conventional pedestrian detection models, the HOG+SVM model and a variation of the YOLOv5 model. The two models can also have similar two-stage process of region selection and classification, and they will be compared with the original detection methods.

5.3.1 Deep learning model

I tested a well-known deep learning method, YOLOv5 for pedestrian detection. Accuracy and computing time performances greatly depend on the input size. Performance evaluation uses the same pedestrian dataset as the dataset used in all experimental parts. Table 5.1 shows the results with varying input sizes. In the YOLOv5 model structure, there are depth multiple parameters that control the number of layers and width multiple parameters that control the number of nodes per layer. Compared to the medium-sized model of YOLOv5m, which has 41 layers, 21.2M parameters and 49.0B flops, the small-sized model has 29 layers, 7.2M parameters and 16.5B flops, and the nano-sized model has 29 layers, 1.9M parameters and 4.5B flops. When the YOLOv5m model is tested with an input size of 640x640 pixel images, the recall performance is 0.924, precision is 0.925, F-score is 0.924, and frames per second (FPS) is 2.980. For an input size of 320x320 pixel images, recall is 0.877, precision is 0.921, F-score is 0.898, and FPS is 8.617. To improve the computational cost, YOLOv5s and YOLOv5n models were used with more detailed tests – see Table 5.1.

Table 5.2: Comparison of OF+YOLOv5 models with various input sizes

Model	OF+YOLOv5s							
Input size	64x64	96x96	128x128	160x160	192x192	256x256	320x320	640x640
Recall	0.597	0.789	0.854	0.887	0.904	0.919	0.924	0.893
Precision	0.945	0.947	0.941	0.938	0.934	0.942	0.940	0.949
F-score	0.732	0.861	0.895	0.912	0.919	0.930	0.932	0.920
FPS	17.501	13.485	11.955	9.073	8.231	6.232	4.698	1.705
Model	OF+YOLOv5n							
Input size	64x64	96x96	128x128	160x160	192x192	256x256	320x320	640x640
Recall	0.506	0.735	0.823	0.861	0.888	0.895	0.898	0.879
Precision	0.937	0.944	0.946	0.946	0.946	0.946	0.946	0.956
F-score	0.657	0.827	0.881	0.901	0.916	0.920	0.921	0.916
FPS	22.129	18.758	15.710	12.837	10.798	8.781	6.629	2.301

The YOLOv5s model (i.e., the YOLOv5 model with a larger number of parameters in the network) shows better accuracy and lower computing speed than the YOLOv5n model. At least the input size of 320x320 is required for good detection performance. The input size determines the number of nodes of the input layer in the deep learning model structure. As the input size decreases, the feature information of an image becomes lost, affecting the performance of object detection. In Table 5.1, the smaller the input size parameter, the lower the detection rate; however, the computing time depends on the number of parameters in the network, leading to faster processing speed.

The YOLOv5 model described above uses the entire image as input data. The lower the dimensions of the input data, the lower the recall performance, since small-sized images tend to lose the feature information for pedestrian. Region selection based on sparse optical flow was tested to see if cropping the ROI has a positive effect on the deep learning model. This combinational model, the optical flow plus deep learning, called OF+YOLOv5, takes the region selection first and then applies the YOLOv5 model to a loose bound of region with wide margin. The method was tested with varying input sizes. Interestingly, Table 5.2 shows that the model produces a high recall performance, of 0.823 with 15.71 FPS even for an input size of 128x128 pixel images. In Table 5.2, a major change compared to the previous experiments (in Table 5.1) is that better performance is observed for a low resolution of images, but with increased computing time.

From the experimental results, I argue that the region selection process is very effective

Table 5.3: SPD model results without region partitioning

Model	Layers	Filters(3x3)	Recall	Precision	F-score	FPS	Params(x10,000)
SPD_1_32	1	32	0.57	0.94	0.71	17.0	1678
SPD_1_64		64	0.58	0.94	0.72	11.2	3356
SPD_2_32	2	32	0.59	0.94	0.73	26.9	420
SPD_2_64		64	0.61	0.94	0.74	21.6	843
SPD_3_32	3	32	0.58	0.94	0.72	31.1	107
SPD_3_64		64	0.59	0.95	0.72	27.7	217
SPD_4_32	4	32	0.57	0.94	0.71	33.1	29
SPD_4_64		64	0.60	0.94	0.73	23.6	64
SPD_5_32	5	32	0.58	0.94	0.71	31.9	10
SPD_5_64		64	0.58	0.94	0.72	23.9	28
SPD_6_32	6	32	0.58	0.94	0.71	32.1	6
SPD_6_64		64	0.58	0.94	0.71	23.5	22

for pedestrian detection. The reason for the increased computational cost is that the YOLOv5 network model was applied repeatedly for all the ROIs. Thus, it needs further effort to organize the classification network efficiently. In this paper, I suggest an SPD network with a small number of layers to the ROI after region selection. The object localization is obtained from the region selection and the CNN structure is built only for object classification, while the YOLOv5 network achieves object localization and object classification simultaneously. The suggested SPD network is customized for the application purpose. In the structure, a fixed (relatively small) size of window frame is prepared for the target object image and no window sliding is needed to search for target objects. Thus, the total computing time can be reduced while maintaining the detection accuracy.

5.3.2 SPD network model

Compared to the YOLOv5 model, the CNN model that performs only classification function, can consist of multiple layers of convolutional filter layers to extract image features and FC layers to classify a given image. The YOLOv5 model simultaneously performs object localization and classification while outputting the location and size of target objects, which can be considered a one-stage detection model. In contrast, the

SPD model tracks the location of moving objects from the optical flow, selects the ROI, and classifies the region whether it includes a pedestrian. Since I do not know how many people are included in the ROI where moving objects are available, I initially evaluated the recall performance by considering the region as an instance of a single pedestrian, regardless of how many people are present, if it is classified as pedestrian.

The effect of layer size in the network for the SPD model is investigated. Table 5.3 shows the result of performance evaluation with various models according to the number of CNN layers and the number of convolution filters in a layer. The overall detection performances are similar, but FPS performances differ depending on the number of layers and the number of parameters of the model. As the number of layers increases, the number of features represented by the CNN layer increases, and the number of weight parameters of the network decreases. Use of three CNN layers shows efficient computational speed performance, and as the number of 3x3 filters per layer increases, the computational speed decreases further, with 32 filters being the appropriate number. In the SPD model, regions including multiple individuals reduce classification performance, and the region with only one person is classified with a high probability. Therefore, if the ROI is partitioned so that each object in a region including multiple people is separately classified, the performance will be further improved, and the detection rate and accuracy performances for each individual object will vary from model to model.

Table 5.4 shows the performance detection results of various SPD models with region partitioning. Compared to the results of the Table 5.3 experiment without region partitioning, the recall performance was significantly improved and the precision and FPS performances were slightly reduced. Though precision performance was a little decreased, the recall performance was greatly improved, and the overall F-score detection performance was improved. If the flaw of the sparse optical flow is improved or other pre-processing techniques are studied, the detection performance through the SPD model will be further improved. From Table 5.3, the SPD_3_32 model with three layers and 32 filters can be chosen as an appropriate model for application, exhibiting excellent detection performance and efficient FPS performance.

Applying the object detection model to the ROI has a benefit in terms of computation time. If I zoom the cropped area and test the region with the detection model, it will be able to capture the feature better. Resizing the region into a fixed size of window is effective in that the machine learning detection model learns the features of an object

Table 5.4: SPD model results with region partitioning

Model	Layers	Filters(3x3)	Recall	Precision	F-score	FPS	Params(x10,000)
SPD_1_32	1	32	0.71	0.81	0.76	12.6	1678
SPD_1_64		64	0.72	0.80	0.76	7.8	3356
SPD_2_32	2	32	0.70	0.83	0.76	23.8	420
SPD_2_64		64	0.72	0.87	0.79	18.1	843
SPD_3_32	3	32	0.71	0.88	0.79	28.2	107
SPD_3_64		64	0.71	0.89	0.79	24.3	217
SPD_4_32	4	32	0.71	0.85	0.77	29.3	29
SPD_4_64		64	0.72	0.84	0.78	19.9	64
SPD_5_32	5	32	0.72	0.83	0.77	29.5	10
SPD_5_64		64	0.71	0.88	0.79	19.8	28
SPD_6_32	6	32	0.71	0.83	0.77	29.6	6
SPD_6_64		64	0.72	0.85	0.78	19.2	22

with a specific size of filter or kernel.

The HOG+SVM model has the learned HOG feature for pedestrian detection and tries to match the HOG feature with the SVM classifier. It basically slides a window frame into the whole image with a given stride length. The SVM classifier detects objects while sliding a window area that matches the input dimension of the HOG vector. When a window slides in an image area and detects an object, it is possible to adjust the parameter depending on the distance it slides and how it performs the classification function. The F-score is an average of recall and precision, and if the precision value decreases, the F-score value may decrease even if the recall value is high. It is observed that for a small stride in the window sliding method, the detection rate becomes higher but the computing time lengthens due to frequent detection attempts. The detection performance significantly degrades for a large stride, since the SVM classifier often fails to find objects matched with the target HOG feature vector. The appropriate stride length should be chosen for the HOG+SVM method. Region selection can be applied to the HOG+SVM model instead of sliding a window, in which case the model needs resizing of the ROI to the SVM input dimensions.

By analyzing the number of objects in the area extracted from the optical flow, I set

Table 5.5: Changes of applying region partitioning method according to aspect ratio (HOG+SVM classifier).

Classifier method	**Labels in ROI**	**Aspect ratio**	**No. of ROI**	**No. of successes /failures**	**Labels detection test**						
					0	1	2	3	4	5	6
HOG +SVM (No sep)	0	0.85(±0.27)	407	119	119	-	-	-	-	-	-
				288	288	-	-	-	-	-	-
	1	0.66(±0.19)	2038	850	311	539	-	-	-	-	-
				1188	555	633	-	-	-	-	-
	2	0.74(±0.18)	699	65	22	28	15	-	-	-	-
				634	190	320	124	-	-	-	-
	3	0.78(±0.27)	141	9	2	3	4	0	-	-	-
				132	26	60	34	12	-	-	-
	4	0.90(±0.23)	35	4	1	2	1	0	0	-	-
				31	2	14	12	3	0	-	-
	5	1.15(±0.27)	2	0	0	0	0	0	0	0	-
				2	0	0	1	1	0	0	-
	6	0.89(±0.04)	2	0	0	0	0	0	0	0	0
				2	0	2	0	0	0	0	0
HOG +SVM (Sep)	0	0.41(±0.33)	1364	119	119	-	-	-	-	-	-
				1245	1245	-	-	-	-	-	-
	1	0.45(±0.24)	3377	873	318	555	-	-	-	-	-
				2504	1135	1369	-	-	-	-	-
	2	0.48(±0.25)	488	42	15	16	11	-	-	-	-
				446	143	221	82	-	-	-	-
	3	0.53(±0.20)	72	5	0	2	3	0	-	-	-
				67	18	26	17	6	-	-	-
	4	0.53(±0.09)	3	0	0	0	0	0	0	-	-
				3	0	1	2	0	0	-	-
	5	0	0	0	0	0	0	0	0	0	-
				0	0	0	0	0	0	0	-
	6	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	0	0	0	0

the threshold according to the aspect ratio, and then divide the ROI. In general, as the number of individuals in the ROI increased, the average aspect ratio of the ROI increased according to the number of individuals. For example, if ROI containing only

one object has an aspect ratio of 0.66, set a threshold of 0.66, and then no separation is attempted if the ROI has an aspect ratio of 0.66 or less. If it exceeds 0.66 but is less than 0.74, separate the ROI into two according to the ratio.

Table 5.5 shows the result of the HOG+SVM model without and with region partitioning. Term of labels in ROI indicates the number of ground truth labels in each moving object ROI. The numbers of label detection tests from zero to six represent the detected number of the ground truth labels in the ROI. Sep indicates region partitioning. The table above shows how successful it could be, if labels included in the ROI are individually well detected, compared to the ROI where two or more individuals were failed to detect. In the table below, it appears that the number of individually divided objects has increased, but the number of objects successfully detected has not increased significantly. Since the number of successful detections of ROI containing one person increased from 539 to 555, the detection performance is not significantly improved. The number of successful detections of ROI where objects do not actually exist indicates false alarms, but since this does not increase, the accuracy performance has not changed significantly. From this, it may be seen that the HOG+SVM model does not effectively detect individual objects even in each partition of ROI. The HOG+SVM model mainly adopts a method of cropping the image area from the sliding window at several-pixel intervals and inputting it to the model. Experiments show that the HOG+SVM model is vulnerable to distortion of the image feature, when the resolution of the input image is fitted.

In previous experiments, the SPD model was expected to improve performance further when applied in each partition of the ROI. Table 5.6 shows the experimental results when the classification model is applied as a lightweight SPD model composed of a few CNN layers. In general, the probability of detection failure increases as the ROI includes a large number of individuals. According to the above table before separation, the detection ratio of ROI with two objects is 649:42, 96:28 for three objects, 24:9 for four objects, and for five or more objects, only one detected. Even detecting multiple people fails, the SPD model shows high detection performance when detecting the corresponding objects individually. From this, the detection performance will be improved by region partitioning to divide the ROI containing multiple people into the areas containing individual objects.

Table 5.6 shows that the number of successful detections of ROI containing one person increased from 1909 to 3105. From this, the detection performance (recall) will

Table 5.6: Changes of applying region partitioning method according to aspect ratio (SPD classifier).

Classifier method	**Labels in ROI**	**Aspect ratio**	**No. of ROI**	**No. of successes /failures**	**Labels detection test**						
					0	1	2	3	4	5	6
SPD (No sep)	0	0.85(±0.27)	407	301	301	-	-	-	-	-	-
				106	106	-	-	-	-	-	-
	1	0.66(±0.19)	2038	1911	2	1909	-	-	-	-	-
				127	3	124	-	-	-	-	-
	2	0.74(±0.18)	699	656	0	7	649	-	-	-	-
				43	0	1	42	-	-	-	-
	3	0.78(±0.27)	141	108	0	0	12	96	-	-	-
				33	0	0	5	28	-	-	-
	4	0.90(±0.23)	35	26	0	0	0	2	24	-	-
				9	0	0	0	0	9	-	-
	5	1.15(±0.27)	2	1	0	0	0	0	0	1	-
				1	0	0	0	0	0	1	-
	6	0.89(±0.04)	2	0	0	0	0	0	0	0	0
				2	0	0	0	0	0	0	2
SPD (Sep)	0	0.41(±0.33)	1364	718	718	-	-	-	-	-	-
				646	646	-	-	-	-	-	-
	1	0.45(±0.24)	3377	3115	10	3105	-	-	-	-	-
				262	8	254	-	-	-	-	-
	2	0.48(±0.25)	488	476	0	12	464	-	-	-	-
				12	0	0	12	-	-	-	-
	3	0.53(±0.20)	72	67	0	0	6	61	-	-	-
				5	0	0	0	5	-	-	-
	4	0.53(±0.09)	3	3	0	0	0	0	3	-	-
				0	0	0	0	0	0	-	-
	5	0	0	0	0	0	0	0	0	0	-
				0	0	0	0	0	0	0	-
	6	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	0	0	0	0

increase significantly, and the accuracy performance (precision) will decrease slightly by increasing the number of detection successes from 301 to 718. However, as the number of failed detections of ROI without objects increases from 106 to 646, it seems

to distinguish the false alarming region very well. The number of cases that could improve recall performance increased from 124 to 254 when number of labels ROI was one. These cases are phenomena that appear from the vulnerability of sparse optical flow. When a part of a human object, such as an arm or leg, moves, a part where a motion vector occurs greatly appears. Accordingly, a part of the human body is frequently captured in the ROI. In addition, an inaccuracy occurs in which one object is divided into two due to an incomplete partition of ROI. The CNN-based neural network model SPD also learns images containing parts of the body and several people, which is insufficient than when entering the entire human body, but shows good robustness. In the experimental results, even when a part of one object is classified, it shows strong detection performance.

5.3.3 Comparison of detection performance

I tested the learning models for pedestrian detection: HOG+SVM, HOG Multiscale, YOLOv5n of 320x320 and OF+HOG+SVM (loose bound of region selection), OF+YO LOv5n_128 (loose bound) and OF+SPD_3_ 32 (tight bound) models. Figure 5.5 shows the comparison of their performances. Performance results are compared by applying the same performance evaluation method, which treats ROI with multiple objects as a single object detection success for all models. For the proposed model, I set the optimal parameters obtained through the experimental results. For the dataset PETS2009, each of conventional models also used its optimized parameters for comparison. The HOG+SVM model is a classification model of combining the HOG features and the SVM classifier. The HOG+SVM model applies the sliding window method to the entire image resolution of the frame, and inputs and classifies the area corresponding to each window into the model. The window size is set according to the dimensions of the learned SVM classifier. The HOG-Multiscale method uses varying scales in an image pyramid (Li et al., 2019).

The YOLOv5n_320 model uses the entire image as input data, and unlike the existing sliding window method, it is a one-stage detector that performs both object localization and classification. The latest deep learning model uses an anchor box to learn the location and size of the object and extracts the location and classification results from the extracted image feature. The OF+HOG+SVM model has the suggested region selection, tracks moving objects with optical flow, and classifies the area of moving objects with a sliding window of HOG feature and SVM classifier. The

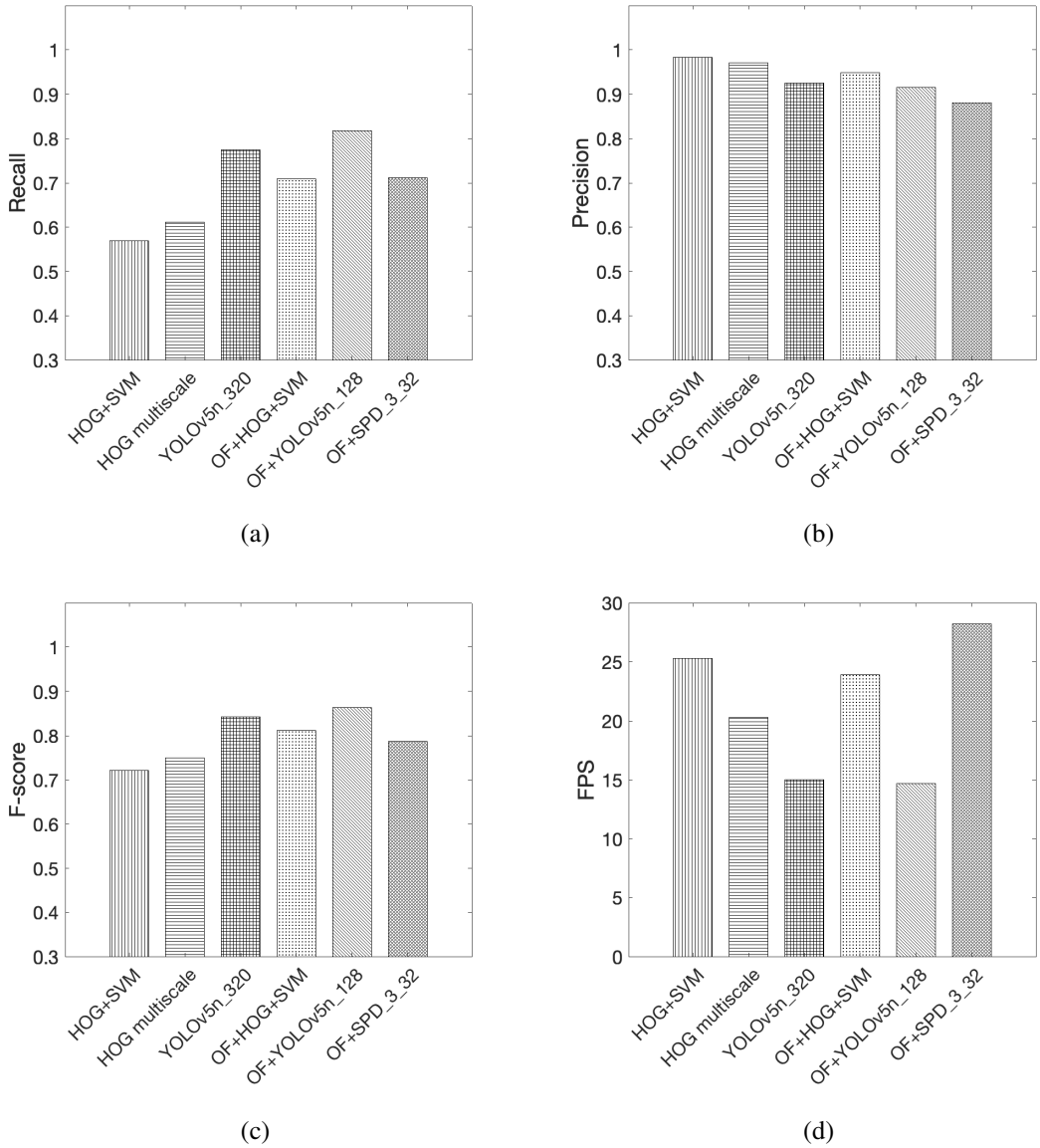


Figure 5.5: Comparison of detection performances (a) recall performance results (b) F-score (c) FPS

OF+YOLOv5n_128 method inputs a moving object region extracted from sparse optical flow to the model, producing robust detection performance even for small image input sizes. The OF+SPD_3_32 model with region partitioning is compared with other learning models. This method replaces object localization with region selection (i.e., tracking moving objects with sparse optical flow) and significantly reduces computation time by applying small-sized CNN model.

Figure 5.5 shows that detection performance recall and F-score values are increasing in the order of HOG+SVM, HOG multiscale, YOLOv5_320, OF+HOG+SVM

(loose bound of region), OF+YOLOv5n_128 models (loose bound of region), and OF+SPD_3_32 (tight bound of region). On the other hand, it is confirmed that, in general, as the detection performance increases, the FPS value gradually decreases. Other conventional models perform both the location and classification of the object at the same time (especially in the process of locating the object), processing the entire image, which results in poor processing speed performance. The OF+HOG+SVM model can quickly process to find the location of moving objects through the sparse optical flow, and although it is a simple function, it performs the function of distinguishing whether the moving object is human through HOG feature and learned SVM classifier. The OF+YOLOv5_128 model shows the best detection performance, but the worst in terms of FPS. On the other hand, the OF+SPD_3_32 (tight bound) model shows the lowest computational speed, and the detection performance is also comparable to the OF+HOG+SVM model. Based on the results, it can be selected to the OF+YOLOv5_128 model if I increase the detection performance in equipment with sufficient computational speed, or the OF+SPD_3_32 model for compliant detection performance in low-cost equipment with low computational speed.

I compared the performances of various models with the laptop computer and portable embedded board. Table 5.7 shows the results with the HOG Multiscale, HOG+SVM, OF+HOG+SVM (loose bound), OF+HOG+SVM (tight bound), YOLOv5_320, OF+YOLOv5n_128 (loose bound), and OF+SPD_3_32 (tight bound) models. The laptop computer has a 2.3 GHz 8-core Intel CPU and the Raspberry Pi 4 Model B has a 1.5 GHz 4-core ARM CPU. Both models perform computational processing only on the CPU, and the Raspberry Pi is a low-cost and low-performance computing device. The results of FPS experiments in the Raspberry Pi environment show a feasibility for real-world applications useful in industrial fields. Depending on the environment of the two computation devices, a compatibility problem between the supported operation functions and the framework may occur. Accordingly, there is a difference in computational speed according to the devices between different framework-based models, Pytorch and Keras of the Tensorflow backend. The YOLOv5n model in the Pytorch environment operates in the capability of the deep learning network library to work in C++ languages and is well-optimized. The proposed SPD model operates in the Keras of the Tensorflow backend environment, and the computational processing method operating in the ARM CPU is less optimized. The computing speed index is mainly divided into optical flow-based ROI extraction time (OF), object classification time (Class.), and

Table 5.7: Performance comparison of various models on laptop Intel CPU and Raspberry Pi 4B; NA means Not Applicable, OF in FPS category indicates the region-selection processing time to obtain ROI, Class indicates classifying regions of moving objects, and Total means a total processing time for image frames.

Model	Recall	Precision	F-score	FPS					
				Intel CPU			Raspberry Pi 4B		
				OF	Class.	Total	OF	Class.	Total
HOG Multiscale	0.61	0.97	0.75	NA	20.3	20.3	NA	2.6	2.6
HOG+SVM	0.57	0.98	0.72	NA	25.3	25.3	NA	6.2	6.2
OF+HOG+SVM (loose bound)	0.71	0.95	0.81	37.7	72.3	24.8	16.1	9.8	6.1
OF+HOG+SVM (tight bound)	0.23	0.94	0.36	36.8	652.2	35.4	16.1	113.3	14.1
YOLOv5n_320	0.79	0.97	0.87	NA	15.0	15.0	NA	3.5	3.5
OF+YOLOv5n_128	0.82	0.94	0.88	36.0	24.6	14.6	15.7	5.4	4.6
OF+SPD_3_32 (tight bound)	0.71	0.88	0.79	38.5	105.6	28.2	16.4	9.0	5.8

overall computing speed (Total), in Table 5.7. The corresponding index is expressed in FPS, and if the reciprocal is taken, it can be expressed as the processing time per frame.

Table 5.7 shows the difference in computational speed performance according to the computational processing capabilities optimized in low-performance embedded devices for each model. First, in the case of the HOG multiscale model provided by OpenCV library, detection processing speed is significantly reduced in the ARM CPU environment compared to the Intel CPU, from 20.3 to 2.6 FPS. The optical flow method part, which is relatively simple but repeatedly performs arithmetic processing, does not show a significant difference between the laptop environment and the Raspberry Pi environment, with 36 and 16 FPS, respectively. The machine learning model OF+HOG+SVM (scaling+sliding Window) applies the sliding windows only to the region extracted from optical flow and scaled, not the entire image such as HOG+SVM model, indicat-

ing that the computation speed has increased considerably.

The OF+HOG+SVM model (with a tight bound of region) follows the same procedure as the OF+SPD model, whose cropped region is resized into a 128x64 pixel image. As seen in Table 5.7, if the HOG-based approach has no window sliding to search for objects and its classifier has a single run of test on the selected region (a tight bound of region selection), then its performance is drastically reduced, though it may have fast computing time. If a restricted zone of region is not fitted well, it easily loses target pedestrians and the HOG feature is not effective within the selected region. The performance evaluation results of the YOLOv5n_320 and OF+YOLOv5n_128 models show that the backend framework of the corresponding detection model is well optimized for the ARM CPU, thus reducing the computing time. However, the OF+SPD_3_32 model with CNN layer has a total FPS of 5.8, indicating that its processing is not as efficiently organized for the Raspberry board as an ARM CPU. It is observed that better computing speed is available on the Intel CPU board.

5.4 Summary of Chapter 5

Detecting pedestrians is one of the challenging tasks in the video surveillance system, and such surveillance is useful in improving safety for automotive applications and traffic flow measurements. I introduce a small-sized pedestrian detection network, called SPD network, which significantly improves the computing time to detect pedestrians with small-sized convolutional neural networks (CNNs). This approach follows a motion-based detection approach for pedestrian detection. Initially moving object tracking with optical flow is applied to the video stream, the region of interest is selected and then a classification is performed on the region to recognize pedestrians. The region of interest is cropped for a sequence of image frames and then resized to a regular size of window for testing the neural networks. In addition, pedestrian detection performance is improved with region partitioning for a group of pedestrians. With the experimental results, region selection based on optical flow and the object detection model show better performance in pedestrian detection than the conventional algorithms with HOG method or deep neural networks. Also, the approach improves the computing time to a large extent, compared to the classical CNN model, YOLOv5. It provides a possibility that the approach can be applied in a low-cost portable embedded device.

Chapter 6

Conclusions

In this dissertation, studies of integrative approach to pedestrian detection based on sparse optical flow and CNN model have been presented. The study of pedestrian detection has been improved for the safety of workers in industrial areas and for monitoring traffic flow in surveillance systems. It is focused on reducing computational cost for application on devices operated using CPUs.

The work has two main and one supplementary streams: moving-object tracking, pedestrian detection, and data selection. The moving-object tracking and pedestrian detection portions concentrate on reducing computational cost and enhancing detection performance based on sparse optical flow and CNN model. The data selection portion concentrates on investigating the influence graph and coverage effects on the custom datasets.

Chapter 3 provides the detailed method of moving time window detector and memorized estimator functions. Chapter 4 suggests the framework to investigate the relationship, influence, and coverage area of training datasets. Chapter 5 describes the system that integrates the moving time window detector and simple CNN-based SPD model to detect pedestrians in a surveillance system. The experiments were conducted on input video data obtained from surveillance camera systems.

The detailed conclusions are described in the following subsections.

6.1 Moving object tracking based on sparse optical flow

In Chapter 3, the approach to moving object detection and tracking is proposed. One of the major attributes is to improve the accuracy performance and reduce the computation time of responding to moving objects or moving pedestrians. The proposed method is based on the sparse optical flow approach, that is, a coarse-grained optical flow, but it includes the corner feature reset with a moving window. A sequence of images effectively finds the flow of moving objects and thus the moving window of image frames easily captures moving targets without wasting much time.

The moving time window detector improves the noise filtering and the detection rate, by looking at a history of optical flows. In a hazardous environment, such as the construction sector, there may be the risk of meeting many obstacles including walls and trees, and various optical flow patterns are often observed, when pedestrians should be detected. The memory-based target estimator plays the role of monitoring the targets or pedestrians without missing when the targets move around or stagger at some positions. Even if a moving target is initially recognized, the target may move continuously with occasional pause. With this estimator, the last position of a moving targets is estimated and this improves the performance of detecting moving objects in a row.

I adapt this detection algorithm in the embedded board system, Raspberry Pi4, for real application. The experimental results demonstrate that the suggested approach is effective for preserving the detection performance even with a low computing power of the embedded device. According to the experimental results, the proposed method shows similar or higher accuracy performance, compared to the conventional algorithms for moving object detection using optical flows or vision processing algorithms: Lucas-Kanade's method and Farneback's method in addition to HOG and Haar-like methods. It also provides a more efficient computing time than dense optical flows and vision processing algorithms. The approach works well even for distorted views from a top-viewed camera and also for blurred images or noisy image frames, and thus it can be robustly applied to various environments.

6.2 Data selection for deep learning model

Many deep learning models have been applied to recognize objects or pedestrians. Fisheye cameras have a wide viewing angle close to 180°, and their image distortions

make it difficult to transfer a deep learning model of object detection to a new fisheye camera environment. Adaptation due to the geometric distortions and perspective changes should be considered. In addition, the environmental conditions affects the performance of deep-learning neural networks at object detection. I collected images from six fisheye cameras mounted on a hydrogen-powered bus and also images from another type of fisheye camera on an excavator vehicle under various environmental conditions and varying camera positions. The collection of the custom image datasets was classified into six subjects depending on the environmental factors, whether their photos are captured indoors or outdoors, in daylight or at night, or at low angle or high angle of the camera position. The bus and excavator vehicles are supposed to use deep learning models to monitor nearby pedestrians or objects to guarantee safe driving.

I suggested a cross-test approach to analyze the relationship among a collection of subjects or image datasets. The cross test is used to build up a deep learning model for one subject and then evaluate the model on the test data for a target subject. Based on the result, I represent the graph relationship between pairs of subjects, and the edge from a source subject to a target subject encodes the information of how much the performance of a target subject is improved by the addition of a source subject. I train a neural network model for the base set alone and another neural model for the source subject plus the base set. The two models are tested on a target subject and the performances are compared to derive the effect of a source subject on the target subject.

From the experimental results, I derived the influence graph among six subjects of the custom image datasets obtained from fisheye cameras of vehicles, and the information of positive or negative effect in the influence is useful to choose an appropriate training subject to improve the validation performance of a target subject. Furthermore, interrelations among subjects can be inferred and analyzed. The approach can reduce the cost of collecting the image data with of the appropriate environmental factors even with fisheye cameras. When a negative aspect is observed, I can analyze its cause or latent variables by drawing the shadow zone.

6.3 Pedestrian detection based on convolutional neural network model

Pedestrian detection is a main subject regarding video surveillance applications to improve safety for automotive applications, CCTV monitoring systems and traffic management. To improve the accuracy performance, deep learning neural networks with GPUs have become popular. However, they need a large amount of computation and the learning models are rarely useful on a low-cost embedded board without a GPU.

In Chapter 5, I consider applications of pedestrian detection running in a portable PC without a GPU and introduce an effective and efficient approach for pedestrian detection, which performs region selection based on optical flow and then applies CNNs to a regular size of window region. It needs cropping and resizing operators after region selection, and the operators operate with relatively low-cost computation. The optical flow initially tracks moving objects in the video stream, and a collection of points with high magnitude in optical flow forms an ROI. The selected region is cropped from the image and resized into a minimum size of window needed to detect pedestrians effectively.

The suggested approach, called the OF+SPD model, can select ROIs with optical flow and apply small-sized neural networks of classifying objects in the regions. It can reduce the time and effort of testing the neural networks on every spot in an image. The region selection as well as the SPD network need relatively small computing cost compared to deep neural networks such as YOLOv5. Experimental results support that the method has good detection performance with a short processing time. The YOLOv5 model with optical flow showed excellent detection performance, but requires high computing time. The customized CNN model for the SPD network has a small number of layers (less than six layers), but keeps detection performance comparable to the deep learning model.

The HOG+SVM method with region selection shows better detection performance than that without region selection. The HOG-based approach has small computing time, but it is much worse in the detection performance than the suggested OF+SPD model or the OF+YOLOv5n model. The CNNs are highly effective in extracting features for detecting pedestrians. In addition, their learning and adaptation is reasonably applicable at new environments with distorted images or occluded images. Region selection with optical flow is a crucial aspect of improving the performance and the com-

puting time. The performance deterioration often occurs in the presence of small-sized pedestrians. The region selection resizes the candidate regions to be fit into a regular form of window. Compared to direct application of the YOLOv5n model, this process improves the detection accuracy even for the OF+YOLOv5n model.

I propose a model with excellent detection performance or computational speed by applying YOLOv5 or a small-sized CNN learning model to the moving object region extracted with sparse optical flow. The OF+YOLOv5 model shows excellent detection performance due to its complexity and large size, but requires long computation time. To improve this drawback, a small-sized CNN network called SPD model is suggested. A high computational speed of detection can be obtained while performing moving object localization with sparse optical flow and keeping detection performance compliant through SPD models composed of lightweight CNN layers. Furthermore, region partitioning for a group of pedestrians appearing in an ROI improves performance of pedestrian detection. Depending on the performance requirements, the accuracy, the computing time or cost, users can choose appropriate models. The suggested approach is an alternative for pedestrian detection with a low-cost embedded board.

6.4 Future works

The proposed optical flow object tracking algorithm has a limitation that the camera must be fixed, and if there are many objects moving on the screen, the object may not be captured individually. Future studies are proposed to improve the object tracking algorithm through the optical flow that I proposed. To capture the moving object as individually as possible, the feature point area in which the optical flow is captured may be further restricted by placing the center of gravity or weight of the feature points. Regarding camera ego-moving restrictions, unlike conventional background subtraction or pixel differentiation, a study is required to select feature points that are distinguished from background movements and have detected movement of objects based on feature points.

The solution is presented to find and solve a region that is hard to grasp on various datasets. There has been suggested no solution to reduce the generalization error revealed by the cross-subject validation. In the future, Incremental learning could be one alternative to improve the performance of object detection for a target subject. A proper choice of source subject in sequence can help enhance the performance. By following

the partial-order relations to influence a target subject, a source subject can be added one-by-one for training a deep neural network model. It could be necessary to check whether the constructing the learning data is sufficient to improve the performance of the detection model in the desired environment by applying the proposed strategy to other unsolved datasets in addition to the dataset prepared. Furthermore, research is required on whether the model learning performance of proposed strategy can be improved through a combination of three or more dataset relationships rather than two combinations.

It is a developed algorithm to classify objects moving with object detection models in the optical flow object tracking area. To improve detection performance, it is possible to consider a study where vectors of a different dimension are used in the SVM model classifying HOG features for each specific optical flow region. If research on individual optical flow box generation, previously considered in the object tracking section, makes further progress, then the tracking area can be sized to fit the input vector dimension of the SVM classifier and it can be applied for a classification model to the area. The advantage of this is that it is possible to reduce the computational requirement by performing classification once for a given input area without the need for classification while moving the detector window. The reason is that the location of the optical flow box is determined by the location of the classified object, reducing the number of operations performed on the object location by the detection models.

Based on the CNN feature extractor, it is possible to develop a deep learning model that is effective for classification functions with less computational cost. By applying developed deep learning or CNN-based detection models in an area extracted from the optical flow method, one achieves a moving object detection model which is more robust, faster, and with improved detection performance. Furthermore, developed separation of ROI to an individual object will improve the detection performance of the proposed OF+SPD model. The detection performance of the proposed model could be further improved if the region obtained from the sparse optical flow is more precisely segmented.

Bibliography

Agarwal, A., Gupta, S., and Singh, D. K. (2016). Review of optical flow technique for moving object detection. In *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, pages 409–413. IEEE.

Aslani, S. and Mahdavi-Nasab, H. (2013). Optical flow based moving object detection and tracking for traffic surveillance. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, 7(9):1252–1256.

Bai, X., Zhang, H., and Zhou, J. (2014). VHR object detection based on structural feature extraction and query expansion. *IEEE Transactions on Geoscience and Remote Sensing*, 52(10):6508–6520.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359.

Bertozzi, M., Castangia, L., Cattani, S., Prioletti, A., and Versari, P. (2015). 360 Detection and tracking algorithm of both pedestrian and vehicle using fisheye images. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 132–137. IEEE.

Blott, G., Takami, M., and Heipke, C. (2018). Semantic segmentation of fisheye images. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 181–196.

Bouguet, J.-Y. et al. (2001). Pyramidal implementation of the affine Lucas-Kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4.

Braillon, C., Pradalier, C., Crowley, J. L., and Laugier, C. (2006). Real-time moving obstacle detection using optical flow models. In *2006 IEEE Intelligent Vehicles Symposium*, pages 466–471. IEEE.

Brutzer, S., Höferlin, B., and Heidemann, G. (2011). Evaluation of background sub-

traction techniques for video surveillance. In *Computer Vision and Pattern Recognition Conference (CVPR) 2011*, pages 1937–1944. IEEE.

Chae, S. and Yoshida, T. (2010). Application of RFID technology to prevention of collision accident with heavy equipment. *Automation in Construction*, 19(3):368–374.

Chan, K. L. (2013). Detection of swimmer using dense optical flow motion map and intensity information. *Machine Vision and Applications*, 24(1):75–101.

Chao, C.-H., Hsu, P.-L., Lee, H.-Y., and Wang, Y.-C. F. (2020). Self-supervised deep learning for fisheye image rectification. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2248–2252. IEEE.

Chen, S., Xu, T., Li, D., Zhang, J., and Jiang, S. (2016). Moving object detection using scanning camera on a high-precision intelligent holder. *Sensors*, 16(10):1758.

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollár, P., and Zitnick, C. L. (2015). Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Chen, X., Li, S., Mersch, B., Wiesmann, L., Gall, J., Behley, J., and Stachniss, C. (2021). Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data. *IEEE Robotics and Automation Letters*, 6(4):6529–6536.

Chen, Y., Nasrabadi, N. M., and Tran, T. D. (2011). Simultaneous joint sparsity model for target detection in hyperspectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 8(4):676–680.

Cheng, G. and Han, J. (2016). A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:11–28.

Chi, S. and Caldas, C. H. (2012). Image-based safety assessment: Automated spatial safety risk identification of earthmoving and surface mining activities. *Journal of Construction Engineering and Management*, 138(3):341–351.

Chiu, Y.-C., Tsai, C.-Y., Ruan, M.-D., Shen, G.-Y., and Lee, T.-T. (2020). Mobilenet-SSDv2: An improved object detection model for embedded systems. In *2020*

International Conference on System Science and Engineering (ICSSE), pages 1–5. IEEE.

Cho, J., Lee, K., Shin, E., Choy, G., and Do, S. (2015). How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? *arXiv preprint arXiv:1511.06348*.

Choi, H., Kang, B., and Kim, D. (2022a). Investigating cross-test approach on fisheye camera environments of vehicles using deep learning model (submitted). *Measurement*.

Choi, H., Kang, B., and Kim, D. (2022b). Moving object tracking based on sparse optical flow with moving window and target estimator. *Sensors*, 22(8):2878.

Choi, H. and Kim, D. (2022). Pedestrian detection based on sparse optical flow and convolutional neural network model (being prepared). *Applied Soft Computing*.

Chou, J.-Y. and Chang, C.-M. (2021). Image motion extraction of structures using computer vision techniques: A comparative study. *Sensors*, 21(18):6248.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE.

Das, S., Mirnalinee, T., and Varghese, K. (2011). Use of salient features for the design of a multistage framework to extract roads from high-resolution multispectral satellite images. *IEEE Transactions on Geoscience and Remote sensing*, 49(10):3906–3931.

De Morsier, F., Tuia, D., Borgeaud, M., Gass, V., and Thiran, J.-P. (2013). Semi-supervised novelty detection using SVM entire solution path. *IEEE Transactions on Geoscience and Remote Sensing*, 51(4):1939–1950.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

Diamantas, S. and Alexis, K. (2018). Optical flow based background subtraction

with a moving camera: Application to autonomous driving. *arXiv preprint arXiv:1811.06660*.

Do, Y. (2020). Human detection in images using optical flow and learning. *Journal of Sensor Science and Technology*, 29(3):194–200.

Eikvil, L., Aurdal, L., and Koren, H. (2009). Classification-based vehicle detection in high-resolution satellite images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(1):65–72.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, pages 363–370. Springer.

Fei-Fei, L. and Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531. IEEE.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., Wiesbeck, W., and Dietmayer, K. (2020). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *The International Conference on Machine Learning (ICML)*, volume 96, pages 148–156. Citeseer.

Fu, C., Duan, R., Kircali, D., and Kayacan, E. (2016). Onboard robust visual tracking for UAVs using a reliable global-local object model. *Sensors*, 16(9):1406.

Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A., and Berg, A. C. (2017). DSSD: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*.

Gao, R., Zhu, H., Liao, Q., Qu, B., Hu, L., and Wang, H. (2022). Detection of coal fire by deep learning using ground penetrating radar. *Measurement*, 201:111585.

Garcia-Garcia, B., Bouwmans, T., and Silva, A. J. R. (2020). Background subtraction in real applications: Challenges, current models and future directions. *Computer Science Review*, 35:100204.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448.

Gong, J. and Caldas, C. H. (2011). Learning and classifying motions of construction workers and equipment using bag of video feature words and Bayesian learning methods. In *Computing in Civil Engineering (2011)*, pages 274–281.

Hariyono, J., Hoang, V.-D., and Jo, K.-H. (2014). Moving object localization using optical flow for pedestrian detection from a moving vehicle. *The Scientific World Journal*, 2014.

Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, 17(1-3):185–203.

Hou, J., Jiang, H., Wan, C., Yi, L., Gao, S., Ding, Y., and Xue, S. (2022). Deep learning and data augmentation based data imputation for structural health monitoring system in multi-sensor damaged state. *Measurement*, 196:111206.

Huang, R., Pedoeem, J., and Chen, C. (2018). YOLO-LITE: A real-time object detection algorithm optimized for non-GPU computers. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2503–2510. IEEE.

Inglada, J. (2007). Automatic recognition of man-made objects in high resolution optical remote sensing images by SVM classification of geometric image features. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(3):236–248.

Ishimoto, H. and Tsubouchi, T. (2013). Stereo vision based worker detection system for excavator. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 30, page 1. IAARC Publications.

KaewTraKulPong, P. and Bowden, R. (2002). An improved adaptive background mix-

ture model for real-time tracking with shadow detection. In *Video-based Surveillance Systems*, pages 135–144. Springer.

Kale, K., Pawar, S., and Dhulekar, P. (2015). Moving object tracking using optical flow and motion vector estimation. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, pages 1–6. IEEE.

Kim, D.-S. and Kwon, J. (2016). Moving object detection on a vehicle mounted back-up camera. *Sensors*, 16(1):23.

Kim, M. S., Seo, J.-H., Kwon, N. K., Song, J.-m., and Park, P. (2016). Real-time moving object detection using a vehicle-mounted monocular rear-view fisheye camera. In *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–6. IEEE.

Kumar, D., Dewangan, A., Tiwari, R., and Bordoloi, D. (2021). Identification of inlet pipe blockage level in centrifugal pump over a range of speeds by deep learning algorithm using multi-source data. *Measurement*, 186:110146.

Kursun, O. and Favorov, O. V. (2010). Feature selection and extraction using an unsupervised biologically-suggested approximation to Gebelein's maximal correlation. *International Journal of Pattern Recognition and Artificial Intelligence*, 24(03):337–358.

Kushwaha, A., Khare, A., Prakash, O., and Khare, M. (2020). Dense optical flow based background subtraction technique for object segmentation in moving camera environment. *IET Image Processing*, 14(14):3393–3404.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view RGB-D object dataset. In *2011 IEEE International Conference on Robotics and Automation*, pages 1817–1824. IEEE.

Leger, P. C., Rowe, P., Bares, J., Boehmke, S., and Stentz, A. (1999). Obstacle detection and safeguarding for a high-speed autonomous hydraulic excavator. In *Mobile Robots XIII and Intelligent Transportation Systems*, volume 3525, pages 146–156. International Society for Optics and Photonics.

Li, E., Zhou, Z., and Chen, X. (2018). Edge intelligence: On-demand deep learning

model co-inference with device-edge synergy. In *Proceedings of the 2018 Workshop on Mobile Edge Communications*, pages 31–36.

Li, J., Zhang, H., Zhang, L., Li, Y., Kang, Q., and Wu, Y. (2019). Multi-scale HOG feature used in object detection. In *Tenth International Conference on Graphics and Image Processing (ICGIP 2018)*, volume 11069, pages 1113–1119. SPIE.

Liang, Y., Huang, H., Cai, Z., Hao, Z., and Tan, K. C. (2019). Deep infrared pedestrian classification based on automatic image matting. *Applied Soft Computing*, 77:484–496.

Lienhart, R. and Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. In *Proceedings. International Conference on Image Processing*, volume 1, pages I–I. IEEE.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2):261–318.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer.

Liu, W., Liao, S., Ren, W., Hu, W., and Yu, Y. (2019). High-level semantic feature detection: A new perspective for pedestrian detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5187–5196.

Liu, Y., Sun, P., Wergeles, N., and Shang, Y. (2021). A survey and performance evaluation of deep learning methods for small object detection. *Expert Systems with Applications*, 172:114602.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157. IEEE.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Lv, Z., Wang, W., Xu, Z., Zhang, K., Fan, Y., and Song, Y. (2021). Fine-grained object detection method using attention mechanism and its application in coal-gangue detection. *Applied Soft Computing*, 113:107891.

Ma, G., Dwivedi, M., Li, R., Sun, C., and Kummert, A. (2009). A real-time rear view camera based obstacle detection. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–6. IEEE.

Mehryary, F., Björne, J., Pyysalo, S., Salakoski, T., and Ginter, F. (2016). Deep learning with minimal training data: TurkuNLP entry in the bioNLP shared task 2016. In *Proceedings of the 4th BioNLP shared task workshop*, pages 73–81.

Meng, L., Peng, Z., Zhou, J., Zhang, J., Lu, Z., Baumann, A., and Du, Y. (2020). Real-time detection of ground objects based on unmanned aerial vehicle remote sensing with deep learning: Application in excavator detection for pipeline safety. *Remote Sensing*, 12(1):182.

Miyamoto, K. (1964). Fish eye lens. *Journal of the Optical Society of America A*, 54(8):1060–1061.

Montero, A. S., Lang, J., and Laganiere, R. (2015). Scalable kernel correlation filter with sparse feature integration. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 587–594. IEEE.

Moore, R. C. and Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference short papers*, pages 220–224.

Piccardi, M. (2004). Background subtraction techniques: A review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 4, pages 3099–3104. IEEE.

Plastiras, G., Kyrkou, C., and Theocharides, T. (2018). Efficient ConvNet-based object detection for unmanned aerial vehicles by selective tile processing. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, pages 1–6.

Prasad, D. K. (2012). Survey of the problem of object detection in real images. *International Journal of Image Processing (IJIP)*, 6(6):441.

Qiu, T., Zheng, K., Song, H., Han, M., and Kantarci, B. (2017). A local-optimization emergency scheduling scheme with self-recovery for a smart grid. *IEEE Transactions on Industrial Informatics*, 13(6):3195–3205.

Qiu, Y. and Lu, J. (2021). A visualization algorithm for medical big data based on deep learning. *Measurement*, 183:109808.

Ramzan, H., Fatima, B., Shahid, A. R., Ziauddin, S., and Safi, A. A. (2016). Intelligent pedestrian detection using optical flow and HOG. *International Journal of Advanced Computer Science and Applications*, 7(9).

Rasul, A., Seo, J., and Khajepour, A. (2021). Development of sensing algorithms for object tracking and predictive safety evaluation of autonomous excavators. *Applied Sciences*, 11(14):6366.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28:91–99.

Roberts, D. and Golparvar-Fard, M. (2019). End-to-end vision-based detection, tracking and activity analysis of earthmoving equipment filmed at ground level. *Automation in Construction*, 105:102811.

Shafiee, M. J., Chywl, B., Li, F., and Wong, A. (2017). Fast YOLO: A fast You Only Look Once system for real-time embedded object detection in video. *arXiv preprint arXiv:1709.05943*.

Shi, J. et al. (1994). Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600. IEEE.

Sobral, A. and Vacavant, A. (2014). A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21.

Son, H., Seong, H., Choi, H., and Kim, C. (2019). Real-time vision-based warning system for prevention of collisions between workers and heavy equipment. *Journal of Computing in Civil Engineering*, 33(5):04019029.

Son, H., Sung, H., Choi, H., Lee, S., and Kim, C. (2017). Detection of nearby obstacles with monocular vision for earthmoving operations. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 34. IAARC Publications.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852.

Szegedy, C., Toshev, A., and Erhan, D. (2013). Deep neural networks for object detection. *Advances in Neural Information Processing Systems*, 26.

Tadjine, H., Hess, M., and Karsten, S. (2013). Object detection and classification using a rear in-vehicle fisheye camera. In *Proceedings of the FISITA 2012 World Automotive Congress*, pages 519–528. Springer.

Teizer, J. (2015a). Safety 360: Surround-view sensing to comply with changes to the ISO 5006 earth-moving machinery-operator's field of view-test method and performance criteria. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 32, page 1. IAARC Publications.

Teizer, J. (2015b). Wearable, wireless identification sensing platform: Self-monitoring alert and reporting technology for hazard avoidance and training (SmartHat). *Journal of Information Technology in Construction (ITcon)*, 20(19):295–312.

Teizer, J., Allread, B. S., Fullerton, C. E., and Hinze, J. (2010). Autonomous pro-active real-time construction worker and equipment operator proximity safety alert system. *Automation in Construction*, 19(5):630–640.

Tong, K., Wu, Y., and Zhou, F. (2020). Recent advances in small object detection based on deep learning: A review. *Image and Vision Computing*, 97:103910.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999.

Viola, P. and 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*, volume 1, pages I–I. IEEE.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.

Von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 319–326.

Wang, K., Liu, M., and Ye, Z. (2021). An advanced YOLOv3 method for small-scale road object detection. *Applied Soft Computing*, 112:107846.

Wang, Z., Shao, Y.-H., Bai, L., and Deng, N.-Y. (2015). Twin support vector machine for clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2583–2588.

Wang, Z. and Yang, X. (2018). Moving target detection and tracking based on pyramid Lucas-Kanade optical flow. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 66–69. IEEE.

Wu, Z.-Z., Wang, X.-F., Zou, L., Xu, L.-X., Li, X.-L., and Weise, T. (2021). Hierarchical object detection for very high-resolution satellite images. *Applied Soft Computing*, 113:107885.

Xiang, X., Bao, W., Tang, H., Li, J., and Wei, Y. (2016). Vehicle detection and tracking for gas station surveillance based on AdaBoosting and optical flow. In *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pages 818–821. IEEE.

Xu, C. and Hu, X. (2020). Real time detection algorithm of parking slot based on deep learning and fisheye image. In *Journal of Physics: Conference Series*, volume 1518, page 012037. IOP Publishing.

Xue, Z., Xue, N., Xia, G.-S., and Shen, W. (2019). Learning to calibrate straight lines for fisheye image rectification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1643–1651.

Yankun, Z., Hong, C., and Weyrich, N. (2011). A single camera based rear obstacle detection system. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 485–490. IEEE.

Yavariabdi, A., Kusetogullari, H., Celik, T., and Cicek, H. (2021). FastUAV-Net: A multi-UAV detection algorithm for embedded platforms. *Electronics*, 10(6):724.

Yin, X., Wang, X., Yu, J., Zhang, M., Fua, P., and Tao, D. (2018). FisheyeRecNet: A multi-context collaborative deep network for fisheye image rectification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 469–484.

Yuwono, E. I., Tjondonegoro, D., Sorwar, G., and Alaei, A. (2022). Scalability of knowledge distillation in incremental deep learning for fast object detection. *Applied Soft Computing*, 129:109608.

Zhang, J., Ding, Y., Xu, H., and Yuan, Y. (2019a). An optical flow based moving objects detection algorithm for the UAV. In *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, pages 233–238. IEEE.

Zhang, S., Benenson, R., Schiele, B., et al. (2015a). Filtered channel features for pedestrian detection. In *CVPR*, volume 1, page 4.

Zhang, S., Wang, T., Wang, C., Wang, Y., Shan, G., and Snoussi, H. (2019b). Video object detection base on RGB and optical flow analysis. In *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*, pages 280–284. IEEE.

Zhang, Y., Tong, X., Yang, T., and Ma, W. (2015b). Multi-model estimation based moving object detection for aerial video. *Sensors*, 15(4):8214–8231.

Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232.

Zhu, J., Wang, Z., Wang, S., and Chen, S. (2020). Moving object detection based on background compensation and deep learning. *Symmetry*, 12(12):1965.

Zhu, X., Vondrick, C., Fowlkes, C. C., and Ramanan, D. (2016). Do we need more training data? *International Journal of Computer Vision*, 119(1):76–92.

Zhu, X., Vondrick, C., Ramanan, D., and Fowlkes, C. C. (2012). Do we need more training data or better models for object detection?. In *The British Machine Vision Conference (BMVC)*, volume 3. Citeseer.

Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.