UNIVERSITY OF CALIFORNIA
RIVERSIDE

High-Fidelity Vehicle Detection, Positioning and Tracking with Infrastructure-Based Fisheye Cameras

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Jiahe Cao

September 2023

Thesis Committee:

Dr. Matt Barth, Co-Chairperson
Dr. Guoyuan Wu, Co-Chairperson
Dr. Salman Asif

The Thesis of Jiahe Cao is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am very grateful to my advisors Dr. Guoyuan Wu and Prof. Matthew Barth for their invaluable advice, continuous support, and patience. Without their help, I would not have been here. Their vast expertise and abundant experience have inspired me. Also, their high standard of academic research taught me to treat research seriously and precisely. I would like to thank Guoyuan deeply not only his great inspiration and supervision of my study but also for how to become an excellent researcher from a student by all means.

I would also like to thank Toyota InfoTech Labs who sponsor the program and contribute to this program. I would like to thank Dr. Yongkang Liu, Zhouqiao Zhao, and Zhengwei Bai, who provide detailed and technical support throughout my pursuit of Master degree. Their fruitful discussion and creative ideas are invaluable to my study.

To my parents for all the tremendous understanding and encouragement. I would not have finished this thesis without my parents' full support.

ABSTRACT OF THE THESIS

High-Fidelity Vehicle Detection, Positioning and Tracking with Infrastructure-Based Fisheye Cameras

by

Jiahe Cao

Master of Science, Graduate Program in Electrical Engineering
University of California, Riverside, September 2023
Dr. Matt Barth and Dr. Guoyuan Wu, Co-Chairpersons

Urban expansion has necessitated the development of a more efficient transportation system, emerging technologies, and services of Intelligent Transport Systems (ITS). ITS aim to offer innovative services for various transportation modes and traffic management, empowering users with better information and facilitating safer, more coordinated, and intelligent utilization of transport networks. This thesis specifically investigates detection based on infrastructure camera sensors, with a focus on fisheye cameras, which are incompatible with conventional detection methods.

Two distinct approaches are proposed to address these challenges. The first employs a traditional method, Background Subtraction, enhanced with a multi-layer structure. This method successfully detects vehicles throughout the whole day as the lighting conditions and weather change slowly or dramatically. In the simulation experiment, compared with state-of-the-art methods, our BS method demonstrates significant improvements by 6.22% in multiple object tracking accuracy (MOTA) and 1.71 pixels in multiple objects tracking precision (MOTP).

The second approach involves deep learning. While there have been numerous attempts to apply deep learning to fisheye camera detection, the absence of a comprehensive and large-scale fisheye image dataset often results in poor detection performance. To overcome this issue, the second method generates a realistic-style dataset using a modified AttentionGAN (a combination of CycleGAN and Transformer) with ground truth labels produced by CARLA. Deep learning object detection techniques can be directly trained on the generated dataset. This project employs YOLOv5 as the detection network. The final experiment demonstrates that the trained network effectively detects 100% vehicles in the generated dataset and 16.78% vehicles in the real-world dataset which is superior to the YOLOV5 trained on the labeled CARLA dataset. The results verify the feasibility of using style transfer for infrastructure fisheye camera sensing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

This thesis focuses on Intelligent Transportation Systems (ITS) and aims to solve and enhance the sensing system to address issues related to efficiency and safety. More specifically, this thesis attempts to propose high-fidelity vehicle detection methods by infrastructure-based sensing (in particular, fisheye cameras) in the urban intersection environment to better monitor and manage the transportation system associated with safety, efficiency, and environmental problems. Also, other traffic systems such as Emergency vehicle notification systems, Variable speed limits and so on would benefit from real-time high-fidelity vehicle detection greatly.

In this study, two kinds of vehicle detection methods are realized for the infrastructure-based fish-eye camera. Each of them represents different technique routes. Both of them are capable of detecting vehicles from the fisheye camera in real time. It should be noted that the first method provides the necessary dataset for the second deep-learning-based method.

1.2 Motivation

1.2.1 Safety

Traffic safety is a crucial element in the pursuit of improving transportation networks around the world. As urban populations continue to expand and the demand for efficient mobility increases, the need for innovative solutions to address traffic congestion and accidents has never been greater. Intelligent Transportation Systems (ITS) hold the key to revolutionizing traffic safety and enhancing the overall quality of life for millions of commuters.

According to the California Office of Traffic Safety (OTS), in 2020, there were 3846 traffic fatalities in California. Furthermore, the OTS reports the 5-year rolling average traffic fatalities to keep increasing from 2016 to 2020.

This alarming data highlights the urgent need for innovative solutions, such as ITS, to minimize the risk of human error and create safer road environments. Also, ITS can significantly reduce the number of traffic accidents and fatalities by using real-time data to predict and prevent collisions. For instance, the US Department of Transportation (USDOT) [4] has reported that Connected Vehicle technology, a critical component of ITS, can potentially prevent or reduce the severity of up to 80% of non-impaired crashes.

Furthermore, ITS can play a crucial role in improving emergency response times, which is vital for saving lives in the aftermath of a traffic accident. According to [5], rapid ambulance response time is associated with a higher rate of survival. By utilizing ITS to manage traffic flow more effectively, emergency vehicles can navigate through congested areas more efficiently, potentially saving countless lives.

1.2.2 Infrascructure-based Sensing

The trend of ITS shows the great demand for precise localization and reliable information from different sensor sources. Thus, infrastructure-based sensing becomes a hotspot for its various specifics and advantages.

Other than onboard sensing, infrastructure-based sensing has a wider perception range. Owing to its fixed position, infrastructure-based sensing can be armed with high-performance servers which provide the presence of excellent detection models and the ability for accurate detection and localization. Also, the different information sources (on-board and infrastructure-based sensing) are complementary to each other so they can work and coordinate together for better detection, localization, and robustness.

Usually, infrastructure-based sensing is connected to the central server and the detection result is broadcasted to the transportation system and users (vehicles, cyclists, and pedestrians). Users (commonly vehicles) can locate themselves more precisely by combining the self-GPS, and IMU with the detection result sent by the central server.

Inside infrastructure-based sensing, there are two main kinds of sensors, camera, and LiDAR. Due to the widespread usage of road surveillance cameras, we choose cameras as the sensor for perception. Thus, our methods can be applied in practice without too much change to the hardware.

1.2.3 Deep Learning

The method we focus heavily on employing is deep learning, specifically its application in computer vision, which has played a significant role in the development of

autonomous driving and Intelligent Transportation Systems (ITS). Object detection and semantic segmentation are essential for autonomous vehicles, allowing them to perceive their environment, detect vehicles, pedestrians, and cyclists, and accurately identify scene details such as traffic lights and lane markings on the road. These advancements have contributed significantly to enhancing the safety and efficiency of ITS.

Apart from pure computer vision, deep learning also has been used on LiDAR sensors to detect, track, and predict the movement of objects in 3D space, providing a more comprehensive understanding of the environment for autonomous vehicles and ITS.

Other than the applications mentioned above, some methods focus on vehicle and pedestrian behavior prediction via Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM). These works predict the future behavior of vehicles and pedestrians, enhancing the ability of autonomous vehicles to anticipate and avoid potential hazards.

AI-generated content, one branch of deep learning, although it is seldom mentioned in ITS and autonomous driving, has gained much attention since 2021. Generative Pre-training Transformer (gpt), especially ChatGPT in NLP has launched the next-generation technique revolution since the amazing dialogue performance. Back to the vision field, Generative Adversarial Network (GAN) and stable diffusion are two major generative methods.

1.3 Main Contributions

This section simply illustrates and summarizes all works in this thesis and related contributions. A more efficient traffic system is required as cities continue to develop, leading to the proposal of Intelligent Transport Systems. The aim of this system is to

provide innovative services related to different modes of transportation and traffic management, enabling users to be better informed and make smarter use of transport networks while increasing safety and coordination. This thesis specifically focuses on detection using infrastructure camera sensors, particularly fisheye cameras, for which previous detection methods cannot be used directly.

Two different methods were developed to solve this problem and they compose a whole workflow as shown in Fig. 1.1. The first method is a traditional one based on Background Subtraction, modified with a multi-layer structure, and capable of detecting vehicles all day, irrespective of weather and illumination changes. In the CARLA experiment, the method showed significant improvement in terms of MOTA and MOTP. In the most complex scenario, the method outperformed the SOTA by 6.22% on MOTP and 1.71 pixels on MOTA.

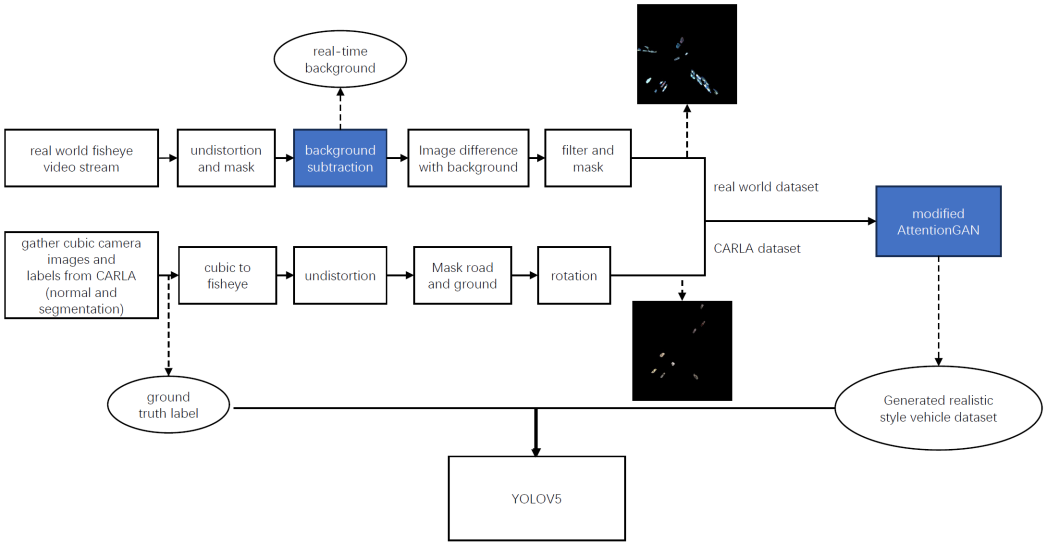


Figure 1.1: The whole workflow of infrastructure-based fisheye camera sensing.

- Develop a traditional fisheye camera vehicle detection method based on Background Subtraction with a multi-layer structure. This method is able to separate vehicles from video frames at the pixel level and obtain all vehicles' positions at longitude and latitude.

- Test this detection method with an ego-vehicle that can get its real-time position.

Deep learning is the focus of the second method, which aims to create a dataset that has a similar distribution as the real world but is generated in a simulator (CARLA) to address the issue of the lack of a large-scale labeled dataset for fisheye camera images. A real-world style dataset with ground truth labels is generated using a modified AttentionGAN (a combination of CycleGAN and Transformer). Deep learning object detection methods can then be directly trained on the generated dataset, with YOLOV5 as the detection network in this study. The experiment shows that the trained network can detect all vehicles in the verified part of the generated dataset and some vehicles in the real-world dataset. Also, this experiment verifies the feasibility of the workflow shown in Fig 1.1.

- Develop modified AttentionGAN to learn the real-world vehicles' style and transfer CARLA vehicle images into real-world vehicle images with the same ground truth labels.

- Train and validate the YOLOV5 model with the generated real-world vehicle dataset and test it in the real-world intersection scenario.

1.4 Structure of Thesis

The detailed structure of the thesis is shown in Fig. 1.2.

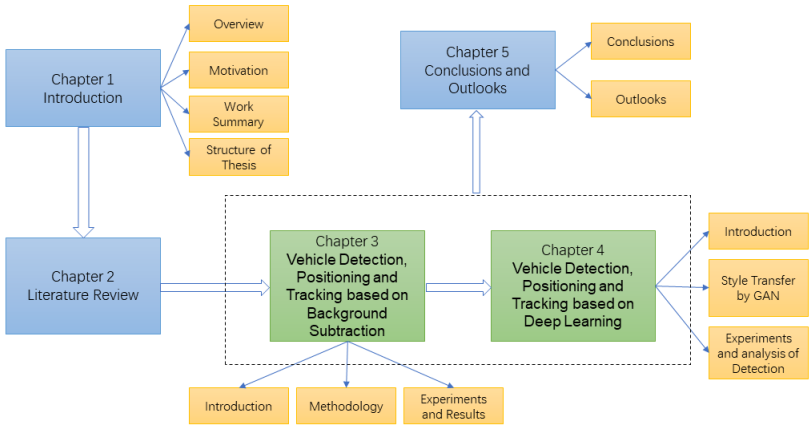


Figure 1.2: Structure of Thesis

Chapter 1 briefly introduces two infrastructure-based sensing methods proposed by this thesis and the motivation of these two methods. Chapter 2 reviews related research fields including ITS, vehicle detection methods (onboard sensing and infrastructure-based sensing), fisheye cameras, and deep-learning-based algorithms. Chapter 3 introduces a traditional vehicle detection algorithm based on Background Subtraction. Chapter 4 focuses on the Style Transfer of the traffic dataset by attention mechanism and GAN, and the deep-learning object detection method is included in this chapter. Chapter 5 summarizes the key takeaways of this thesis and presents future work related to infrastructure-based fisheye camera sensing.

Chapter 2

Literature Review

Many studies have been conducted on the fisheye camera model, object detection, and infrastructure-based sensing in the past few years. This chapter reviews necessarily related information and studies.

2.1 Infrastructure-based Traffic Surveillance

As mentioned above, this thesis mainly targets at solving the infrastructure-based sensing problem posed by the fisheye camera. Infrastructure-based traffic sensing or roadside-based traffic sensing plays an important role in the Intelligent Transport Systems mentioned above. Infrastructure sensors are installed or embedded on the road or near the road. Especially, vehicle-sensing systems may include the deployment of infrastructure-to-vehicle and vehicle-to-infrastructure(Optional) electronic beacons for identification communications.

2.1.1 Sensor Types

This section mainly introduces different types of roadside-based sensors according to the basics of how they work.

The first traffic recorder, operated off a strip across the street was invented in 1937 [6]. Later on, vehicle detection loops, called inductive-loop traffic detectors were invented. They should be embedded under the pavement and can detect vehicles passing or staying above detectors. Due to their reliability, they are widely used at intersections or along freeways, cooperating with traffic lights or ramp metering. When vehicles approach detectors, traffic lights or freeway lights could switch on or off.

Wireless detection includes Bluetooth, WLAN, RFID, Ultra-Wide Band (UWB), etc [7] [8]. All wireless detection requires a vehicle-side tag for roadside sensor perception. When vehicles carrying on wireless devices pass through sensing devices along the road, sensing devices are able to detect and perceive wireless devices or tags inside vehicles.

Bluetooth detection works in the 2.4-GHz ISM band. Bluetooth sensors are able to detect tags within 30 meters which requires technologies for higher accuracy. Traditionally, Multi-sensors and Received Signal Strength Indicator (RSSI) analyses are used here. Combined with Multi-sensor installation, it is able to calculate the tag's location by triangulation.

RFID is composed of RFID readers and tags, they are designed with RF-compatible integrated circuits which can transmit electromagnetic waves. Readers can send also receive radio frequency waves from tags, and the wave is modulated with identification information. RFID tags are classified into two categories, passive and active. Passive RFID tags

work without a power supply. They reflect waves sent from RFID readers and modulate signals with information. However, due to its working method, RFID readers cost much, the reflected wave is weak and its valid reading distance is within 1-2 m. Active tags are different from using batteries so they can receive and send electromagnetic waves. Because active tags can send signals actively, they have a larger detection distance.

Ultra-Wideband (UWB) is a technology for transmitting information across a wide bandwidth (>500 MHz). It allows communication on the short-range, high frequency (3.1-10.6 GHz) with a low energy need by pulse position or time modulation. Time of arrival (TOA)/Time Of Flight (TOF) and time difference of arrival (TDOA) could be used to precisely locate tags at a very high precise accuracy (20 cm).

Video camera traffic surveillance is another well-spread infrastructure-based sensing since video detection systems such as those used in automatic number plate recognition. The video from cameras is fed into computation units such as CPU, GPU, or FPGA to analyze the content of the video image. Based on various video processing algorithms and perception algorithms, infrastructure-based sensing is realized with even many complex functions such as lane-by-lane vehicle speeds, counts, lane occupancy readings, and so on. Also, unlike other traffic detection methods, video cameras do not require installation directly into the road surface or roadbed, which is known as a "non-intrusive" method. These benefits enable infrastructure-based sensing based on cameras to be well-used in the world. More unique cameras are reviewed according to the camera types and their working principle.

Fisheye camera is widely applied in Intelligent Transport Systems for its large field of view. Compared to highways and rural roads, urban traffic situations can be more complex and more challenging, especially at intersections, than those on highways and rural roads. The large field of view enables the fisheye camera to capture more content than the traditional pin-hole camera. However, the great edge distortion bought from the large field of view leads to new challenges to existing object detection methods. More details about the fisheye camera are introduced in Chapter 2.4.

The thermal camera [9] perceives environments through infrared and creates a crisp image based on subtle temperature differences. It does not need any visible light whatsoever as well as does not blinded by direct sunlight. Thus, it can work under extreme conditions such as sun glare, headlights glare, or low light at night. The thermal camera's independence of lighting enables it to be widely applied in infrastructure-based sensing.

The event camera [10], also known as a dynamic vision sensor (DVS), is a type of vision sensor that operates differently from traditional cameras. Instead of capturing static frames at fixed intervals, event cameras detect changes in brightness at each pixel and report these changes as they occur. They are therefore capable of capturing very fast movements and can operate under a wide range of lighting conditions. The pixels of an event camera operates independently and asynchronously, meaning that whenever there is a significant change in light intensity at a particular pixel, an event is triggered, which contains the coordinates of the pixel, the time of the event, and the sign of the change in intensity. With its wide working range of light conditions and high-speed response, it is ideal for detecting incidents such as accidents or road obstructions.

LiDAR [11], or Light Detection and Ranging, is an optical remote sensing technique that measures the distance of a target by illuminating it with light, often a pulsed laser, and measuring the reflected light. The differences in laser return times and wavelengths can then be used to produce a digital 3-D representation of the target. Using LiDAR, it is possible to obtain high-resolution point clouds that provide high-resolution images. As a result, it can be used to measure distances and to draw maps. LiDAR is widely used to monitor the speed and volume of traffic, identify individual vehicles, and detect incidents such as accidents or traffic jams by scanning the road surface.

2.1.2 Comparison of Onboard Sensing and Infrastructure-based Sensing

According to the sensors' installation location, sensors can be assorted into two kinds, on-board and infrastructure-based. Also, most vehicle companies focus on onboard sensing to realize automatic driving. The production models from vehicle companies are armed with GPS, cameras, LiDARs, Rader, and other sensors in order to realize onboard sensing. Prior knowledge of HD maps contributes to self-localization.

For onboard sensors such as video cameras, they are installed around vehicles so as to cover all-around FOV. Benefiting from the large field of view, it is able to establish a bird-eye view image for vehicles so drivers can notice any obstacles around the vehicles which is commonly used in reverse gear.

Beyond that, onboard sensing has the below advantages.

- Onboard sensing has a low time delay of perception for ego vehicles.
- Onboard sensing perceives surrounding and the nearest objects directly.

- Onboard sensing localizes itself directly.
- Onboard sensing could independently work without communication to ITS

However, onboard sensing also has these limitations.

- It is hard to upgrade sensors and computation ability for onboard sensing.
- It is expensive and time-consuming to upload real-time raw data to the central server of ITS
- The mobility working conditions of onboard sensing may bring instability to sensors

Because sensors are mounted on vehicles, it is expensive and difficult to update sensors and computation platforms and optimize configuration. To address this problem, vehicle companies launch several models every year to catch up the state-of-the-art techniques, both in software and hardware. Although these companies may update the autonomous vehicle system (detection, perception, communication, and so on), the past vehicles still can not match the newest vehicles.

At the same time, the severe working condition (vibration and high speed) requires automotive-grade sensors, which are high cost to manufacture and repair. For ITS, it is difficult and high-cost to transfer huge amounts of raw data from onboard sensing to the central server.

The aforementioned problems of onboard sensing limit ITS development. Thus, V2X, IoT, and infrastructure-based sensing are introduced in ITS. Different from onboard sensing, infrastructure-based sensing is position-fixed which brings so many benefits.

- Infrastructure-based sensing always is precise and high-resolution.
- Infrastructure-based sensing has large perception fields.
- Infrastructure-based sensing is easy to be upgraded.
- Infrastructure-based sensing has high transmission bandwidths to ITS.

The fixed position enables the high-resolution or performance sensors which can not be used on vehicles because of the size to be installed on infrastructure. Benefiting from the large sensor volume, these sensors have higher resolution, working frequency, and lower noise than sensors used on vehicles.

Generally, infrastructure-based sensors are installed at a high height, etc on the traffic light. The height gives a relatively large perception so only several infrastructure-based sensors are sufficient to cover one specific scenario.

Also, infrastructure-based sensors are all upgradeable. For onboard sensing, onboard sensors are designed to be small or thin and fit into the installation slot and they are placed inside the vehicles commonly. Thus, it is hard and complex for onboard sensors to upgrade. On the contrary, infrastructure-based sensors are installed on the surface of infrastructure so they can be upgraded conveniently.

The last and most beneficial point is the high data transmission bandwidth. Due to the fixed position, infrastructure sensors can be connected by Ethernet wire or other wire communication which is reliable and high-performance on data transmission. The high data transmission bandwidth enables the connection to the high-performance central server, and numerous methods for data fusion, processing, and perception.

2.2 Traditional Approaches to Vehicle Detection

Vehicle detection has been researched since the 1990s. In the past two decades, it is widely accepted that two kinds of vehicle detection methods are developed in chronological order. Traditional vehicle detection started in the 1990s [12]. Deep learning-based object detection started in 2014 [13] and then occupies mainstream status. In this section, some traditional and statistical methods including but not limited to object detection are reviewed and introduced first, and then deep-learning methods are reviewed.

2.2.1 Feature Descriptor

Features descriptors are representations of images or image patches that simplify the images by extracting useful information and removing unnecessary information. In general, a feature descriptor describes the current window in terms of a high-dimensional vector. The purpose of a feature descriptor is to extract unique features from an image so that the algorithm can detect the same object regardless of whether the scene changes or if the same object appears in another image.

Histogram of Orientated Gradients (HOG) [14] feature descriptor was originally proposed in 2005, and HOG was used to detect pedestrians at that time. HOG is advanced by its scale-invariant feature transform and shape contexts. To eliminate the influence of scale and illumination, HOG descriptor is operated on a dense grid of uniformly spaced cells and local contrast normalization is used to improve accuracy. Benefiting from the scale-invariant, HOG can descript objects in different sizes.

Local binary patterns (LBP) [15] is another type of feature descriptor. It compares the current pixel to each of its 8 neighbors. Where the center pixel's value is greater than the neighbor's value, "0" is written to a vector. Otherwise, "1" is written. By this process, an 8-digit binary number is computed for the current pixel. Then, for all pixels in the current window, the histogram which is a 256-dimensional feature vector of all 8-digit numbers is computed. Same to the HOG descriptor, machine learning methods are used for post-processing. In addition, other descriptors such as wavelet, Haar features, DPM features, PCA, SIFT, and SURF are proposed for better performance. Generally, the combination of features could improve the generalization ability of descriptors.

After feature extraction, machine learning such as K-nearest neighbors (KNN), support vector machine (SVM), AdaBoost algorithm, and so on are used to classify objects. Generally, the feature descriptor and machine learning methods traverse the full image to extract features. However, generally most part of the image does not include vehicles. To reduce the waste of computing power, region of interest (ROI) is used here for reducing workload and improving real-time performance.

2.2.2 Background Subtraction and Optical Flow

For traffic scenarios, some specified methods are introduced for reference, such as background subtraction and optical flow, in addition to object detection. In contrast to the methods mentioned above, these two methods can only detect moving objects with a static camera, which makes them ideal for traffic surveillance.

The background subtraction method [16] is widely used for detecting moving objects in static camera videos. By measuring the difference between the current frame and

the reference frame, also known as the "background image" or "background model", this approach can be used to detect moving objects. In general, the background image should be a representation of the scene without any dynamic elements. To be able to adapt to changing lighting conditions and geometry settings, it must be regularly updated.

The same as background subtraction, optical flow [17] can also be used to detect moving objects in static camera videos. With optical flow, each pair of adjacent frames in a video produces a vector field representing its velocities and directions. Moving objects can be detected using image proposals such as image thresholds, filters, and morphological transformations.

2.3 Deep Learning-based Approach to Vehicle Detection

Deep learning is a subfield of machine learning, which is an integral part of artificial intelligence (AI), and has been developed for about 50 years [18]. Applications of deep learning include natural language processing, computer vision, speech recognition, and various other tasks that require advanced pattern recognition capabilities. According to the algorithm classification, deep learning algorithms include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), Deep Reinforcement Learning (DRL), and Attention mechanism (Transformer). Among these algorithms, CNNs are the most commonly used algorithms in detection.

CNNs have gained significant success in object detection such as handwritten numeral recognition and ImageNet competition since backpropagation [19] was introduced in CNNs. Most deep learning methods for object detection are based on CNNs, and they

can be classed into 2 categories, one-stage algorithms, and two-stage methods. Two-stage methods generate all windows (bounding boxes) that include objects in the first stage, then localize, minimize the bounding boxes, and also classify the correlated categories of objects in the second stage. Two-stage methods include variations of CNN such as R-CNN [13], Fast R-CNN[20], Faster R-CNN[21], and so on. One-stage methods try to abandon generating region proposals and refinement. This type of method such as YOLO [22] series and Single Shot Detection (SSD) [23], realizes object identification and detection in one network for simplicity. Two-stage methods are superior to one-stage methods in accuracy. However, due to the complex structure, two-stage methods always require more time for inference, so one-stage methods are used for real-time tasks which are important in traffic surveillance.

Other than target-level object detection, semantic segmentation can be used for object detection too by assigning a label to each pixel in an image. Semantic segmentation is more accurate and precise than target-level object detection. VGG [24], GoogleLeNet [25], and ResNet [26] are some famous models in semantic segmentation. Both these networks have two major parts, convolution layers, and fully connected layers. ResNet is meaningful because it first proposes the residual block that can increase network depth hugely. Residual blocks enable layers to learn new and different information rather than information from encoded input and also overcome the vanishing gradients problem. Fully Convolution Network (FCN) [27] then exceeds all previous models by replacing fully connected layers with convolutional layers to output spatial maps instead of classification scores. These output feature maps are then upsampled by fractionally strided convolutions to assign pixel-wise labels.

The transformer architecture, originally proposed for natural language processing (NLP) tasks, has been applied in the Computer Vision field and has proven to be very effective for different complex tasks such as object detection, semantic segmentation, and AI-generated content. Transformers work on the principle of self-attention - the ability to attend to different parts of the input when producing an output. For images, a common approach is to divide the image into a set of patches and flatten each patch into a sequence of vectors (generally by CNNs), each representing a different part of the image. Then, the sequence of vectors is fed into the transformer.

A key part of the transformer architecture is the self-attention mechanism, which allows the model to weigh the importance of different parts of the input when producing an output. In the context of images, this means the model can focus on the parts of the image that are most relevant to the task at hand. The significant advantage of transformers over CNNs-based methods mentioned above is their ability to capture long-range dependencies in the input data. In the context of images, this means they can understand the relationships between distant parts of the image, which can be crucial for tasks like object detection. For object detection, the transformer can be trained to output a set of bounding boxes and class probabilities. Vision Transformers (ViT) [28] and DEtection TRansforme (DETR) [29] have showcased how transformer mechanism could be effectively applied in object detection.

2.4 Fisheye Camera-based Vehicle Detection and Tracking

The fisheye camera differs from the normal camera because of its large field of view. Generally, the fisheye camera has a 180-degree field of view so it can cover much more area

than the normal camera. Thus, fewer fisheye cameras are needed for traffic surveillance or self-driving vehicles. However, some problems are introduced in these application scenarios.

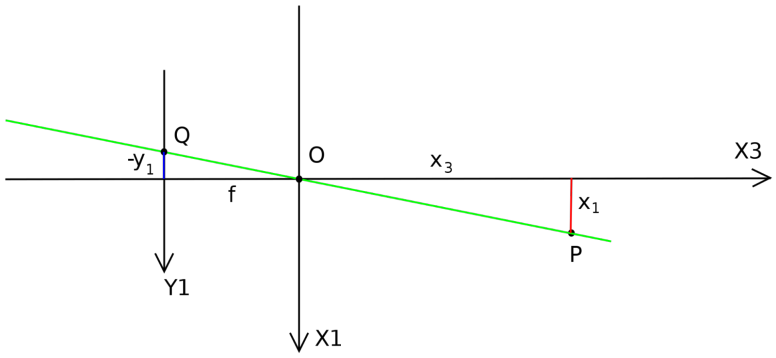


Figure 2.1: Pin-hole model of normal camera

The fisheye camera has a different camera model rather than the pin-hole model. In the pin-hole model Fig. 2.1, the intersection angle of incident light is the same as the angle of the projection line and optical axis. They follow that

$$\frac{-y_1}{f} = \frac{x_1}{x_3} \tag{2.1}$$

Most existing detection algorithms or datasets are based on normal cameras by default. One reason is that there have been a great number of normal cameras installed on the roadside for real-world applications. However, these algorithms can not be transferred to the fisheye camera directly due to model differences. Thus, it is necessary to review and study the fisheye camera model.

Commonly, there are two kinds of fisheye cameras. The first one is the circular fisheye camera and the second one is the full-frame fisheye camera. The only difference is the field of view. The full-frame fisheye camera crops the edge parts of the circular fisheye

camera so only the central part is left. In this study, the fisheye camera is referred to as the circular fisheye camera.

The fisheye camera model is different from the normal camera model as the angle of the projection line and the optical axis are different from the angle of incident light. The radius of the image point on the image plane has the following relationships to the angle of incident light. Thus, it enables a fisheye camera to capture edge incident light of 90 degrees incident angle into the field of view.

Here are the four widely used projection relationships for mapping the surface of a half sphere to the image plane. As shown in figure 2.2, the mapping function gives R, the position of the object from the center of the image, as a function of f, the focal length, and θ, the angle from the optical axis. θ is measured in radians.

- Equidistant Fisheye $R = f\theta$
- Stereographic $R = 2f \tan \frac{\theta}{2}$
- Orthographic $R = f \sin \theta$
- Equisolid (equal-area fisheye) $R = 2f \sin \frac{\theta}{2}$

The nonlinear projection relationship brings great distortion and related problems. Some model-based or learning-based methods such as YOLO, SSD, and Vision Transformer are not suitable for fisheye camera surveillance because of the lack of pixel-wised labeled or accurately labeled datasets and the distortion.

Another problem is that one object shows and scratches differently when it locates in different directions in the fisheye camera's view. In some fisheye camera scenarios, fish-

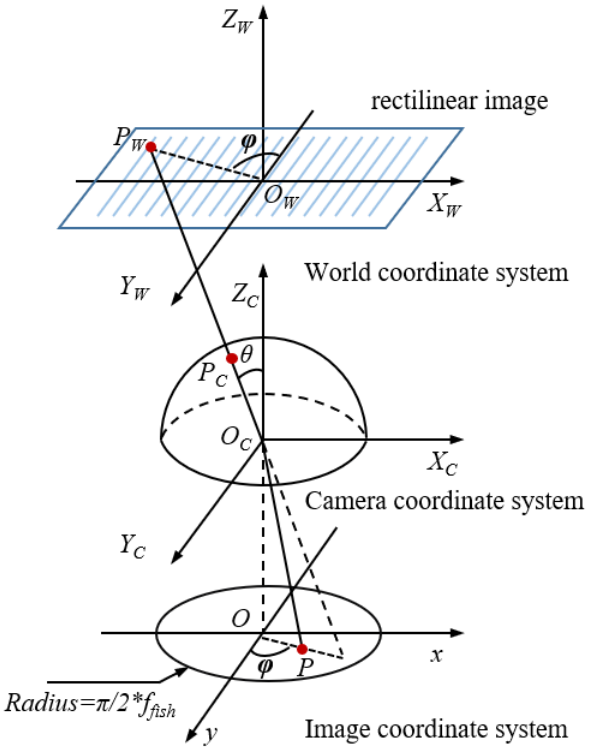


Figure 2.2: Fisheye camera model (adopted from [1])

eye cameras are mounted vertically so that they face the ground. In such conditions, one object in the left FOV will have a reverse scratch direction compared to the same object in the right FOV although it keeps the same direction in the real-world coordinate. Rather than the fisheye camera, the normal camera only has translation invariance. The perspective difference leads to the problem that some model-based networks cease to be effective. Also, some data augmentation techniques which are commonly used for model training lose efficacy too because of the translation variance.

The perception and detection methods for the fisheye camera are from those applied to the normal camera. Background subtraction is a kind of well-researched statistical method for object detection.

2.5 Style Transfer

Style transfer is a kind of software algorithm which manipulate digital images or videos for different appearances or visual style. Before CNN was brought into this area, the most relative research was image-based artistic rendering. Generally, two datasets are compulsory, one is the input dataset and another one is a target-style dataset.

The first kinds of example-based style transfer algorithms were image analogies and image quilting [30]. Both of them were based on patch-based texture synthesis algorithms, such as Stroke-based rendering (SBR), Region-based Techniques, Example-based Rendering, and Image Processing and Filtering.

SBR refers to a process of applying strokes (e.g, brush strokes, tiles, stipples) on input images. SBR algorithms define the style or strokes inside the algorithms so one algorithm refers to one style. They only need the input dataset and drop the style dataset, but they are not capable of learning arbitrary styles, which is inflexible.

Region-based Techniques exploit the style transfer based on content. They try to apply different strokes based on the shape of regions. The shape of regions is considered to classify the content of regions. Region-based Techniques try to synthesize or incorporate region segmentation, but they are still the same as SBR which one algorithm relates to one style.

Example-based rendering aims to learn a mapping between an exemplar pair. Different from SBR, it is a supervised-learning algorithm. The training dataset is composed of content images (unstyled) and the corresponding stylized images. Image-analogy algorithm learns the transformation from the paired training dataset. The need for the paired dataset

is critical and necessary. However, these kinds of datasets are unavailable in most times and scenarios.

Image Processing and Filtering borrow theories from signal processing. It considers images as signals and applies 2D Fourier transform, wavelet transform or other transforms on images. Then it applies filters on corresponding results on the frequency domain and combines different frequency results from the content dataset and the style dataset.

Later on, the neural network was introduced into this area. Leon Gatys proposed Neural Style Transfer (NST) [31] by borrowing VGG-16 from object recognition.

NST assumes input image x and style image y. NST tries to extract content and style separately. x is fed into the network first. $C(x)$ is the output sample, which is called the 'content' of the input x. Style image y is then fed into the network. $S(y)$ is the output sample, which is called 'style'. Content and style are sampled in different parts of the CNN activation so it is possible to extract content and style respectively. The synthesis image is p, and the target is $C(p) = C(x), S(p) = S(y)$. Here is the loss function $L(x) = |C(p) - C(x)| + k|S(p) - S(y)|$.

The main principle of NST is to separate content and style independently. It also assumes that arbitrary content can combine with arbitrary style.

Generative Adversarial Network (GAN) [32] was proposed by Ian Goodfellow and his colleagues in 2014 and also is used in style transfer. The main principle is that two neural networks (generator and discriminator) contest with each other, and one agent's gain is another agent's loss. Given two datasets and corresponding labels, the generator aims at generating fake images that look like real images and the discriminator aims at recognizing

the generated fake images. Here are 2 players: the generator G and the discriminator D. The generator's strategy set is $\mathcal{P}(\Omega)$, the set of all probability measures μ_G on (Ω). The discriminator's strategy set is the set of Markov kernels $\mu_D : \mathcal{P} \to [0, 1]$, where $\mathcal{P}(\Omega)$ is the set of probability measures on $[0, 1]$. The GAN game is a zero-sum game, with the objective function

$$L(\mu_G, \mu_D) := \mathbb{E}_{x\sim\mu_{ref}, y\sim\mu_{D(x)}}[\ln y] + \mathbb{E}_{x\sim\mu_G, y\sim\mu_{D(x)}}[\ln 1 - y] \tag{2.2}$$

Despite the self-contained principle and mechanism, the complex structure tends to fail because of asymmetry training and suffers the following problems.

- Non-convergence: the model parameters oscillate, destabilize, and never converge
- Mode collapse: the generator collapses and produces a limited variety of samples
- Diminished gradient: the discriminator gets too successful that the generator gradient vanishes and learns nothing
- Imbalance between the generator and discriminator causes overfitting
- GAN is highly sensitive to the hyperparameter selections

Apart from the problems mentioned above, GAN always learns "unwanted" styles from the background. Thus, to overcome these problems as well as for better transfer ability, attention mechanism [33] [2] is introduced in CycleGAN. Attention mechanism enables networks to distinguish foreground from background so

Attention mechanism has been proposed to improve the performance of encoder-decoder models for machine translation. As a result of the attention mechanism, the decoder

is able to utilize the most relevant parts of the input sequence in a flexible manner, by weighting all of the encoded input vectors, with the most relevant vectors being given the highest weights.

Recurrent Attention Model (RAM) [34] first introduced attention mechanism into vision. By applying a policy gradient, RAM embedded with attention mechanism recurrently predicts the important regions and updates the entire network. Later on, self-attention [33] was first proposed and rapidly provided great advances in the field of natural language processing (NLP). Then self-attention was introduced to computer vision by novel non-local network [35] with great success in video understanding and object detection. Self-attention shows its ability to distinguish target zones and their properties with attention scores. AttentionGAN [2] introduced self-attention into CycleGAN in order to better discriminate targets from the background and transfer the style of targets. The structure of AttentionGAN is shown in Fig. 2.3.

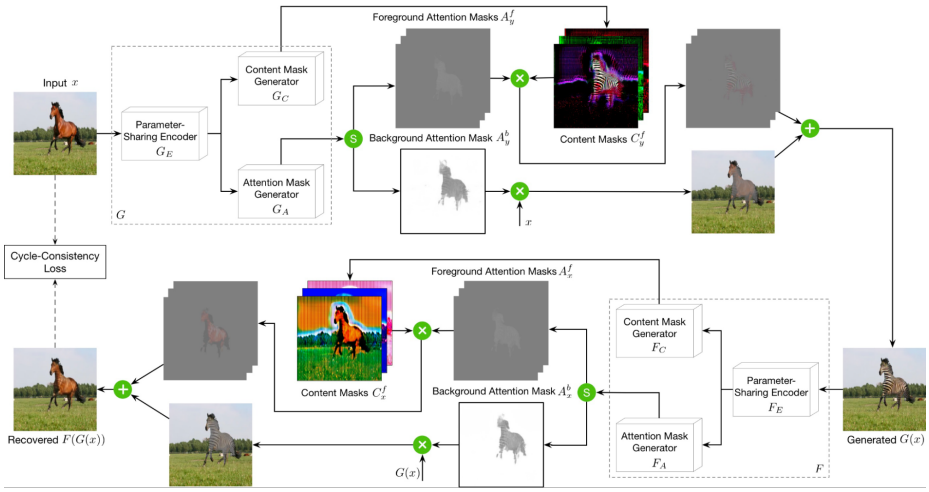


Figure 2.3: The structure of AttentionGAN, (adopted from [2])

Two generators and two discriminators are included in AttentionGAN. The pro-

posed generator in AttentionGAN could generate attention masks and content masks as shown in the figure. Generator G in the figure is composed of a parameter-sharing encoder G_e, an attention mask generator G_a, and a content mask generator G_c. G_e aims to extract both low-level and high-level deep feature representations. G_c aims to produce multiple intermediate content masks. G_a generates multiple attention masks. It should be noted that both attention mask generation and content mask generation have their own network parameters and will not interfere with each other. Also, generator G and F are independent so that they will not interfere with each other too.

The attention mask generator G_a aims to generate both $n-1$ foreground attention mask and one background attention mask. This configuration allows the proposed network to learn the novel foreground while preserving the background of input images at the same time. One of the key points of success of the proposed structure is that it generates both foreground and background attention masks. This allows the model to modify the foreground while simultaneously preserving the background.

It should be noted that the attention masks represent the weights of each content. Thus the pixel-wise addition of attention masks should be 1. They follow the equation

$$A_y^f = \text{Softmax}(mW_A^f + b_A^f), \quad \text{for} \quad f = 1, \cdots, n \tag{2.3}$$

where m is the feature map from Ge. The channel-wise softmax function used for normalization can guarantee that the channel-wise (pixel-wise) weight addition is 1. In the same way, content masks are reserved and extracted by Gc. $N-1$ content masks represent the foreground content. Content masks are generated by the equation

$$C_y^f = \text{Tanh}(mW_A^f + b_A^f), \quad \text{for} \quad f = 1, \cdots, n \tag{2.4}$$

In this way, the foreground content with the target style aligned with the original background is synthesized.

Not only attention mechanism contributes to style transfer, but the transformer network as a high-level structure of multi-head self-attention is also used in style transfer too. StyTr^2 [36] uses transformer encoders to separate style from content and generate stylized images by combining content with another style with transformer decoders. Also, content-aware positional encoding scheme (CAPE) is proposed to learn the positional encoding based on image semantic features and dynamically expands the position to accommodate different image sizes and guarantees the style-transferred content still keeps the same position as it is in the original image. Later on, STyle TRansformer (STTR) [37] tries to use ResNet-50 as the style/content extractor inversely and use transformer encoders to encourage style or content to group together.

2.6 Game Engine-based Driving Simulator

As various methods are introduced above, it is necessary to build a solid and powerful experimental platform to test methods. However, building a traffic experimental platform costs high and is insecure while testing some potentially unstable methods which propose a great need for driving simulators. Game engine-based driving simulators are programs that try to replicate the experience of driving a vehicle as realistically as possible.

Different game engines provide different advantages when it comes to creating

driving simulators. For instance, Unity and Unreal Engine are two popular engines used in the creation of these simulators. They both support advanced physics simulations and high-quality 3D graphics, which are essential for a realistic driving experience.

2.6.1 CARLA

CARLA is developed from the ground up to support the development, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely. The simulation platform supports flexible specifications of sensor suites and environmental conditions.

2.6.2 LGSVL

LGSVL Simulator is based on the Unity engine. The simulator has an API that enables passing messages between the simulator and an Autonomous Driving stack (AD stack) such as Cyber RT and Baidu Apollo, the two most popular open-source AD stacks. The API is responsible for communication between the localization in the AD stack and HD map inside LGSVL as well as the control system in AD stacks and vehicle simulation in LGSVL. Also, LGSVL support supports creating, editing, and exporting HD Maps.

Chapter 3

Vehicle Detection, Positioning and Tracking based on Background Subtraction

3.1 Introduction

Despite the emergence of deep learning-based CV approaches in recent years, conventional inference pipelines remain popular, especially for roadside sensing, mainly because they do not require labeled data and are more generalizable. One of the key steps along the pipeline is background subtraction (BS), which is intended to separate moving objects (i.e., "foreground") from stationary images (i.e., "background"). It should be noted, however, that most existing BS algorithms are not capable of reliably separating stationary objects from the background, which is a common scenario at signalized intersections. Additionally,

to meet the requirement for 24/7 traffic surveillance, a dynamic BS strategy is required to adapt to changing weather and lighting conditions.

Based on the Adaptive Frame Difference algorithm and modified Gaussian Mixture Model (GMM), we propose a cascaded adaptive background subtraction algorithm. The algorithm dynamically updates regions without objects and reserves regions with objects, resulting in fewer background trials. Compared to previous BS algorithms, our method tries to remove objects regardless of whether they are moving or not. As for the fish-eye camera, which has gained much attention recently, some feature-based methods, such as LBSP and LOBSTER, cannot separate the foreground from the background.

To validate the proposed algorithm, we apply it to both the CARLA simulation environment and the real-world testbed. The major contributions of this study are summarized below.

- Propose a hierarchical BS algorithm that can efficiently filter out temporally static objects frequently observed in traffic scenarios at signalized intersections.
- Compare with state-of-the-art BS algorithms using both simulation and real-world datasets.
- Create a fish-eye camera dataset with ground truth labels in CARLA simulator.

3.2 Methodology of Dynamic Background and Contour-based Detection

The key idea in this paper is to dynamically update the regions without objects and reserve the regions with objects. Thus, the mask of foreground and background should be used to choose which region needs to be updated. The masking strategy needs to be carefully designed to balance system performance for updates between moving and static objects, as well as for changes in illumination or environment. Some static objects will be included in the background if the masking strategy is too aggressive. Nevertheless, if the strategy is too conservative, some regions may remain the same even when the illumination changes.

3.2.1 Cascade Structure

To address this problem, we propose a hierarchical strategy based on both Adaptive Frame Difference (AFD) and Gaussian Mixture Model (GMM). Neither AFD nor GMM is capable of achieving satisfactory background subtraction results. The cumulative errors in AFD may result in some image faults that cannot be updated in some regions; therefore, a GMM may mix the objects and background together, and there may still be noise in the generated background. In order to overcome the disadvantages of each method, we combine these two methods into a cascaded structure, as shown in Fig.3.1, where AFD accounts for choosing the updated region while GMM is responsible for updating the region.

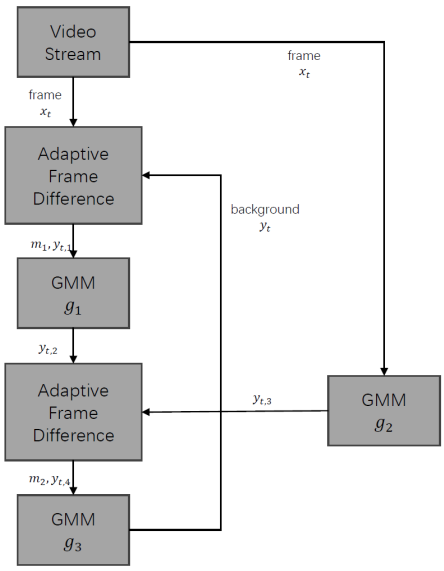


Figure 3.1: The cascaded structure

With the first (AFD) module, the primitive low-quality background y_{t1} is obtained with fewer foreground trails. In the second module, a GMM model g_1 is applied, which takes the mask and defective background as inputs and generates a relatively clean background y_t2. The third module is another AFD module, in which the output y_{t3} obtained from the second GMM model g_2 is used as the "background" for obtaining the mask. This module allows some mild blurring of shadows and vehicles as the result of g_2 keeps being updated directly. Its result has the same global color distribution as the ground truth. The trail region from y_{t1} is identified and updated with the background y_{t2} from g_1 by AFD. y_{t3} can bring more robust information to the background in the color space while keeping the detail in y_{t2}. The output y_{t4} is fed into the third GMM model g_3 and the final background y_t can be obtained. The detailed function of each module, i.e., AFD or GMM, is introduced in the rest of this section.

3.2.2 Adaptive Frame Difference (AFD) Module

The existing AFD methods only update the region without moving objects, which does not remove static objects. We, therefore, change it to update the region without objects. To obtain a mask of the foreground M, we compare the current frame I to the current background B and use threshold truncation to obtain a mask of the background M. It should be possible to exclude objects from the reverse of mask M_{bg}.

$$\begin{aligned} d &= (B^R - I^R)^2 + (B^G - I^G)^2 + (B^B - I^B)^2 \\ M &= Threshold(d) \\ M_{bg} &= \bar{M} \end{aligned} \tag{3.1}$$

Then the updated region and maintained region can be obtained and updated by

$$\begin{aligned} B_{new,bg} &= (1 - lr)B \cdot M_{bg} + lr \cdot I \cdot M_{bg} \\ B_{new,fg} &= B \cdot M \\ B &= B_{new,bg} + B_{new,fg} \end{aligned} \tag{3.2}$$

The AFD module is designed to remove objects. However, experiments indicate that the AFD module has a relatively low tolerance for illumination changes, noise, and moving objects. A high threshold will include the edges of some objects; a low threshold will not include camera noise and illumination. In order to compensate, a GMM module is applied following the AFD module.

3.2.3 Gaussian Mixture Model (GMM) Module

In the GMM module, N Gaussian distributions are used to model the distribution of pixels inside the image, and each Gaussian represents a different feature or texture within the image. The likelihood of a pixel being a background pixel is:

$$P(I_t) = \sum_{n=1}^{N} \frac{\alpha_n}{(2\pi)^{3/2}|\Sigma_n|^{1/2}} \exp^{-\frac{1}{2}(I_t-\mu_n)^T\Sigma_n^{-1}(I_t-\mu_n)} \tag{3.3}$$

where α_n is the associated weight for the n-th Gaussian component and $\Sigma\alpha_n = 1$. For computational simplicity, the covariance matrix, Σ_n, is set to be diagonal because the R,G,B channels are assumed to be independent.

The standard deviation and mean of the measurements are used to determine if the pixel is the background. If all channels' values (RGB) are within the threshold $||\mathbf{Z}_t - \mu_n|| < k\sigma_n$, the pixel is regarded as the background. Next, the model of the background is updated by

$$\begin{aligned}
\alpha_n^{'} &\leftarrow (1-\delta)\alpha_n^{'} + \delta \\
\mu_n^{'} &\leftarrow (1-\rho_n^{'})\mu_n^{'} + \rho_n^{'}\mathbf{Z}_t \\
\sigma_n^{2'} &\leftarrow (1-\delta)\alpha_n^{'} + \rho_n^{'}(\mathbf{Z}_t - \mu_n^{'})^T(\mathbf{Z}_t - \mu_n^{'}) \\
\rho_n^{'} &\leftarrow \sigma\mathcal{N}(\mathbf{Z}_t|\mu_n^{'}, \sigma_n^{'})
\end{aligned} \tag{3.4}$$

where ρ is an updating parameter and alpha is the user-defined learning rate. In this step, the weight of the background model is strength and the other models' weight is shortened by $\alpha_n \leftarrow (1-\alpha)\alpha_n$. If the pixel is judged as foreground, the models are initialized as current measurement $\alpha_n = \sigma, \mu_n = \mathbf{Z}_t, \sigma_n^2 = \overline{\sigma}^2$.

Algorithm 1: Algorithm for Modified GMM

Input: Initialized Background X_0, Current Video Frames x_t, Learning Rate lr

Output: Background y_t

if (start video) **then**

set $y_{t-1} = X_0$

end if

Calculate pixel-wise distance $d_1 = (x_t^R - y_{t-1}^R)^2 + (x_t^G - y_{t-1}^G)^2 + (x_t^B - y_{t-1}^B)^2$

Apply threshold function to get the binary mask. $m_1^{'} \leftarrow Threshold(d_1)$

Fill the inner holes in the mask and apply image morphology transformation on the mask. $m_1 \leftarrow MorphoTrans(m_1^{'})$

Update the background region by $y_t^1 = (1 - lr) \cdot y_{t-1} \cdot \bar{m_1} + lr \cdot x_t \cdot \bar{m_1}$, while the foreground region keeps as last background.

$y_{t,2} \leftarrow g_1(y_{t,1})$

For k iterations, $y_{t,3} \leftarrow g_2(x_t)$

Calculate pixel-wise distance $d_2 = (y_{t,3}^R - y_{t,2}^R)^2 + (y_{t,3}^G - y_{t,2}^G)^2 + (y_{t,3}^B - y_{t,2}^B)^2$

Apply threshold function to get the binary mask. $m_2^{'} \leftarrow Threshold(d_1)$

Fill the inner holes in the mask and apply image morphology transformation on the mask. $m_2 \leftarrow MorphoTrans(m_2^{'})$

Update the background region by $y_t^4 = (1 - lr) \cdot y_{t,2} \cdot \bar{m_2} + lr \cdot y_{t,3} \cdot \bar{m_2}$, while the foreground region keeps as last background.

$y_t \leftarrow g_3(y_t^4)$

return the background y_t

3.2.4 Contour-based Detection, Localization, and Tracking

Contour-based Detection

After adaptively generating the dynamic background, contours representing the detected objects can be calculated by subtracting the current frame from the background. Sometimes, the adjacent vehicles have conglutination which leads to bad detection. To further purify the detection result, post-processing is needed. First, the morphology transform is used to remove the burr, glitch, and some slight conglutination. Concave point detection is then used to eliminate severe conglutination. If a contour has more than one concave point and its size and shape satisfy the predefined threshold, the contour is identified as a conglutinated one. Finally, we connect the two most concave points so that the contour is cut into two sub-contours.

Localization

It is not possible to calculate the speed and orientation of the vehicles using only the detected position information because of the camera distortion and projection error. As a result, we use the average contour optical flow to represent the movement of the vehicles. The optical flow of feature points (e.g., SIFT or Shi-Tomasi corner), which leads to unstable results, should be tracked using a dense optical flow calculation. The dense optical flow calculation is computationally demanding and is not suitable for real-time applications. During this process, we downsample raw images in order to speed up the process without compromising accuracy.

As illustrated in the previous subsection, object detection is based on a contour representation of the object. In this case, the contour center can be determined by calculating the center of the minimum bounding rectangle of the contour. Ideally, if the vehicle is right under the camera, the contour center point (the yellow point in Fig.3.2) should align well with the chassis center point (the green point in Fig.3.2). In spite of this, we cannot use the contour center directly to determine the position of the vehicle due to the projection effect of the camera. As shown in Fig.3.2, the positioning error of a point with height h is given by $\frac{d_{perspective}h}{H}$, where H is the height of the camera. Assuming a vehicle is a cube, a BEV camera can see around half of the vehicle outline pixels on the ground, which does not have any positioning error (h=0). Therefore, the overall positioning error of a vehicle can be approximated by equation 3.5.

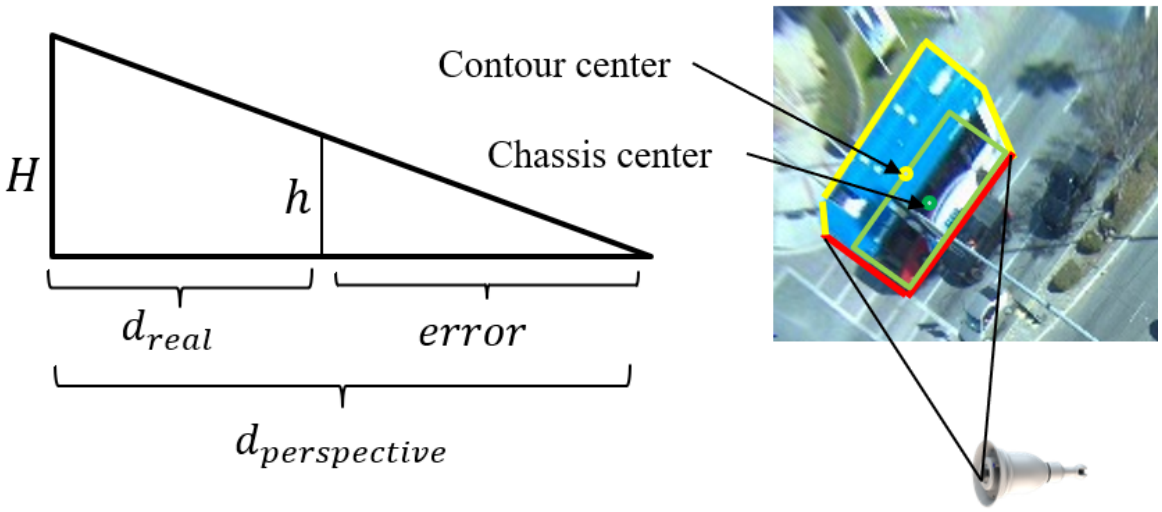


Figure 3.2: Chassis center estimation

$$\text{Position Error} = \frac{d_{perspective}h}{2H} \tag{3.5}$$

The average projection height of a vehicle h is approximated using equation 3.6

$$h = \frac{h_{max} - h_{min}}{2} |\cos(\theta - \varphi)| + \frac{h_{max} + h_{min}}{2}$$
$$(\theta, \varphi) = \text{card2pol(x,y)} \tag{3.6}$$

where θ is the position angle in the camera coordinate, and ϕ is the yaw angle of the vehicle calculated from the previous step. h_{max} and h_{min} are the maximum and minimum heights of a vehicle, respectively, which are the predefined parameters.

Point projection is the next step, aiming to project the corrected center points to the world coordinates using the calibrated camera parameters and the Haversine formulation. Based on the assumption that a road surface is a plane in 3D space, we project the points from the image plane into the plane of the road surface in camera coordinates. As a result, the Haversine formulation 3.7 gives the projection between two points' longitudes and latitudes as a function of their great-circle distance.

$$d = 2r \arccos\left(\sqrt{\sin^2\left(\frac{\varphi_1 - \varphi_2}{2}\right) + \cos\varphi_1 \cos\varphi_2 \sin^2\left(\frac{\lambda_1 - \lambda_2}{2}\right)}\right) \tag{3.7}$$

Tracking

Our object tracker is based on the DeepSort algorithm, one of the most popular methods for tracking objects in real time. Since the algorithm requires the bounding box of the detected objects as input, we calculate the minimum bounding rectangle of the contours. Due to this, all bounding boxes are assigned consistent IDs by the tracking module.

3.3 Experiment

In order to validate the performance of the adaptive cascaded BS algorithm, we conducted experiments both in a simulation environment and in the real world. Moreover, we focused on background subtraction on fish-eye cameras that are already installed on the real-world testbed in this study. Comparison of the background mask with the ground truth background is an effective method of measuring the performance of the system. However, it is difficult to obtain real-time ground truth at the pixel level in a real-world setting. Thus, we use the simulation-based dataset from CARLA for the quantitative analysis, where the ground truth of the pixel-wise segmentation is available.

For the real-world experiment, a fisheye camera is installed at a real-world intersection of University Avenue and Iowa Avenue in Riverside, California, to obtain a continuous video stream of traffic. In order to provide the broadest coverage of intersections and roads, the fisheye camera is angled vertically to the ground.

Furthermore, fisheye cameras have a unique field of view (FOV) and distortion that causes objects at the edge of the image to appear stretched. As a result of these features, feature-based methods also fail to perform well. It is imperative and necessary to calibrate fisheye images. Furthermore, wide FOV lens distortion calibration cannot be applied to fisheye cameras as they have a relatively narrow FOV (generally 120). In fact, fisheye camera calibration is mature and widely used. We use fisheye camera calibration in both the simulation dataset and the real-world dataset in our experiment.

3.3.1 Simulation Experiments

In order to examine the feasibility of the proposed algorithm, we first implemented it in a CARLA-based simulation environment. During the simulation, we used the Town10HD map provided by CARLA, which illustrates a typical urban arterial network with a few signalized intersections. One intersection was selected in this map for the purpose of collecting video frames at the traffic light where vehicles may stop to wait for the signal to turn green. In the simulation environment, 32 vehicles were randomly spawned on the map and traversed the intersection at random. Additionally, since CARLA does not have a ready-to-use fish-eye camera model, we adopted a transformation method called cubic2fisheye to create fish-eye video by combining multiple regular cameras from different perspectives. The frame rate is 10 frames per second and the resolution is 960 by 960 pixels.

Environment Setup: Cubic to Fisheye

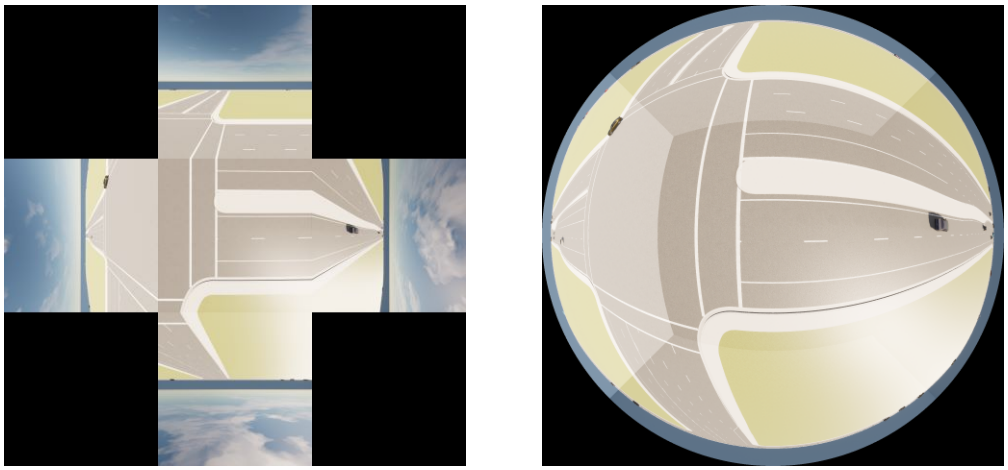
As shown in Fig. 3.3a, five cameras are placed at the same position at different angles in order to capture images of the surfaces of the cube. Each of these five cameras is located in the center of the cubic structure and faces the five sides, except for the bottom. To ensure that each camera's field of view (FOV) is not overlapped or deficient, we set the field of view (FOV) of each camera to 90 degrees, allowing the images from all cameras to be aligned. In this configuration, the cubic camera is able to obtain RGB information from every direction except the bottom one.

After that, we projected every pixel from the top half of the cubic surface onto a hemispherical surface. We then applied a polynomial fitting model to project this transfor-

mation onto a 2-D plane. Fig. 3.4 shows the relationship between azimuth angle and radius on the 2-D plane, which is considered a polynomial fitting model:

$$r(\theta) = k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + k_5\theta^9 \tag{3.8}$$

As is shown in Fig. 3.3b, the fish-eye image can cover 180 degrees of FOV. The outer blue circular ring is the sky which means this image includes an all-direction skyline.



(a) The sewed-up cubic camera image and there is no edge error because of 90-degree FOV

(b) Fisheye image

Figure 3.3: Cubic to Fisheye

Results and Analyses

To measure the detection and tracking performance of the proposed method, two experiments are needed. For the detection part, other state-of-the-art BS-based detection algorithms such as GMM, ViBe, PBAS, LOBSTER, as well as SuBSENSE are compared with this method at the pixel level. For object tracking, Multiple Object Tracking Precision

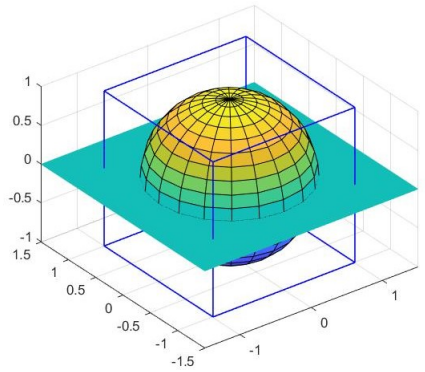


Figure 3.4: The cubic camera model and the hemisphere

(MOTP) and Multiple Object Tracking Accuracy (MOTA) are adopted as the metrics.

Fig. 3.5 shows a sample frame with a synthesized fish-eye image, ground truth label, and the BS results obtained using the proposed algorithm, the GMM algorithm, ViBe, PBAS, and LOBSTER. For the CARLA dataset, we adopted the confusion matrix as the metric. The generated foreground masks were compared with ground truth labels at the pixel level after applying the BS algorithms. The overlapped and non-overlapped pixels are obtained for each frame in order to determine the confusion matrix. Then the True Positive (TP), False Positive (FN), True Negative (TN), and False Positive (FP) are normalized and averaged across all the pixels in the dataset along the entire simulation time span. Other induced metrics, including precision, recall, FPR, FNR, and F-Measure are introduced for comparison.

- Precision (Pr): $TP/(TP+FP)$, larger is better
- Recall (Re): $TP/(TP+FN)$, larger is better
- False Positive Rate (FPR): $FP/(FP+TN)$, smaller is better
- False Negative Rate (FNR): $FN/(FP+TN)$, smaller is better

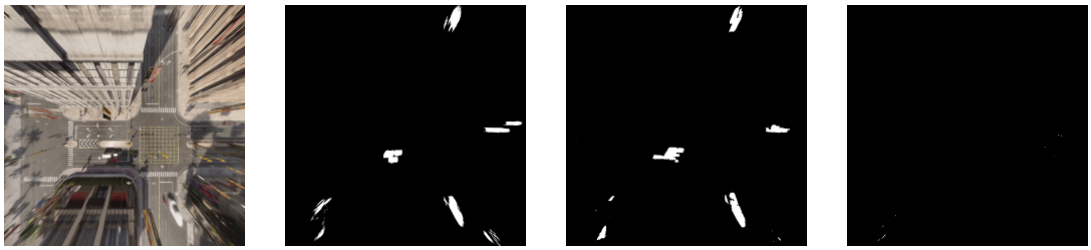
- Percentage of Wrong Classification (PWC/PBC) : $100 \cdot (FP+FN)/(TP+FN+FP+TN)$, smaller is better

- F-Measure ($F1$): $(2 \cdot Pr \cdot Re)/(Pr+Re)$, larger is better

By segmenting the fish-eye camera, the ground truth labels (foreground masks) exclude vehicles' shadows. Most BS algorithms cannot distinguish shadows from objects, so shadows are regarded as part of objects. It is for this reason that the FP values are so high.

Table 3.1 shows that our method is able to significantly retain and subtract static foreground, whereas other methods may recognize some objects as background. Therefore, our method has a lower FN and a higher recall than all other methods. Our method shows better TP and FN than SOTA (LOBSTER). According to one hypothesis, LOBSTER may be influenced by distortion from the fish-eye camera and may identify some foreground in the edge region as background since it is a feature-based method. It should be noted that this study conducted measurements at the pixel level. Despite the fact that the average precision is not very high, it is sufficient for applications requiring accurate detection and localization. As a result, our method outperforms LOBSTER in terms of F-Measure (a comprehensive metric) by 0.2096, demonstrating its superiority.

The tracking performance is shown in Table 3.2, where MOTA is measured in pixels. The method proper in this thesis achieves the best performance in all scenarios for all metrics. Also, because all other methods fail when vehicles stop at the intersection for the red light, MOTP and MOTA only measure the frame when vehicles are moving.



(a) One frame of CARLA dataset (b) Ground truth label (c) Our method (d) GMM

(e) ViBe (f) PBAS (g) LOBSTER

Figure 3.5: CARLA dataset and experiments

3.3.2 Real-world Experiments

In order to evaluate the performance of the system under dynamic illumination and weather conditions (e.g., windy), video clips were obtained from the fish-eye camera installed at the intersection of University Avenue and Iowa Avenue in Riverside, California. This video clip lasted for seven hours and was collected between the hours of early morning and late afternoon on February 21, 2022. Approximately half of the dataset was collected under cloudy conditions, while the remaining half was collected under sunlight conditions, so both shadows and sunlight variations are included. The frame rate is 10 FPS and the resolution is 960*960.

Fig.3.6 illustrates a sample frame taken at 35 seconds showing an undistorted image and the BS results of the proposed algorithm, GMM algorithm, as well as the ViBE algorithm. However, despite the challenge of quantifying performance with real-world datasets,

Table 3.1: BS-based detection performance on simulation environment

Method	TP	FN	TN	FP	Precison	Recall
Our Method	**1.242%**	**0.415%**	98.243%	0.618%	0.668	**0.750**
GMM	0.265%	1.426%	**98.665%**	**0.196%**	0.575	0.157
ViBe	0.509%	1.182%	**98.665%**	0.206%	0.712	0.301
PBAS	0.662%	1.029%	98.37%	0.489%	0.561	0.392
LOBSTER	0.616%	1.075%	98.665%	0.206%	**0.749**	0.364

Method	FPR	FNR	PWC/PBC	F-Measure	FPS
Our Method	0.006	**0.004**	**0.010**	**0.707**	8.569
GMM	**0.002**	0.014	0.016	0.250	23.898
ViBe	0.002	0.012	0.014	0.430	29.689
PBAS	0.005	0.010	0.015	0.470	8.44
LOBSTER	0.002	0.011	0.013	0.497	3.297

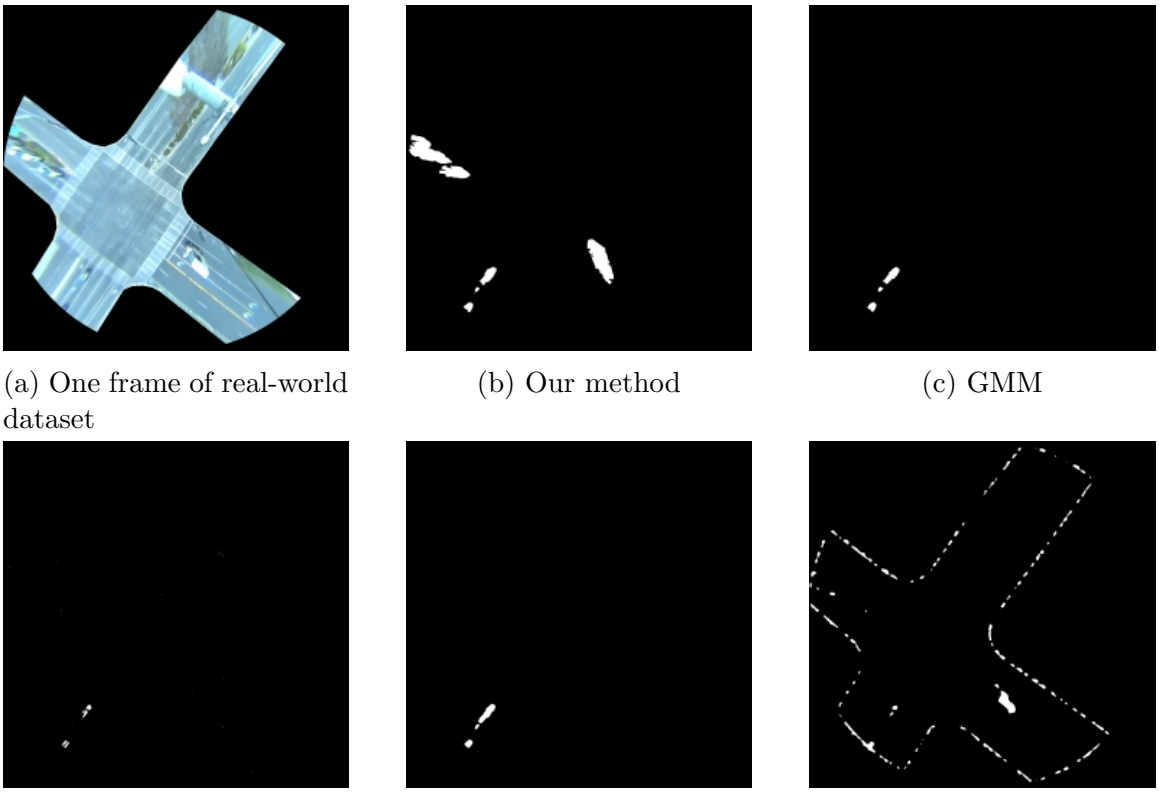
Table 3.2: Vehicle tracking performance on simulation environment

Method	MOTP	MOTA
Our Method	**82.11%**	**9.94**
GMM	29.04%	14.49%
ViBe	80.25%	10.49%
PBAS	49.06%	12.96%
LOBSTER	75.36%	10.68%

the visualization results indicate that our method remains reliable to identify vehicles waiting at signals, trees swaying in the wind, and rapid illumination changes as clouds move. Due to iterative updating, none of GMM, ViBe, PBAS, or LOBSTER can detect temporally static objects. They may consider such vehicles as background. The longer the vehicles wait, the more likely they are to be classified as background. Due to the lengthy initialization process, LOBSTER also includes some edge blobs in the foreground masks.

Fig.3.7 illustrates a sample frame of shadow variation after 5.05 hours and the

related BS results of the proposed algorithm. Using our method, shadow and illumination variations are successfully recovered into the background, along with separating vehicles from the frame regardless of whether they are moving or idling. Shadows can also be recovered with other methods, but many stationary vehicles or slowly changing vehicles are included in the background. Furthermore, our method can process real-world datasets in real time. GMM, ViBe, and PBAS can also meet the real-time requirement, while LOBSTER takes 3 times as long as our method.



(a) One frame of real-world dataset

(b) Our method

(c) GMM

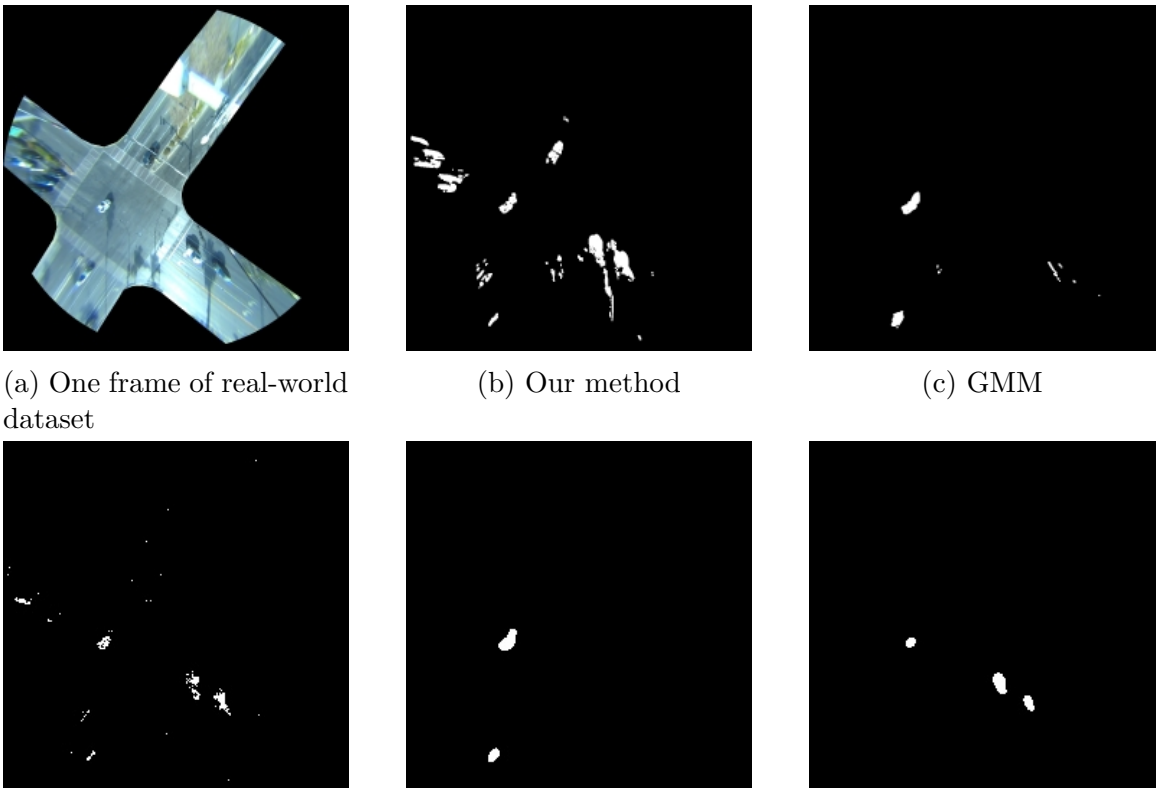
(d) ViBe

(e) PBAS

(f) LOBSTER

Figure 3.6: Real-world dataset and experiment results (at the 35th second)

Apart from the experiment about the background, we also do an experiment about



(a) One frame of real-world dataset (b) Our method (c) GMM

(d) ViBe (e) PBAS (f) LOBSTER

Figure 3.7: Real-world dataset and experiment results (at the 18,180th second)

the quantitative performance of detection. Since it is difficult to the ground truth position of all the vehicles in public traffic, the quantitative performance is only measured for the test vehicle. The test vehicle is a connected vehicle equipped with Real-Time Kinematic (RTK) GPS device. RTK system can achieve centimeter-level accuracy at the target intersection. Therefore, in this research, it is used for acquiring the real-world ground truth location.

Fig. 3.8 shows all vehicles' bounding boxes and locations in one frame and the memories of all vehicles' locations as trajectories with a static background. All vehicles' locations are searched for their one-to-one nearest locations in the last frame so as to track vehicles.

Fig.3.9 shows the collected trajectory of the ego vehicle from both the Dynamic Background detection and the RTK ground truth. As can be seen from the figure, the detection points (in red) align well with the ground truth (in blue). It should be noted that the average processing time of the RSPU is 90ms, and the communication delay from the RSPU to the connected vehicle is 160ms.

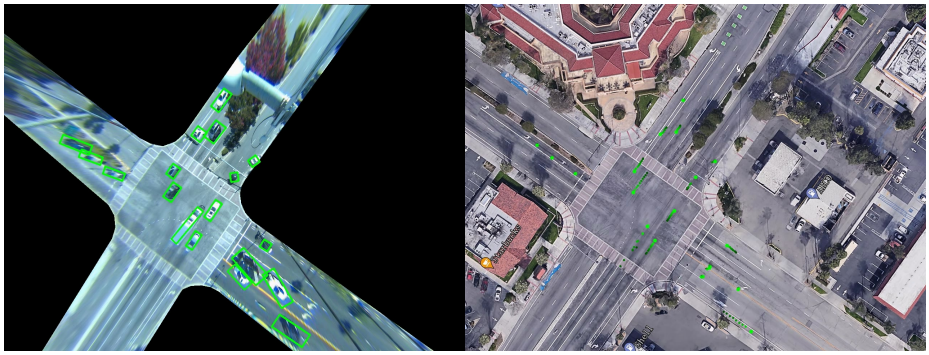


Figure 3.8: Detected bounding boxes and detected trajectories

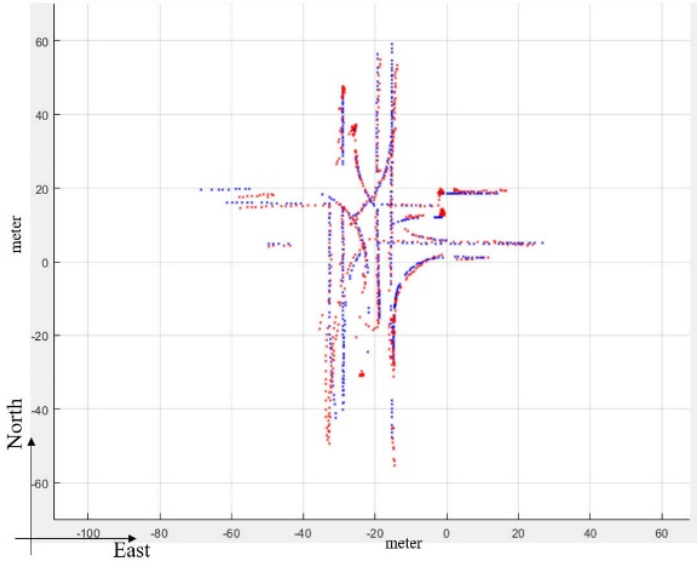


Figure 3.9: Trajectory of the ego vehicle, ground truth in blue and detection in red

The tracking performance is shown in Table 3.3. Detection accuracy is at its highest within a 10-meter range. However, as the detection range increases, the accuracy is expected to decrease due to radial distortion. This decrease in accuracy can be attributed to factors such as communication delay and the characteristics of the intersection being monitored. For instance, the average speed between 30-40 meters is higher than between 20-30 meters and 40-50 meters, as there is no stop line in the former range. As a result, the

Table 3.3: Quantitative results of real-world experiment

Range(m)	Overall	0 10	10 20	20 30	30 40	40 50	50
Accuracy (m)	2.28	1.22	2.11	2.85	3.02	2.16	2.31

detection accuracy within the 30 to 40 meters range is lower than in the other ranges.

Chapter 4

Vehicle Detection, Positioning and Tracking based on Deep Learning

4.1 Challenges of Deep Learning-based Approaches for Fisheye Camera

Although object detection based on deep-learning methods has been put into practice and gained much success on the normal camera, it fails to be transferred to the fisheye camera. The great distortion caused by the wide FOV of the fisheye camera leads to the failure that previous CNN-based methods can not be applied to detection because these methods assume that the geometric transformation of the camera is fixed and known. Meanwhile, rectangular bounding boxes are not appropriate to localize objects in the fisheye camera for the objects being stretched along the circumference. [38] proposes a generic polygon representation that can better approximate contours in fisheye images by polygon

representations in the loss function. FisheyeDet [39] designs a No-prior Fisheye Representation Method to extract valid distortion features and construct an effective fisheye feature pyramid adaptively. It also tries to integrate distortion features representations learning and tighter bounding box locations refinement into the detectors.

Another critical problem is the lack of annotated fisheye-camera dataset. FisheyeDet synthesizes a fisheye-camera dataset (VOC-Fisheye dataset) by transforming PASCAL VOC. However, the VOC-Fisheye dataset is transformed from a rectangle images dataset into round fisheye images dataset by radial distortion [40]. This transformation can not guarantee the same distortion along different radial directions. The distortion along the diagonal direction in rectangular images is greater than in other directions. Thus the simple transformation from normal images to fisheye images in this dataset is insufficient and untruthful.

To detect vehicles in fisheye images by deep-learning methods, a well-annotated fisheye-camera dataset is required for training. In this thesis, this target is realized by style transfer. For more detail and reality, the hemisphere model and polynomial projection model are used for synthesizing annotated CARLA fisheye-camera images which have been introduced in Chapter 3. It should be noted that all images are calibrated from fisheye-image into bird-eye view images and all follow-up works or processes are down in the bird-eye view images so that previous deep-learning methods can be directly used for training and inference without distortion in fisheye-images. Meanwhile, another real-world vehicle dataset also is obtained by the Dynamic Background mentioned above. After collecting these two datasets in CARLA and the real world, it is feasible to synthesize another dataset that looks like real-world vehicles but still has ground truth labels by style transfer. Then

a YOLOV5 detection network is trained by the synthesis dataset for real-world vehicle detection.

4.2 Style Transfer by GAN

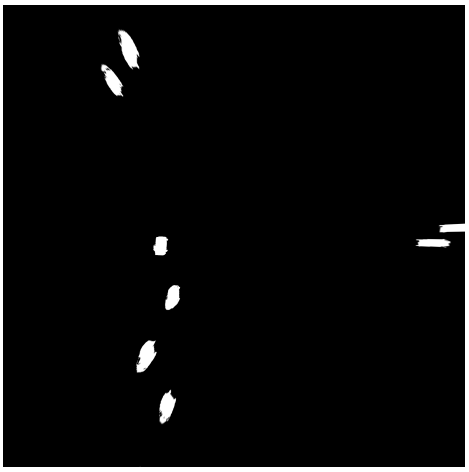
As the problem is the lack of annotated fisheye-images dataset, this section aims at synthesizing an annotated calibrated fisheye-images dataset by style transfer. Thus deep-learning detection methods can be directly applied.

4.2.1 Obtaining Dataset

It is necessary to obtain two symmetrical datasets with the same content but different styles for style transfer. For the simulation dataset, CARLA provides the segmentation camera which can directly save semantic segmentation images but only the normal FOV segmentation camera is available. In the same way, mentioned in the Chapter 3 experiment, fisheye semantic segmentation images are synthesized from five semantic segmentation cameras which have the same location and orientation as cubical cameras in CARLA. After calibration, pairs of images including the raw image as Fig. 4.1a and the semantic mask as Fig. 4.1b could be collected from CARLA. By simply covering up all other parts of the image, vehicle contours could be extracted from raw images as shown in Fig. 4.1c. For more reality, the vehicle contours images are all rotated in a direction that is the same as the real-world fisheye camera. Also, CARLA provides the APIs of all vehicles' locations and the length/width of the bounding box. After screening out vehicles out of image perspective and normalizing vehicles' location, ground truth labels of vehicles

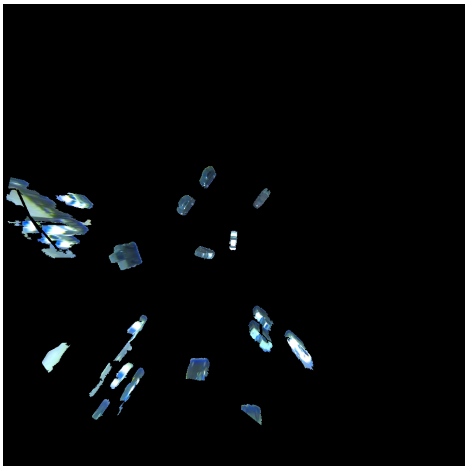
are collected for each calibrated image. Through this approach, the first CARLA vehicle dataset is obtained.

(a) One bird-eye view image of Carla dataset



(b) Semantic segmentation of bird-eye view image

(c) Vehicles of bird-eye view image



(d) Vehicles of real-world image

Figure 4.1: The process of obtaining CARLA vehicle dataset and real-world style vehicle dataset

Dynamic Background mentioned above can obtain the vehicle mask, so real-world vehicle contour shown in Fig. 4.1d can be obtained by covering up the ground in the same way.

Style transfer refers to a class of algorithms that manipulate images or videos to adopt the appearance or visual style of another image. Since GAN is developed in the form of a zero-sum game with two agents, its unique structure results in its good performance in style transfer. The two agents (generator and discriminator) may compete with each other for more detail and better style representation. Variations of GAN, such as DCGAN [41], PGGAN [42], StyleGANV1 [43], StyleGANV2 [44], and SAGAN [45] all focus on generating stylized images.

One milestone in the development of GAN is CycleGAN [3] as shown in Fig. 4.2, it firstly proposes a symmetric structure that includes two GANs (F and G) and introduces cycle consistency loss into GAN. Each GAN targets transferring style from style A to style B or from style B to style A. Apart from the traditional GAN loss function, CycleGAN designs cycle consistency loss of image x and the cycle generated image $F(G(x))$.

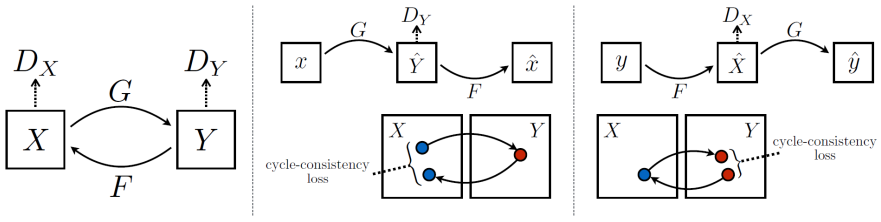


Figure 4.2: The mapping functions of CycleGAN, forward cycle-consistency loss and backward cycle-consistency loss (adopted from [3])

However, most CycleGAN variants [46] [47] [48] [49] [50] [51] can be easily affected by unwanted content and cannot focus on the most discriminative semantic part of images during the translation stage. Usually, unwanted contents are transferred style too which leads to bad visual effects while the loss function in GAN indicates the training has conver-

gence. Especially those methods focus on transferring the style of one main body in each image while this section focuses on transferring the style of multiple objects in each image.

In this study, multiple vehicles and a static background require the discriminating ability of style transfer algorithms. Thus, a technique of distinguishing the wanted part and the unwanted part is needed here. Attention mechanism which has been reviewed before fulfills the discrimination need precisely. AttentionGAN [2] introduced self-attention into CycleGAN in order to better discriminate targets from the background and transfer the style of targets. Based on AttentionGAN, one style transfer algorithm is introduced for transferring CARLA vehicles into the real-world side with ground truth labels.

4.2.2 Network Structure

The proposed method is based on AttentionGAN. AttentionGAN is derived from CycleGAN so it is symmetrical too. Two generators and two discriminators are included in AttentionGAN. The proposed generator in AttentionGAN could generate attention masks and content masks as shown in the figure. Generator G in the figure is composed of a parameter-sharing encoder G_e, an attention mask generator G_a, and a content mask generator G_c. G_e aims to extract both low-level and high-level deep feature representations. G_c aims to produce multiple intermediate content masks. G_a generates multiple attention masks. It should be noted that both attention mask generation and content mask generation have their own network parameters and will not interfere with each other. Also, generator G and F are independent so that they will not interfere with each other too.

The attention mask generator G_a aims to generate both $n-1$ foreground attention mask and one background attention mask. This configuration allows the proposed

network to learn the novel foreground while preserving the background of input images at the same time. One of the key points of success of the proposed structure is that it generates both foreground and background attention masks. This allows the model to modify the foreground while simultaneously preserving the background.

It should be noted that the attention masks represent the weights of each content. Thus the pixel-wise addition of attention masks should be 1. They follow the equation

$$A_y^f = \text{Softmax}(mW_A^f + b_A^f), \quad \text{for} \quad f = 1, \cdots, n \tag{4.1}$$

where m is the feature map from Ge. The channel-wise softmax function used for normalization can guarantee that the channel-wise (pixel-wise) weight addition is 1. In the same way, content masks are reserved and extracted by Gc. $N-1$ content masks represent the foreground content. Content masks are generated by the equation

$$C_y^f = \text{Tanh}(mW_A^f + b_A^f), \quad \text{for} \quad f = 1, \cdots, n \tag{4.2}$$

In this way, the foreground content with the target style aligned with the original background is synthesized.

However, the structure of AttentionGAN is much more complex and redundant for transferring the CARLA vehicle to the real-world side since the masks of vehicles are available in CARLA and the real-world side.

As shown in Fig. 4.3, the modified AttentionGAN takes a three-channel RGB image and its vehicle mask as input and outputs N-1 content masks, thus we fuse all of these masks and the input image to produce the final results. The generator architecture

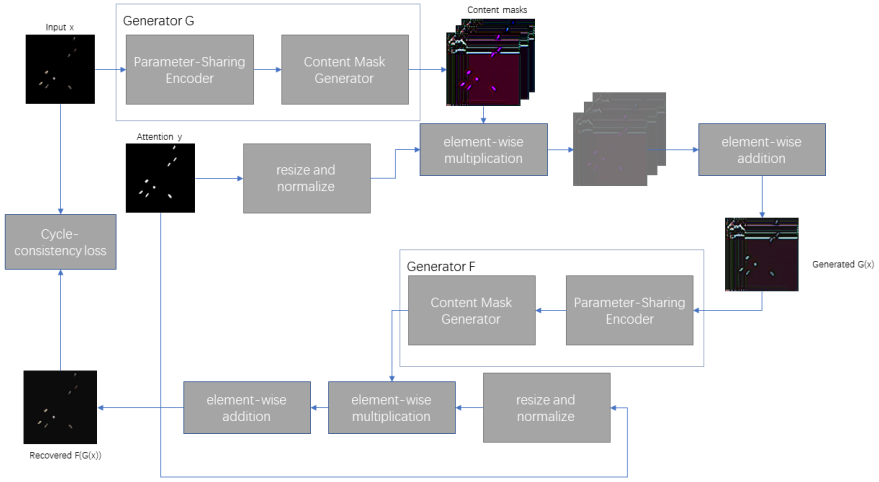


Figure 4.3: The structure of modified AttentionGAN

and the vanilla discriminator architecture are employed from CycleGAN.

The modified AttentionGAN simplifies the generator and abandons the attention mask generator. After reshaping and normalization, the vehicle masks collected from CARLA or the real world can be treated as attention masks of the foreground which are vehicles in images. It should be known that in this section, the background is omitted in style transfer but will be merged in the later synthesizing dataset process. The generated image follows this equation.

$$G(x) = \sum(C_x y) \tag{4.3}$$

4.2.3 Loss Function

To better regularize mappings, CycleGAN uses two-cycle generation. The main idea of cycle-consistency loss is that the image transferred from style A to style B and then back to style A, should be the same as the original input image, i.e., $x \rightarrow G(x) \rightarrow$

$F(G(x)) \approx x$. The cycle-consistency loss can be described by

$$\mathcal{L}_{cycle}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1] \tag{4.4}$$

Here, $F(G(x))$ and $G(F(y))$ are two cycles that are expected to match x and y in order.

Apart from the cycle-consistency loss, discriminator loss is required too. Discriminator is expected to distinguish whether an image is a style-untransferred or style-transferred image. Discriminator should output a higher score if the input image is a real image and output a lower score for the generated image. The adversarial loss can be formulated as follows

$$\begin{aligned} \mathcal{L}_{GAN}(G, D_Y) &= \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))] \\ \mathcal{L}_{GAN}(F, D_X) &= \mathbb{E}_{x \sim p_{data}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)}[\log(1 - D_X(F(y)))] \end{aligned} \tag{4.5}$$

In $\mathcal{L}_{GAN}(G, D_Y)$, G tries to minimize the adversarial loss, while D_Y attempts to maximize it. By contrast, G aims to generate an image $G(x)$ that resembles the images from domain Y, while D_Y aims to distinguish between the generated image $G(x)$ and the actual image Y. The overall loss function is shown here.

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda_{cycle} * \mathcal{L}_{cycle} + \lambda_{id} * \mathcal{L}_{id} \tag{4.6}$$

$\mathcal{L}_{GAN}, \mathcal{L}_{cycle}$ and $\mathcal{L}_{id}$ are GAN, cycle-consistency, and identity preserving loss [52], respectively.

4.2.4 Parameter Settings and Training

For the balance between time and generalization ability of transferred images, all datasets are reshaped to 256*256. The two datasets all have 20000 images for training and 2000 other images for the test. Adam optimizer with momentum terms $\beta_1 = 0.5$ and $\beta_2 = 0.999$ is used for optimization. $\lambda_{cycle} = 10$ and $\lambda_{id} = 10$. The epoch is set to 200.

The modified AttentionGAN is trained on one RTX3090 and it takes about 240 hours for training. It should be noted that the background is ignored in this network and some noises are generated, so another mask operation and background addition are applied here for synthesizing the dataset. Fig. 4.4 shows the generated images.

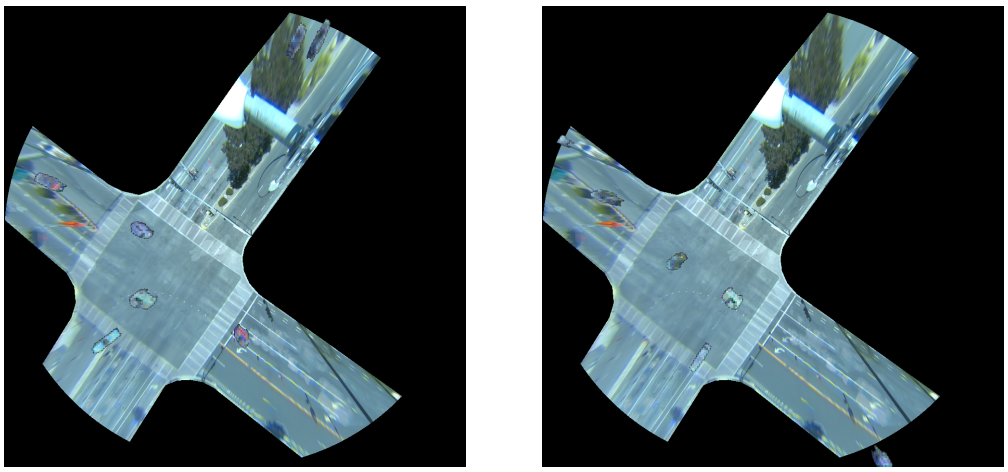


Figure 4.4: Two generated real-world side images form modified AttentionGAN

4.3 Detection

The generated dataset has a similar vehicle appearance to dataset generated from CARLA. Thus some deep-learning-based detection methods can be directly used here. In this section, YOLOV5 is trained from a pre-trained parameter. It should be noted that due to the rotation of datasets, the length and width of the bounding box are ignored here.

All parameters keep the same as YOLOV5 original settings. YOLOV5 is trained on RTX 3090 and it takes 30 hours for training.

4.3.1 Experiments and Analysis

Another same-source generated test dataset which is from modified AttentionGan is fed into the trained YOLOV5. Ignoring the orientation of the bounding box, all style-transferred vehicles are detected and assigned with a confidence score as shown in Fig. 4.5. The final experiment is to test the detection performance of YOLOV5 on the real-world dataset. The results are shown in Fig. 4.6.

Only parts of vehicles are detected, and most of them locate in the center of the image.

In order to calculate the detection rate, 200 frames are labeled manually and there are 2128 vehicles among 200 frames. YOLOV5 trained on the labeled real-world style dataset could detect 357 vehicles among frames. However, YOLOV5 trained on a labeled CARLA vehicle dataset can not detect any vehicle among frames. Compared to the 0% detection rate (True Positive) of YOLOV5 trained on the CARLA dataset, YOLOV5

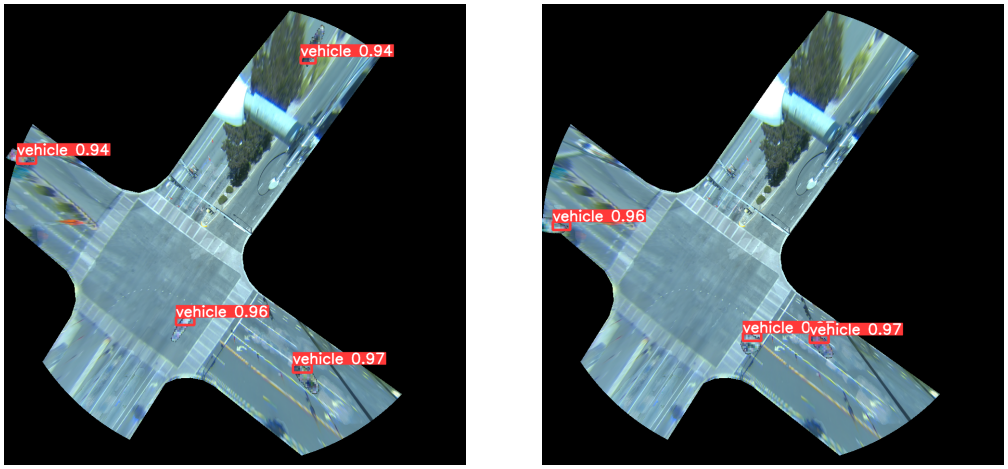


Figure 4.5: The results of YOLOV5 detection network on the test dataset, all generated vehicles are detected with independent confidence.

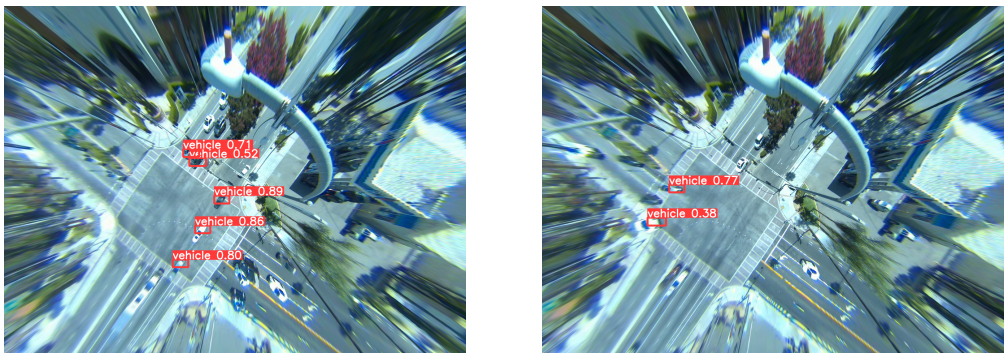


Figure 4.6: The results of YOLOV5 detection network on the real world dataset, only parts of vehicles are detected, especially vehicles in the central area.

trained on the labeled real-world style dataset successfully achieve a 16.78% detection rate (True Positive).

Although the YOLOV5 trained on labeled real-world style dataset is able to detect vehicles, it still misses most vehicles. One explanation is that although calibration eliminates partial distortion around the center parts, the great distortion from the real-world side is still left in calibrated images.

Another reason is the low generalization of current style transfer. Although style transfer tries to imitate the realistic vehicles' appearance, it fails to transfer CARLA vehicles with the severe real-world distortion style as well as the color shift form lens.

Also, the imbalance of CARLA dataset and the real-world dataset leads to low-quality generated vehicles. CARLA images always include fewer vehicles than real-world images. What's more, CARLA spawns less diversity of vehicles. Only 32 sedan vehicles are spawned in CARLA, while trucks; buses, and other kinds of vehicles appear in the real-world dataset.

Chapter 5

Conclusions and Outlook

5.1 Conclusions

This thesis focuses on infrastructure-based sensing and the fisheye camera detection methods and applies the latest advances in the field of Intelligent Transportation Systems (ITS) to achieve real-time high-fidelity vehicle detection and safety, high-efficiency ITS.

This thesis proposes a workflow for infrastructure-based sensing which is composed of two different methods. The first method applies background subtraction algorithms to vehicle detection in the infrastructure-based sensing scenario. Our modified background subtraction algorithm successfully solves the problem that all previous background subtraction algorithms could only separate moving objects from the background by cascade algorithm structure and introducing several time-series background models. Our algorithm shows its great ability on separating the foreground from the background and recovering the background although long-stay objects obscure the background in the simulation ex-

periment. It surpasses the SOTA method as LOBSTER by 34.68% on recall without too much loss on precision. At the same time, it can run in real time which the SOTA method can not. As for real-world experiments, the results show our method could maintain a clean background all day long and separate all vehicles on the road.

Without a labeled real-world dataset, it is impossible to train a deep-learning model to detect objects. The second method adopts style transfer by a modified AttentionGAN for data augmentation and then trains available deep-learning object detection algorithms. However, most existing style transfer algorithms can not identify the target objects so many unwanted background style is transferred. With the help of the attention mechanism, a modified AttentionGAN based on CycleGAN is able to generate a real-world style dataset with ground truth labels. The training datasets for the modified AttentionGAN are collected from the real world and the CARLA simulator. Then, a YOLOV5 is trained on the generated dataset from the modified AttentionGAN and then used for vehicle detection. The real-world experiments and results show it is able to detect parts of vehicles. Although some vehicles are not detected, the results still verify the fidelity of the whole process.

5.2 Outlooks

This section mainly lists future directions for the traditional vehicle detection method and the process of style transfer and vehicle detection based on deep-learning methods.

- The real-world vehicle dataset has blurry vehicles and conglutinations, so the background subtraction algorithm could be optimized for better-detecting vehicles. Also, this dataset still lacks diversity for some special kinds of vehicles missing. Thus, more different kinds of vehicles could be collected in the real-world vehicle dataset.

- The CARLA vehicle dataset also has the same problem that lacks diversity, so more vehicles could be initialized and collected in this dataset.

- The modified AttentionGAN can only generate low-resolution images due to the GPU memory size limit. It is expected to optimize the modified AttentionGAN structure to generate high-resolution realistic vehicle images.

- YOLOV5 was proposed in 2020 and other novel deep-learning-based detection methods are proposed continuously. It is expected to replace YOLOV5 with the new SOTA methods for better detection ability.

Bibliography

[1] Yaozu ye, Kailun Yang, Kaite Xiang, Juan Wang, and Kaiwei Wang. Universal semantic segmentation for fisheye urban driving images. pages 648–655, 10 2020.

[2] Hao Tang, Hong Liu, Dan Xu, Philip H. S. Torr, and Nicu Sebe. Attentiongan: Unpaired image-to-image translation using attention-guided generative adversarial networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–16, 2021.

[3] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.

[4] U.s. dot advances deployment of connected vehicle technology to prevent hundreds of thousands of crashes.

[5] Andreas Bger, Jan Wnent, Andreas Bohn, Tanja Jantzen, Sigrid Brenner, Rolf Lefering, Stephan Seewald, Jan-Thorsten Grner, and Matthias Fischer. The Effect of Ambulance Response Time on Survival Following Out-of-Hospital Cardiac Arrest. *Dtsch Arztebl International*, 115(33-34):541–548, 2018.

[6] Traffic count. `https://en.wikipedia.org/wiki/Traffic_count/`.

[7] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.

[8] Junyi Zhou and Jing Shi. Rfid localization algorithms and applications—a review. *Journal of Intelligent Manufacturing*, 20(6):695–707, 2008.

[9] Rikke Gade and Thomas B. Moeslund. Thermal cameras and applications: A survey. *Machine Vision and Applications*, 25(1):245–262, 2013.

[10] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008.

[11] Yingji Xia, Zhe Sun, Andre Tok, and Stephen Ritchie. A dense background representation method for traffic surveillance based on roadside lidar. *Optics and Lasers in Engineering*, 152:106982, 2022.

[12] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey, 2019.

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[14] N. Dalal and B. Triggs. 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 vol. 1, 2005.

[15] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996.

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

[17] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–466, 1995.

[18] C. M. BERNERS-LEE. Cybernetics and forecasting. *Nature*, 219(5150):202–203, 1968.

[19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[20] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[28] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[29] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[30] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review, 2017.

[31] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[32] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[34] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention, 2014.

[35] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks, 2017.

[36] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Xingjia Pan, Lei Wang, and Changsheng Xu. Stytr^2: Image style transfer with transformers, 2022.

[37] Jianbo Wang, Huan Yang, Jianlong Fu, Toshihiko Yamasaki, and Baining Guo. Fine-grained image style transfer with visual transformers, 2022.

[38] Hazem Rashed, Eslam Mohamed, Ganesh Sistu, Varun Ravi Kumar, Ciaran Eising, Ahmad El-Sallab, and Senthil Yogamani. Generalized object detection on fisheye cameras for autonomous driving: Dataset, representations and baseline, 2020.

[39] Tangwei Li, Guanjun Tong, Hongying Tang, Baoqing Li, and Bo Chen. Fisheyedet: A self-study and contour-based object detector in fisheye images. *IEEE Access*, 8:71739–71751, 2020.

[40] Jianglin Fu, Saeed Ranjbar Alvar, Ivan V. Bajic, and Rodney G. Vaughan. Fddb-360: Face detection in 360-degree fisheye images, 2019.

[41] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[42] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[43] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[44] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020.

[45] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2019.

[46] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation, 2017.

[47] Hao Tang, Dan Xu, Wei Wang, Yan Yan, and Nicu Sebe. Dual generator generative adversarial networks for multi-domain image-to-image translation, 2019.

[48] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks, 2017.

[49] Yaxing Wang, Luis Herranz, and Joost van de Weijer. Mix and match networks: cross-modal alignment for zero-pair image-to-image translation, 2019.

[50] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. 2017.

[51] Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping, 2017.

[52] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation, 2017.

[53] Kiyoung Shin, Ryan McConville, Oussama Metatla, Minhye Chang, Chiyoung Han, Junhaeng Lee, and Anne Roudaut. Outdoor localization using ble rssi and accessible pedestrian signals for the visually impaired at intersections. *Sensors*, 22(1), 2022.

[54] L.M. Ni, Yunhao Liu, Yiu Cho Lau, and A.P. Patil. Landmarc: indoor location sensing using active rfid. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).*, pages 407–415, 2003.

[55] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.